SLIG

Sign Language Interpreter Glove

Department of Electrical and Computer Engineering
University of Central Florida
Dr. Samuel Richie & Dr. Lei Wei

Senior Design I
Spring 2015

GROUP 24

Christopher Delgado	Christopher.Delgado@knights.ucf.edu	Electrical Engineer
Emmanuel Hernandez	Emmanuelh@knights.ucf.edu	Electrical Engineer
Jason Balog	JasonBalogucf@knights.ucf.edu	Electrical Engineer
Ramon Santana	Santana@knights.ucf.edu	Electrical Engineer

Table of Contents

1. Executive Summary

For its senior design project, Group 24 has come together to develop SLIG, a sign language interpreter glove. As the name suggests, this project is meant to be a glove that can be worn on a person's hand and have the ability to recognize the American Sign Language (ASL) signs the person may be signing. Whatever sign the glove has recognized will be displayed through the designated user interface. The project was chosen with two purposes in mind; the first being to help ASL speakers communicate with those who do not understand ASL and the second to help non-ASL learn and practice signing. SLIG will be made to the same specifications that characterize most modern electronics; i.e. it will be a lightweight, energy efficient and inexpensive device with wireless connectivity.

Through research, the team has developed a design that will allow the SLIG to function according to the objectives and requirements previously mentioned. SLIG will require the use of flex sensors, an accelerometer, a gyroscope and force sensors to capture all of the necessary information to identify each and every ASL sign for the English alphabet. All the information collected through these sensors will be processed by an MCU, which will be part of a PCB that will bring all the components together. The glove will be Bluetooth capable and will transmit all data to an Android smart phone application where the final interpretation will be displayed visually. A lithium-ion battery will power all the electronics on SLIG. The rest of the supporting electronics will include voltage regulators, analog to digital converters and others of that nature.

The team expects this project and the current design to be further influenced by a number of realistic design constraints. Among all the possible constraints, only a few will be highly relevant to the nature of this project. The most pertinent constraints will be from an economic, a health, an ethical and a manufacturing perspective. This project will after all be a glove meant to be worn by people and meant to facilitate interactions between an ASL speaker and someone who does not understand ASL. This brings about the health and ethical implications that will shape this project. The economic and manufacturing constraints are strongly in effect because this is a project run solely by engineering students who have limited resources, experience and connections.

Another influential force that has and will continue to shape SLIG are existing standards on the various technologies that the current design intends to employ. The standards the team has chosen to consider have been standards regarding Bluetooth technology, lithium-ion battery technology and android application creation. Although adherence to these standards is not mandatory, it would most likely be beneficial to the overall performance of SLIG. Thus, this document will guide the reader through the process of research and design that group 24 has followed in its endeavor to produce the best sign language interpreter glove possible.

2. Project Description

The sense glove is a lightweight, thin, Bluetooth-enabled glove that allows the user to translate the American Sign Language (ASL) sign of the letters of the alphabet to an external display. This glove is equipped with a series of flex sensors, an accelerometer, an embedded processor on a printed circuit board and Bluetooth technology that combine to give the user the ability to accurately communicate to any individual who doesn't understand sign language. The original motivation to pursue this project comes from one of our team members who has experienced the difficulty of communicating with his speech-impaired sister. This project got us thinking along the lines of wearable technology and opened the door for our extensive research in the area.

Our objective is to establish communication between a sign language speaker and a non-sign language speaker. Through the use of flex sensors and an accelerometer, any letter the user signs will be displayed through a user interface where the non ASL-speaker can read the letter.

2.1 Motivation

2.2 Goals and Objective

2.3 Requirements and Specifications

There will be some limitations and constraints that we need to be ready for if we want to make the glove lightweight, portable and energy efficient. There is so much data that we can collect out of the sensors, but making sure that the software and the machine both understand the data we are sending to them will be our greatest challenge. Anyone that uses the glove will do the gestures slightly differently, even if it's the same person. This means that the machine and the software both need to be dynamic enough to be able to make the correct command.

In order to accomplish this “smart” glove, we need to take into consideration how many parts we will need. We anticipate to use at least five flex sensors (one for each finger), one accelerometer since is just one glove, one microcontroller and one li-ion battery. Since the glove should be energy efficient, we want the user to just have to charge it for two hours for every 24 hours of normal usage.

Specifications:

- Color – Black
- Wearing Style - Right handed glove
- Dimensions - 151mm x 196mm x 33mm
- Glove Weight - 1.8 lb
- Connectivity - Bluetooth 4.1 (50 Ω antenna connector)
  - Wireless Transmission Range: 100 feet
  - Operating Temperature: -40 degrees Celsius ~ +85 degree Celsius
  - Operating Frequency Band: 2.4GHz ~ 2.48GHz
  - Operating Voltage: +3.3 V and +5.0 V
  - Microcontroller – Arduino (Dimensions: 6.8cm (L) x 5.3cm (B) x 2.3cm (H) / Weight: 24 grams)
- Battery Type: Lithium-ion
- Device Battery Life: 12 hours
- Flex Sensor
  - Flat Resistance: 25K Ohms
  - Resistance Tolerance: +/- 30%
  - Bend Resistance Range: 45K to 125K Ohms
  - Power Rating: 0.5 Watts continuous. 1Watt Peak

3. Research

3.1 Similar Projects

The group researched other similar Senior Design projects in order to determine how feasible it was going to be to accomplish the design of the Sign Language Interpreter Glove in just 2 semesters. The group was able to find previous senior design projects that used the same or very similar technology to the one that is required by the Sign Language Interpreter Glove. The group then narrow all those projects and focused on projects that related to the idea of a “smart” glove. This section will talk about two of the projects that the group though were the most interesting.

3.1.1 High Six

During the spring semester of 2014, a group of students from the University of Central Florida created a project called High Six. High Six consisted of three main subsystems which are hand gesture detection, Bluetooth communication and an android application. The High Six glove was able to interpret data being fed from several sensors and decide which letter of the American Sign Language was being gestured by the glove. In order for High Six to determine which of 26 American Sign Language letters was being gestured by the glove, it needed to distinguish the different parameters that make each of those letters unique. High Six was able to accomplish this task by using several sensors that would provide

the orientation of the palm, the hand shape being made and the movement of the hand.

The main difference between the High Six project and the Sign Language Interpreter Glove project is the fact that High Six was only able to translate letters. The Sign Language Interpreter Glove project will not only translate ASL letters, but also it will be able put this letters together to create words and put the words together to create sentences. The Sign Language Interpreter Glove project will also use other parts and components that we think might be best for the implementation of the SLIG design.

High Six was much expensive than expected, they estimated an initial cost of $518 but the actual cost was $919. The Sign Language Interpreter Glove project is expected to cost around $450, meaning it would be about half of the cost of the High Six project. High Six also offers a 20 hour battery life (from one charge), the Sign Language Interpreter Glove is shooting for at least 24 hours of battery life. Last but not least, the Bluetooth connection range was up to 50 meters, SLIG wants to double that distance to 100 meters. The main takeaway from High Six is that the project will take lots of time and dedication to complete. The group was able to see the hard work they put into the High Six design.

3.1.2 GloveSense

During the spring semester of 2011, a group of students from Boston University created a project where they were able to communicate with non-verbal gestures. Their goal was to create an electronic communication device that would silently send signals inside a building or across walls, through the use of hand gestures. Team GloveSense was able to send information to emergency personnel or military personnel with this technology. Team GloveSense emphasized there was a lack of reliable communication systems and said non-verbal communication was needed in order to help keep first responders out of danger.

Team GloveSense used National Instruments hardware and software in order for their project to be able to recognize and wirelessly transmit the hand gestures. Their project required to be capable of detecting movement, as well as the incorporation of a lightweight design. Their project also required a library of gestures and to be able to send signals over a long distance. This project is very similar to the Sign Language Interpreter Glove because SLIG is also a lightweight design that via hand gestures will wirelessly communicate by sending its signals over a far distance. The Sign Language Interpreter Glove will also include a library of gestures just like GloveSense. Basically both projects are pretty much using similar technologies that will accommodate different purposes. GloveSense focused on helping firefighters and police officers while the Sign Language Interpreter Glove focuses on helping the deaf community.

In order to accomplish their goal, they decided to split the project into two main parts. Their first design was a glove connected to a PC. To facilitate reading data, they utilized a supplementary board which also powered their aux components. To alert the users of an oncoming message, they used a small vibration motor. Their glove software was able to distinguish between finger gestures and hand gestures motions, then using the pre-computed library it determined what message to send. They even had an option to select to who send the message to over a ZigBee protocol.

Their final prototype consisted of a microprocessor that was used to process signals and provided wireless output. Combining the microprocessor with many different sensors together and you get a “smart” motion capturing glove capable of recognizing gestures using a LabVIEW software interface. Team GloveSense was so successful with their project, that they received the top prize award at their school. The P.T. Hsu Outstanding Senior Design Project was given to all the members of the team, Luke Anderson, Anna Evans, Patrick Henson, Jonathan Kwan, and Angelo Luo.

In conclusion, when it comes to technology, just about everything is getting smart. We have smart phones, smart watches, smart TV’s, smart cars, smart homes, smart wallets and the list goes on. This project shows that the concept of a “smart” glove is probably going to be added to that list very soon.

3.2 Hardware

3.2.1. Flex Sensors

3.2.1.1 Functionality

Flex sensors will be the primary sensors employed in this projects. They will be used to detect the degree to which each finger is bent on the hand performing the sign language. The combination of different degrees of flex for each finger will be the identifying mark for most of the letters. This will be the main method in determining which letter of the alphabet the user is trying to sign. For example, if the user wanted to form the sign for the letter 'A' he or she would have to completely bend down every finger except for his or her thumb. The flex sensors corresponding to these four fingers would increase their internal resistance as they are bent and in turn they would output a minimum voltage that could be measured and ultimately recognized as the signal configuration for the letter A.

3.2.1.2 Models and Specifications

In the market there is a plethora of different flex sensors available; each one with features that would benefit our project. For the scope of this project the chosen

flex sensors must meet certain requirements to ensure SLIG will be an efficient and manageable device. The proposed specifications are as follows:

- Flat Resistance: 25K Ohms
- Resistance Tolerance: +/- 30%
- Bend Resistance Range: 45K to 125K Ohms
- Power Rating: 0.5 Watts continuous. 1Watt Peak

SpectraSymbol Long Flex Sensor- This is a one-directional flex sensor with a base resistance (resistance when unflexed) of about 10Kohms. When the sensor is fully flexed, the resistance can increase to as much as 110Kohms. This can be connected to the analog input of a microcontroller or a digital input if a 0.1uF capacitor is used. It is reported to have a power rating 0.50 Watts continuous, a resistance tolerance of +/- 30%, a bending resistance range of 60K to 110K Ohms and a length of 112.5 mm. It is important to note this flex sensor is reported to have a fragile bottom piece where any unnecessary strain could rip the part away from its contacts. Below is a visual representation of how the flex sensor functions.

Figure 3.1: Bend range of Flex Sensor

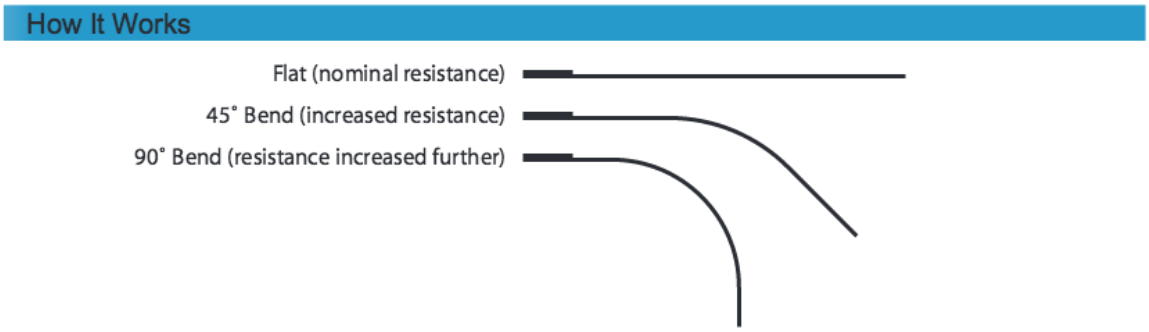


Figure 3.1 was reprinted with permission from SparkFun Electronics.

The main advantage of this flex sensor would be its wide availability, being sold by a multitude of vendors all at competitive pricing. It has been used in a multitude of different projects and application including but not limited to robotics, virtual motion gaming, medical devices, computer peripherals, musical instruments and physical therapy. The team will consider purchasing this product from either Digikey or Sparkfun, which are both well-established companies with reputable support services. Also, it is compatible with most microcontroller units.

Images Two-Directional Bi-Flex Sensor – This sensor has an un-flexed nominal resistance of 10 kilohms. This sensor has two leads that can be bent in both directions and still have its resistance changed. Unlike the one-directional flex sensors, bending the sensor actually decreases its resistance and correspondingly increases the voltages that would be measured. The sensor has a length of 4.5", width of 0.375" and a thickness of .038". This sensor's main advantage is that it is also pressure sensitive and may be used as a force or

pressure sensor. This could possible eliminate the need to purchase a separate pressure sensor as will be discussed further on. This particular sensor has been useful in many applications including collision avoidance on moving robots, virtual reality gloves and suits and physics experiments. Images Scientific Instruments may not be as well-known as the previous two vendors but it has been functioning for over thirty years and sells a wide range of instruments including gieger counters. Again, this sensor will definitely be compatible with any of the circuit boards and part we choose.

littleBits Bend Sensor – The littleBits bend sensor is like most other sensors. It activates when the long strip is flexed and an output signals are sent to the output bits. This exact model has been used in many student created projects including but not limited to 'Soccer Accuracy Trainer', 'Waste Paper basketball cheering machine', a dancing robot, a virtual table tennis opponent and a waiving hand. The manufacturer littleBits is completely geared towards small independent projects and has even made it is mission to support open source hardware. It is especially helpful in providing ideas on how to best use this flex sensor and other parts. Although, it may be optimal to match it with other littleBits parts it is also compatible with other parts.

Tactilus Flex – This sensor is designed with screen-printed resistive ink think films sensors that give the sensor the ability to repeatedly measure the degree of bending movements. According to experiments performed by the manufacturers, bending the sensor ninety degrees can generate 200,000 different voltage levels. They are intended to be integrated into an existing feedback and control system or to a multimeter or oscilloscope. The recommended applications include but are not limited to human body interface, biomechanics, air fluid flow and industrial controls. The manufacturers claim it has the following benefits over other technologies: it has a lightweight, thin and low profile, it is available at a fraction of the price of conventional actuators, it has a non-mechanical solid state and it is extraordinarily durable (>35 million cycles).

Each sensor is sequentially serialized and quality tested to minimize the chance of any defects and maximize the repeatability and durability of the product. On top of that the sensors make use of high quality Berg connectors, which should reduce the problem that many other sensors have with the base connection. The manufacturers claim they can build these bend sensor to the buyer's specific requirements. On top of all of these advantages, the manufacturer Sensor products has been in business for approximately 25 years and specializes in sensors starting with pressure and surface sensors.

Below is a summary of the standard specifications for this model.

Table 3.1: Physical Specification of Tactilus Flex Sensor

Physical Specifications	
Min and max bend:	180°
Query speed:	< 1 MHz
Durability	> 35 million cycles (0.24" (6 mm) radius 90° bend)
Material:	Polyimide or Polyester films
Temperature range:	-76°F to +221°F (-60°C to +105°C)
Gauge:	0.003" to 0.01" (0.076 mm to 0.254 mm)
Principle of leave:	Resistive ink with introduced micro fractures
Voltage:	3 VDC to 12 VDC
Accuracy:	± 1°
Repeatability:	± 3%
Hysteresis	7%
Non-linearity	±3% (subject to test conditions)

Table 3.1 was reprinted with permission from Sensor Products INC.

3.2.2 Accelerometer & Gyroscope

3.2.2.1 Functionality

Though flex sensors are a great way to capture many useful pieces of information from the physical state of the user's hand, they are limited in the range of motions that they are capable of sensing. The flex sensors can only detect how much bending they are experiencing and so they neglect motions like tilting the hand back and forth at different angles. Instead, accelerometers and gyroscopes can be used to measure this type of motion, which are crucial in identifying certain sign language letters such as "j" and "z". The two parts work in conjunction with one another where accelerometers can sense the orientation or tilt of the users hands and fingers while the gyroscope actually measure the angular motion of the wrist movements. Below are some figures that partially illustrate how an accelerometer and how a gyroscope would function.

Figure 3.2: Output Response vs. Orientation to Gravity

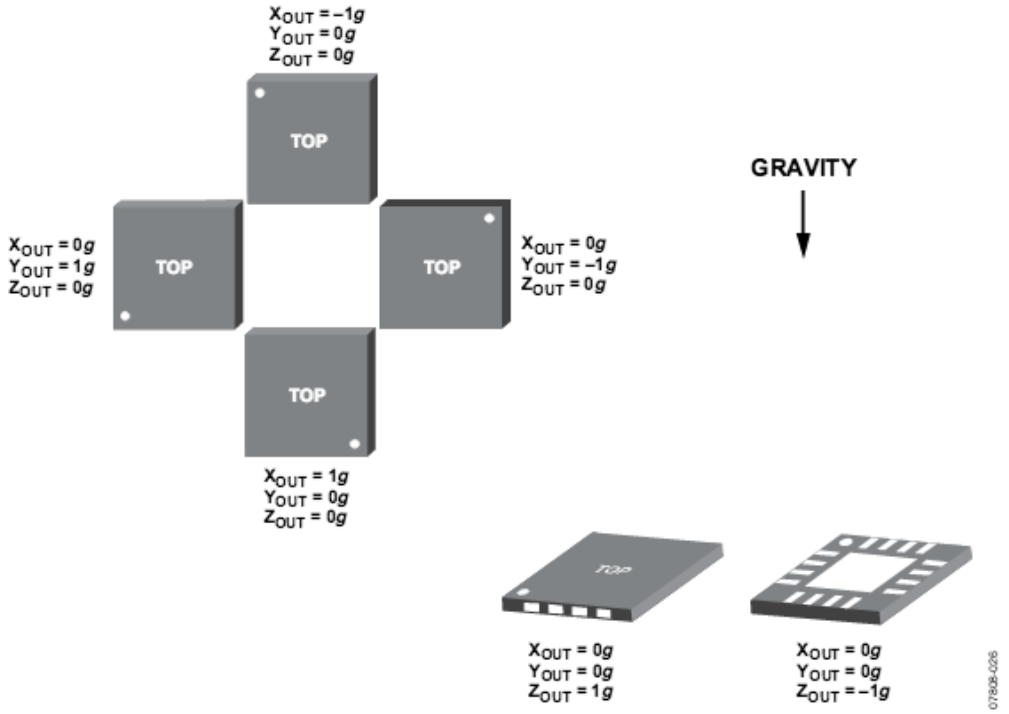


Figure 3.2 was reprinted with permission from SparkFun Electronics.

Figure 3.3: Pin Diagram showing measurable angular velocities

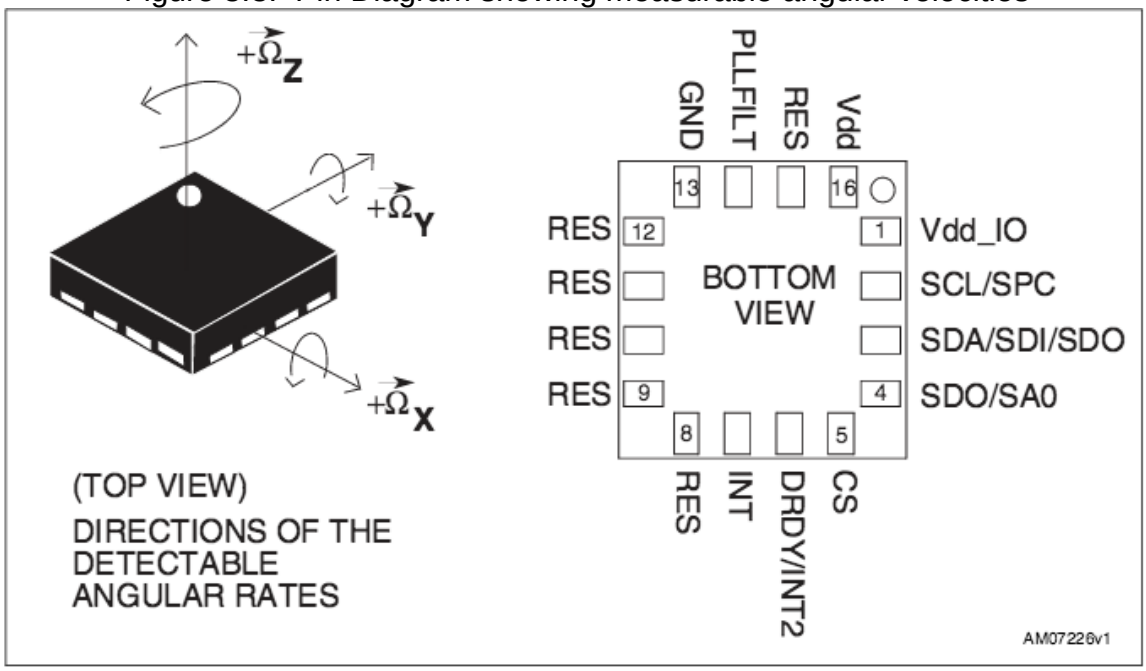


Figure 3.3 was reprinted with permission from SparkFun Electronics.

3.2.2.2 Models and Specifications

For the scope of the project the accelerometer we choose must meet certain requirements to ensure SLIG will be an efficient and manageable device. The candidate models are listed below.

SparkFun Triple Axis Accelerometer ADXL335 – This is a triple axis MEMS accelerometer from Analog Devices with a competitively low noise and power consumption - only 320uA. It also boasts a full sensing range of +/-3g. Because it does not have any integrated voltage regulation, the power provided should be between 1.8 and 3.6 VDC. The board brings 0.1uF capacitors that set the bandwidth of each axis to 50Hz. This specific model measures 0.7” by 0.7”. Accelerometers of its type has been used in a wide range of projects including collision analysis projects, guiding systems for mobile creations and a seemingless endless number of smart phone applications. The supplier as mentioned before is a reliable source with a multitude of accelerometers with different specifications. They also supply plenty of documentation and support and provide this component both on its own and as part of a breakout board to make integration flexible and straightforward. Below is functional block diagram of this accelerometer.

Figure 3.4: Functional Block Diagram

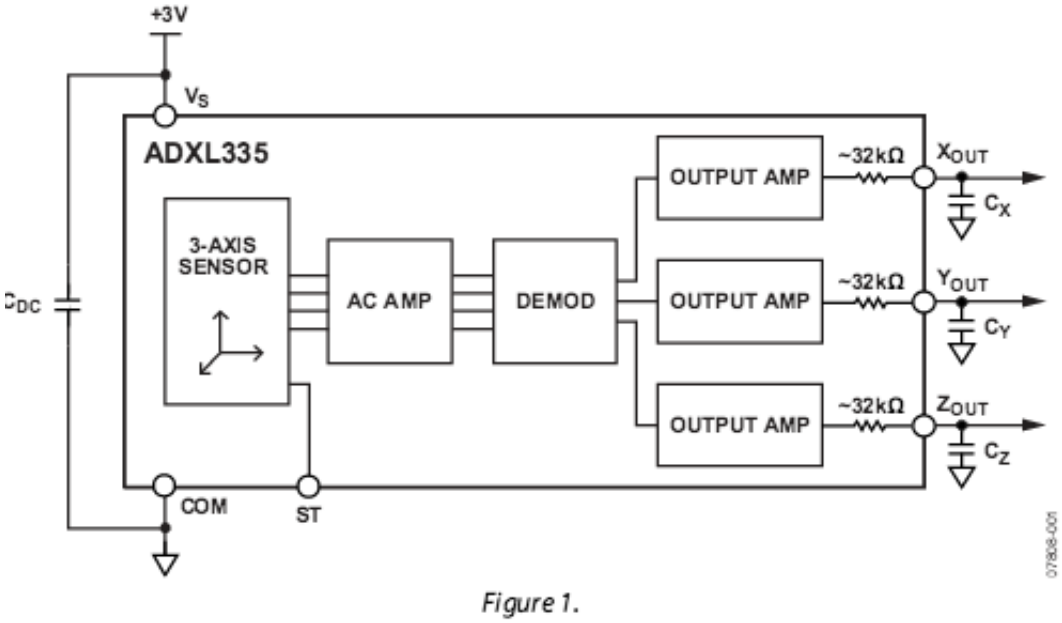


Figure 3.4 was reprinted with permission from SparkFun Electronics.

SparkFun Triple Axis Accelerometer ADLXL345 – This version of the SparkFun accelerometer features 2 standoff holes along with an extra decoupling capacitor. It is small, thin, energy efficient with a high-resolution measurement at up to +-16g. The output data is digital and formatted in a 16-bit two complement accessible through either a SPI or 12C digital interface.

This accelerometer effectively measures the static acceleration of gravity in projects that require tilt sensing and dynamic acceleration from movement. It can measure precise changes in inclination less than 1.0 degree. It also makes use of special sensing functions. It can detect the presence or lack of motion and if the acceleration in any direction passes a user-set maximum value. It can also detect if a device is free falling. All of these sensing functions can be mapped to up to two interrupt output pins. IT comes with a 32-level FIFO buffer designed to store data in order to reduce host processor intervention. Some of the top features are the 2.0-3.6 VDC supply voltage requirement, the ultra low power consumption of just 40uA in measurement mode, 0.1uA in standby mode at 2.5V, tap/double tap detection, free-fall detection and SPI and I2C interfaces.

SparkFun Triple Axis Accelerometer MMA8452Q – This model is a smart energy efficient, triple axis, MEMs accelerometer with 12 bits of resolution. It comes with certain functions including user programmable options that can be configurable to two interrupt pins. These functions are valuable because they enable power saving, keeping the host processor from repeatedly polling data. It has multiple dynamically selectable scales ranging from +/- 2g to +/- 8g.

Other important specifications include the 1.95 V to 3.6 V supply voltage requirement, an interface voltage of 1.6 V to 3.6 V, output data rates (ODR) from 1.56 Hz to 800 Hz, 12-bit and 8-bit digital output choices, I2C digital output interface (operates to 2.25 MHz with 4.7 kΩ pullup), two programmable interrupt pins for six interrupt sources, three embedded channels of motion detection, an orientation (Portrait/Landscape) detection with set hysteresis, high pass filter data available real-time and a current consumption of 6 µA – 165 µA.

SparkFun Triple Axis Accelerometer ADXL362 – This is another energy efficient model that possesses more or less the same functionality as the ones before. Some important specifications are the 3-Axis capability with selectable measurement ranges of ±2, ±4, or ±8g, its ultralow power consumption, an SPI digital interface, a high resolution of 1 mg/LSB, low noise as low as 175 µg/√Hz, a wide voltage range of 1.6 V to 3.5 V, an adjustable threshold for motion activation and the ability to select measurement ranges via SPI commands. Accelerometers of its kind have been used extensively in a wide array of applications including but not limited to Hearing aids home healthcare devices, motion enabled power save switches, wireless sensors and motion enabled metering devices.

SparkFun Triple Axis Accelerometer LIS331 – This is comparably low power full-scale linear accelerometer. The top features are a 2.16-3.6V input, the ultra low-current mode which can decrease consumption down to 10uA, selectable ranges of 6g, 12g, 24g, a I2C/SPI digital output and a16 bit data output.

Gyro Breakout Board IDG500 Dual 500°/s – This product is no longer available for purchase but is an excellent base model to which other gyroscopes can be compared. It measures angular velocity on two axes and has all necessary electronics built into one chip. The top specifications are the 3-7V single-supply operation, integrated X- and Y-axis gyros on a single chip, two separate outputs per axis for standard and high sensitivity on the X-/Y-Out pins of 500°/s, full scale range 2.0mV/°/s sensitivity X/Y4.5 out pins of 110°/s and full scale range 9.1mV/°/s sensitivity. Moreover, it features integrated amplifiers and low-pass filters, an auto-zero function, an on-chip temperature sensor, a high vibration rejection over a wide frequency range, a high cross-axis isolation by proprietary MEMS design, being hermetically sealed for temperature and humidity resistance and being 10,000 g shock tolerant.

SparkFun Tri-Axis Gyro L3G4200D – This is a more advanced model that has a greater degree of user customizability when it comes to measurements. The specifications for this part are three selectable full scales (250/500/2000 dps), I2C/SPI digital output interface, a 16 bit-rate value data output, an 8-bit temperature data output, a wide supply voltage ranging from 2.4 V to 3.6 V, a low voltage-compatible IOs (1.8 V), an embedded power-down and sleep mode, an embedded temperature sensor and a high shock survivability. Below is the block diagram for this gyroscope.

Figure 3.5: Block diagram for the L3G4200D

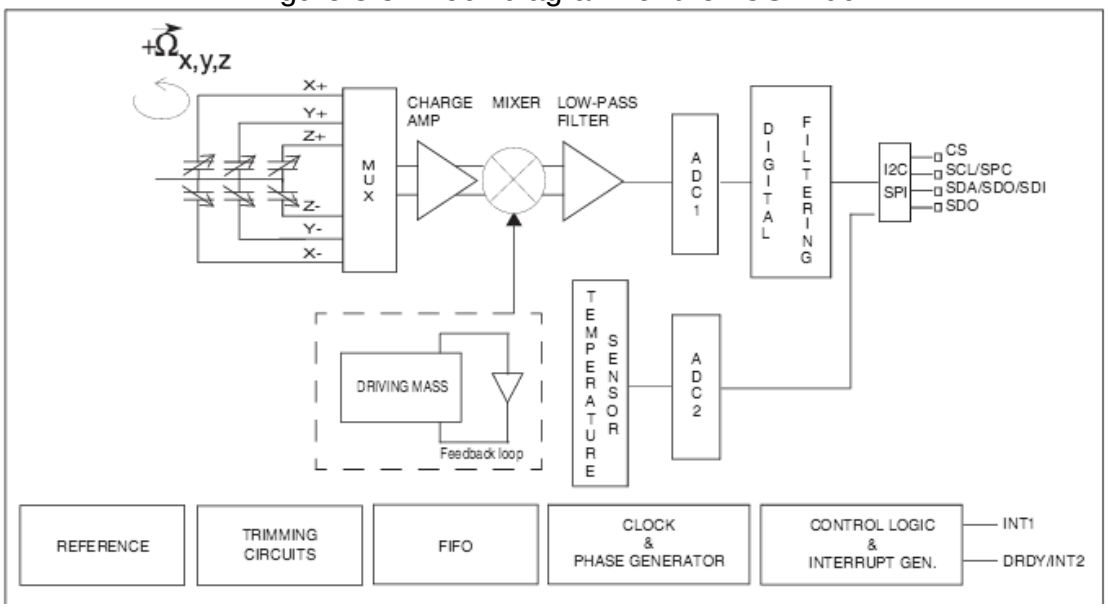


Figure 3.5 was reprinted with permission from SparkFun Electronics.

SparkFun 6 Degrees of Freedom IMU Digital Combo Board
ITG3200/ADXL345 – This board provides a full 6 degrees of freedom by combining the integrated accelerometer and gyroscope while keeping the size to a minimum.

SparkFun 9 Degrees of Freedom Sensor Stick – This option provides the functionality of an accelerometer, a gyroscope and a magnetometer all in one board. This part possibly provides more precision and functions than is necessary for our project but would definitely reduce any inconsistencies in measurements. It is a carefully designed combination of the ITG-3200 - triple-axis digital-output gyroscope, ADXL345 -13-bit resolution, ±16g, triple-axis accelerometer and HMC5883L - triple-axis, digital magnetometer fitted onto a single, flat board for a total of 9 degrees of Freedom.

Some of the top specifications are the fact the outputs of all sensors are processed by on-board ATmega328 and sent out via a serial stream, the autorun feature and help menu integrated into the example firmware, the output pins that match up with FTDI Basic Breakout, Bluetooth Mate, XBee Explorer, the 3.5-16VDC input and the ON-OFF control switch and reset switch.

3.2.3 Contact and Pressure Sensors

3.2.3.1 Functionality

Due to the fact there are a few pairs or groups of sign language letters that are not distinguishable by the degree a person's finger is bent nor any tilting motion, we will need to implement contact sensors or pressure sensors. The former can identify when two or more fingers are touching and may be precise enough to detect where along each finger the contact is being made. The latter in the form of piezoresistive force sensors are typically used to measure any type of applied force. However, it may be necessary to have these types of measurements for this project and not just determine whether two fingers are touching. Either type of these sensors will be helpful if not crucial in telling apart the following pairs of sign language: R and U, S and T and M and N.

3.2.3.2 Models and Specifications

For the scope of the project the contact sensors or pressure sensors we choose must meet certain requirements to ensure SLIG will be an efficient and manageable device. The candidate models are listed below.

Flexiforce Pressure Sensor 25lbs – Different levels of pressure created by any means, including pressure between fingers that are touching, will lower the sensor's resistance. These varying resistances can be used measured to

determine which finger is making contact. This model has resistances that range from infinite to about 300kohms and can measure from 0 to 25lbs of pressure. This a product sold by Sparkfun Electronics, which has a large inventory of sensor with a similar purpose. The following is a table detailing the physical specification of this model.

Figure 3.6: Physical Specifications of Flexiforce Sensor

PHYSICAL PROPERTIES

Thickness	0.203 mm (0.008 in.)
Length	191 mm (7.5 in.)* (optional trimmed lengths: 152 mm (6 in.), 102 mm (4 in.), 51 mm (2 in.))
Width	14 mm (0.55 in.)
Sensing Area	9.53 mm (0.375 in.) diameter
Connector	3-pin Male Square Pin (center pin is inactive)
Substrate	Polyester (ex: Mylar)
Pin Spacing	2.54 mm (0.1 in.)

Figure 3.6 was reprinted with permission from SparkFun Electronics.

Force Sensitive Resistor 0.5" – This would be the product that would precede a full-fledged pressure sensor. It works similarly to a pressure sensor and sends its output through two pins at the end. They are less expensive than the typical pressure sensor but are also less accurate and are best used to tell if they being pressed and not for exact measurements. For the scope of this project, this may be sufficient but actual testing is the only way to be sure. This part has an overall length of 2.375", an overall width of 0.75" and a sensing diameter of 0.5". The following two figures show the inner workings of this type of sensor.

Figure 3.7: The construction of an FSR

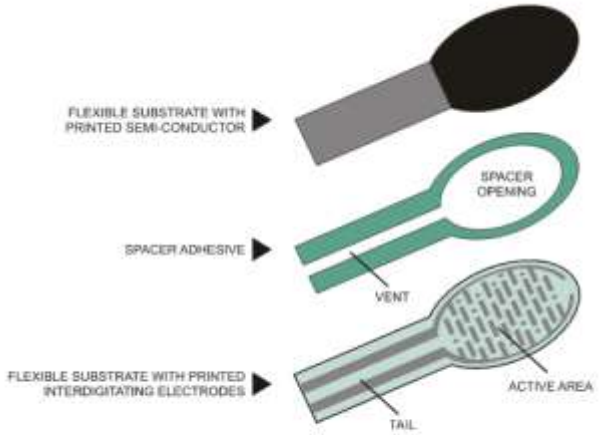


Figure 1: FSR Construction

Figure 3.7 was reprinted with permission from SparkFun electronics.

Figure 3.8: Force Applied Vs Resistance

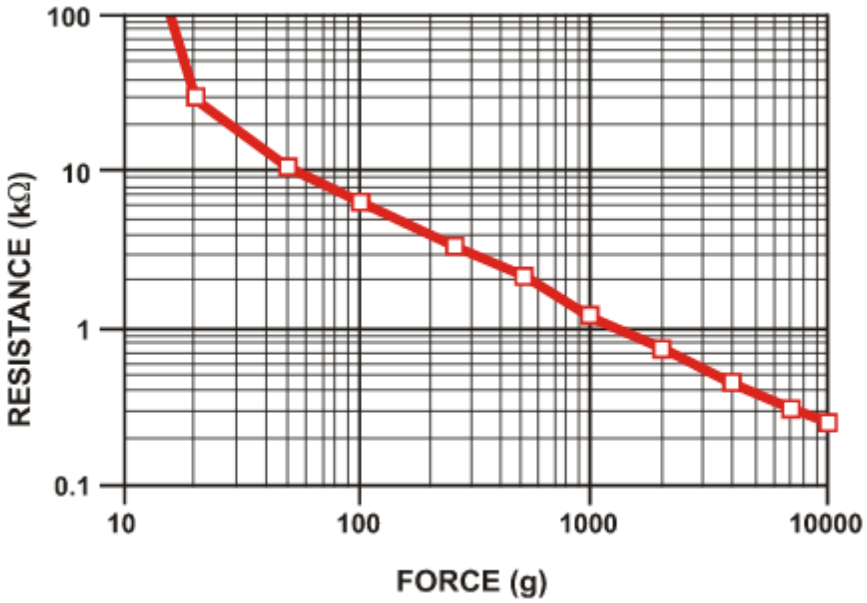


Figure 3.8 was reprinted with permission from Sparkfun Electronics.

Phidgets Touch Sensor – This is a capacitive touch sensor that can detect contact through different pieces of material such as plastic, glass or paper or in the case of this project the lining of the SLIG glove. It works well at close distances, detecting any object with half an inch of the board in any direction. Some of the most noticeable features are the recommendable material thickness of up to 1/2", attachability of the sensor to metallic objects to allow for a larger touch pad, the sensor's ratiometric nature and the standard 3-pin cable that comes with the sensor.

Phidgets Linear Touch Sensor – This sensor can measure changes in capacitance between electrodes on the device and the objects touching the board. It is meant to be mounted behind a sheet of glass or plastic close to an eighth inch in depth. It can alter its input value from 0 to 1000 in about 125 different steps as contact is made across its surface. This analog input isn't used unless the two digital inputs are set to one, which occurs only when contact is made and an object is in the close proximity of the board. This sensor also comes in a circular version, which may prove more useful for this project.

Phidgets Force Sensor – It is a typical force/pressure sensor in most ways. It has the capability of measuring forces relative to 3 kilograms. The output is relayed as a varying resistance value that spans all the way from 100kohm down to 1kohm. Its resistance is in a voltage divider arrangement with a 7.5K ohm

resistor. The part has the following physical dimensions: in length and width it measures 2.8 x 3.0 cm and it has mounting holes of 2.0 x 2.3 cm.

Softpot Linear Potentiometer – This is a series of linear potentiometers that come in a range of sizes including 50mm, 100mm, 150mm and 200mm. These are competitively thin variable potentiometers. Pressing down along any part of the sensor will linearly shift the resistance anywhere from a small 100ohms to 10,000ohms giving very precise information of the position where contact was made.

Figure 3.9: Electrical Specification

Electrical Specifications	SoftPot	ShieldedPot *(Preliminary data)*	HotPot
Resistance, **standard**	10kΩ (lenght >300mm 20kΩ)	10kΩ (lenght >300mm 20kΩ)	10kΩ (>300mm 20kΩ)
Resistance Range, customized	1kΩ to 100kΩ	1kΩ to 100kΩ	5kΩ to 100kΩ
Resistance Tolerance	± 20%		
Effective Electrical Travel	8 to 2400 mm	10 to 1200 mm	10 to 1200 mm
Linearity, independent	± 3% for rectilinear, ± 5% for rotary		
Power Rating (depending on size) (varies with length and temperature)	1 Watt max. @ 25°C	1 Watt @ 25°C	1 Watt @ 25°C
Resolution	Theoretical-Infinite; resolution dependant on contact wiper thickness and construction		
Dielectric Value	No affect @ 500 VAC, 1 minute	No affect @ 500 VAC, 1 minute	No affect @ 500 VAC, 1 minute

Figure 3.9 was reprinted with permission from SparkFun Electronics.

Softpot Rotary Potentiometer- This is the circular equivalent to the potentiometer above. The possible advantage of a potentiometer based sensor over the contact, force and pressure sensors discussed previously is that these will probably allow for the easiest way of pinpointing the exact place of contact. This will be useful to distinguish between the trickiest pairs of sign language letters where whether one finger is slight under the second calls for a completely different sign than if the finger was directly adjacent to it.

Figure 3.10: Dimensions of Rotary Potentiometer

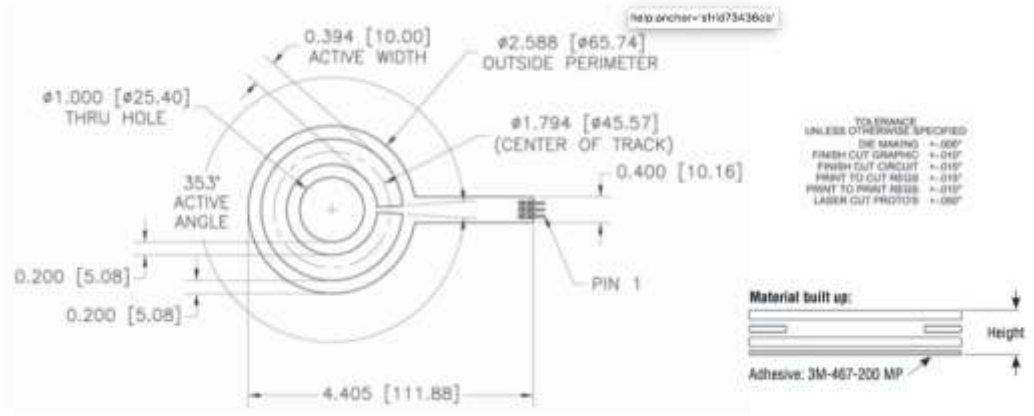


Figure 3.10 was reprinted with permission from SparkFun Electronics

3.2.4 Microcontroller Unit

There are many options available to us when it comes to choosing an appropriate microcontroller unit for the SLIG. The unit that is chosen will have to employ at least 6 analog input pins (with analog to digital converters for each), have digital outputs, and also be able to output information through the use of serial communication. When considering the microcontroller unit, it is also important to note that units that come with their own native programming environments, as well as pre-written functions such as the UART initiation functions would be overwhelmingly more convenient than a unit that does not employ these conveniences.

The group has considered several options, including Texas Instruments' MSP430, the ATMega32 series which is popular in Arduino development boards, as well as the popular Atmel AVR microcontroller. These microcontroller units all have their advantages and disadvantages, and those will be discussed at length in this section.

For the scope of this project, the initial leaning of the group was towards the MSP430 from Texas Instruments. In comparison to the next best option, the ATMega32, the MSP430 seems to have many significant limitations. For one, the MSP430 can only be used for developments on the Microsoft Windows platform and can only be programmed using the integrated development environment provided by Texas Instruments, Code Composer Studio. On the other hand, ATMega32 allows many forms of cross-platform development that includes development on Windows, iOS, Linux, and others.

This may seem like a big issue to when beginning to look into the subject, but upon further research the group decided on developing the entirety of the control system for the Sign Language Interpreter Glove on a Windows machine anyways, so this limitation is not a deterrent from using the MSP430. In fact, all four group members already have some experiencing developing on the MSP430 and using Code Composer Studio because they have all had to use this same setup while completing the laboratory experiments in the class, "Embedded Systems". However, the MSP430 has a significant advantage over the ATMega32 microcontroller in the sense that it consumes significantly less power than the ATMega32 does.

This is important because this glove will only have one battery powering all of the electronic devices. A processor that has a high power consumption will be inadequate for our purposes because we will need to allocate more power to the microcontroller and it will require a higher amount of power to be available for all of the circuits that will be employed in the project. From this standpoint, the MSP430 would be superior to the ATMega32 for the purposes of the project and what is waned from a microcontroller unit.

There are other factors in play that have helped tip the scales towards the MSP430 in favor of the ATMega32. Both microcontroller units operate at about the same speed of about 16 MHz, however the MSP430 has higher capabilities with this 16 MHz because it has double the size of the data bus than that of the ATMega32. The ATMega32 microcontroller unit is an 8-bit unit, while the MSP430 is a 16-bit unit. This allows significantly more data to be processed at a time while using the same exact clock speed. This can be significant because there will be a lot of data being processed simultaneously: there are five different flex sensors which will constantly be supplying data into the microcontroller unit, there is an accelerometer an gyroscope, which will also be constantly supplying data into the microcontroller, there are pressure sensors also supplying data. All of this leads to the importance of having more bits available so that every time that the clock 'hits', more data can be processed.

Even if the ATMega32 was a bit faster than the MSP430, it would have to be significantly faster in order for the clock speed to make up for the MSP430's ability to simply crunch much more data with each clock cycle. On the other hand, the ATMega32 has much more random-access memory available than the MSP430 does. The MSP430 has 512 bytes of random-access memory, while the ATMega32 has 2.5 kilobytes. This is a significant difference. However, the group does not plan on having to use the machine learning algorithm to decipher what hand gesture the user is trying to make, and rather use a more intuitive method by creating a function (described is sections above). Because of this, it is not necessary for the microcontroller unit that is used in the SLIG to employ a high amount of memory. If the machine-learning algorithm was to be used, more memory would be required because the machine learning algorithm would be based on storing many different iterations of the same gesture and referencing those iterations later on when the system is asked to make a decision on a current hand gesture input. For this reason, the group believes that the 512 bytes of memory offered by the MSP430 should be sufficient to supply the memory needs of the control system of the SLIG.

The biggest, and perhaps most significant factor in choosing between these two very comparable microcontroller units is the price. The ATMega32 is offered by many different vendors, for an average price more than double that of the MSP430. The ATMega32 is offered for an average price of about $25. The MSP430, on the other hand, is offered for $9.99. This is significant given that the group is on a slightly tight budget and would benefit greatly from having a microcontroller unit that is that much more affordable than the next option. For these reasons, the MSP430 is the microcontroller unit that the group chose, and the vendor of choice is Mouser Electronics.

It should be noted that the group already has a significant amount of experience developing on the MSP430 which should reduce the time that it takes to learn the specifics about the microcontroller as well as the development environment in which that microcontroller must be developed (Code Composer Studio). These,

as well as the other reasons mentioned above, make it so that the use of the MSP430 seems like the most reasonable choice to make. Below is a table illustrating the mentioned differences between the ATMega32 and the MSP430.

Table 3.2: MSP430 vs. ATMega32 comparison

Feature	MSP430	ATMega32
Analog Input Pins	8	12
Digital Input Pins	8	20
Random Access Memory	512 Bytes	2.5 Kilobytes
Data Bus	16 bits	8 bits
Speed	16 MHz	16 MHz
Cost/Vendor	$9.99 → mouser.com	$24.95 → ebay.com

As seen in the table above, the MSP430 seems to be the most reasonable option for the purposes of the Sign Language Interpreter Glove. It is significantly cheaper, it has double the size of the data-bus than that of the ATMega32, and all members of the group are intimately familiar with it. Although the ATMega does have more RAM than the MSP430, that is not enough of a deterrent to make the group go against the MSP430.

Figure 3.11: MSP430 Functional Block Diagram

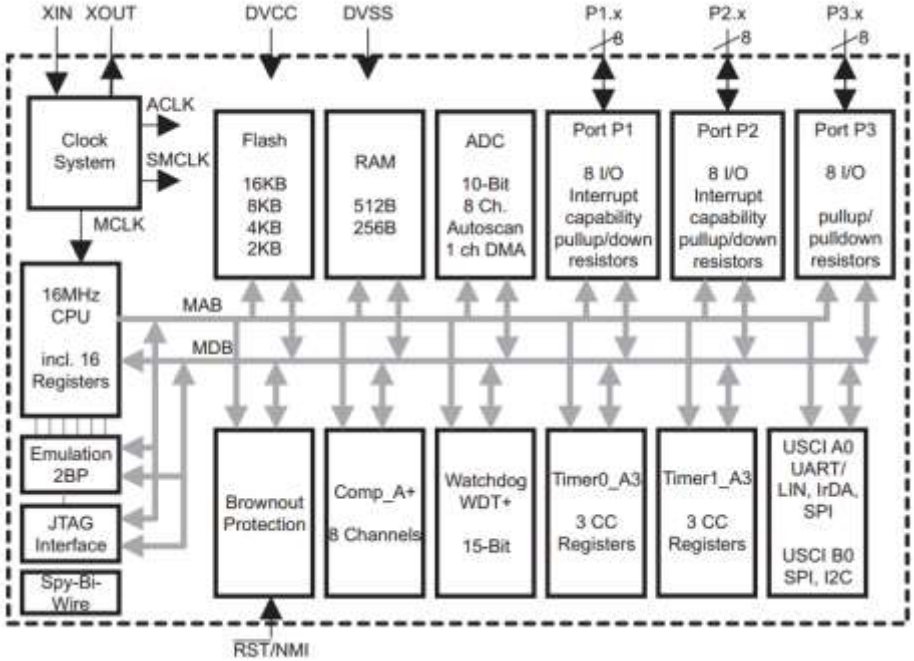


Figure 3.11 was reprinted with permission from TI.com

3.2.5 Wireless Communication

This section talks about the different types of wireless communication that were researched by the group, how this different types of wireless communication works and which of the thousands of wireless communication technology the group decided to use for the implementation of the Sign Language Interpreter Glove. The group would like to connect to an external display without the use of any cables or wires, because the group wants the project to have a clean and modern look. Using cables or wires could also be a bad idea because the cables or wires are at possibility of bending or disconnecting and might cause the project to fail. Therefore, it is a must that the Sign Language Interpreter Glove uses some type of wireless technology to establish the communication between the glove and the external device.

As stated earlier, there were so many different options available in the market that the group could of have chosen for this project, when it comes to deciding what type of wireless technology to use. Through researched, the group was able to narrow down the options to just three choices. These three choices were, Near Field Communication also known as NFC, Wi-Fi and Bluetooth. Keeping in mind that efficient and portability are the two biggest factors that is required by the Sign Language Interpreter Glove, the group was able to make a final decision in which technology would be best for implementation of this design.

3.2.5.1 Wi-Fi and Li-Fi

Wi-Fi, also known as Wireless Fidelity or WLAN ‘Wireless Local Area Network” is one of the most popular types of wireless communication used today to transfer and receive data, like surfing on the web. Wi-Fi uses radio waves frequencies that operate between the 2.4 - 5 Gigahertz ranges. The use of high frequencies is use to reduce the possibility of interference with other devices like a car radio, mobile devices or even walky talkies. Also high frequency is use so that Wi-Fi can provide more data at faster speeds.

Wi-Fi signal could be interrupted or the speed could lower if other devices connect to the same router. The solution to avoid other devices to connect to the same router, is giving the user the option to create a password. One of the most secure ways to protect a Wi-Fi network is via WPA2 which stands for Wi-Fi Protected Access 2. WPA2 gives the user the power to control who connects to their network and at the same time it has an encryption mode to provide the user with extra security. Bluetooth devices, security cameras, cordless phones, and many more devices can also cause a significant amount of interference to Wi-Fi. The Wi-Fi range for indoor use is about 70 feet since wall also cause interference, but it’s range is much greater when use outdoors.

Li-Fi, is a new type of technology that will soon be available in the market. Li-Fi is a uses visible light or IR light to communicate at high speeds and carry even

much more information than Wi-Fi. Li-Fi works by turning the led bulb on and off to send pulses within nanoseconds, too fast for the human eye to notice. Even more amazing is the fact that the led bulbs could be dimed enough, to the point that the bulbs are still able to transmit data and look like if they are turned off to the human eye. Li-Fi signals can bounce of the wall, which means a direct line of sight is not necessary. The downside of Li-Fi is that is not able to penetrate walls, but at the same time it means is more secure. Li-Fi will be a great technology in places where electromagnetic interference is a problem, such as airplanes and nuclear power plants. Li-Fi is expected to transmit data at the speed of 10 Gigabits per second and its cost to be 1/10 cheaper than Wi-Fi. Unfortunately, Li-Fi is still very new for it to be implemented in our design.

In conclusion, the group knows there will be no problem as far as connectivity if Wi-Fi is use as the primary way of transmitting the data from the glove to the external device. The only problem is that the group wants the design to be portable and light weight. If the group uses Wi-Fi the group would need a router, which means the project will be more expensive, more time consuming and not very portable friendly like the group want it to be. Also, Wi-Fi consumes lots of power compare to the other wireless communication devices that the group research. All the benefits that Wi-Fi provides is not really necessary and the power consumption is very critical for the battery life of the project. For this reason the group eliminated Wi-Fi from being used in the design. Table 1 below, shows some of the parameters of Li-Fi vs Wi-Fi.

Table 3.3: Li-Fi vs Wi-Fi

Parameters	Li-Fi	Wi-Fi
Speed	High	High
Range	Low	Medium
Data Density	High	Low
Security	High	Medium
Reliability	Medium	Medium
Power Available	High	Low
Transmit/Receive Power	High	Medium
Ecological Impact	Low	Medium
Device-to-device Connectivity	High	High
Obstacle Interference	High	Low
Bill of Materials	High	Medium
Market Maturity	Low	High

3.2.5.2 Near Field Communication

Near field communication also known as NFC, has become very popular now at days. It can be found not only in cellphones, but also in laptops and cars. Near field communication can send and receive data, but like its name implies, the

digital devices using this technology have to be very close to each other. Near field communication works with electromagnetic radio fields unlike Wi-Fi and Bluetooth that use ultra-high frequency radio waves. NFC comes in three different types of forms, Type A, Type B and FeliCa. These forms are very similar but communicate in different ways.

NFC devices could be either passive or active. Passive means that the device contains information that another device can read. Pretty much like an NFC tag, in other words a passive device does not read any information from other vices. A good example of this can be the nutrition facts label found in the foods we buy; anyone can read the information but the nutrition facts label cannot read anything. All it does is provides the information.

In the other hand, an active near field communication device is much smarter. It can do the same thing a passive device does, so not only it can transmit information but it can also read information. Active near field communication is found in most mobile devices. When you send a picture, video or any other file via near field communication to a friend, your device is transmitting the information. When your friend sends you a file back via near field communication, your device needs to read the information. Therefore, cellphones have an active near field communication.

Security is very important in all types of wireless communication. Especially now at days that near field communication is being used in the industry of automobiles, public transportation, banks, credit cards, and much more. For example you can open your car and turn it on by just having your keys near you. Make a payment by just waving the credit card or phone without having to touch or swipe the card. Near field communication can even store personal information to give you access to a secure building. For this reasons and many more, near field communication uses a secure channel with encryption when it’s transmitting or sending information.

In conclusion, although near field communication is contactless and very simple to use, because of the fact that the operating range (distance) is so limited, the group automatically eliminated near field communication from being used as the primary form of communication in the design. Nevertheless, the group might still use near field communication tags to possibly provide a unique feature to the Sign Language Interpreter Glove.

3.2.5.3 Bluetooth Classic & Bluetooth Low Energy

Bluetooth could be considered like a combination of near field communication and Wi-Fi. Bluetooth will transmit data at a much lower frequency than Wi-Fi, therefore it will consumes less power. The operating distance for Bluetooth ranges between 10 meters to 100 meters depending on the manufacture, which

is way more than what the design requires. Bluetooth is also very affordable and reliable. Bluetooth is not only power efficient, but also is easy to use.

Bluetooth works by using ultra high frequency radio waves. Ultra-high frequencies are stablish in a range of 300 Mega Hertz and 3 Giga Hertz. This ultra-high frequencies unfortunately are not good for transmitting through objects such as hills or tall buildings, but the good news is that the walls in our homes can't stop this signal. Wi-Fi, baby monitors, garage door openers, cordless house phones, portable speakers and other numerous devices use this type of frequency.

Bluetooth is use in almost everything nowadays, and is perfect to use for devices that only need to communicate over a short distance, like the Sign Language Interpreter Glove. There are several different types of Bluetooth, with the two most common being Bluetooth Low Energy (BLE) and Bluetooth Basic Rate / Enhanced Data Rate (BR/EDR). The difference between the enhanced data rate and basic rate is very simple; the enhanced data rate supports a bit rate of 2 Megabits per seconds and basic rate only supports a bit rate of 1 Megabits per second. Their similarities are that they both use a six-digit passkey that is much safer and more secure so that the possibility of another device interfering or intercepting the information is reduced. They both are also optimized to send high quality data while using the minimum power possible in order to save battery.

Bluetooth low energy is what allows developers to create tiny sensors that can run off a small coin cell battery for months or sometimes even years. The main difference between Bluetooth Low Energy and BR/EDR is that not only BLE is much more energy efficient, but also is built on a new development framework. Bluetooth low energy is also known as Bluetooth Smart because is being used to power the Internet of Things (IoT). Bluetooth Smart allows users to quickly send large data, like videos at very high speeds only when needed it, which means a longer battery life.

Pairing is what is needed when two Bluetooth devices wish to communicate with one another. The pairing between two devices must be a trusted; this can be accomplished by a passkey. Most devices are pair so often that the passkey or password is saved in order to avoid having to enter it each time the devices wished to communicate. Pairing is a very important and essential for the project, since there is only one sign language interpreter glove but multiple devices that should be able to connect to it. It is crucial to understand how pairing works because pairing is the part of Bluetooth that maintains a list of the devices that have made successful connections in the past. Nevertheless, pairing can also make sure to not permit the previous devices to connect if the user wishes to do this.

Bluetooth can connect to multiple devices at the same time without causing interference with one another. This is made possible thanks to spread-spectrum frequency hopping technique, which allows the transmitters to change frequencies about 1600 times per second. Even if there was interference, it will only last for a fraction of a second. In other words, Bluetooth meets every demand that the group is looking for the design. Therefore, the group decided to use Bluetooth Low Energy (BLE) for the design of the SLIG. Table 2 below, is a comparison of Classic Bluetooth vs Bluetooth Low Energy Technology.

Table 3.4: Classic Bluetooth vs. Bluetooth low energy

Technical Specification	Classic Bluetooth technology	Bluetooth low energy technology
Radio Frequency	2.4 Ghz	2.4 Ghz
Distance/Range	30 meters	50 meters
Over the air data rate	1–3 Mbit/s	1 Mbit/s
Application throughput	0.7–2.1 Mbit/s	0.2 Mbit/s
Active slaves	7-16,777,184	Unlimited
Security	64/128-bit and application layer user defined	128-bit AES and application layer user defined
Robustness	Adaptive fast frequency hopping, FEC, fast ACK	
Adaptive fast frequency hopping		
Latency (from a non-connected state)	Typically 100 ms	6 ms
Total time to send data	100 ms	<6 ms
Government Regulation	Worldwide	Worldwide
Certification Body	Bluetooth SIG	Bluetooth SIG
Voice capable	Yes	Yes, with some limitation
Network topology	Scatternet	
Star-bus		
Power consumption	1 as the reference	0.01 to 0.5 (depending on use case)
Peak current consumption	<30 mA	<15 mA
Service discovery	Yes	Yes
Profile concept	Yes	Yes
Primary use cases	Mobile phones, gaming, headsets, stereo audio streaming, automotive, PCs etc.	Mobile phones, gaming, PCs, watches, sports and fitness, healthcare, security & proximity, automotive, home electronics, automation, Industrial, etc.

Table 3.4 was reprinted with permission from Intelligent Systems Source

3.2.6 Power Source

The power source is one the most important parts of this project. Without a power source, none of the components use in the project would be able to operate. Also if the group does not give it the right amount of power then the components will fail. It is crucial to also make sure the group gets the most out of the battery. A charging station will be needed to recharge the battery of the Sign Language Interpreter Glove.

3.2.6.1 Batteries

In this section the group will talk about some of the different types of batteries available in the market. Powering the microcontroller is one of the most crucial parts of this project. The battery use to power the design will make a significant contribution; therefore, the group needs make sure the group uses the best battery that's available in the market. The group will only research rechargeable batteries that can be use with the microcontroller. The group will also talk about charging the rechargeable battery, this is very important because using a cheap charger could kill off the cells in the battery.

3.2.6.1.1 Nickel Cadmium & Nickel Metal Hydride Batteries

Nickel Cadmium batteries also known as Ni-Cad batteries, where popular in the late 19th century, until nickel metal hydride (Ni-MH) batteries took over. Nickel cadmium batteries are very inexpensive and retain their charge for long if left alone. They also offer great cycle life and low temperature performance when compared to other types of rechargeable batteries. The charging rate for nickel cadmium batteries depends on how the cell was manufactured, but regardless they offer a great charge and discharged life cycle. However, nickel cadmium batteries are low on power density. This batteries need to be frequently exercised in order to prevent the memory effect. Nickel cadmium batteries contain toxic metals, causing them to be environmentally unfriendly. Therefore, because of this reasons the group decided to not use it in the project.

Nickel metal hydride batteries are more popular than nickel cadmium because of the much higher power density. They also contain mild toxins which makes it more environmental friendly than nickel cadmium batteries. Nickel metal hydride batteries are usually used to replace non-rechargeable alkaline batteries and have about 30% – 40% more capacity than nickel cadmium. These batteries also have their cons, they cost more than nickel cadmium and their service life cycle does not last very long. They also self-discharge very quickly and their performance drops when they are exposed to high temperatures. To prevent crystalline formation, this nickel metal hydride batteries requires to be frequently

fully discharged. For this reason the group decided not to use Ni-MH batteries for the Sign Language Interpreter Glove.

3.2.6.1.2 Lead Acid Batteries

Lead acid batteries were invented in the late 18th century. It is the oldest type of rechargeable battery and it is still being use today in cars, marine and other power machines. Most of the battery consists of soft lead, but other small metals are used to get much better electrical properties and to improve its mechanical strength. Lead acid batteries come with an even number of volts since each cell is approximately 2 volts. Lead acid batteries should not be discharged completely because it will cause permanent damage on the battery. A full discharge will take away a small amount of capacity from the battery. Lead acid batteries provide about 250 discharge and charged cycles, depending on the depth of the discharged. When temperatures are high or when the battery draws high currents, corrosion and depletion can occur.

Recently, there has been advancement in lead acid batteries. In the past, it was known that lead acid battery performance was affected by sulfate accumulation. Scientist discovered that by adding carbon to the negative plate, the charging and discharging performance of the lead acid battery increased tremendously. This type of battery is known as Advanced Lead-Carbon (ALC). ALC batteries can operate between 30% - 70% state of charge, unlike the regular lead acid batteries.

Figure 3.12: Advance lead-carbon battery

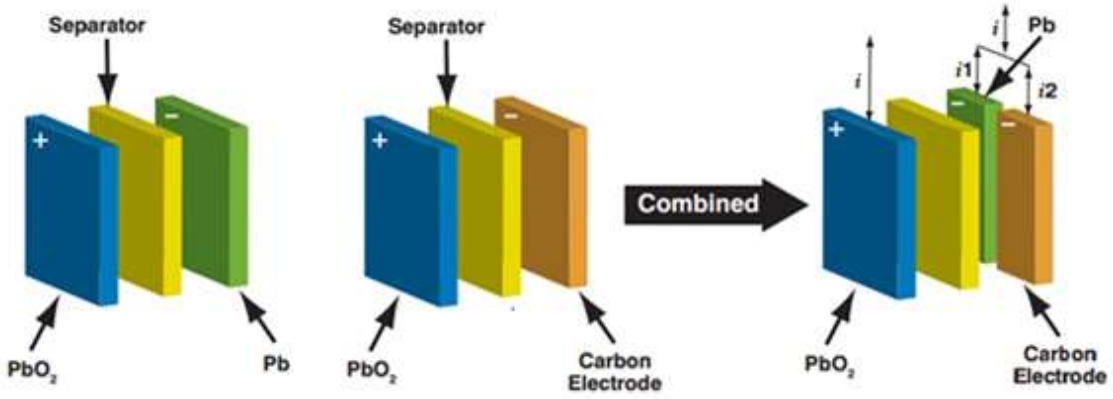


Figure 3.12 was reprinted with permission from Battery University

Lead acid batteries are very popular and are currently one of the bestselling batteries in the market. Advantages of lead acid batteries are that they take about a year for it to lose about 40 percent of its stored energy. They also work

well in cold temperatures. Lead acid batteries are very powerful and cheap. Nevertheless, the battery is very bulky and is less durable than nickel cadmium and lithium ion batteries. The lead content inside this batteries also damage the environment. This type of battery is mostly used in projects where weight is not an issue and lots of power is needed.

3.2.6.1.3 Lithium Ion Batteries (Li-ion)

If you open the back of cover of your phone, most likely you will find a li-ion battery powering your device. Li-ion batteries are the most common batteries used in electronics at the moment. The pros of li-ion batteries are that they are very lightweight and have high power and energy density, which makes them great for portable devices. Li-ion batteries are like a combination of nickel cadmium batteries and nickel metal hydride batteries. They are very similar to nickel cadmium batteries when it comes to not losing much charge when not in use, but also very similar to nickel metal hydride when it comes to having small memory effect.

Compared to nickel cadmium, li-ion's energy density is twice as much. One of the limitations of using li-ion batteries is that a protection circuit is required in order to limit de voltage and current. Also, li-ion batteries are more expensive compared to Ni-Cad and Ni-MH. In conclusion, there are many different types of batteries available in the market. They all have their advantages and disadvantages. Nevertheless, at the moment the group believes that the lithium ion battery will be best for the design.

Figure 3.13: Ion flow in lithium-ion battery

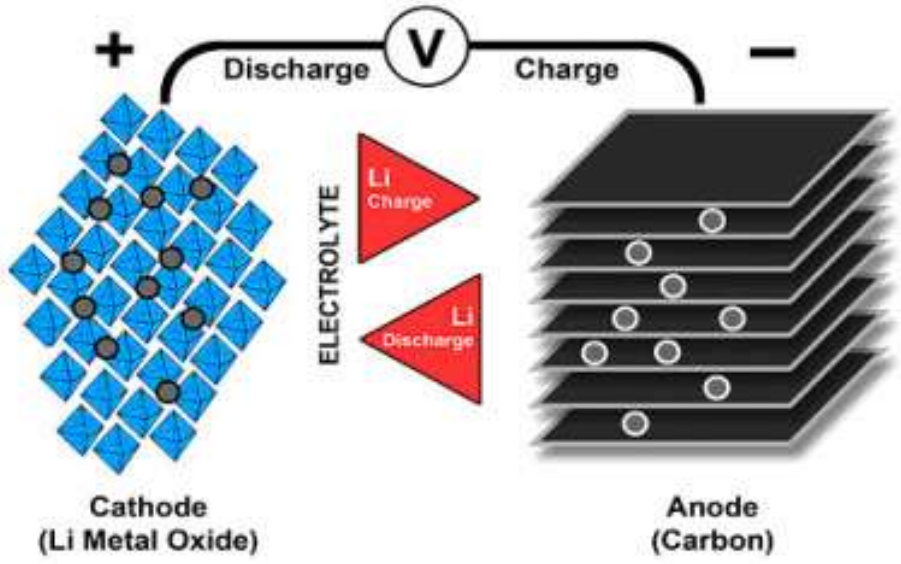


Figure 3.13 was reprinted with permission from Battery University

Table 3.5: Comparison of the batteries researched

Specifications	Lead Acid	NiCd	NiMH	Li-ion[1] Cobalt	Li-ion[1] Manganese	Li-ion[1] Phosphate
Specific energy (Wh/kg)	30–50	45–80	60–120	150–250	100–150	90–120
Internal resistance	Very Low	Very low	Low	Moderate	Low	Very low
Cycle life[2] (80% DoD)	200–300	1,000[3]	300–500[3]	500–1,000	500–1,000	1,000–2,000
Charge time[4]	8–16h	1–2h	2–4h	2–4h	1–2h	1–2h
Overcharge tolerance	High	Moderate	Low	Low. No trickle charge		
Self-discharge/ month (room temp)	5%	20%[5]	30%[5]	<5% Protection circuit consumes 3%/month		
Cell voltage (nominal)	2V	1.2V[6]	1.2V[6]	3.6V[7]	3.7V[7]	3.2–3.3V
Charge cutoff voltage (V/cell)	2.40 Float 2.25	Full charge detection by voltage signature		4.20 typical Some go to higher V		3.60
Discharge cutoff voltage (V/cell, 1C)	1.75V	1.00V		2.50–3.00V		2.50V
Peak load current Best result	5C[8] 0.2C	20C 1C	5C 0.5C	2C <1C	>30C <10C	>30C <10C
Charge temperature	–20 to 50°C (–4 to 122°F)	0 to 45°C (32 to 113°F)		0 to 45°C[9] (32 to 113°F)		
Discharge temperature	–20 to 50°C (–4 to °F)	–20 to 65°C (–4 to 49°F)		–20 to 60°C (–4 to 140°F)		
Maintenance requirement	3–6 months[10] (toping chg.)	Full discharge every 90 days when in full use		Maintenance-free		
Safety requirements	Thermally stable	Thermally stable, fuse protection		Protection circuit mandatory[11]		
In use since	Late 1800s	1950	1990	1991	1996	1999
Toxicity	Very high	Very high	Low	Low		
Cost	Low	Moderate		High[12]		

Table 3.5 was reprinted with permission from Battery University

3.2.6.2 Charging

3.2.6.2.1 Charging Ni-Cad and Ni-MH Batteries

First let's talk about some of the benefits of slow charging a battery. Cells inside a nickel-cadmium battery may have self-discharged so it is important to slow charge the battery to bring all the cells to an equal charge level. It is always recommended to slow charge the batteries for about 24 hours before first time use. Slow charging the battery also helps when the electrolytes are at the bottom of the cell, which happens when the battery is stored for a long period of time. Slow charge will help redistribute the electrolytes and eliminates the dry spots on the separator.

These types of batteries don't reach optimal performance until they have been charge and discharged several times. Some batteries, if made with good quality could reach optimal specification requirements with just about five to seven cycles. Nevertheless, it could also take about 50-100 cycles if is a cheap battery. Other important factor is that these rechargeable batteries should not be charge incorrectly. Most batteries will come with a safety vent to make sure it releases extra pressure if is ever incorrectly charged. For nickel-cadmium batteries the vent opens between 150-200 psi. The vents are re-sealable, but damage could happen if vent keeps opening up, causing a leakage due to the electrolytes escaping the battery. The battery should be charge correctly to avoid the battery becoming in a dry-out condition.

Figure 3.14: Charge characteristics of a nickel-cadmium battery

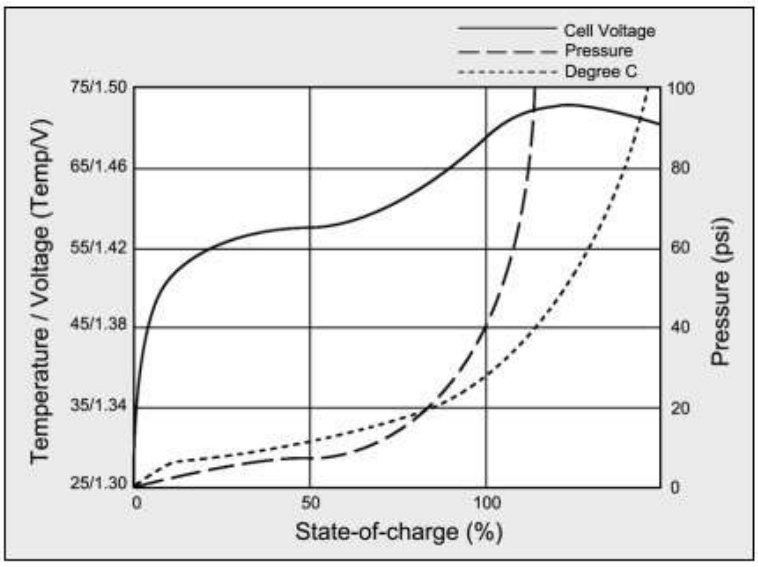


Figure 3.14 was reprinted with permission from Battery University

The way that some cheap charger know that the battery is fully charged is by measuring or sensing the skin temperature of the battery. Once the battery temperature is 50°C (122°F) the charger will stop charging the battery. Nevertheless, that's not a very accurate way of deciding a fully charged battery because the core of the cell is much warmer than the skin of the battery. This can cause over charging. Also, temperatures above 45°C (113°F) are harmful to the battery. A better quality charger or more advance charger will determine when to stop charging a battery by sensing the rate of temperature increase over time. This method is much accurate than waiting for maximum temperature to occur and it also keeps the battery cooler when charging. These advance chargers will stop charging the battery once the temperature rises 1°C or (1.8°F) per minute. If unable to detect this rate of change in temperature, the charger will also stop once battery reaches 60°C (140°F). Other advanced chargers use a defined voltage to determine when to stop charging a battery. The reason for this is to have more accurate full charge detection. This method is also known as the negative delta V (NDV). This types of chargers also include an absolute temperature and time out timer for back up, just in case is unable to determine the voltage drop across the battery. A major advantage of nickel cadmium batteries with ultra-fast charging cells is that they can be charged extremely fast and cause minimal stress on the battery.

3.2.6.2.2 Charging Lithium-ion Batteries

Charging lithium-ion batteries requires a voltage-limiting device, very similar to the same charger use in lead acid batteries. The exception is that a lithium-ion battery charger will have a higher voltage per cell with more voltage tolerance and no trickle at full charge. Lithium-ion batteries do not accept overcharge, therefore manufactures are very strict when it comes to the voltage cut off unlike lead acid batteries which offer some flexibility. A prolonged charging above 3% (4.3 Volts) on a 4.20 volts per cell lithium-ion battery will cause plate metallic lithium on the anode. The reason for this is because the cathode material loses stability and becomes an oxidizing agent that produces carbon dioxide. Also a prolonged charging will cause the cell pressure to rise.

Current interrupt devices (CID) are responsible for the safety of the battery, they should stop current from flowing at about 145-200 psi. Some lithium-ion batteries even have a safety membrane to avoid the battery from getting on fire, which opens the battery at about 500 psi. Just like lead acid batteries, nickel cadmium batteries and nickel metal hydrate batteries; lithium-ion batteries will melt down and might get on fire when overcharged.

A lithium-ion battery is done charging once the current drops to 3% of rated current or when it drops to a set level. Sometimes, elevated internal resistance can cause the temperature of the lithium-ion battery to rise by 9º F. Nevertheless, the lithium-ion battery and/or the charger should be decreasing when a rise of

18º F occurs. As previously stated, it is not desirable to fully charge a lithium-ion battery because high voltage stresses the battery. A portable device should be turn off when charging to avoid stress on the battery because a parasitic load confuses the charger, causing it to continue charging a battery although the battery is already full charged.

Figure 3.15: Charge stages of lithium-ion

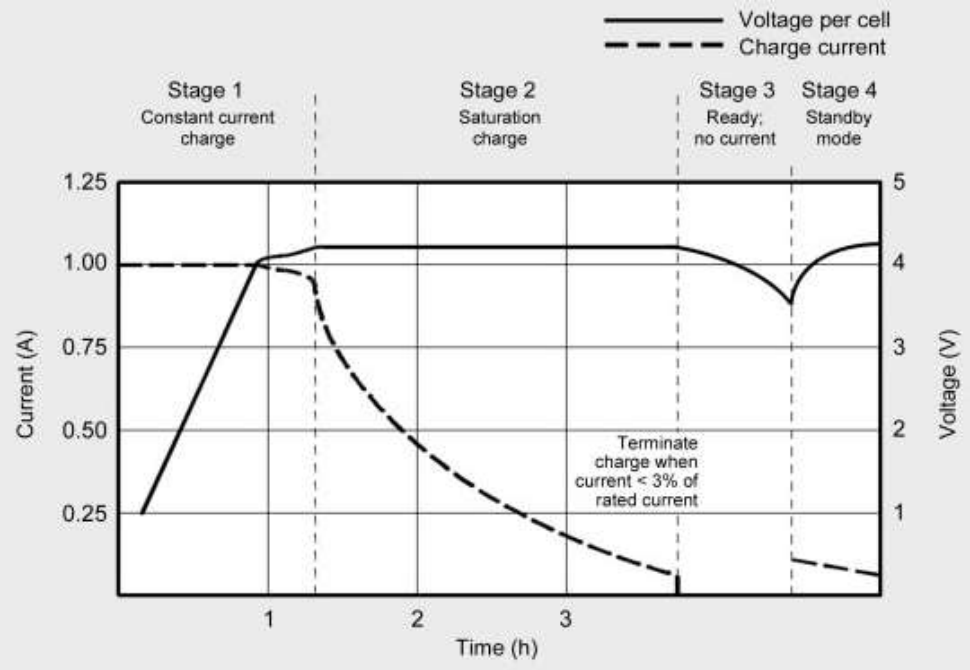


Figure 3.15 was reprinted with permission from Battery University

When charging lithium-ion batteries, is better to have the device turn off to allow the battery to reach threshold voltage and lower saturation current when full. Lithium-ion battery should not be charged while below freezing or at very high temperatures. Lithium-ion batteries and chargers need to be discontinued if the battery gets excessively warm while charging. Before storing, a 40-50 percent charge is recommended. Partial charge is better than a full charge to prolong the lithium-ion battery life.

In conclusion, the current and voltage limitations on lithium-ion batteries are much easier to analyzing than complex voltage signatures because the current and voltage will not change as the battery ages. The lithium-ion batteries do not need saturation and they do not need to be fully charge to operate. In other words, charging lithium-ion batteries is simpler than charging other types of batteries. The advantages of charging lithium-ion batteries are the absence of float charge and equalizing charges is not necessary.

3.2.6.2.3 Charging Lead Acid Batteries

The most common method of charging a lead acid battery is by the method called constant current constant voltage (CC/CV). A current that is regulated is used to increase the terminal voltage, once it reaches upper charge voltage limit it saturates and the regulated current will be reduce. One of the downsides of lead acid batteries is that they require a long time to charge. Depending on their size, it could take between 12 hours to 48 hours for a full charge. A multi-stage charge method can charge a lead acid battery much faster, sometimes reducing the charge time by 8-10 hours. Nevertheless, the multi-stage method cannot fully charge the battery to its 100% capacity.

A lead acid battery goes through 3 stages when it is charging. The first stage is called the CCC also known as the constant current charge. During this stage, the battery is capable of charging up to 70% in just 5 to 8 hours. The next stage is call the TC, toping charge. At this stage the current lowers and provides saturation. This stage is necessary so that the battery does not lose its ability to accept a full charge and to keep the battery performance at its peak. Toping can pretty much be compared to resting after a hard and long workout. The final stage is the FC, float charge. This is what makes possible for the battery to maintain its full charge.

Figure 3.16: Charge stages of a lead acid battery

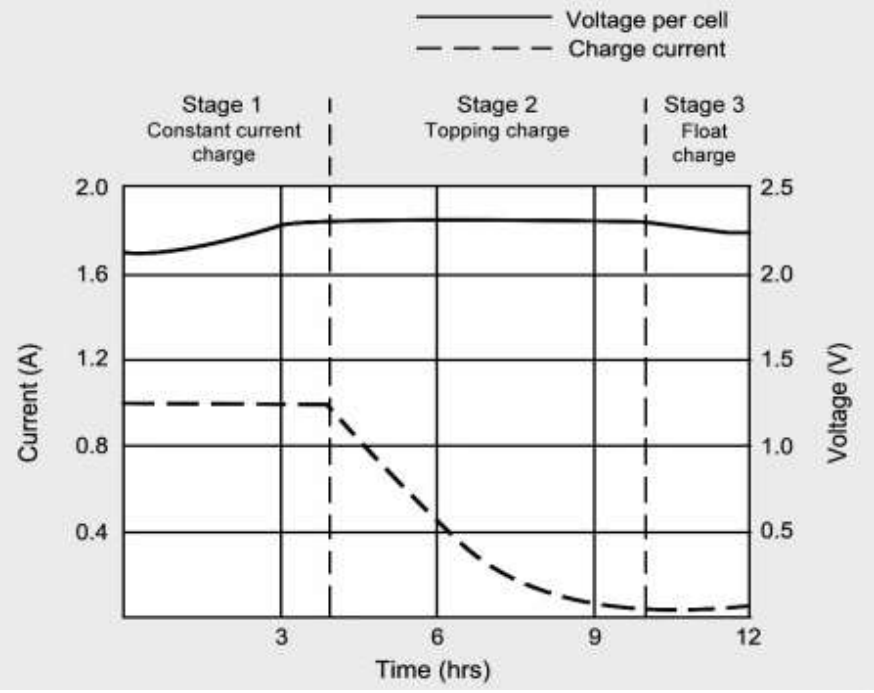


Figure 3.16 was reprinted with permission from Battery University

When a lead acid battery is not being used, then it must be kept on float charge. Float charge makes sure to stop the float current when the battery has reach full charge and is at standby. This is good to do in stationary batteries that do not draw any load. However, to prevent sulfation a lead acid battery needs a topping charge at least twice a year. In order to determine the state of charge of the battery, the open circuit voltage (OCV) must be measured. If the value measured of 2.10 volts at room temperature indicates that the battery is about 90% charged. Meaning the battery is in great conditions and only need a quick charge prior to use.

For optimal charging, a lead acid battery should be charged in a well ventilated location because the hydrogen gas that builds up is explosive. Lead acid batteries should always be fill with distilled / de-ionized water. Lead acid batteries should not be overfilling when it is on a low charge because it can cause the acid to spill. If hydrogen appears on the negative plate or oxygen on the positive plate, this is an indication that the battery is reaching a full stage of charge. Float charged should be minimized when the ambient temperature exceeds a temperature greater than 29°C (85°F). Also, lead acid batteries should not be charged for temperatures above 49°C (120°F) neither should they be charged if frozen. This is a good reason to not let the battery discharged too low, because the battery would freeze sooner than a fully charged battery. Most important, is to watering the battery. A new lead acid battery should only be inspected every few weeks for watering maintenance. Never add electrolyte because this will cause corrosion.

3.2.6.3 Voltage regulation

This section talks about the most popular types of voltage regulators and which one is the best choice for the project. The purpose of voltage regulators is to keep a constant voltage level. Voltage regulator can be used to regulate either alternating current or direct current voltages. Computer power supplies use electronic voltage regulators to stabilize the direct current voltages that is used by the processor. In a distribution substation, large size voltage regulators are used to make sure that the customers receive stable voltage no matter how much power is taken away from the line. The reason why voltage regulators will be use in this project is to make sure that each component gets the appropriate voltage it needs to function and operate properly.

There are a couple things the group needs to consider when deciding what type of voltage regulator the group plans on using for the design. First whether the group needs a fixed voltage regulator or adjustable voltage regulator. Most of the times the fixed voltage regulator is the right choice if there is no need to trim the output voltage. Also a fixed voltage regulator has less parts than the adjustable. Another thing to consider is whether the group needs a linear voltage regulator or a low voltage regulator, also known as ULDO. The major difference between the

two is that a linear regulator needs a minimum of about 3 volts change between the output and the input voltages. The ULDO in the other hand, only needs between 0.035 volts to 1 volts difference.

The group needs to consider not only the dropout voltage, but also the maximum output current. This is critical because it can cause instability issues in the design if the output current rating is not within the same ratings of the maximum required current by the circuit. The group also needs to be careful that the voltage regulator rating is not too high, because this can also cause the short circuit current to be high as well. The Power Supply Rejection Ratio also known as the PSRR is another factor to consider.

Voltage regulators also sometimes produce output noise, which could interfere with the sensitive components in or design. Stability can also cause poor performance because poor stability degrades the power supply rejection ratio. Finally the last factor the group needs to consider is the output impedance. If the regulator has low output impedance then it will perform better and the lower the chances of instability.

3.2.6.3.1 Series Voltage Regulators

Series voltage regulators, also known as series pass regulator, is one of the most popular types of voltage regulators. The way it works is by having a variable element in series with a load in order to provide effective voltage regulation. When the resistance of the series element is changed, the voltage across the load remains constant because the regulator varies the voltage drop. Series voltage regulators provide voltage regulation within a linear power supply. The advantage of a series voltage regulator is that is doesn’t draw the full current and the amount of current that it draws is as effective as the current used by the load, making it more efficient than other types of regulators.

3.2.6.3.2 Shunt Voltage Regulators

Shunt voltage regulators use variable resistance to provide a path from the power supply voltage to ground. Shunt voltage regulators are mostly used when the amount of wasted current is so small that it is not even consider. This types of voltage regulators are very simple and usually consist of just voltage reference diodes. Shunt voltage regulators are found in many DC power supplies and in many voltage reference circuits. Shunt voltage regulators come in different packaging type with different operating temperature ranges, accuracy and even reference voltage. The most popular being the 1.24 volts. Some applications of a shunt regulator are used for precision current limiters, voltage monitoring, error amplifiers, low output voltage, current source, analog/digital circuits and many more. One of the disadvantages of a shunt voltage regulator is that it draws maximum current from the source.

3.2.6.3.3 Switching Voltage Regulators

Switching voltage regulators are very similar to a linear regulator, except that thanks to a feedback mechanism is able to turn on and off devices in series. This is great because then the group can either have fully conducting series elements or switched off series elements with no power dissipation. One of the advantages of a switching voltage regulator is that its output voltage can be greater than its input voltage. There are several types of switching regulators and they come with different input voltages, maximum output currents and maximum switching frequency. The most common frequencies are between 300 kilohertz and 4 megahertz. Its maximum switching frequency could be 2.5 GHz. With a maximum output current of up to 3 amps. Most switching regulators are used for direct current to direct current conversion.

3.2.7 FPGA

As mentioned in above sections, certain parts of the project may require the use of logic gates to perform hardware implementations of certain things. For example, logic gates can be used to determine when the glove is in 'standby' mode and not actually being used so that it does not randomly send outputs when the user is not trying to communicate anything, but their hand moves slightly, causing the SLIG to output erroneous characters.

A great solution for logic gates is a Field Programmable Gate Array. This is an integrated circuit that can be programmed with a computer and simulates the presence of physical logic gates. The group would use a developing environment for the FPGA such as Xilinx. In this way, the FPGA can be programmed in any way that is desired such that it represents the actual functionalities of physical logic gates. Many different circuits that would require a large amount of physical logic gates can be implemented through the use of one chip in the FPGA. An FPGA development board that all members of the group have experience using and are proficient with is the Basys and the Basys II from Digikey.

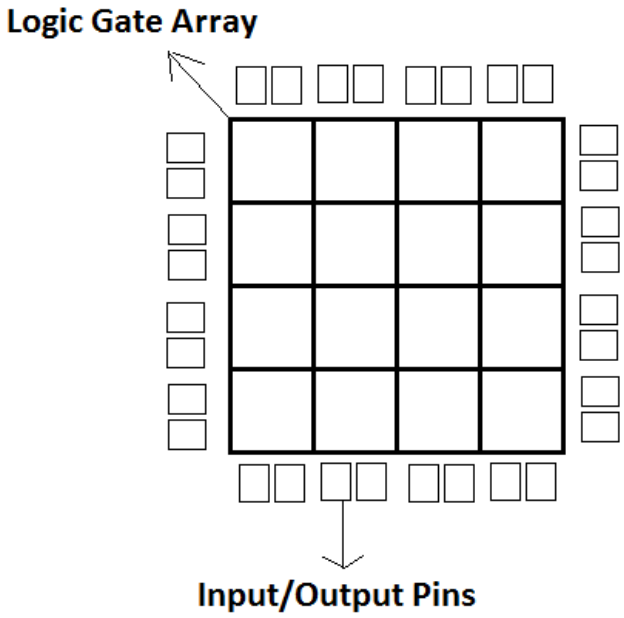
This board comes equipped with input and output devices, as well as LED lights, and other equipment that allows the user to interact with the FPGA. For the purposes of the SLIG, it would be unnecessary to have all of that extra equipment, since the project has its own input and output equipment (the sensors, data, etc.). Therefore the development board would be unneeded and only the chip (the actual gate array) will be needed, which can be included on the printed circuit board and use the input and output that is present on the glove from the sensors. This chip that comes on the Basys board is the Xilinx3E.

To program the FPGA so that it simulates any circuit that involves physical logic gates, the group can employ the use of a programming language known as Verilog. Verilog would allow the circuits to be built by the way of typing out the

logic, and not necessarily having to physically connect all of the components together to build the circuit. This makes it much more convenient to implement the circuit because the circuits can get quite large and having wires and busses running through the entire workspace can be daunting and make the design of the circuits be difficult and tedious. However, if desired, programming environments for FPGA design such as Xilinx also allow the developer to use the workspace area to actually draw out the gates that are wanted to implement the digital circuit. Below is a schematic diagram that was generated by the group that shows how an FPGA array is laid out, as well as the input and output pins that accompany it.

The group is not yet certain if the use of the Field Programmable Gate Array will be employed because it has not yet been decided if the group will implement the 'standby' mode through the use of a hardware solution or a software solution. In the case that it is a software solution, the group would simply program in the 'standby' mode into the microcontroller unit without the need to employ the use of the FPGA and use hardware to determine if the glove is not being used at the time.

Figure 3.17: FPGA Schematic



3.2.8 Printed Circuit Board

The group has many options when it comes to choosing the manufacturer for the printed circuit board that will be employed by the SLIG. The group would need to keep in mind that that printed circuit board must be small (to fit on the glove, not be too bulky, and also allow the end user to move their hand around freely

without having to worry about damaging or even sensing the presence of the printed circuit board.

Because of the constraints present upon the project such as economic and time, the group needs to find a compromise with a manufacturer that will provide the printed circuit board at a reasonable price, but whom can also have the PCB available in a relatively short amount of time. Of the many printed circuit board manufacturers that are available, most of the ones that best fit the economic constraint are shipping the board from overseas. This makes it imperative that the group has the printed circuit board ordered at an early time because otherwise it could arrive too close to the deadline, not giving the group ample time to test and implement the printed circuit board fully into the project. The main manufacturers that the group has been inquiring about are: Smart Prototyping, ShenZhen2u, PCB Zone, Express PCB, among others.

Of all the possible manufacturers that can provide the group with the printed circuit board, PCB Zone seems to be the manufacturer that has the most reasonable price. To produce a 4-layer PCB, PCB Zone is charging $74.64 plus shipping and handling. For the same 4-layer PCB, Smart Prototyping has it for $88.85, ShenZhen2U has it for $91.97, and Express PCB has it for $204. All of these prices are plus shipping and handling. The manufacturers will not provide the final cost of the PCB until after it has been designed and ordered, upon which point the group can have a better idea of how much it will actually cost from each manufacturer and how long it would take for the board to arrive from each manufacturer. Given the information that the group has at the moment, it seems as if PCB Zone will offer the group the best deal in manufacturing the PCB. Below is a table with the four manufacturers that are under consideration, the price from each, and the location from which the PCB would be shipped.

Table 3.6: PCB Manufacturer Comparison

Manufacturer	Price (Before Shipping)	Manufacturer Location
Express PCB	$204	USA
Smart Prototyping	$88.75	China
ShenZhen2U	$91.97	China
PCB Zone	$74.64	New Zealand

The printed circuit board will also have to be designed using some type of CAD software. If the group was to go with Express PCB, they provide free CAD software in which the design of the printed circuit board can be created. This is very helpful because in this case the PCB will be able to be designed in their native software and the group will not need to use some type of external CAD software to design the printed circuit board. However, if the group decides to go

with one of the other printed circuit board manufacturers that do not provide a CAD design software tool, there are many options out there wat would provide the service of designing the circuit board. The most popular software tool for designing the printed circuit board is EAGLE. This is software in which the printed circuit board can be fully designed. This would allow the manufacturer to just read the design that was made in EAGLE and build the PCB in the exact way that is needed for the particular application at hand.

Besides EAGLE, there are a number of other free options in which the printed circuit board can be designed. Some of these software tools are: PCBWeb Designer, ZenitPCB, Osmond PCB, DesignSpark PCB, Fritzing, as well as others. These software tools have the ability to import specific parts and design the printed circuit board to include the exact parts in the exact locations that is needed for the project. The group will have to gain proficiency in using these CAD software tools and design the printed circuit board that will go into the SLIG. At this point in time, the group thinks that the software design tool that is most likely to be used are either EAGLE or Fritzing. These are two very well-known and well respected software tools that have credibility and have a plethora of resources available online to learn how to use and troubleshoot in creating the design.

3.2.9 Serial Communication

Serial communication is when data is transmitted one bit at a time. This data uses a specific channel with a known standard that is used to transmit data through a known and tested method. Serial communication will be used in the implementation of the SLIG to transmit data from the microcontroller unit to the Bluetooth module. The Bluetooth module can be configured using serial communication and use the data received to transmit it through the ether and out to the receiving end of the Bluetooth setup (the Android smartphone). The most popular form of serial communications is the use of RS-232. This is a standard that is very popular in the electrical engineering industry and is used world-wide. Through RS-232, data is transmitted through a cable that has a number of wired connections in it. There are different types of cables that can be used to transmit data via the use of serial communication, and it all depends on the type of setup that is being used that will determine the type of cable that is used. For example, if a regular, “straight” cable is used, that means that each pin on each end of the cable is a straight shot and corresponds to the same pin on the other side of the cable.

This type of setup requires the use of a modem. A modem in between the two sides of the communication medium is helpful at changing the send and receive bits and aligning them with the corresponding bit that would go on the other side of the communication line. This is the more traditional method for using serial communications. In this scenario, the setup consists of: the transmitter, which is producing an output which is the data that is trying to be transmitted, a

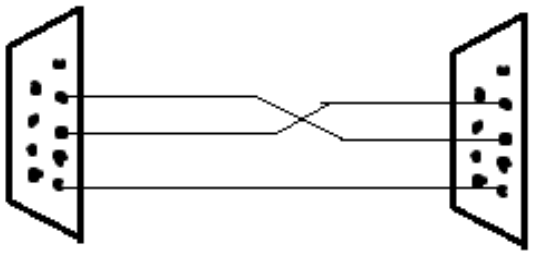
transmitting modem, which does the job of “changing” the send bit from the sending side of the cable to the “receive” bit on the receiving end of the communication, a straight shot RS-232 cable that connects from the transmitting modem to the receiving modem, a receiving modem which takes in the data that was transmitted through the straight cable, and finally the receiver, which is the intended target that the information was trying to be sent to in the first place.

The alternative to using a straight cable with modems to do the job of switching the orientation of the send and receive bits and make the communication possible, is through the use of a “null-modem” cable. A null-modem cable is a cable that is specifically designed to not need a modem. This type of cable can be connected from the transmitter straight through to the receiver without the need of any interference.

The way that this type of cable works is that within the cable, the send and receive bits are actually changed, internally within the insulation of the cable. For this reason, this type of cable is sometimes colloquially referred to as a “twisted cable”. This type of cable makes it a lot more convenient to use serial communication because an engineer or technician need only to have a null-modem cable in their bag and they can simply connect to whatever they are trying to communicate with without the use of large amounts of hardware such as modems.

Below is a simple schematic diagram that was generated by the group that illustrates how the null-modem cable works and switches the wires internally so that pin 2 on one end is actually continuous with pin 3 on the other end, and not the same pin 2. As is evident below, the null-modem cable can send information without the use of a modem because the wires inside of the cable are set up in such a way that the send from the transmitting side is twisted to match the pin on the receive pin of the receiving side of the link. This makes it more practical to use the null-modem cable and not have to have modems in the system.

Figure 3.18: Illustration of Null-Modem RS-232 Cable



3.2.9.1 Analog to Digital Converters

The implementation of the Sign Language Interpreter Glove will need the use of analog-to-digital converters. This is because the output that is received from the sensors is a continuous, analog voltage level. The microcontroller naturally operates at the digital level and it would be impossible to perform digital computations on an analog signal. For these reasons it is necessary to convert the analog voltage that is taken from the flex sensors, pressure sensors, and accelerometer into a corresponding digital voltage level.

There are many different types of analog to digital converters. They all use a slightly different method to provide an output that is a digital voltage but that is proportional to the analog voltage that was input. One type of analog to digital converter is the “flash” analog to digital converter. This type of analog to digital converter uses a series of comparators that all operate simultaneously to determine what the voltage level of the analog signal coming in is. The “flash” converter has a specific voltage level “assigned” to each comparator, and each comparator is constantly examining the input analog signal to determine if that signal matches up to its assigned level. Eventually one of the comparators will have a match with the analog input signal, and it will output its digital version of that voltage level.

3.2.10 Glove

3.2.10.1 Functionality

The glove of course will serve the purpose of holding all of the different electronics and sensors and making this project a usable device. In choosing a glove we must consider certain important features of said glove to make sure the electronics function properly and that the end results is a comfortable but efficient product. These features include but are not limited to the size, type of the material and number of layers of the glove. To elaborate, this project would require a glove that could fit most people but definitely be large enough to hold all required electronics. Furthermore, different materials could possibly conduct heat differently and so some may prove less than optimal for working along with electronics and sensors. Lastly, the team has noticed that there are some gloves available with two layers which would be convenient in the sense that the electronics of the place between the two layers improving the overall aesthetics of the glove.

3.2.10.2 Models and Specifications

As could be expected there is a plethora of different gloves for all types of purposes available in the market. As mentioned before it is important to pick a glove that best matches the needs of this project. Below are some of the models.

Under Armour Yard Baseball Glove – This glove makes use of very high quality, soft cabretta leather for optimal feel. This would probably make a very popular choice among users and improve the overall comfort of using the SLIG glove. Along with the leather this glove makes use of synthetic materials to increase the durability and flexibility of the glove which is a great advantage when it comes to applying electronics this glove and the repeated use that will have to endure.

Under Armour ColdGear Infrared Fleece – This is glove is made from micro-fleece which is a thermal conductive material that is very efficient in keeping onto heat without adding bulk of the glove. Although this may be helpful to a consumer who may need the glove for outdoor purposes it may prove troublesome for this project as electronics used for the Slagle most likely generate heat and trapping this he would only make it uncomfortable for the user. However, it should still be noted that it seems to be more flexible with the use of spandex than most other leather gloves, which will enable the user to fully flex their hands and form the sign language letters correctly.

Under Armour ColdGear Infrared Engage Run Gloves – These gloves are not made of fleece like the previous but they do retain the same heat absorbing properties of the ones before. Again, this is this will probably be a disadvantage for indoor use where he will just make it more comfortable for the user however this particular model has a very prominent distinct feature. It claims to have touchscreen compatible fingertips which would mean that the user can make use of the screen will be displaying the information using the same hand that they're using to to make the sign language letters. Moreover, this is a very aesthetically pleasing glove out visually match our PCB board and all the electronics. This would make the end product more satisfying to the user.

Under Armour Strike Skin Tour – This was originally designed to be a golf single glove that features premium high-performance leather that can stay soft and flexible after heavy use. It is made of the top quality cabretta leather that enables this glove to deliver the promised level of control and flexibility, which will be crucial in allowing the user to fully form every single sign language letter properly. Plus, since it is sold as a single glove, the team would be saving the cost of buying two gloves. Finally, it has small incisions all throughout the fingers that will most likely be useful if we have to tie any of the electronics through the glove.

Under Armour Charge Will Run – This glove is made out of very soft wool that is advertised to dry much faster than traditional material. But like many of the previous gloves, this model calims a high-level of flexibility that is always a necessary feature to allow the user to fully form the sign language letters and like ones before, it has techtouch technology that allows you to use touchscreen

devices without having to remove the glove. Wool is said to be an insulator so it should not interfere with any of the activities from the electrical on the device.

Under Armour Motive Batting Gloves – This glove is made of very fast drying technology that wicks away sweat, which will be useful in helping the user keep his or her hand dry and comfortable while he or she is using the glove. Furthermore, the manufacturer has incorporated their four-way stretch technology meaning that the construction can both stretch latterly and vertically for the maximum range of mobility and accelerated dry time. The elastic cuffs deliver additional wrist support, which most likely will come into play when the user has to create the signs for the letters J and Z, which require movement of the wrist. Again, this glove is shown with a few sets of perforations that could be helpful in attaching different components to the glove.

Under Armour Camo HeatGear Line – The smooth and durable fabric used to create this glove ensures that the user's hand will remain cool and dry while he or she is using the product. Like most of their other gloves, this one claims a high level of flexibility and mobility that along with the anti-odor technology will make make this a decent choice for this project. The anti-odor technology is actually more important than it may seem at first. From a health perspective, it is actually preventing the growth of odor-causing microbes, meaning it is can still keep the glove sanitary while multiple people use it.

Under Armour Camo ColdGear Liner – This specific model was originally designed to be a men's hunting love. Following previous trends, it was designed to provide the user with extreme breathability, warmth and increased dexterity. On top of that, it also has moisture management technology to manage sweat, which has its benefits as mentioned before. Perhaps the most notable feature of this glove is the extra-long cuff that was originally designed to add protection to the customer's forearm. For the scope of this project, this extra-long cuff will be essential to helping fully carry all the electronics in case these prove longer than the length of the hand. This will make the final prototype more secure and stable since the electronics will be firmly attached to the whole glove and not just fixed at the top.

Under Armour Tactical Fire Retardant Liner – This glove goes one step further than the one previously mentioned. This glove sports the extra-long cuff that would be helpful in fully supporting any circuitry longer than the user's hand; a moisture transfer system to keep it dry and whisk away sweat; materials to prevent the growth of odor-causing microbes and a high level of flexibility and comfort. The most notable feature of this glove is the fire retardant technology; which would make this glove an excellent choice in terms of safety for the user because there is always a possibility that the electronic components could malfunction and somehow result in flames. This glove would definitely protect the user the best out of all other models.

Outdoor Research Men's Versaliner Gloves – These versaliners offer breathable insulation and waterproof protection, which were originally for outdoor activities. The feature that stands out the most on these gloves on the double layer where the outside is a ripstop fabric shell the covers the inner fabric. The space between these two layers would be a perfect place to place some of the electronics in the fact that is waterproof would make it safe for any user to use. Below is a summary of the different gloves and specifications covered thus far.

Table 3.7: Summary of Glove Options

Summary of Features					
Model	UA Yard Baseball	UA Coldgear Infrared	UA ColdGear Infrared Engage Run	UA Strikeskin Tour	UA Charged Wool Run
Price	$47.99	$29.99	$24.99	$21.99	$49.99
Material	Leather	Polyester, Nylon, Polyurethane	Polyester	Sheepskin Leather, Nylon	Nylon, Wool, Elastane
Top Feature	Soft Leather	Tech Touch technology	Thermo-Conductive	Maximum Control	Tech Touch Technology
Summary of Features					
Model	UA Motive Batting Gloves	UA Camo HeatGear Liner	UA Camo ColdGear Liner	UA Tactical Fire Retardant Liner	MEN'S VERSALINER
Price	$39.99	$19.99	$29.99	$49.99	$52.00
Material	Synthetic leather	Polyester, Elastane	Polyester, Elastane	Acrylic, Rayon, Aramid	100% nylon 40D stretch ripstop shell, Radiant Fleece™ 95% polyester
Top Feature	4-Way Stretch	4-Way stretch	Extra-long cuff	Extra-long cuffs	Dual Layer

3.2.11 Onboard LCD Display

Although the main method for the end user to view the text that is being generated by the Sign Language Interpreter Glove is the use of an Android application that will display the output in real time. However, the group has considered the idea of including an LCD display on the actual glove itself. This

can be helpful to the user because with the Android application, the person with whom the user is trying to communicate with can see what is being said, but the actual user cannot know if the gesture that they have made actually produced the correct output on the display, unless they are right next to the person with whom they are speaking and can physically view the cell phone.

For this reason it would be helpful to have a display that is native to the glove. The user can be performing their gestures, and simultaneously monitoring that the control system is producing and transmitting the proper character. In the unlikely case of an error being made by the control system, the user can communicate with the receiving person, and let them know that they in fact were not trying to say what was displayed on the cell phone, and try to correct it. The on-board LCD module would have several pins, normally about 16. These pins would be configured with the output from the microcontroller and the display would show the corresponding character that was outputted from the microcontroller unit and also sent out through the Bluetooth module. The onboard LCD display would employ the use of the pins that are on it and it would interpret the characters through the use of hexadecimal values that illuminate specific parts of the display. In this way the SLIG will show the hand gesture that the user is trying to make and display it right on their glove, so that the receiver of the message can be sitting away from the user (within Bluetooth's range) and the user can perform their hand gestures with confidence that the receiver is reading the actual character that the user is trying to communicate.

Though the addition of the LCD display to the SLIG would add an extra layer of convenience for the end user, given the economic, health, safety and time constraints that are imposed onto this project, the group will have to go through an in-depth process of determining whether or not the implementation of the LCD would be a worthy endeavor when designing the Sign Language Interpreter Glove. On top of this addition taking quite a bit of time to design and implement, it would also cost more money to purchase the display and additional parts needed to seamlessly implement it into the project. In addition, this display would also add another piece of electronic equipment to the glove, and as mentioned in another section, the group would like to keep the glove as sleek as possible, avoiding any bulging electronic components that the end user can inadvertently damage and/or hurt themselves.

3.2.12 Bluetooth Low Energy Module

The job of the Bluetooth module is to collect all the information from the external sensors, could be either digital or analog data. Once it collects the information it should send it via radio frequency to the external device. There are several types of Bluetooth modules. This section will talk about the pros and cons of several different types of Bluetooth modules.

One of the first decisions that the group needed to make in regard to the Bluetooth module was whether a single mode or a dual mode was need it. Single mode is when only a Bluetooth low energy module is used and dual mode is when a Bluetooth Classic and a Bluetooth low energy module is used. Because of cost and simplicity of the Sign Language Interpreter Glove, the group decided to go with just a single mode Bluetooth module. Dual mode would have allowed older devices that do not support Bluetooth low energy to be able to communicate with our project. Nevertheless, if the group would of have used dual mode, the power consumption of the Bluetooth module would have been much greater and more expensive.

Bluetooth modules also come in two different packages options, which are Quad Flat No Leads package also known as QFN and Wafer Level Chip Scale package also known as WLCSP or CSP. Some differences between the Quad Flat No Leads package and the Wafer Level Chip Scale package is that the QFN is much larger and cost more because it contains more material than CSP. Nevertheless, using a WLCSP will be more expensive when used in a PCB design because it requires tighter tolerances and more than two layers. So QFN might be more inexpensive when use in a PCB design. CSP is more suitable for really small product design where QFN doesn't fit.

3.2.12.1 Nordic Semiconductor nRF8001

The group is considering the Nordic Semiconductor nRF8001 Bluetooth low energy chip. The nRF8001 runs the Bluetooth low energy stack internally. The nRF8001 Bluetooth chip features a very simple serial interface that is compatible with many microcontrollers. The peak current of the nRF8001 Bluetooth chip could be as low as 12 mA. Having such low peak current allows the nRF8001 Bluetooth chip to have a battery lifetime that last months and depending on the application could even last years. The nRF8001 Bluetooth chip supports security functions as well as GAP role, server role and client role. Basically the nRF8001 Bluetooth chip is design for the slave role (peripheral operation).

The nRF8001 Bluetooth chip comes in a 32-pin 5 by 5 Quad Flat No Leads package. The nRF8001 Bluetooth chip also has an analog to digital converter which can be used for managing the level of the battery. The nRF8001 Bluetooth chip also includes a low tolerance 32 kHz RC oscillator which is used to remove the need for an external crystal. The nRF8001 Bluetooth chip also includes a DC to DC voltage regulator and a linear voltage regulator. The DC to DC voltage regulator is used to lower the current consumption when using a 3 V battery by 20 percent. The linear voltage regulator is used to provide a voltage supply range of 1.9 – 3.6 volts.

Below are some of the features the nRF8001 BLE Chip offers.

- Ultra-low power consumption
    - 11mA Active TX peak current at 0dBm output power
    - 12.5mA Active RX peak current
- 2.4GHz Radio
    - Fully Bluetooth Smart v4.0 compliant
    - 0, -6, -12, and -18dBm programmable TX output power
- System Peripherals and I/O
    - Temperature sensor
    - UART for DTM
- Embedded Bluetooth Smart stack
    - LL, L2CAP, GAP, SM, ATT and GATT mandatory features for peripheral role operation
    - GATT Client and GATT Server
- Temperature range
    - -40 to +85 ºC

For the complete list of features and diagrams please visit:
https://www.nordicsemi.com/eng/Products/Bluetooth-Smart-Bluetooth-low-energy/nRF8001

Figure 3.19: nRF8001 Pin Diagram

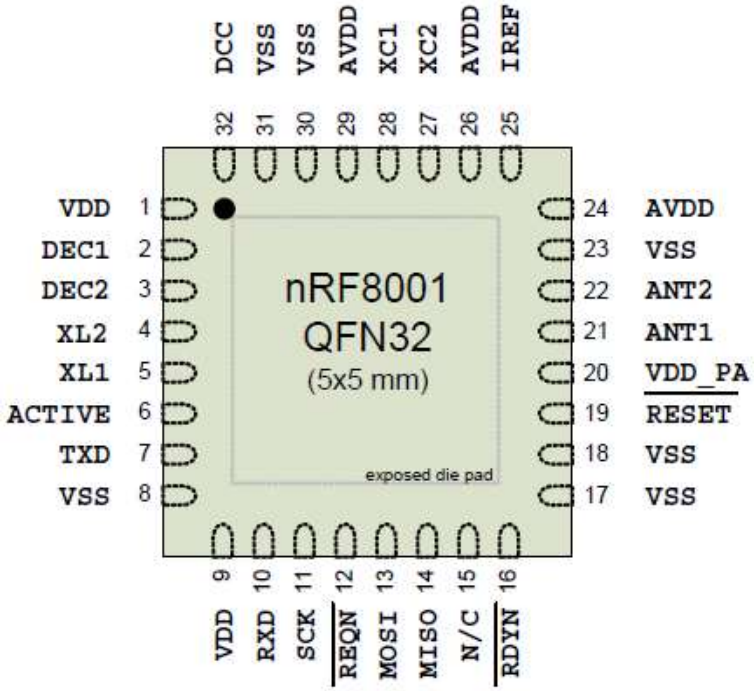


Figure 3.19 was reprinted with permission from Nordic Semiconductor

3.2.12.2 Microchip RN4020

The second option would be the RN4020 by Microchip. The RN4020 Bluetooth low energy chip is compatible with many of the affordable microcontrollers in the market today. The RN4020 offers internal scripting capabilities to accomplish those basic functions, avoiding the need for software development tools or an external host MCU. The RN4020 Bluetooth chip features digital analog inputs and outputs, ASCII command interface API over UART, MCU and it includes all Bluetooth SIG profiles. Users can remotely control the RN4020 Bluetooth chip using a secure connection with another Bluetooth chip. Updating the RN4020 Bluetooth chip is easy; it can be done over the air or even via the UART interface. Optimize for long range of over 100 meters, the RN4020 Bluetooth chip offers a built in high performance printed circuit board antenna. The RN4020 Bluetooth chip is the perfect size for the Sign Language Interpreter Glove, only 11.5 by 19.5 by 2.5 mm.

Some of the features of the RN4020 are:

- GAP,GATT,SM, L2CAP and integrated public profiles
- Data streaming with Microchip's Low Energy Data Profile (MLDP)
- 7 dBm transit power for 100m+ range
- Software configurable role as peripheral or central, client or server
- UART interface, GPIO, ADC
- 64KB internal serial flash
- Castellated SMT pads for easy and reliable PCB mounting

For the complete list of features and diagrams please visit:
http://www.microchip.com/wwwproducts/Devices.aspx?product=RN4020

Figure 3.20: RN4020 Pin Diagram

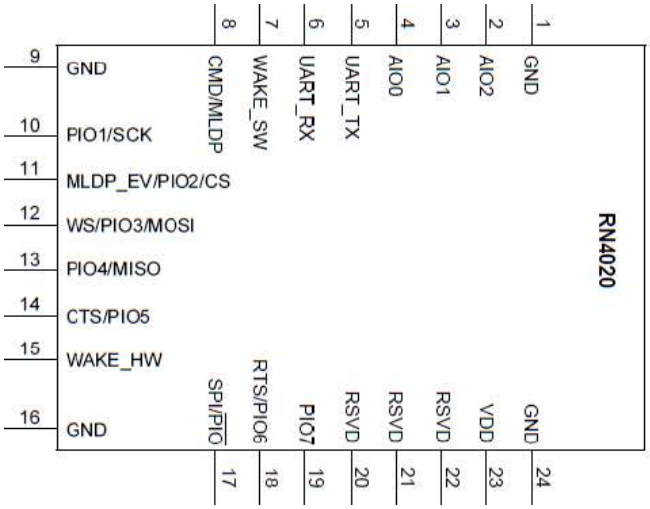


Figure 3.20 was reprinted from the Microchip RN4020 datasheet

3.2.12.3 Texas Instruments CC2541

The group is also considering the CC2541 chip by Texas Instruments. The CC2541 Bluetooth chips has several applications but are mostly used in mobile phone accessories, home automation, lighting control, alarms, wireless sensor networks and many more. The CC2541 Bluetooth chip is one of the most recent Bluetooth chip that Texas Instruments has manufactured. The CC2541 Bluetooth chip does not requires much power to function, which means that it can operate on a small coin cell battery and it can provide excellent battery lifetime. Overall the CC2541 Bluetooth low energy chip provides high performance, it's very affordable.

The CC2541 is composed of several different parts. The CC2541 comes with the option of a 128 KB Flash Ram or 256 KB Flash Ram. The CC2541 contains a single cycle 8041 CPU core. The CC2541 Bluetooth chip connects all the hardware to the memory via an SFR bus. The CC2541 SRAM is able to keep its information even when is powered off thanks to its ultralow power. The CC2541 also offers a 5 channel DMA controller. The CC2541 also comes with a watchdog timer and a sleep timer. The main difference between the CC2540 which is another popular TI Bluetooth low energy chip and the CC2541 is that the CC2541 has an I^2C device. I^2C stands for inter-integrated circuit, and its purpose is to support the slave and master operation. Multiple master devices are able to connect thanks to the bus design of the I^2C. With a simple command, I^2C allows slave and master devices to switch their roles.

Below are some of the features that the CC2541 Bluetooth Chip offers.

- RF
    - Excellent Receiver Sensitivity (–94 dBm at 1 Mbps), Selectivity, and Blocking Performance
- Layout
    - 6-mm × 6-mm QFN-40 Package
- Low Power
    - Active-Mode RX Down to: 17.9 mA
    - Active-Mode TX (0 dBm): 18.2 mA
    - Power Mode 1 (4-µs Wake-Up): 270 µA
    - Power Mode 2 (Sleep Timer On): 1 µA
    - Wide Supply-Voltage Range (2 V–3.6 V)
- TPS62730 Compatible Low Power in Active Mode
    - RX Down to: 14.7 mA (3-V supply)
    - TX (0 dBm): 14.3 mA (3-V supply)
- Peripherals
    - Powerful Five-Channel DMA
    - Battery Monitor and Temperature Sensor

Below is a picture of the pin diagram for the CC2541 BLE chip.

Figure 3.21: CC2541 Pin Diagram

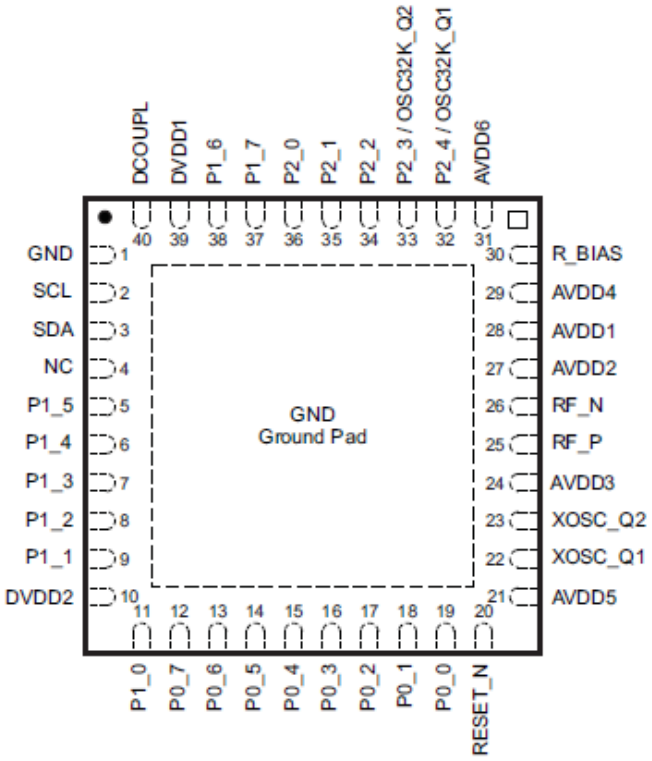


Figure 3.21 was reprinted with permission from Texas Instruments

3.3 Software

There are multiple software components required for the sign language glove which are explained in detail in the following few sections. The first section will explain the mobile application being created for the user interface. The mobile application is responsible for wirelessly displaying hand gestures performed by the glove onto a mobile phone screen as text. The next section will discuss the control systems portion of the software, which includes programming the microcontroller and writing machine learning algorithms used for hand gesture recognition. Some of the programming languages will be considered too so that we are sure we make the most informed choice for the mobile platform and microcontroller.

3.3.1 Mobile Application

In the next few sections, we will compare the top three mobile platforms available to use for our mobile application.

3.3.1.1 Mobile Application Overview

The mobile application is an important feature of this project that will serve as the user interface for the sign language glove. The functionality of the glove will be determined by the success of the mobile application which is easily forgotten in many technical applications. Many designers are concerned with meeting the technical specifications and requirements of their design but forget that the user (consumer) is the end-goal of the project. In order for the mobile application to be successful, the design of the app should consider the user, be simple and elegant, and meet all design requirements. A more detailed discussion of the design process for the mobile app user interface is explained later in the Mobile Application section under Design (6.2.2.3 Menu Layout).

The purpose of the mobile application is to provide the user with a visual interface for the sign language glove that displays feedback of the gestures in real time. The mobile application will not be performing the gesture recognition processes; it will mainly be responsible for receiving data from the glove via Bluetooth and displaying it on the phone screen. Therefore, the amount of processing power required for graphical user interface (GUI) of the mobile app will not be very demanding and will not be a major concern when choosing a mobile device platform. The three main mobile platforms available are Android™, iOS, and Windows Mobile which all have their advantages and disadvantages.

When choosing a mobile platform, each design requirement for our sign language glove cannot be looked at individually. For example, designing the GUI for an iPhone would be much easier than designing the GUI for an Android phone because iPhone displays only come in a few standard sizes and Android displays can range from a mobile phone screen to a tablet. However, the approval process of putting an app in the app store for Android is much easier than iOS. Therefore, choosing the mobile platform that supports the most important design features will be the best option. One factor that will make the decision a little easier is that some of the design constraints for the mobile application come with the mobile platform which means that there is no choice. One example where this comes in handy is when choosing a programming language because Android, iOS, and Windows Mobile all have different languages. These languages all have about the same capabilities and performance but it doesn't really make a difference which one is used except for the developer's preference.

In order to determine which mobile platform is the best fit for the sign language glove, a more detailed analysis needs to be done on each platform. As mentioned above, only a select few features make a significant difference for the mobile platform and these features need to be identified before proceeding. In general, developers choose their mobile platform based on the hardware/software compatibility, programming languages, audience base, security, and available resources. Not all of these considerations are weighted as

heavily as others but need to be included in the decision process. Determining which features are important and which are not, the functionality of the sign language glove comes into play. Taking a look at the hardware and software needed is the first step because it is the most prominent part of the design.

Some common mobile phone hardware components consist of a combination of the following: display, keypad, battery, memory (ROM/RAM), microprocessor, USB, speaker, microphone, camera, Bluetooth, GPS, antenna, volume control switch, and on/off switch. From this list, the mobile application for the sign language glove will only use the battery, display, Bluetooth, microprocessor, and possibly memory components, so these are the main hardware features that will be compared when choosing a mobile platform. As far as the software goes, the design considerations include programming languages, developer access to the mobile development environment and mobile phone, and access to the app store. The biggest factors are whether the developers (us) have access to the mobile development environment and which mobile phones the developers own because if the majority of our team has Android it would not make sense to use iOS as our mobile platform since we would have to get ahold of more iPhones. The next step is to take a look at the other criteria required for making a mobile application like the audience base, security, and available resources because these can make a big influence in our decision too. However, these factors must be discussed in more detail for each mobile platform since they are considered facts more than they are design options.

3.3.1.2 Potential Mobile Platforms

3.3.1.2.1 Android

Android is a very popular and powerful operating system that runs on phones, tablets, watches and more. Having the largest installed base of any mobile platform, Android provides a platform for creating apps and games for Android users across the globe as well as a well-supported developer environment. The Android Developer Tools offer a complete Java Integrated Development Environment (IDE) called Android Studio, which is basically an application that provides tools for developing, debugging, and packing Android apps. In addition, Android Studio provides developers with the option of running their apps on any available Android device or creating a virtual device that emulates any hardware configuration. Therefore, Android apps are compatible with all Android supported phones, tablets, and other devices and it automatically adapts to your UI. Another feature that makes Android so attractive to developers is the fact that it is open source, which means that there are better quality apps, more documentation, and more customization. The Android Open Source Project is led by Google and ensures that all Android developers using their open source community also maintain the Android Compatibility Program. This means that every app on Google Play will be compatible with most Android devices and keeps Android developers all on the same page.

All of these bells and whistles that make Android so popular are great but now it is time to assess how Android can satisfy the needs of the sign language glove. As mentioned above, the display is not a compatibility issue for Android applications since their framework is designed to adapt to the UI of the device. The biggest hardware compatibility issue predicted will be communicating between the glove and the mobile app, so we will start there. Standard hardware on and Android phone offer a few different methods of sending/receiving data by antenna like radio frequency (RF), Wi-Fi, Bluetooth, and GPS but the decision process for choosing the best option is discussed in the communication section of the report. Therefore, the sign language glove will only be using Bluetooth as its means of transmitting data. Most new Android devices are Bluetooth compatible so this should not be an issue; however, the latest version of Bluetooth is not supported by all mobile devices. Making sure that the data sent from the glove to the mobile app is an essential element of the project but it will be one of the more challenging obstacles during mobile development and thus, the easier we can make this process the better. The newest Bluetooth technology, Bluetooth Low Energy (BLE), offers the best reliability and efficiency so we will be taking advantage of this feature if possible. Thankfully, all of Android's latest mobile devices are al BLE compatible so it passes the test.

Another hardware consideration for the mobile application is the power consumption of the app on the mobile device because battery life is an important factor for anything to do with mobile devices. However, since using Bluetooth Low Energy is the best option for wireless communication, there is no need to worry about the power consumption demanded by Bluetooth because it is already designed to be very power efficient. The other two hardware components that come into question are the processor and the memory on the mobile device. After learning more about Android's Compatibility Program, we discovered a compatibility test that all Android apps must be evaluated by during the development process which makes our job that much easier. The test ensures that an app is compatible with all software and hardware requirements of a compatible Android device, so as long as our mobile app is not pushing the limits of the processor and memory it should be fine. Most of the processing will be done on the glove's MCU anyways so the processing power required will be minimal.

Now that the hardware compatibility options are exhausted, we will test Android for software compatibility issues. The first main concern with software is whether the developers have access to the mobile development environment and a compatible Android mobile device. After surveying the team, it appears that three out of four members own an Android mobile phone and one owns an iPhone. With three compatible Android devices in the group, the next question is whether we have access to the mobile development environment. Having access to Android's mobile development environment simply means that you have access to a computer capable of running Android studio and testing the mobile

application. With four senior Electrical Engineering undergraduates at UCF, we have plenty of options for computers such as our own personal computers to the computer labs open on campus and other locations that should be able to run the software. Android Studio is capable of running on any of the popular operating systems like Windows, Mac, and Linux and therefore will be easy to find access.

The last few considerations which have been briefly mentioned already are security, audience & availability, and resources. In the introduction paragraph to Android, it claimed to have the largest install base of any other mobile platform. So, not having enough users will not be an issue; especially since UCF is a large public university and has one of the largest concentrations of mobile phone users in the world. Building on top of that, the resources available for Android developers are abundant since Android is open source and comes with tons of documentation and third-party applications that will make developing a new app much easier than starting from scratch. However, with such a big Android community comes lots of security threats which can be a major issue for certain types of applications like banking and online shopping which require a lot of personal information to be entered into their system. Luckily, our mobile application does not need any vital personal information and our project scope does not reach past building a prototype so we should not have to concern ourselves with security.

3.3.1.2.2 iOS

iOS is another very popular mobile platform that is just one of Apple's operating systems (OS). Apple has an OS for mobile phones, tablets, computers, watches, and TVs which are all written in the same programming languages but applied to each platform differently. The global market share of iPhones is only about 20% but the U.S. market share is about 50:50 compared to Android. This means that iPhones are about just as popular as Androids are in the U.S. to date and is a viable option for our project based on user availability. Just like Android, iOS comes with its own integrated development environment (IDE) called Xcode that allows apps for Mac, iPhone, and iPad to be built and tested in one program. There are some pros and cons between using different IDEs but these reasons are not as important as the other differences between mobile platforms. The iOS Developer Library is another important bonus for developers; this library has API references, programming guides, sample code, and many other resources to help build your apps. iOS mobile app development is just as popular as Android development, if not more in the U.S., and according to the market, iPhone apps generate more revenue on average than Android apps. The reason for this is that iOS users tend to buy more apps and there is research that shows that iPhone owners generally belong to an above average social class. A big portion of Android's user base is in third-world countries because Android phones are generally less expensive than other mobile devices; this means that a lot of Android's users either don't buy apps or don't spend much money on apps. In most cases, more revenue from apps is appealing to new mobile developers, but

the mobile app created for this project is not intended to be sold on the market for profit which makes this unbeneficial for us. Another road block for iPhone apps is that the process of getting an app approved in the Apple app store is notoriously difficult and takes a long time.

Moving on to the more important developer constraints, the hardware and software compatibility of iPhones needs to be considered. As mentioned in the mobile platform overview, the most important hardware features required for the sign language glove's mobile app are the display, Bluetooth, battery, processor, and memory components. The display for iPhones is not an issue because iPhones only come in a few standard sizes, unlike Android, and will be relatively easy to work with. One note is that although iPhones and iPads both operate using iOS, the apps for each platform are different because of the screen size and therefore require developers to make an app for each platform individually. Luckily, the scope of this project does not need an iPad app because that would just be an additional feature which doesn't add significant value. Another feature that won't be an issue with iPhone compatibility is Bluetooth because iPhones are all manufactured by Apple who only produces a couple different models of their phone. Thanks to Apple's high quality standards, all of the latest iPhones have Bluetooth Low Energy (BLE) which will be used for communicating with our glove. Using BLE also means that our battery life will not be a big concern for the mobile app because the power consumption from the other features will be negligible in comparison to the Bluetooth. The last piece of hardware to consider is the memory, but we know that since the mobile app will not be doing many computations there should be plenty of memory available. So now a look at the software side of mobile development on iOS will determine its feasibility.

When talking about the software side of mobile development, this is referring to the operating system required to run the IDE and the type of mobile phones that our team has access to. The types of mobile phones that our team owns are discussed in the Android section so it will not be brought up again here. The IDE for iOS is not compatible with any other computer OS other than Mac however, which will cause an inconvenience because nobody on our team owns a Mac computer. There are Macs available in the Engineering building on campus, but having a personal computer for mobile app development will be necessary for a proper testing environment. The only other main consideration for choosing the mobile platform is the security of iOS which can be ignored because this project will only be building a prototype of the sign language glove. The mobile app for this project will not be collecting personal information or storing data onto servers so there should be no need to worry about security anyways like was discussed in the Android section.

3.3.1.2.3 Windows Mobile

Windows Mobile is the last mobile platform that we will be considering for the mobile application of our sign language glove. After going through the design

considerations for the last two mobile platforms, Android and iOS, we can already deduce that Windows Mobile will not be a viable option for our mobile platform mainly because none of our team members own a Windows phone. However, it is still important to discuss what Windows Mobile is and some of the reasons why it might be a good option for a mobile platform if the circumstances were right. There will be no need to look at the Windows phone's hardware/software compatibility though since that won't affect our decision.

Windows Mobile is a mobile platform created by Microsoft and it is the third most popular mobile platform in world. Based on the Windows operating system (OS), Microsoft tried to merge the computer based interface of Windows with the mobile interface and create a new experience for mobile users. If you take a look at the newest Windows operating systems, Windows 8 and Windows 10, their GUI looks very similar to the GUI of Windows Mobile which is no surprise. Microsoft's goal is to have universal Windows applications that can be accessed on all Windows devices like the PC, phone, tablet, and more so that developers target device families and not just an OS. This allows developers to easily make applications for multiple platforms and allows users to have access to all of their apps on any Windows device. Now users have a convenient and productive way of getting the most out of their applications without having to spread themselves thin over multiple platforms and applications that aren't cross compatible.

Taking a deeper look into the Windows Mobile platform, the development tools for Windows Phone applications are actually very good. Just like the other mobile platforms, Windows Mobile has its own integrated development environment (IDE) called Visual Studio which is the preferred IDE for Windows Phone development. "It has built-in support for version control, code analysis, TDD (Test Driven Development), and even UML (Unified Modeling Language) diagram generation" (). These tools, in addition to the Windows phone community, make developing for the Windows Mobile platform a great option for developers looking to support Microsoft's vision.

Another big factor for mobile developers is the programming language used to create the mobile application. Microsoft gives developers the flexibility of working with a few different languages such as C#, C++, and Visual Basic in addition to the markup language XAML (these languages will be discussed in more detail in the programming languages section). Having three programming language options allows Windows phone developers to choose which language will give them the best performance for their mobile app's purpose or choose the one that is most comfortable to use.

The downsides of using Windows Mobile are that the user base is a small fraction compared to that of Android and iOS and the IDE, Visual Studio, requires purchasing a license which starts at about $500. In the global market today, Android and iOS own over 90% of the mobile devices which leaves Windows Mobile and Blackberry at about only 4-5% each. Therefore, getting access to a

Windows phone will be difficult, especially because none of our team members own one. Also, the Windows Mobile community is not as strong as Android's or iOS's because the small user base leads to a small developer community. Even though developing on Windows Mobile is still possible, it is very inconvenient and therefore will not be used.

3.3.1.3 Conclusion

In the Android section, it was already concluded that three out of four of our members own Android phones so it would be much easier to use Android as our mobile platform for this reason alone. However, detailed research of each mobile platform was necessary to ensure that there was not something preventing us from using Android or advantageous to using one of the other platforms for the sign language glove. So, now it is time to compare how each mobile platform performed against the given constraints and make a final decision. Looking at the more basic features first, there were no issues with security or audience base for either Android or iOS which means that we can focus on the other more prominent issues like the hardware and software. From a hardware standpoint, if one of the mobile platforms does not support all of the vital features of our sign language glove, then there will be a big problem with that platform. Thanks to our simple design, both Android and iOS support all of the hardware components that the mobile application will use. The only hardware compatibility difference is that Android devices can have a wide variety of screen sizes compared to iPhones, which have only a couple standard screen sizes. Although developing a variable sized GUI for mobile apps is more work and can cause applications to not look as they were intended, it is not a serious issue.

A more prominent decision is the software aspect of each mobile platform because developing the mobile application requires access to the IDE and a mobile device for testing, which our team does not have for all three platforms. These requirements seem fairly simple, but iOS and Windows both don't give free access to their IDE (iOS notably because Apple only supports their IDE on Mac OS) whereas Android not only has free access to its IDE but it is open source and has tons of free resources available. It only makes sense to use Android for a short-term project such as this one, given our teams' circumstances, where the cost and accessibility of the mobile platform's software are the second priority behind hardware compatibility. In order to make the situation a little clearer, consider using iOS instead of Android from our perspective. Switching to a new computer OS and getting a mobile device for testing would be a huge waste of resources (time and money) if there is no significant benefit to the design process or functionality. Another point to consider is the goal of our project, which is to create a working prototype, not a fully functional product that demonstrates our knowledge of electrical engineering. With the goal in mind, our investment into this project should only be for educational purposes and not for a profitable business model. Therefore, creating a mobile application for educational purposes makes Android an even

better choice because all of the open source resources that are available to our developers. These resources will help smooth out some of the more time consuming parts of the mobile app development process and give more time for learning and helping out with other parts of the design; plus, we can give back by adding our finished mobile app to the Android open source community. Therefore, with everything in consideration, we feel the best option for our mobile application platform will be Android.

3.3.2 Control System

While developing the best approach plan for the control software of the SLIG, the first step was formulating a plan and approach. First, the group had to determine whether the group wanted to process the data on the glove using the microcontroller, or if the group wanted to send the raw data to a device with more computing power. Before the group could answer this question, the group had to have a better idea as to which path the group wanted to take to interpret the hand-gesture data provided by the sensors. First, the group looked into the sector of computer science called “machine learning”. Machine learning uses highly specialized algorithms which “train” the system to “learn” from the data to avoid mistakes (see section “Machine Learning”).

If the group decide to use machine learning in our project, the group would likely process the data externally (not on the microcontroller) due to the higher computing power. Given that the SLIG will also be used with a stronger processor (Android phone for user interface), the group figured it would be possible to use this processor to interpret the hand gestures. In that case, the group could potentially use the microcontroller unit to collect the raw data from the sensors, normalize the data, and bundle it up into “packets” that the group can send to the Android phone via Bluetooth. Then, the group would be able to implement the machine learning algorithm on the phone to interpret which letter the raw data represents.

Given the relative complexity of implementing a machine learning algorithm, and how that complexity gets a little bit more complex by the fact that our project is a wearable, with a small microprocessor, the group looked into other methods of deciphering the data provided by the sensors. If the group went through with the machine learning, the group would likely use what is referred to as “supervised learning”. This method relies heavily on repetition, as there is a long “training” period, in which the developer would have to perform every single unique hand gesture, multiple times. This is almost like a “brute-force” method, in which the processor would have a massive pool of data from different hand gestures in the past. The processor would rely on this past data when making decisions. This requires a lot of storage on the processor, and becomes more accurate as the pool of data gets larger (more and more iterations of the same gesture, at different times and performed by different people).

The group figured that there had to be a better, more intuitive way to determine which gesture was being made by the user of the glove. The group purchased some flex sensors and performed some simple experiments with them to determine how precise they would be. What the group found was that the data coming from the sensors was not necessarily very precise, or even consistent. This was a bit surprising, but it also gave us some new ideas as to where to go with our software design approach. When the group say "inconsistent" the group mean that the output from the sensors is not always the same for a given amount of bend. It was always within the same ballpark, but it would non-deterministically vary $\pm$ 2kΩ-4kΩ. However, the group also noticed that the change in impedance between the different ranges of bend on the sensors was so great that it is essentially impossible to mistake the position of the sensor. For example, when the sensor is completely straight with no bend, the impedance measured across the sensor was about 20 kΩ. When the sensor was bent about half way, the impedance was about 35 kΩ, and when the sensor was fully bent the impedance was about 50 kΩ. This became more apparent when the group hot-glued three sensors onto a glove and built a simple voltage divider circuit on a breadboard. As the group wore the glove and measured the output at different levels of bend, the group discovered that the changes in output voltage were large enough to determine whether a finger was straight, halfway bent, or bent all the way. This didn't change for different people wearing the glove. The output was consistently about 2V when the finger was straight, and it would go down to less than 1V whenever the finger was fully bent.

This experimentation with the flex sensors lead us to realizing that the specific amount of bend in the fingers, down to the degree, was not that important to determine what gesture the user is trying to make. If the group know the range of values that can be considered "fully bent", "not bent", etc. then the group can proceed to programming the MCU without having to implement a machine learning algorithm. The group can write a function that takes in the input from each sensor as its parameters, and determines the levels of bend in each sensor. At this moment, the group think that three different "levels" would be able to accurately depict which ASL hand gesture the user was trying to perform. This is because almost every letter in the alphabet (excluding special ones which will use an accelerometer and pressure sensors) either have a finger being completely unbent, slightly bent, or "curled", and fully bent. The output from the function would be an integer, for example "1", "2", or "3" which represent the different "positions" of the fingers. If the group can successfully create this function to accurately output an integer for each "range" of bend, then the problem is simplified quite a bit. At this point, the process of actually computing what letter is being gestured by the user is a little bit easier to do because the inputs that the group would be working with are all discrete. This would essentially eliminate the problem which lead us into looking at machine learning: different iterations of the same hand gesture producing a different output due to discrepancies in the exact value of the input. Once the group have the data from the flex sensors as a discrete integer, the group can go through a series of

conditional statements which to interpret the hand gestures. These conditional statements will determine if the value from each finger is an exact combination of values that represents a certain letter (i.e.: Thumb = 1, Index= 3, Middle = 3, Ring = 3, Pinky = 2).

Another potential issue that the group had to consider was the glove sending outputs to the user interface while the user was not trying to actively make a hand gesture. This is an issue that can be mitigated by some way to put the microcontroller on “standby”. This would be a mechanism that disables the functionality of the device while it is not intended to be used. A potential solution to this can be a push button on the printed circuit board that disables the device. This push button can potentially be placed between the output of the MCU and the Bluetooth module. Under these circumstances the MCU is always analyzing the data from the hand gestures and sending outputs to the user interface, but the button breaking the line between the output and Bluetooth unit would stop the Bluetooth module from constantly sending outputs while the glove is on this “standby” mode. Alternatively, the push button could be placed on the printed circuit board in between the resistors for the voltage divider, and the input pins on the microcontroller. This would be breaking the line that provides the input to the MCU. Under these circumstances, the MCU would simply not be computing any values because the data from the user’s hand gestures would never be reaching the microcontroller.

Potentially, software can provide another solution that can be used to prevent the glove from constantly putting out outputs when the glove is not intended to be used by the user. Above, mentioned, a push button that would physically interrupt the circuit and in this way put the device on ‘standby’. Perhaps there can be a hand gesture that is used to put the device in this ‘standby’ mode. In this scenario, there would be no need to have actual hardware which ‘breaks’ the circuit. This would be a bit challenging because the microcontroller is continuously running an infinite loop, and even if the group can come up with a gesture that puts the device into this ‘standby’ mode, it would be a bit of a challenge to hold this standby mode because the glove would be constantly reading any movements that may be being made by the user. This can potentially be mitigated by making the ‘standby’ gesture a gesture that can be easily held. In this case the user can make a hand gesture (for example, a fist), and hold this gesture until they are ready to use the glove again.

The third, and perhaps least complicated of the three methods that the group can use to put the device in some sort of ‘standby’ mode can be to simply cut off the power from the microcontroller. Naturally, the MCU will be equipped with a button or switch to turn the power to it on or off. The group would have to do some research and experimenting with this to determine in what ways cutting the power on and off on the microcontroller unit would affect the software. Naturally, the MCU would have to start from the beginning, and go through its ‘initial setup’

before it goes into its infinite loop, every time that the device is brought back from the standby mode.

Regardless of which method the group uses to put the glove on ‘standby’ mode while it is not intended to be used, the glove will have some type of indication that it is in this standby mode. Perhaps an LED on the glove that changes state, depending on the state of the glove. This LED can be connected to one of the digital output pins on the microcontroller, and can be commanded by a conditional statement in the main loop. For this purpose it would be beneficial to us if the group is able to implement the ‘standby’ mode on the glove using a software solution. This would allow us to implement into the software a conditional statement that would activate the LED which indicated to the user that the glove in on ‘standby’ mode.

If the group is not able to implement the ‘standby’ mode by using software, then the group would have to somehow use hardware to determine when the circuit is broken by the push button that the group would have, and to activate the indication light without the use of software and the microcontroller.

This can maybe be achieved through the use of hardware integrated circuits like logic gates which can determine when a certain branch of a circuit is ‘hot’. If we were to go this route we can implement technologies such as Field Programmable Gate Arrays and use a programming solution such as Verilog in order to implement the task at hand, which is to activate the indication light without the use of hardware, but using hardware solutions such as logic gates. In this case the group would use a solution such as a Field Programmable Logic Array, which is essentially an integrated circuit on a chip that can be programmed on a computer and allows the developer to simulate any combination of physical logic gates to create a number of different circuits that would take a large number of physical gates to implement, and it would allow the developer to implement it all on the one FPGA board. This is significant because the SLIG is a small device, and if the group had to find space to implement a number of physical logic gates, plus the printed circuit board, sensors, and other equipment, the glove would get quite crowded and not be as practical. The next page contains general schematics of what the hand gesture interpretation system of the SLIG will look like and how it will interact with the other parts of the project.

Figure 3.22: General Schematic of Gesture Interpretation

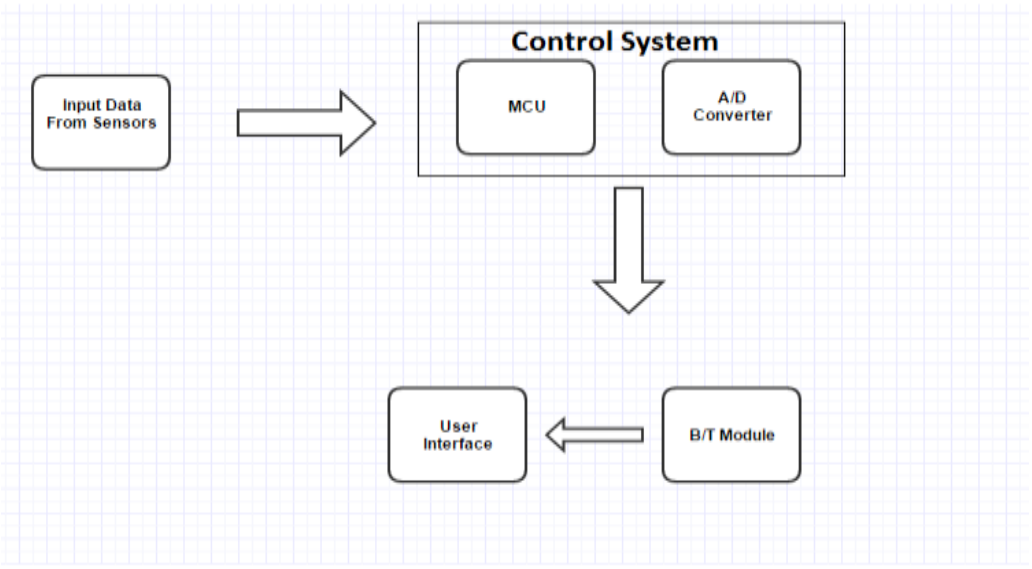
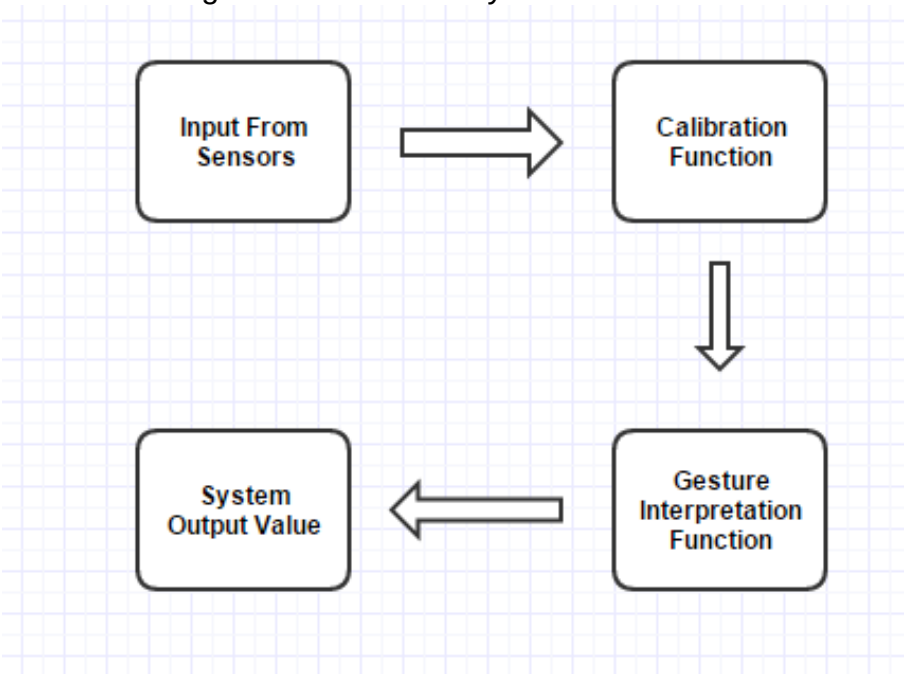


Figure 3.23: Control System Schematic



3.3.2.1 Machine Learning

The process of beginning to design the control system for the SLIG lead us to research into the subfield of computer science called “Machine Learning”. Machine learning is a subfield of computer science which, in essence, deals with computers making their own decisions based on their current environment and past events. This area of computer science is used in the implementation of things like artificial intelligence and pattern recognition. To implement Machine Learning into a project, the developer must use what is known as a “Machine

Learning Algorithm". These algorithms are used to learn from past data in order to make accurate decisions in the future. Normally, when implementing machine learning into a project, one uses one of the many established machine learning algorithms which can be adapted to the particular application at hand.

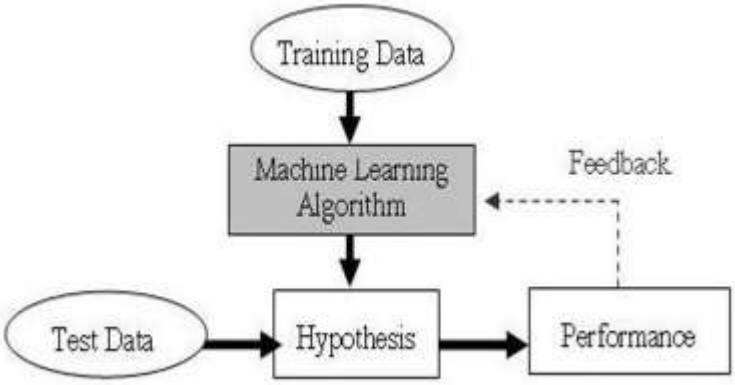
Although all machine learning applications deal with trying to use past data to make decisions and predict future events, there are different types of machine learning algorithms. These algorithms are different in the way in which they "learn" the information. One main type of learning style that is used in machine learning is the "Decision Tree Learning". This is actually a very expansive subject matter in and of itself. Besides machine learning, decision trees are used in many other areas, specifically in data-mining. This type of algorithm uses a style similar to a flowchart. The system makes an output decision on certain inputs, based on a series of "yes or no" decisions. "This or that". The system finally makes its way to the end of the tree and arrives at a decision.

There are many more types of learning styles available for a machine learning application including "Reinforcement Learning" and "Association Learning". However, if the group decides to go with a Machine Learning algorithm to decipher hand gestures coming in from the user of the SLIG, the group would likely use what is known as "Supervised Learning". This is a method where the developers perform many different iterations of the same "input" and "teach" the system what to do in the case of said input. In the case of our sign language glove, the group would go through "training" where the group performs the same letter many times, at different times, and by different people. This gives the processor memory of many different forms of the same gesture, so that when a user makes a certain hand gesture, there is a big enough pool of data inside the microcontroller that it can make an accurate decision.

After the "training" has been completed, the group would put the system through a "testing" phase. This is a time in which the group performs many different hand gestures and has the microcontroller make a decision. Naturally, there would be many errors in the decisions made at this phase. This is sort of a phase to "fine-tune" the system, after it has a big enough pool of data to make decisions. Eventually, after thousands of iterations of each gesture during the training phase, and thousands more during the testing phase, the errors made by the system should be at a minimum. Ideally, the best way to make sure that the system is flexible enough so that it can make the correct decision despite the differences in the hands of the users and their ranges of motion, is to have as many different people as possible involved in the training and testing phases. Doing this ensures that the microcontroller has a big enough sample size that it can make the correct decision nearly every time, regardless of who the user is.
Though the group has done some extensive research in the area of machine learning, and the group is also researching other methods to decide whether something so powerful like machine learning will be necessary in the implementation of our sign-language interpretation glove. Below is a simple

schematic that shows the basic working principle of implementing a machine learning algorithm into a system.

Figure 3.24: Working Principle of Machine Learning Algorithm



3.3.3 Programming Languages

The programming languages used by the software components will be discussed in this section. We will look at the programming paradigm of each language, what they will be used for, and how it will affect our other decisions concerning the mobile application and microcontroller.

3.3.3.1 Mobile Application Languages

3.3.3.1.1 Java

The main programming language used for Android development currently is Java, which is a very popular language created in 1995 by James Gosling and others at Sun Microsystems. Java was established to be used for creating internet applications and other software programs. It is an object-oriented programming (OOP) language just like Objective-C, C++, and C#, which means that it is based on the concept of “objects”. These objects are data structures that contain properties just like a real object does. For example, a dog has a breed, hair color, age, weight, and more which are all properties of that dog. OOPs also have methods which are essentially a procedure created in the code that can be called upon just like the properties of the objects. One of the most important features of object-oriented programming languages is that they are class-based, meaning that objects are instances of classes. A class is simply like a template or blueprint for creating objects that provides initial values for state variables like the properties and methods. Using classes, properties, and methods are essential to programming in an object-oriented language like Java and they can be very useful for certain applications.

Taking a more advanced look into object-oriented programming, there are other capabilities that these languages have to offer. One of the most commonly used features is inheritance, which is when an object or class is based on another object or class. Inheritance takes on the same implementation as the object or class that it is based from and changes something about it to make it unique but still have the same basic properties and methods as before. Inheritance in OOP is just like inheritance with people; a child inherits many of the same genes as their parents. The benefit of inheritance is to minimize the amount of repetitive code so it is easier to create new classes and reuse old ones without writing them again.

3.3.3.1.2 Swift & Objective-C

The two programming languages used for iOS mobile phone applications are Swift and Objective-C. Swift is the new language that Apple is pushing iOS developers to start using more because it is supposed to be much easier to work with than Objective-C; however, Objective-C is still available for those to prefer. With Swift, developers are still using a language that is based on the best features of C and Objective-C, but it removes the compatibility issues that come with them. It is a multi-paradigm programming language that was created not only for iPhone developers but for all Apple software like iOS, OS X, watchOS, and tvOS. Swift was Apple's way of creating a new programming language that was simpler and easier to use than Objective-C and was described as "Objective-C without the C".

Since we don't know anything about Objective-C, we will take a look into it so we can understand the differences from Swift. Objective-C used to be the primary programming language used for writing applications and software for iOS and OS X. Just like Swift was a spin-off of Objective-C, Objective-C is based off of the language C. The difference between Objective-C and C is that Objective-C is an object-oriented programming language that has many more capabilities and better runtime.

Taking a look into the compatibility issues, we have to make sure that the programming languages used are compatible with the version of the operating systems and the hardware on the phones. The old software operating systems that Swift supports are iOS 7, iOS 8, and iOS 9. This is good news for us because Apple releases updates for their iPhones relatively frequently and users will most likely have upgraded from any phone that doesn't support iOS 7 or higher. As far as the hardware goes, Swift will support iPhone 4, 4s, iPhone 5, 5C, 5S, and iPhone 6, 6 Plus. Therefore, we have no issues with compatibility and we don't have to worry about this aspect of using iOS as a potential mobile platform.

3.3.3.1.3 C#, C++, and Visual Basic

The programming languages used by Windows Mobile development are C#, C++, and Visual Basic. When you begin developing on Windows mobile you have to consider which programming language will be best used for your application because all three have advantages and disadvantages. We will begin by looking at C++ because it is the native language for Windows Mobile.

The term *native language* means that the language talks directly to the hardware; so in this case, C++ talks directly to the hardware on the Window's phone. There are many benefits to using the native language of a device because it has more control of the hardware, the application size is smaller, and it is capable of better performance since there are no intermediate steps during execution. However, native languages are usually a lower-level language which means that it is more complicated to use and can cause issues during execution like a memory-leak that will cause the device to crash. Knowing these limitations, developers are able to create applications that run quickly and efficiently for improving the performance of apps such as fast-action games.

When a developer is creating an application that does not require high-performance, there are other programming languages available like C# and Visual Basic (VB). These languages are both *managed* development languages, meaning that they have built-in services that handle memory-management and garbage collection. These services come at a cost of performance, but using an unmanaged language like C++ requires more attention to memory management and security. A couple other features available to developers using C# and VB are the libraries and developer tools. Both of these languages support the .NET Compact Framework – a library of common classes that simplify many tasks that a developer may encounter. There are also developer tools for the visual interface designer that allows developers to drag and drop buttons, text boxes, and more. For more advanced interface design, there is another window that has the auto-generated code from the drag and drop interface and all of the properties that can be modified as needed.

3.3.3.1.4 XML & XMAL

The programming languages used for the entire graphical user interface (GUI) design are XML and XMAL. XMAL is just Microsoft's version of XML used for Windows phone development only. These markup languages are designed for formatting documents, web pages, and more just like the language HTML is used for most web pages. These languages don't really do anything, meaning that they just carry information wrapped in tags and require some software application to send, receive, store, or display it. Below is an example of XML source code that displays a note to Mike, from Jones, stored as XML. As you can see, the XML code is just formatting the information inside of the note by using tags that interpret the content of the page. The IDE Android Studio comes with its own

XML graphical layout editor which will be used to design the interface of the mobile application. This editor comes with drag and drop capabilities and allows you to preview your design as you create the layout. Also, if there needs to be more advanced editing of the layout there is a text editor interface in addition to the drag and drop interface that looks like the XML source code sow on the next page in Figure 25.

Figure 25: XML Source Code and its Output

```
<note>
  <to>Mike</to>
  <from>Jones</from>
  <heading>Invoice</heading>
  <body>You owe me $20 for dinner.</body>
</note>
```

Note

To: Mike

From: Jones

Invoice

You owe me $20 for dinner.

3.3.3.2 Control System Languages

The control system part of the software for the SLIG can be implemented using a number of different languages. The group has considered several languages, including Java, C++, C#, Python, etc. One thing that came up several times while conducting our research was that though it is possible to implement an object-oriented language on a microcontroller system, it is not the most practical and is often decided against.

The control system in our project will be consisting of several functions (seen in sections above) which the group will write. However, there does not seem to be a need to create multiple classes, and have a hierarchy of files containing all of the parts of the program. For this reason, the group has shied away from employing an object-oriented language for the control system portion of our project. Given the requirements of the control system, the group believe that using a language such as “C”, will provide us with enough resources to decipher what hand

gesture is being performed by the user. C allows us to structure the program the way that the group want, in the way that the group can have an infinite loop contained in our main routine, write and call functions as the group need, and contain the entire program for the control system easily in one file.

Since the control system that the group is writing is going to be implemented on a microcontroller unit, it will be close to the hardware and believe that using a language such as C would be the best option. The program could also be written using assembly language, but it would be quite the challenge to keep track of stacks and individual registers, so C seems to be the best option for the control system.

4. Constraints

There will be many constraints that will keep the group from implementing this project as efficiently as hoped for. For one, there is an obvious time constraint. The project must be built within the time frame of two semesters. This means that there is not enough time to perform really extensive research that could yield a much more efficient and desirable result. Also, the time constraint prevents the group from being able to really test with several different solutions to make the glove the best that it can possibly be. Ideally, with more time, the group would be able to build several prototypes. In each prototype, something new can be improved on from the previous iteration. After several prototypes, the final result should be something that is completely free of errors and provides the user with a bug-free experience that has been tested many times over and is ensured to be functioning at the best level possible. With the time that is given, there will inevitably be several setbacks, and there will only be time to build one, maybe two prototypes. At which point the group will have to go with what is built by then, and live with the limitations.

There will also be some health constraints to the project. This is a product that will ideally be operated by persons with some form of speech impediment which keeps them from performing efficient oral communication. This product will be very helpful to said individuals. However, it must be ensured that the user can safely use the product without potential to affect their health in a negative way. For one, the materials that may be used for the actual glove itself can cause an allergic reaction on some users. This makes the selection of the material used on the glove very important (discussed in the “Glove” section). It is also important to note that the glove is going to be worn on people’s bodies. This is a glove that has live electronic parts functioning on it and it can be very possible for a user to experience a shock if the electronic components are not properly placed in an area where it will be unlikely for the user to inadvertently touch them and hurt themselves. For these reasons, the health of the end user of the Sign Language Interpreter Glove needs to be taken into consideration when finalizing the design.

Manufacturability is also a very real constraint that will be faced when implementing the SLIG. This is a fairly complex product, and to build it as sleek and efficient as would be liked, some rather specialized materials will be needed. For example, in an ideal world, the glove would look like any other glove that somebody would wear. Nothing about it would indicate that it is in fact an electronic device with electrical parts inside of it. This would require some handy textile skills and tools. For example, we would be able to sleekly sew over all of the sensors. To hide the printed circuit board, battery, and all other electronic components, the glove would have to have some sort of hidden compartment, perhaps around the sleeve, or somewhere where it would be difficult to see and also where it would be safe for the user to wear the glove and not have to worry about damaging the glove or potentially hurting themselves by the way of an electrical shock encountered by touching an electronic component.

An alternative would be to use two gloves. An inner glove, which has all of the electronic components mounted on top of it, and an outer glove which goes over the inner glove, and over all of the electronic components such as to “hide” the electronics. This, however, is fairly inefficient. The user would experience the “sag” between the gloves. Also, as the outer glove moves and glides over the inner glove, there is a real potential for electronic components to get damaged by being rubbed against the outer glove constantly.

There are ethical constraints involved in implementing the SLIG as well. This project and its design are fully original from the group. However, the idea of a glove that translated sign language into text has been done several times. In fact, a UCF group implemented the same idea several semesters back. Following the ethics of engineering, it is imperative that the group makes clear that although the idea for this project has been done in the past, the Sign Language Interpreter Glove has been fully designed by the group. There is also the issue of permission to use certain images. The group has emailed all appropriate entities to ask permission for the use of certain images, as needed. At the end of this document, the appendices will have more information regarding permissions and will include all emails sent.

Economic constraints also have to be taken in to account in the implementation of the Sign Language Interpreter Glove. The group must be sure that every step taken and every part purchased is correct. If things have to be purchased twice, or if a wrong part is purchased, or any other setback that causes the group to have to spend more money than anticipated would be a big blow. For this reason it is important for the group to be very meticulous and determine that each part that is ordered is in fact the part that is actually needed for the project. It will also be beneficial to try to purchase certain parts together and from the same vendor, so as to save some money when it comes to shipping and handling.

Ultimately, every part should be researched by every possible vendor that offers it, and should be purchased from the most economically viable option possible. This may mean that some parts will need to be ordered from overseas and will take quite some time to arrive. This ties in with the time constraint mentioned above. The parts will have to be ordered in a timely fashion (ideally before the end of the Senior Design I semester) because sometimes parts ordered from overseas vendors can take up to a month to arrive. In case of a setback, the group will try to refer to domestic vendors which can have the parts delivered within a short period of time. However, this may affect the economic constraint, as it will cost more.

5. Standards

5.1 Bluetooth

IEEE Std 802.15.1™-2005(Revision of IEEE Std 802.15.1-2002): This standard discusses Wireless Body Area Networks (WBAN) which have become an important technology in providing real-time health monitoring of a patient and diagnosing many life threatening diseases. IEEE 802 has established a Task Group called IEEE 802.15.6 for the standardization of WBAN. The purpose of the group is to establish a communication standard optimized for low-power in-body/on-body nodes to serve a variety of medical and non-medical applications. Bluetooth technology falls right in this field of communications and so the team should heed any pertinent recommendation in this standard.

5.2 Android Applications

Google's Android developers have set up a multitude of standards and qualification on the different aspects of an android application. These aspects include visual design, user interaction, functionality, performance, stability and interaction with Google Play. Each standard has a four character ID and details specific instructions on how a finalized application should look or function. These standards as a whole are an excellent rubric by which the group's application should be measured.

The details on "Standard Design" UX-B1 are shown below as an example.

Area	ID	Description	Tests
Standard design	UX-B1	App follows Android Design guidelines and uses common UI patterns and icons: a. App does not redefine the expected function of a system icon (such as the Back button). b. App does not replace a system icon with a completely different icon if it triggers the standard UI behavior. c. If the app provides a customized version of a standard system icon, the icon strongly resembles the system icon and triggers the standard system behavior. d. App does not redefine or misuse Android UI patterns, such that icons or behaviors could be misleading or confusing to users.	CR-all

5.3 Lithium-Ion Batteries

The group decided on using a polymer li-ion battery to power the SLIG. Thus, available standards on the safety of Lithium-Ion batteries would be very relevant to the project as it continues from the design stage to the construction stage. The team may use these standards as guidelines for the proper management of the SLIG's battery. Separate international standards organizations have generated the following safety standards for Lithium-ion batteries (International Electrotechnical Commission (IEC) and International Organization for Standardization (ISO). The most relevant standards are listed below.

- IEC 62133-2: safety requirement for portable battery cells
- IEC 62660: batteries for EV/HEV applications
- IEC 61427: secondary cells and batteries for renewable energy storage

6. Design

6.1 Hardware Design

6.2.1 Flex Sensors

6.2.1.1 Selection

For this project the team decided to choose the spectrasymbol flex sensor of 4.5 inch length supplied by SparkFun electronics. This model was chosen for its wide availability from multiple vendors and its strong reputation as a reliable part. It should be the ideal length to fully cover the SLIG glove and allow a full range of flexing motions for the user. The only possible fault with this model was the problematic base connection that was reported to break easily if not handled with care. Below is a picture of the flex sensor showing its size relative to a person's hand.

Figure 6.1: Flex Sensor

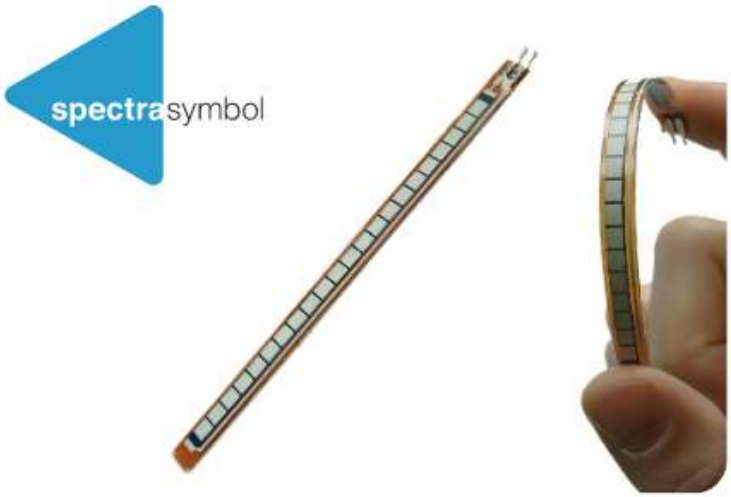


Figure 6.1 was reprinted with permission from SparkFun Electronics.

6.2.1.2 Integration and Schematics

This project will call for the use of five separate flex sensors, one for the finger of the glove. They will run from the tip of the fingers to approximately the wrist area where they will be secured with some sort of adhesive as it stands now. As the sensors are bent, their varying resistance will yield different output voltages, which can be measured and interpreted to determine the configuration of the user's hand.

There are a few ways the flex sensors can be integrated in the circuit. The most basic flex sensor circuit makes use of a voltage divider and an impedance buffer. The impedance buffer is a single sided operational amplifier, used with these sensors because the low bias current of the op amp reduces error due to source impedance of the flex sensor as a voltage divider. Suggested op amps are the LM358 or LM324. Refer to the figure below.

Figure 6.2: Basic Flex Sensor Circuit

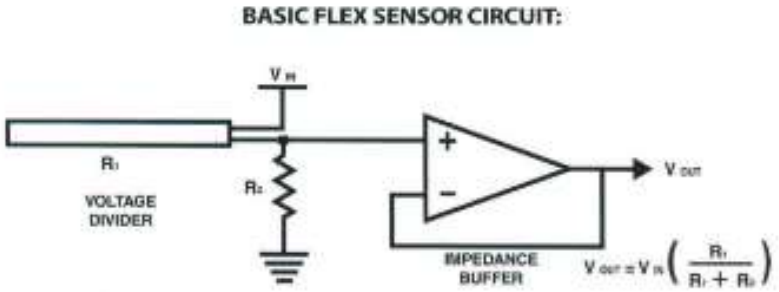


Figure 6.2 was reprinted with permission from SparkFun Electronics.

There are other configurations that can be used to achieve certain results. With the Adjustable Buffer circuit a potentiometer can be added to the circuit to adjust the sensitivity range. Refer to the figure below.

Figure 6.3: Adjustable Buffer Circuit

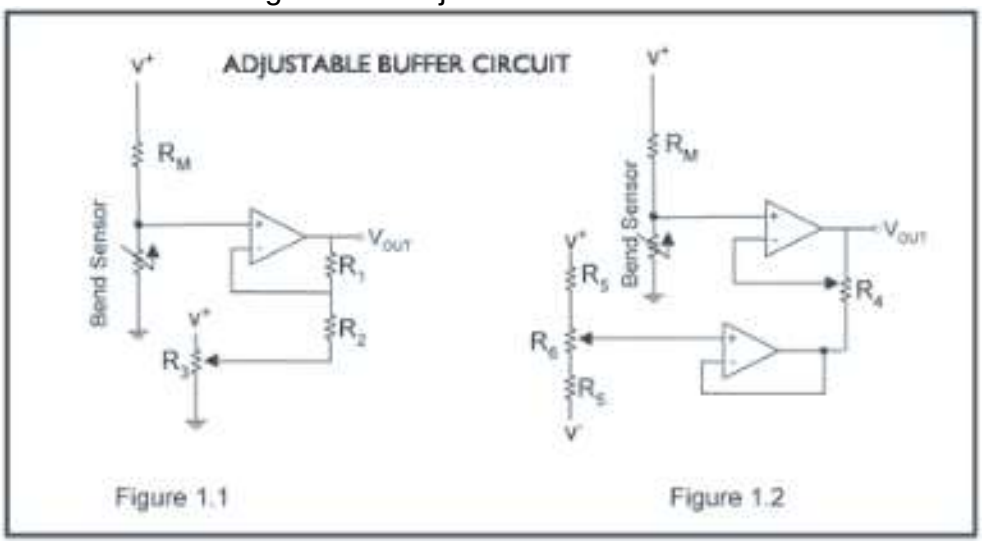


Figure 6.3 was reprinted with permission from SparkFun Electronics.

In the "Resistance to Voltage Converter" circuit, the sensor is used as the input of a resistance to voltage converter using a dual sided supply op-amp. A negative reference voltage will give a positive output. This should be used in situations when you want output at a low degree of bending. Refer to the figure below.

Figure 6.4: Resistance to Voltage Converter

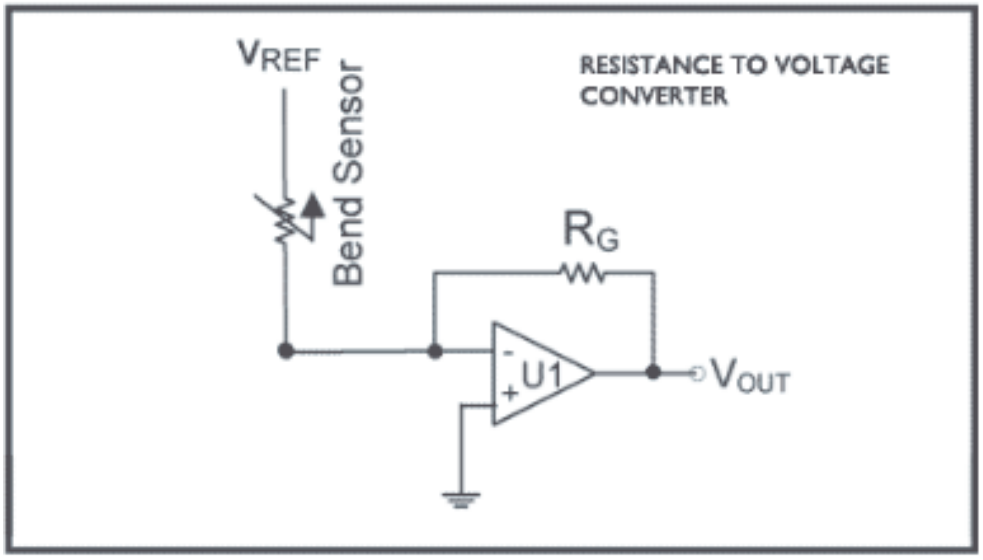


Figure 6.4 was reprinted with permission from SparkFun Electronics.

6.2.2 Accelerometer and Gyroscope

6.2.2.1 Selection

For this project the team has decided to choose the SparkFun 6 Degrees of Freedom IMU Digital Combo Board - ITG3200/ADXL345 supplied by SparkFun electronics. This is a simple breakout board that provides two important functions in one. For the accelerometer, it utilizes an ADXL345 accelerometer and for the gyroscope it utilized an ITG-3200 MEMS gyro. This combination will allow the SLIG to measure movements with a full 6 degrees of freedom. The sensors communicate over I2C and one INT output pin from each sensor is broken out. Since this meets our requirements for a simple and tiny board that offers 6 degrees of freedom, this is definitely a good choice. The board is shown below.

Figure 6.5: Relative size of Combo Board

Figure 6.5 was reprinted with permission from SparkFun Electronics.

The two components for this IMU combo board each offer impressive specifications as briefly mentioned in the sections before. The ADXL345 is a small, thin, low power, 3-axis accelerometer with high-resolution (13-bit) measurement at up to ±16 g. It can readily measure the static acceleration of gravity in tilt-sensing applications, which would be the most pertinent to this project. Its high resolution (4 mg/LSB) enables measurement of inclination changes less than 1.0°. The ITG-3200 likewise provides high resolution, detailed level of programmability and competitively low power consumption.

6.2.2.2 Integration and Schematics

Due to the nature of this breakout board, integrating it to the main circuit board should not be difficult. It will have to go on the circuit board and collect the relevant data from there, which will be somewhere in center of the glove. Below is a breakdown of each individual part of this combo board.

Figure 6.6: Schematic for Combo Board

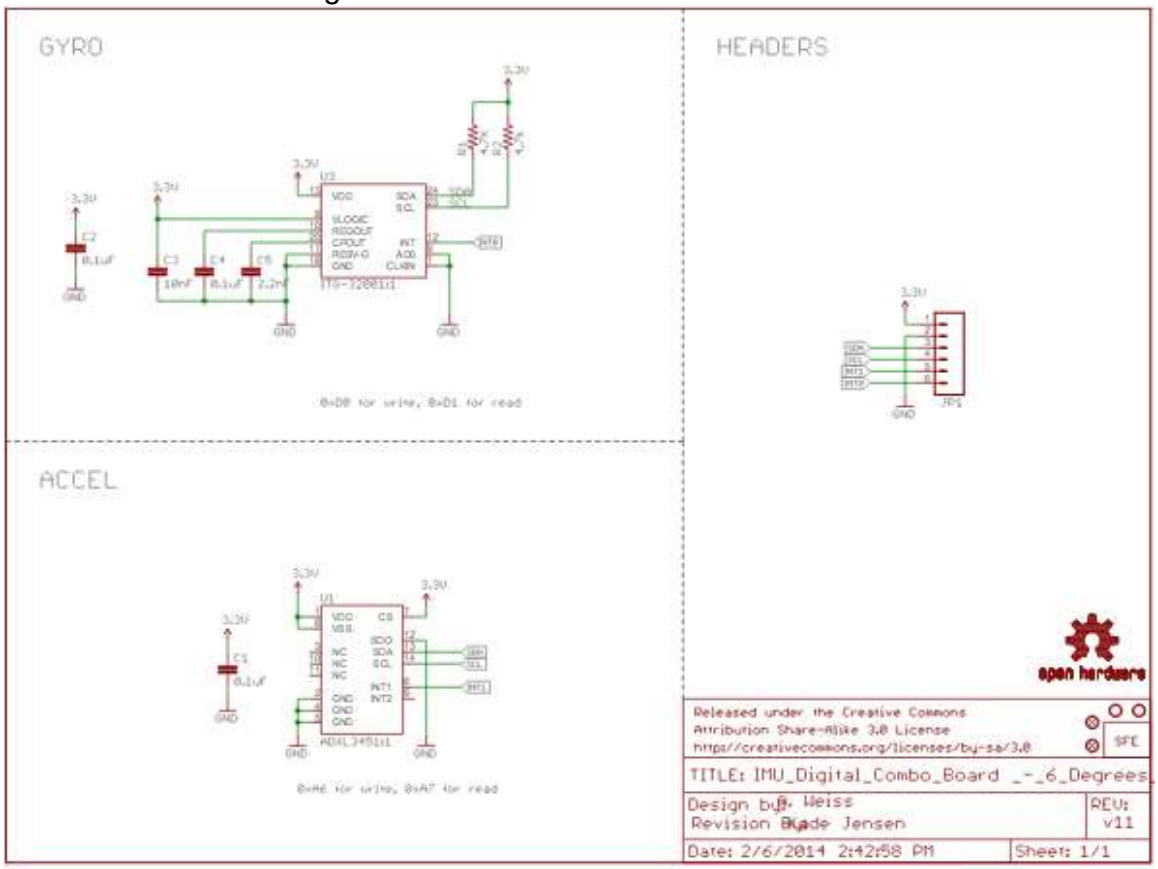


Figure 6.6 was reprinted with permission from SparkFun Electronics.

6.2.3 Contact and Pressure Sensor

6.2.3.1 Selection

Due to the limited models of contact sensors available, the team has chosen to use force sensitive resistors to determine whether contact is being made between any two fingers when this is important in determining the sign language letter being signed. The most important points of contact will be between the middle finger, index finger and thumb of any one hand. More specifically, the team has chosen the Force Sensitive Resistor 0.5" from Sparkfun. This model has a 0.5" diameter sensing area and a length roughly over 2 inches. Similar to the flex sensor, applying any force or pressure to this resistor's sensing area will reduce the resistor's resistance accordingly which in turn causes changes to the voltage across the resistor. This FSR (Force Sensitive Resistor) can sense applied force anywhere in the range of 100g-10kg and has two pins extending from the bottom of the sensor with 0.1" pitch. Below is a visual representation of the size of the sensor.

Figure 6.7: Size of FSR sensor

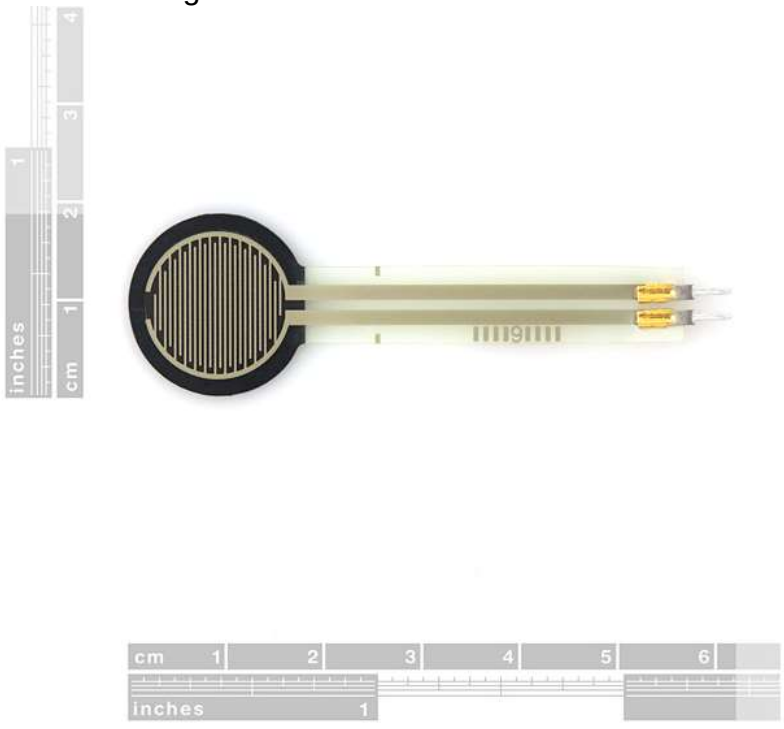


Figure 6.7 was reprinted with permission from SparkFun Electronics.

6.2.3.2 Integration and Schematics

The SLIG glove will possibly require up to five different FSRs but it is possibly that this number could be reduce down to three after further testing is done. The most likely locations for the sensing areas will be either the innermost side or bottom of the middle finger, index finger and the thumb. Like the flex sensors,

these FSR's would require a voltage divider circuit and the like to make its output voltages readable by the rest of the device. Below is a schematic of the most basic voltage divider circuit.

Figure 6.8: Basic voltage divider circuit

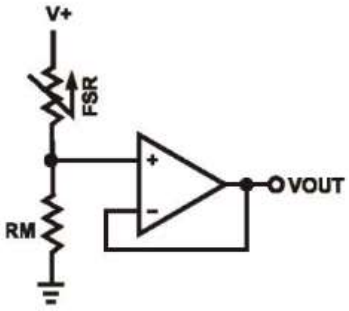


Figure 6.8 was reprinted with permission from SparkFun Electronics.

Similar to the FSR Voltage Divider above, the adjustable buffer interfaces below isolate the output from the high source impedance of the Force Sensing Resistor. However, these alternatives allow adjustment of the output offset and gain.

Figure 6.9: Alternate Circuits

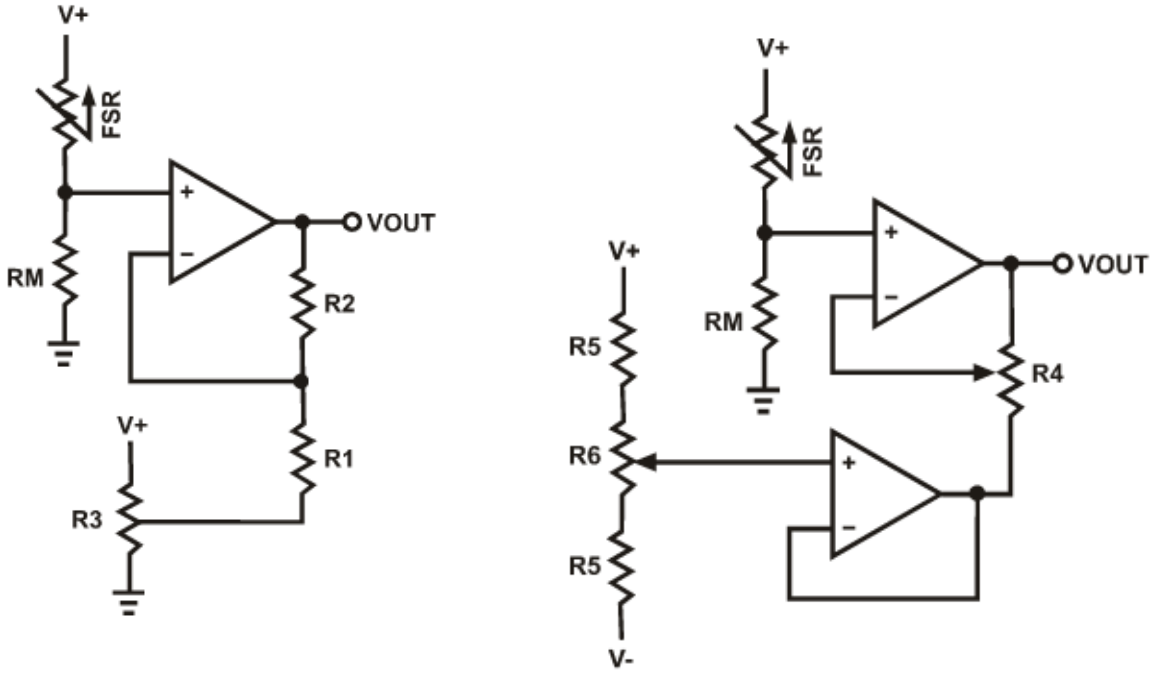


Figure 6.9 was reprinted with permission from SparkFun Electronics.

6.2.4 Glove

6.2.4.1 Selection

The team has decided to go with the UA Strikeskin Tour glove as the base glove for the SLIG. The primary motive behind this decision was the team's stringent budget and the fact that the team only has to pay for one glove because this glove does not come as a pair. Nonetheless, this is still a quality glove that is mostly made out of leather (so there are virtually no concerns with allergies) and boasts a high level of flexibility and movement control, which is crucial to the proper functioning of the SLIG.

6.2.4.2 Integration

The glove will be the base to which the sensors, the PCB and all other electronics will be attached. This will be accomplished using a combination of adhesives, ties, sowing and possibly other methods. Below is a picture of the glove, so that the overall design can be seen.

Figure 6.10: Selected Glove

Figure 6.10 is still pending permission from Under Armour.

6.2.5 Hand Gesture Recognition

6.2.5.1 Interpreting Flex Sensor Data

Regardless of which method the group chooses to use to decipher the hand gestures performed by the user, the group will first need to interpret the data coming into the microcontroller from the flex sensors. The way in which the flex sensors work is, they are essentially variable resistors which change their resistance in proportion to the amount of bend present on the flex sensor. In order for the microcontroller to sense resistance from the sensors, it would need to have some intricate things going on such as a constant current source and then perform some calculations to determine what the resistance is across the sensor. To mitigate this, the group simply hooks up a voltage divider circuit to the sensors. To do this the group places the sensor in series with a fixed resistor, and put a constant voltage across the two resistors. Depending on how much the sensor is bent (and, as a result the resistance across it), the ratio of the total voltage that will be across the sensor will change. In this regard, the group can simply say that the voltage across one of the resistors is our "output". Note, that even though the sensor is a variable resistor, the 'output' of the sensor that the group sends into the microcontroller is a voltage amount.

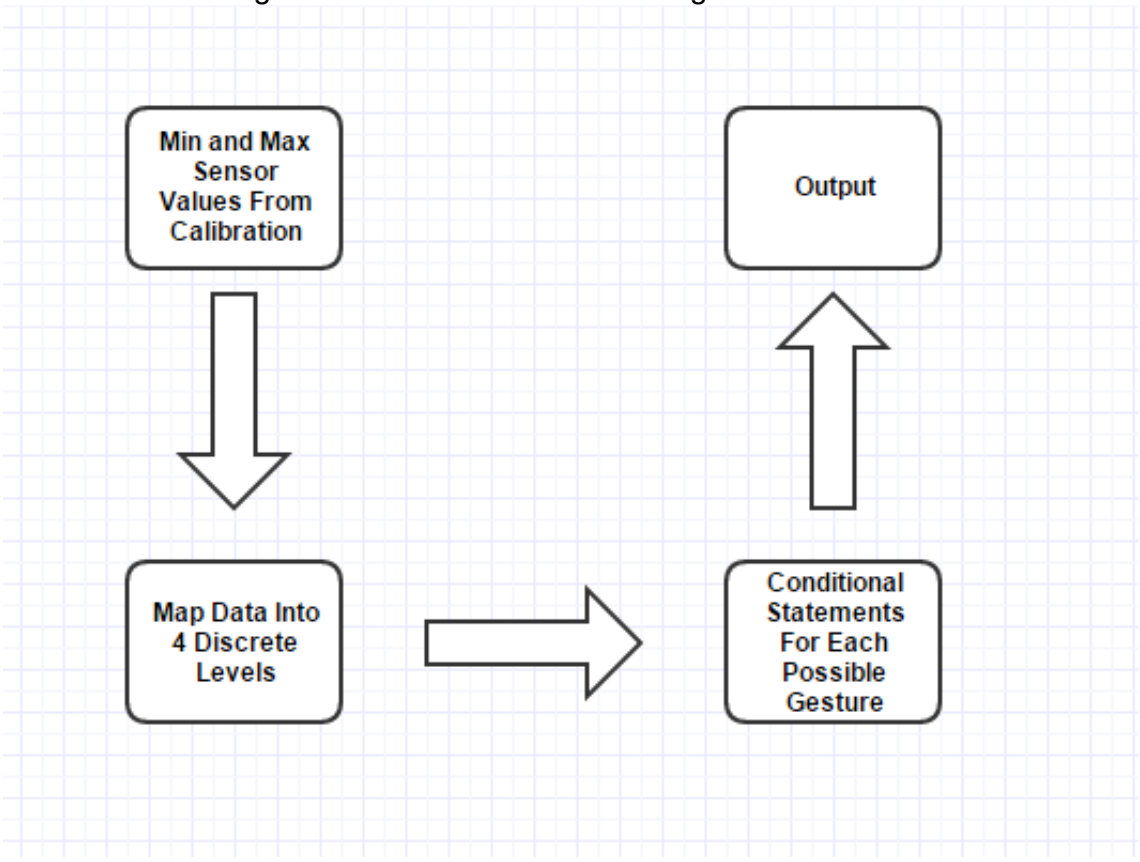
Once the group has the output of the sensors as a voltage, this voltage will go into one of the pins on the microcontroller. The group would have to either choose a microcontroller that comes equipped with analog-to-digital converters or the group would have an external analog-to-digital converter elsewhere on the printed circuit board. This is because the microcontroller is a digital processor, and it can only interpret digital data. The voltage coming out of the sensors is an analog signal, so the group would need to convert that signal into a digital signal.

However, before the group can implement the design for the sensor circuit, the group would need to decide what the optimal voltage level is to hook up to the voltage dividers. One thing to take into consideration when deciding what the optimal voltage level is the voltage available from our power source (lithium-ion battery). At this point the group is still undecided on whether or not the group will need to have regulation at different voltage levels. The group would also need to take into consideration the optimal voltage level that the microcontroller can read and interpret. Also, another important factor in interpreting the sensor data is choosing the correct resistor value for the voltage divider. The flex sensors that the group are using have a resistance that fluctuates somewhere between about $20\ k\Omega$ and $50\ k\Omega$. This means that the group would have to optimize the voltage level hooked up to the two resistors (the sensor and the fixed resistor), and also wisely choose the resistor that is fixed.

The group has performed some elementary experimentation with flex sensors and resistors on a breadboard. Our results show that if the group use about 5V as our source for the voltage divider and use $10\ k\Omega$ for the fixed resistor, the

group get a nice range of values in our output. When the group say a ‘nice’ range of values, the group mean that the output voltage from the voltage divider is small enough to be able to go into the microcontroller, but also large enough so that there is a large, discernable difference between the voltage levels when the sensor is completely straight, about half-way bent, and fully bent. For example, with a 5V source and a 10 $k\Omega$ resistor, the output from the voltage divider circuit is about 3.5V, which is right in the wheelhouse of ‘preferred’ voltage levels of most microcontroller units. The values mentioned above may change as the group proceed with the building and prototyping of our project, but at the moment it seems as if these are the optimal values for best performance from the flex sensor circuit. After these things have been established, the microcontroller will be receiving the data from the flex sensor as a voltage level, and then the group can proceed with the software solution that the group decides is best for our project. Below is a schematic of the function that will go through and determine which hand gesture is currently being performed.

Figure 6.11: Hand Gesture Recognition Function



6.2.5.2 Determining Ideal Voltage Level

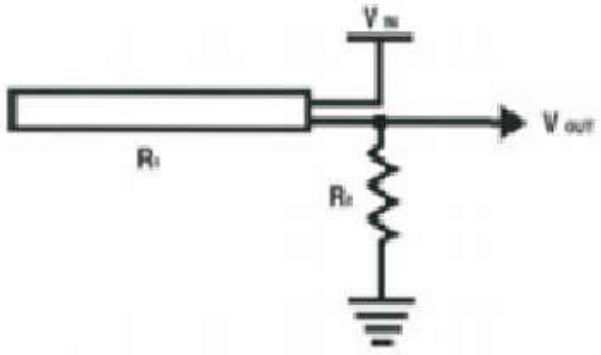
Another big, important factor in designing the system with the flex sensors is determining the ideal conditions for the voltage divider that the group is building for each finger. As mentioned previously, the flex sensor is a variable resistor, so

two things are constant in the circuit: the total voltage level, and the resistance of the resistor which combines with the sensor to create the voltage divider. It is important to choose the correct values for these components, because if not, it will be difficult for the microcontroller to make correct decisions if the group are not providing it with an ideal representation of what is going on at the sensor level.

The first thing to consider here is that the output voltage will be sent into the microcontroller unit. So, the voltage divider has to be designed in a way so that when the flex sensors are fully straight, (the output voltage will be at its maximum), the voltage is low enough as to not damage or be out of the range that the microcontroller can handle. But, it also cannot be too low, because when the fingers begin to bend, the output voltage will begin to drop and the group want it so that when the fingers are fully bent (the output voltage is at its minimum) the voltage is still high enough so that the microcontroller can accurately read it and make a decision on it. For most microcontroller units, an ideal voltage level for its input and output pins are at about 3.5 volts. The group wants to choose levels that do not exceed this amount of 3.5 volts, but also large enough to be somewhere near that range when the output voltage is at its maximum value (fully straight).

Clearly, these values will have to be tweaked along the way the group actually have the system built and the group see what results the group are getting. But, from our calculations and early experimentation, a total voltage into the voltage divider of about 5 volts seems to be ideal. With this voltage level, the group can pair it will a resistor of about $25\ k\Omega$. This will create results that send a good output out to the microcontroller. When the fingers are fully straight, the output voltage will be about 2.5 volts, and while the fingers are fully bent, the output voltage will be at about 1.67 volts. This creates a clearly identifiable change in the output from when the hand is fully straight and when it is fully bent. This also ensures that the output voltage is always within a safe range that will not harm the microcontroller. Below is a schematic that shows how the flex sensor will be connected to the voltage divider.

Figure 6.12: Sensor Voltage Divider Circuit



6.2.5.3 Calibration

Everybody has hands of different shapes and sizes. Some people have larger fingers, and others have shorter ones. Also, different people have a different range of bend in their fingers. This would create some degree of difficulty when it comes to determining what hand gesture a user is trying to make. Each hand gesture will be a set, universal instruction, while the users are inputting slightly different inputs due to the differences in their hands and bending motions. To mitigate this, the group can come up with a way to calibrate the system prior to first use every time that it is powered on. In this way, the group can ensure that the system knows how this particular person's hands move and it can tell what gesture the person is trying to make, based on the characteristics of their hand.

The calibration process will be completed by a function dedicated to calibrating the glove. The first big piece of information that the group want to retrieve from the user, is what their range of motion is. To do this, the group will have the user put on the glove, and power it on. At this point, the control system software will, as almost every other microcontroller unit does, have a set of 'preliminary' instructions that take place before the main, infinite loop is initiated. In this step is where the group will include the function for calibration. The group will have the user bend each finger all the way, almost as if making a fist. The function will take this information and record it in certain variables that the group will use when determining which hand gestures are being performed. One variable that the group would need to record is the maximum amount of bend in each one of this particular user's fingers. This would inform the system how far this particular user can bend their fingers. Had the group not done this, the system wouldn't be able to tell when one user is trying to perform a certain gesture which required a certain finger to be fully bent, while another user may perform a hand gesture that has the same amount of bend on that finger (from the flex sensor's perspective) while that is not 'fully bent' for that user because of the characteristics of his hand. Now that the system has a reference as to what the current user considers his 'maximum bend', it can make decisions accordingly when it comes to deciding what hand gesture that user is making. Similarly, another key piece of information that the calibration process would get from the user is the current user's minimum bend amount.

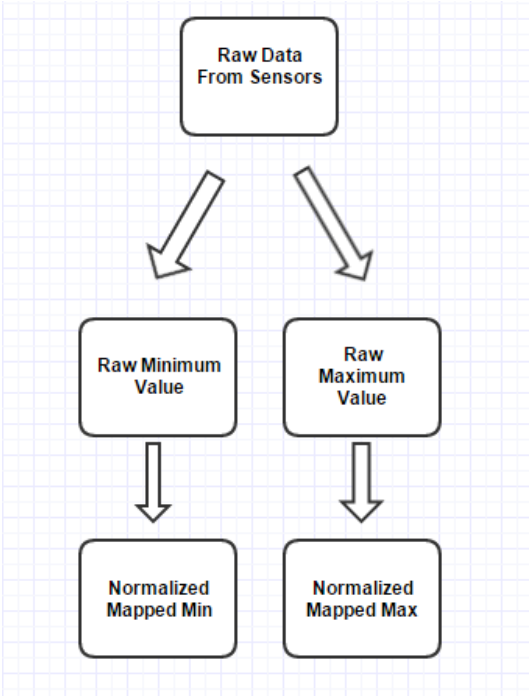
Some people have some degree of bend on their fingers even when they are not actually 'bending' them. This can cause errors if the system does not have an accurate representation as to what the current user's 'fully straight' hand position is, in terms of degrees of bend felt by the flex sensors. Having the calibration process measure the minimum amount of bend found in that user's hand when they just relax their hand with no particular intention of bending make the program assume that when this amount of bend is present on the fingers, for the purposes of the control system the hand is 'fully straight'.

Once the calibration process acquires all of this data from the user, it now has to 'normalize' it, in a sense, so that the main program (infinite loop) knows what to do with the live data that it will be receiving later. To do this, the group will write a function that does something similar to the 'mapping' that the group will also be using to decipher the actual hand gestures. As mentioned earlier in the "Control System Research" section, the typical amount of resistance found on the flex sensors as a result of an average person wearing the glove and bending their fingers ranges from about $25\ k\Omega$ to $50\ k\Omega$. As you know, these ranges of resistances will produce a specific range of output voltages from the voltage divider (depending on the total voltage chosen and the constant resistors).

The group will 'map' this data collected from the initial calibration process to the 'reference' level, which the group will choose based on experimentation when the group choose our voltage level for the voltage divider and the constant resistor. The calibration function is essential because this will provide the gesture interpretation function with normalized information that the function can use to make more informed decisions. Below is a simple schematic displaying the working principle of the calibration function.

The calibration process will begin when the microcontroller is first initialized. This will occur before the infinite loop begins its course. To re-calibrate the glove, the user will have to re-start the microcontroller by pushing the reset button on the PCB or turning power off and on to the microcontroller. This will allow the microcontroller to begin all of its processes again, and in this way re-initiate the calibration process for a new user.

Figure 6.13: Calibration Function



6.2.5.4 Non-Standard Hand Gesture Recognition

One big issue faced when designing the control system for the SLIG is that there are a number of letters in the American Sign Language alphabet that are not able to be captured fully by the flex sensors. This is because they are either very similar to another letter, and the flex sensors will not be able to determine which of the two letters are being performed, or because the actual letter requires some specific motion that is not necessarily just a certain bend in the fingers. To mitigate these issues, the group will need to equip the glove with other hardware and implement this hardware into our software.

One main piece of hardware that can help us with many things is the accelerometer. As described in another section, the accelerometer essentially measures the hand's position with respect to Earth's axis. This device can give us an (x, y, z) representation of the orientation of the hand. With this information the group can determine whether the hand is being tilted in a certain direction. For example, the letter "J" will be the biggest component of our project that will make use of the accelerometer. This letter literally requires the user to draw out a "J" in the air with their pinky.

In these cases, there is not much that can be captured by the flex sensors. So, the accelerometer can measure the change in position of the orientation of the hand and determine when the user is making that "J" motion with their pinky. From a software perspective the group would be able to measure when the change in orientation coincides with that of the "J" motion. To do this the group would have to experiment with the accelerometer once the group have it on our glove, and see what kinds of fluctuations happen when that specific hand motion is made. This change in orientation would be combined with the combination of bend on the flex sensors to accurately make a decision on what hand gesture is being made. The output from the accelerometer that would go into the microcontroller is a voltage that is proportional to the change in orientation of the accelerometer (see the "Piezoelectric Accelerometer" section above).

Another difference that the group will have to deal with is the fact that the letters "U, V, and R" are very similar. Looking at it from the perspective of the amount of bend experienced by the flex sensors, these three letters are almost indistinguishable. The glove would have to be further equipped with a little more additional hardware to distinguish between these three very similar gestures. The letters "U and V" are almost identical: they both have the index and middle fingers fully straight, while the thumb, pinky, and ring fingers are fully bent down. The only thing that distinguishes these two letters in American Sign Language is that the letter "U" has the index and middle fingers separated (like making a "two" or "peace" sign).

The letter "V" on the other hand, is essentially the same gesture, but with the two fingers touching each other. It can clearly be seen that the amount of bend

present on the fingers as a result of these two similar hand gestures is identical. The only way for us to be able to distinguish the two is to insert some type of hardware that can determine if the two fingers are separated or not. The simplest way of determining this would be to simply apply some metal contacts on the side of the fingers. These contacts would touch each other whenever the letter "V" was being performed, and through our software solutions the group can identify the gesture as a "V". From the software perspective, the group can simply add a condition in the conditional statement in the main loop that determines if the letter being performed is a "V", or an "R" and the statement would check to see if the two contacts are touching (they would complete a closed circuit, of which one end would be provided as an input to the microcontroller).

Another potential solution for this problem of determining between the "U" and the "V" would be to use a pressure sensor. Essentially the group would be getting the same information from the glove (are the two fingers separated or not) without them having to be in flush, solid contact. The contacts would work fine, but only under the condition that the two fingers are in perfect contact with each other. This can be accommodated by the user, who would make sure to consciously try to bring the fingers together fairly strongly. However, a pressure sensor would eliminate the need for the two fingers to be actually touching. The output of the pressure sensor would tell us the position of the two fingers relative to each other.

This dilemma between the "U" and the "V" also extends to the letter "R". The "R" also consists of having the index and middle fingers being fully straight, while the rest of the fingers are fully bent down. In the case of the "R", the index and middle fingers are also touching, as in the letter "V". However, this time instead of the two fingers being touching side-by-side like in the "V", the index finger goes in front of the middle finger. This provides us with a very similar dilemma that was described above. The group can also use contacts to determine if the fingers are this position. But, again, the fingers would have to be in that exact, precise position in order for the system to identify that the user is trying to perform an "R". In addition, the differences between the shapes and sizes of the hands of different people would provide a challenge as to choosing the ideal location on the glove to place the contacts. This can also potentially be made up for by using the pressure sensors instead of going with the contacts. Either way, the group need to accurately make sure that the group can depict whether the user is trying to perform a "U", a "V", or an "R", while getting the same exact readings from the flex sensors.

6.2.5.5 Networking With User Interface

The control system and user interface parts of the software for the SLIG will be designed and produced independently from each other. The control system has the job of reading in the sensor data from the flex sensors, accelerometer, and

pressure sensors/contacts. The user interface has the job of reading whatever information is being sent to it from the control system, and displaying this information on the screen of the user's device (Android phone) in a way that is both visually appealing and easy to operate. There needs to be a bridge to get the information from the control system to the user interface, and this bridge needs to be robust so that changes in one system or the other does not negatively affect the other system. To communicate between the control system and the user interface, the group will be using Bluetooth communications. The information will leave the control system as an output from the microcontroller unit.

The microcontroller will send the information out to the Bluetooth module through the use of one of its UART pins (which will be described in further detail below). This information will go through the Bluetooth module and through the ether into the device where the user interface will be running. At the receiving end (the user interface), the device will need to be paired with the Bluetooth module on the glove, and establish a strong connection. Through the strong Bluetooth connection, the group would send the data to the user interface from the control system. This data would be sent through the use of hexadecimal characters to depict the letter being represented by the hand gesture. The user interface will be designed so that it can receive the data that is being sent to it from the control system through the Bluetooth module.

6.2.5.6 UART (Universal Asynchronous Receiver/Transmitter)

Once the microcontroller has made a decision on what letter the user was trying to perform with a given hand gesture, that data will be transmitted out through a Bluetooth module (see "Networking with User Interface" section above). However, this information will need to be sent through some type of receiving and transmitting system, such as RS-232 (serial). This is typically done using some hardware like a Universal Asynchronous Receiver/Transmitter (UART). This device is typically embedded in the microcontroller unit and is used to send and receive serial data to and from external sources. The UART takes a large chunk of data (such as a byte or word) and sends (or receives) it one bit at a time, at a speed dependent on the clock speed of the microcontroller. In our case the group would be sending data out using the UART.

As the group has experienced in the course "Embedded Systems", the group will have to write a function that initialized the UART for our microcontroller, and the group will be using the UART pins as the output that send out the data that is being sent out by the microcontroller to the Bluetooth module. The UART uses a series of shift registers and other internal components to properly time the data and be able to accurately send (or receive) the data. In the process of choosing a microcontroller unit, the group would have to ensure that the group chooses one that employs the use of a UART or another type of transmitting and receiving

hardware. From the research that the group have done, it seems as if the familiar MSP-430 is actually a microcontroller that can potentially serve the purpose of our project.

This microcontroller has a UART embedded in it, has the analog inputs that the group would need, and seems to have plenty computing power to perform the work that the group need from the microcontroller. Another microcontroller that can potentially also be a fit for out project is one from the ATMEGA series. These microcontroller units are typically used in many Arduino applications and can also be a good fit for our project. There are many advantages and some disadvantages to both, but ultimately the MSP-430 seems like it might be the best fit because the group are more familiar with it, it is readily available, it comes with its own programming environment, and it is the more powerful of the two.

6.2.6 Power Source

6.2.6.1 Battery

The group decided on using a polymer li-ion battery to power the SLIG. The reason for using this type of battery is because polymer li-ion batteries are one of the thinnest batteries available in the market. Polymer cells are much thinner than prismatic cells which means they are much lighter. At the same time, polymer li-ion batteries are able to store more energy than nickel-based batteries. Polymer li-ion batteries also retain their charge for longer.

The group decided to purchase the 3.7 volts polymer lithium ion battery – 2000 mAh from SparkFun. The batteries have a 5 start rating and are said to be super slim, which is why the battery are so light weight. The battery includes a build in protection circuit for minimum voltage, over voltage and over current. Rated at 2C continuous discharged, this polymer lithium ion battery has what it takes to power the Sign Language Interpreter Glove.

The battery model is a 585460 with cell dimension of 5.8 by 54 by 60 mm^3.
The following are cautions that need to be taken when charging or discharging the battery.

- The battery charging current must be $1C_5A$ or lower.
- The battery charging voltage should be 4.25 V or lower.
- When discharging the battery, the discharging current should be $2C_5A$ or lower.
- The discharging voltages has to be 2.75 V or greater.

Below are some of the battery specifications:

Table 6.1: Polymer lithium ion battery specification

Item	Specifications	Remarks
Nominal Capacity	2000 mAh	0.2 A discharge
Charge Current	Standard 0.2 C_5A Max 1 C_5A	Working temp. 0° to 40° C
Charge cut-off Voltage	4.2±0.03 V	N/A
Standard Discharge Current	0.2 C_5A	Working temp. -20° to 60° C
Max Discharge Current	2.0 C_5A	Working temp. 0° to 60° C
Discharge cut-off Voltage	2.75 V	N/A
Cell Voltage	3.7 – 3.9 V	When leaves factory
Weight	Approximate: 37 grams	N/A
Storage humidity	65±20% RH	N/A

6.2.6.2 Charging

The group decided to use the Tenergy TLP-4000 Universal 1A Smart Charger. This is a great charger for our 3.7 volts polymer lithium ion battery. The charger offers constant current and constant voltage charging like our battery requires. The charger will be able to charge our polymer li-ion battery at a capacity of 1000 mAh. This means it should be able to charge our battery in just 120 minutes.

The charger comes with an MCU control and to ensure no over-charging, the charger also includes an accurate voltage detection system. Nevertheless, the charger also offers reverse polarity protection and short circuit protection. With integrated LED that turns red to indicate when the battery is charging and green when the battery is done charging. The charger was design to for worldwide usage with an alternating current input voltage of 100 to 240 volts.

There are some precautions the group needs to take when using the Tenergy TLP-4000 battery charger. Tenergy warns users to only charge 3.7-14.8 V LIPO Li-ion batteries that are rechargeable only. Tenergy also advises the users to not expose the charger to rain or humidity.

Below are some of the specifications of the Tenergy TLP-4000 li-ion charger.

Table 6.2: Tenergy TLP-4000 battery charger specification

Input	AC 100-240V 50-60Hz 1.0A
Output	DC 4.2-16.8V Auto-detection
Charging current	1A
Dimension	199*W*61 H38 mm
Weight	284g

Figure 6.14: Tenergy TLP-4000 battery charger

6.2.7 Bluetooth Module

The group decided to go with the HM-10 4.0 Bluetooth low energy module from the JNHuaMao Technology Company. The reason the group decided to go with the HM-10 it's because it uses the TI chip CC2541. The HM-10 is affordable and reliable based on customer's reviews. In this section the group will discuss some of the parameters and the specifications of the HM-10 BLE module. Below are some of the important parameters provided by the HM-10 datasheet.

- Working frequency: 2.4GHz ISM band
- RF Power: -23dbm, -6dbm, 0dbm, 6dbm, can modify through AT command AT+POWE.
- Power: +3.3VDC 50mA
- Power: In sleep mode 60uA~1.5mA, Active mode 8.5mA.
- Working temperature:–5 ~ +65 Centigrade
- Size: HM- 10 26.9mm x 13mm x 2.2 mm; HM-11 18*13.5*2.2mm

Below is the HM-10 schematic also provided by the datasheet.

Figure 6.15: HM-10 Schematic

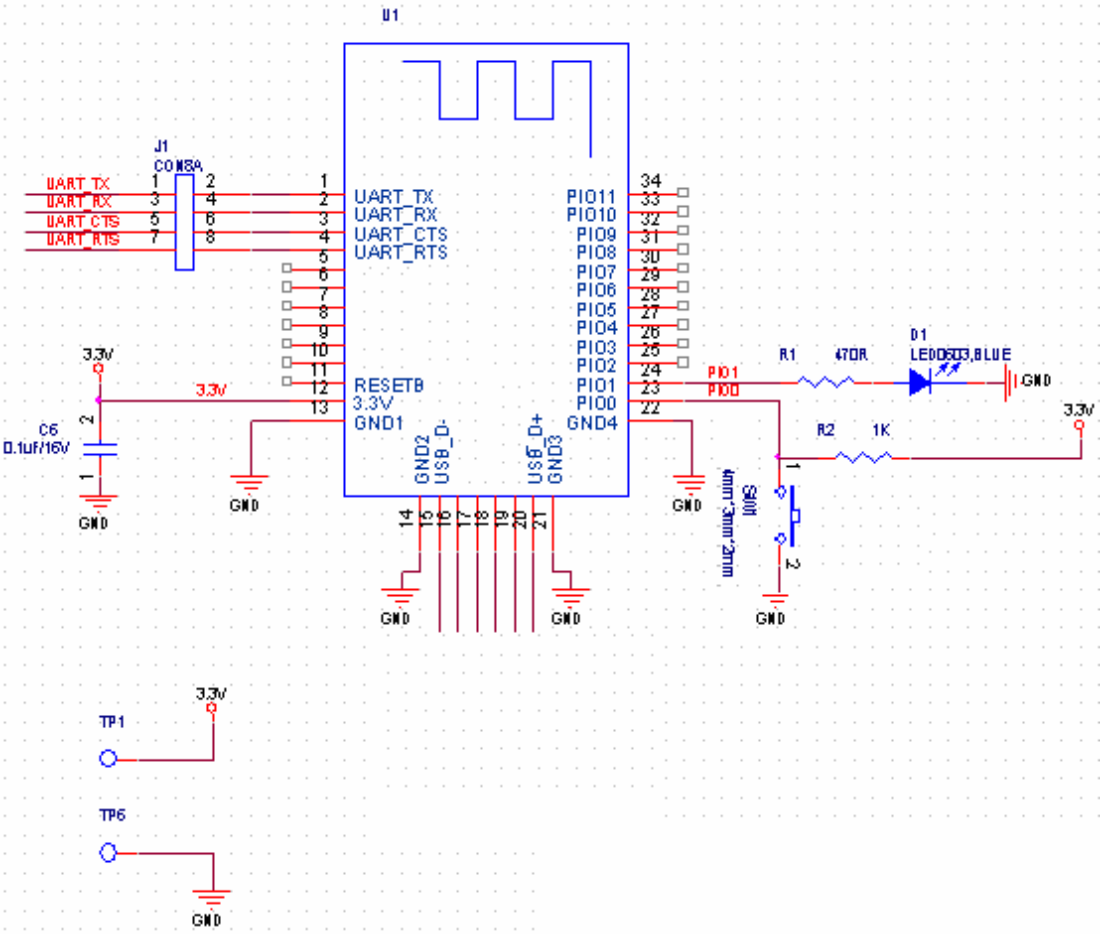


Figure 6.15 was reprinted from the HM-10 datasheet

Figure 6.16: TI CC2541 Block Diagram

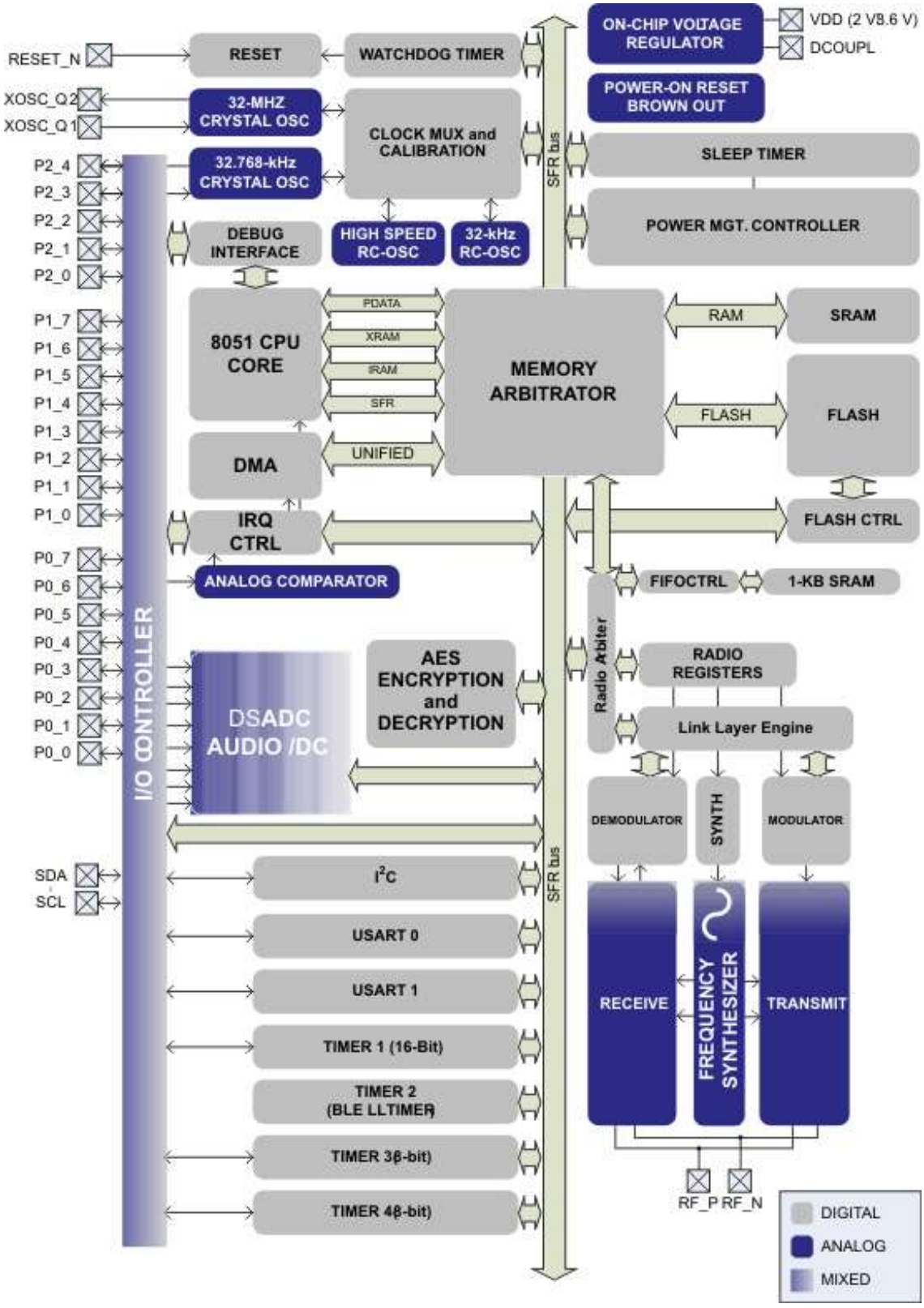


Figure 6.16 was reprinted with permission from Texas Instruments

6.2.7.1 Bluetooth Generic Access Profile

Generic Access Profile, also known as GAP, is what makes a Bluetooth device visible to other devices. Basically the Generic Access Profile controls the connection and advertising in Bluetooth Low Energy. GAP is also responsible for deciding how the devices will interact with each other. Just like the Generic Attribute Profile, the two main roles in GAP are the central devices and the peripheral devices.

Scan response payload and advertising data payload are the two ways that the Generic Access Profile can send advertising. Both the scan response payload and the advertising data payload are similar with the exception that the advertising data payload is obligatory. The reason why the advertising data payload is mandatory is because it should always be transmitting so that the central device is able to recognize the peripheral device. The scan response payload is optional and is used by the device designers to be able to fit more information in the advertising payload. Basically a string for the name of a device would go in the scan response payload. Unlike the Generic Attribute Profile, in the Generic Access Profile a peripheral device is able to connect to more than one central device. Broadcasting in Bluetooth Low Energy is when a peripheral device sends data to other central devices within range. Because the data sent and received can only be seen by two connected devices, this type of configuration can only be possible by using the advertising packet.

Figure 6.17: Bluetooth Advertising Profile

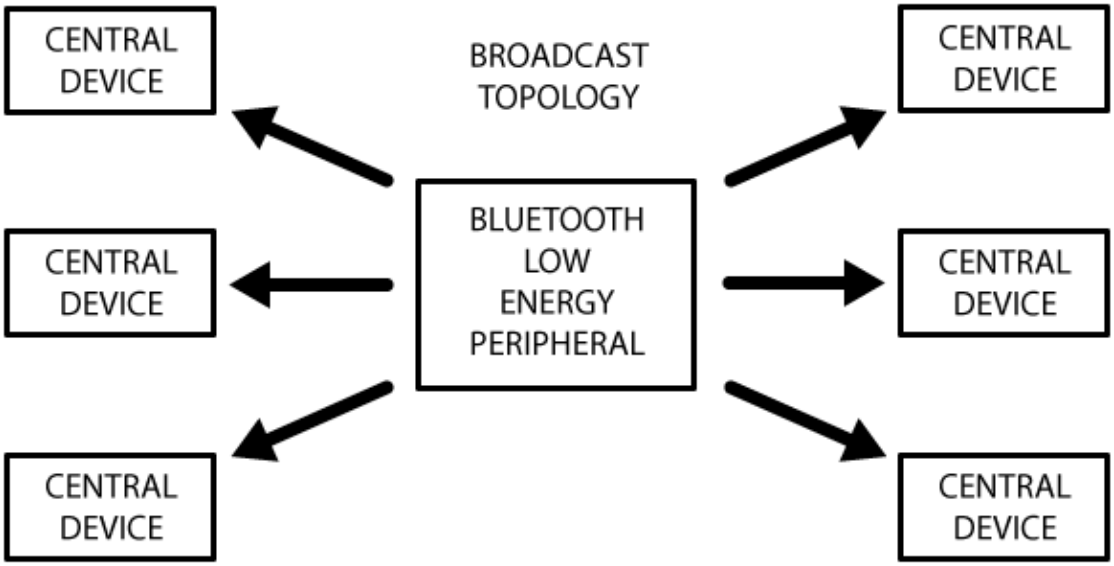


Figure 6.17 was reprinted from adafruit.com

6.2.7.2 Bluetooth Generic Attribute Profile

As previously implied, Bluetooth paring requires at least to devices. One device will be transmitting the data while the other device will be receiving the data. In order to understand how and why the devices need to play a different role in Bluetooth communication, this section will talk about the Generic Attribute Profile also known as GATT. The Generic Attribute Profile explains how Bluetooth Low Energy or Bluetooth Smart devices are able to send data back and forth.

Generic Attribute Profile explains that there is a central device and a peripheral device. Examples of central devices could be a phone, computer, or a tablet. An example of a peripheral device could be a Bluetooth speaker or Bluetooth headphones. In other words, a central device has more processing power and memory and the peripheral device are low power and resource constrained devices. In GATT the difference between a central device and a peripheral device is that a central device can be connected to multiple peripherals devices, while peripheral devices can only be connected to one central device. Once a central device and a peripheral device are successfully paired, the communication between these devices can take place in either direction. A Bluetooth “mailbox” system is needed when data needs to be exchanged between two peripheral devices, this means that all the data needs to pass through the central device.

The Generic Attribute Profile also explains who the peripheral device is known as the “GATT Server” while the central device is known as the “GATT Client”. The GATT Client sends requests to the GATT Server and the GATT Server looks up data and sends back a response. In other words the GATT Client is the master device while the GATT server is the slave device.

Usually the central device is the one that initiates the request to connect to the peripheral device. To avoid causing interference with other devices, the central device will limit its radio transmission when scanning for connection with a peripheral device. Another important feature used to limit the radio transmission of a central device is using short intervals to accomplish a faster detection and connection. The tradeoff of using short intervals is a much greater power consumption.

Slave latency is sometimes limited in order to lower the power consumption on the GATT Server. The slave latency is what defines how many times the GATT server can ignore a consecutive connection. Ignoring a consecutive connection means that the peripheral device is not sending data back to the central device, so it can stay in sleep mode for a longer period of time.

6.2.7.3 Allowing a Bluetooth Connection

In order for the Sign Language Interpreter Glove and the external device to communicate, permission must be granted by both devices. Requesting a connection, transferring data or even accepting a connection requires permission.

When the Sign Language Interpreter Glove is turned on, an established device is needed to connect. The microcontroller is responsible for sending the Bluetooth transceiver the command for it to find an android device to pair up with. The peripheral device, which is the Sign Language Interpreter Glove will send out a signal known as “advertisement”. This advertising signal is needed so that the central device, in our case the android phone, knows that the Sign Language Interpreter Glove would like to establish a wireless connection. Once a connection has been made, the peripheral devices sends another signal to the central device letting it know about the successful connection. The Sign Language Interpreter Glove will implement a single device configuration. Single device configuration consumes less power than the network processor configuration. Single device configuration is the most common and is easier to use. The Sign Language Interpreter Glove should be smart enough to stop scanning once it has been pair up with a device. When scanning for an external device, the Sign Language Interpreter Glove should have a time limit to avoid the battery to drain. Also, if the Sign Language Interpreter Glove does not have any data to send, it should skip the number of connection acknowledgements and go into sleep mode. This feature will help extend the battery life of the SLIG design.

Figure 6.18: Bluetooth Low Energy connection procedure

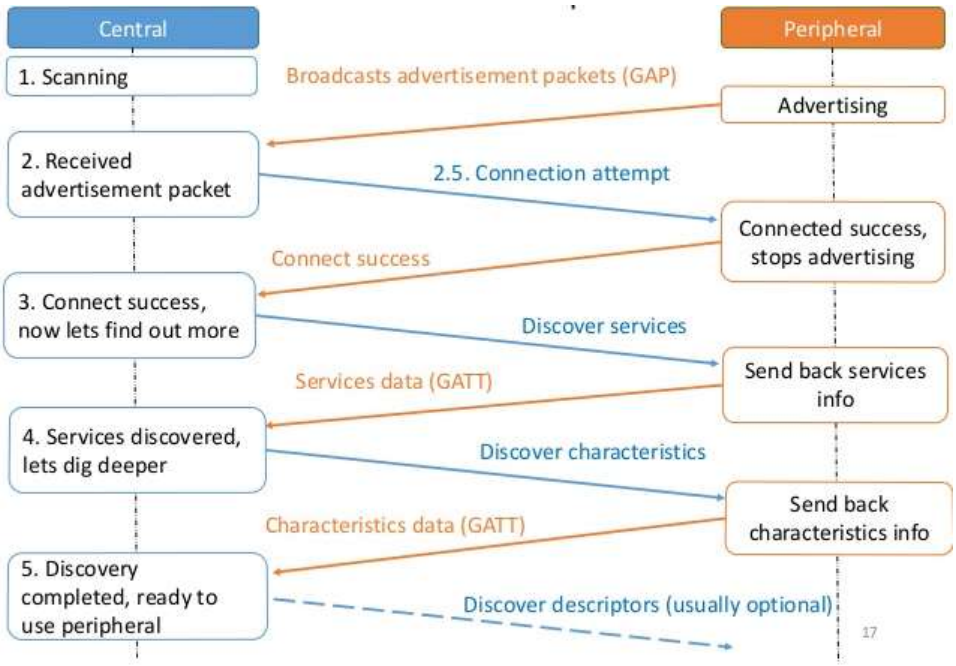


Figure 6.18 was reprinted with permission from the author Mr. Meng

6.2.8 TI LP2985 Regulator

The group decided to use regulators from the TI LP2985 family. These regulators are not only affordable, but also provide excellent performance for all types of applications. The LP2985 regulators are available in many different variations, ranging from 1.8V – 10V. The LP2985 comes in two different version. Version A offers an amazing output tolerance of just 1% and the standard version provides 1.5% output tolerance. Nevertheless, both versions deliver 150-mA of continuous load current.

The LP2985 regulators feature ultra-low dropout, ranging from 280 mV at a full load of 150 mA and 7 mV at 1 mA. The LP2985 can accept a maximum input voltage of 16 V. The LP2985 offers low I_Q, 850 µA at a full load of 150 mA. The shutdown current is 0.01 µA and thanks to a 10-nF bypass capacitor it provides a low noise of just 30 μV_{RMS}. The LP2985 includes overcurrent and over-temperature protection. LP2985 regulators also feature high peak current capabilities. Overall the LP2985 regulators will provide great voltage regulation for the Sign Language Interpreter Glove.

Figure 6.19: LP2985 Functional Block Diagram

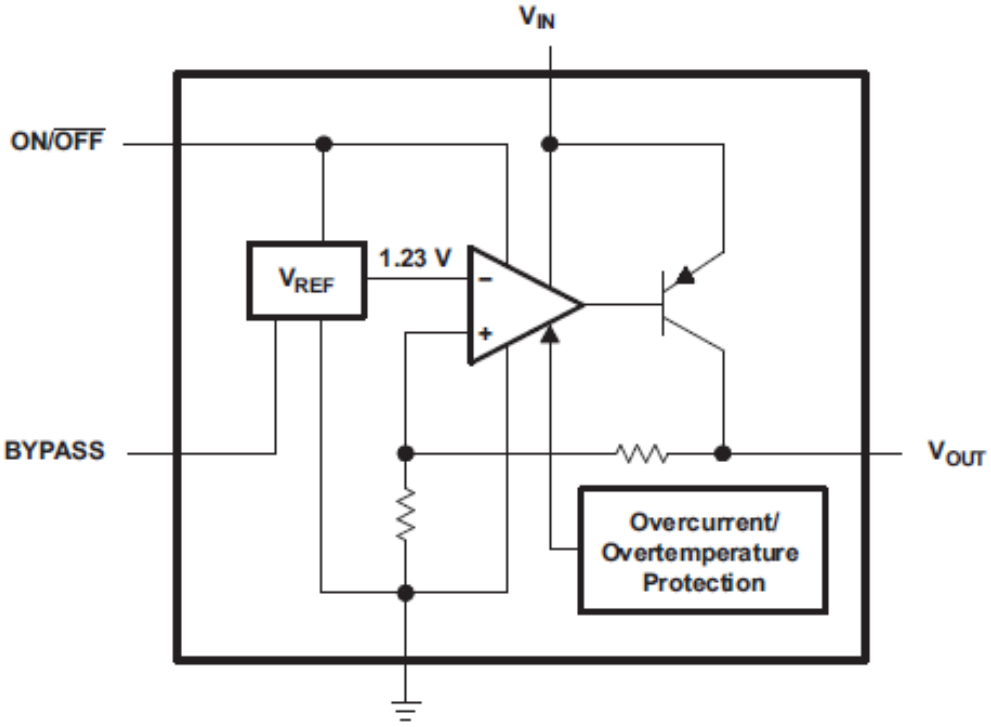


Figure 6.19 was reprinted with permission from Texas Instruments

6.2 Software Design

6.2.1 Control Systems

6.2.1.1 On-Board vs External Processing

One of the biggest decisions that need to be made in regard to the control system of the SLIG is whether the group will process the data using the microcontroller unit on the glove or if the group will send the data to an external processor with more computing power to process the data. This depends highly on which approach the group are going to take to interpret the sensor data. If the group choose to go with “Machine Learning” (see “Machine Learning” section), it is highly likely that the group would decide to send the data out to be interpreted by another computer. This is because efficiently implementing a machine learning algorithm requires quite a bit of computing power and especially memory. The group can recall from the “Machine Learning” section that a machine learning algorithm essentially stores a large amount of previous iterations of each ‘input’ and uses all of that information to make a decision and produce an output. During the training process, thousands of iterations of each hand gesture would be performed, and then the control system would be expected to very rapidly interpret the input, “scan” through all of the data stored during the training and testing process, and make a decision on the output. This would be quite a challenging task for a small microcontroller unit.

If the group decides to send the data to an external computing device, the group would still be making great use of the microcontroller unit. The MCU can be used to receive the data coming in from the sensors, and interpret what the values are. The group can then use the microcontroller to essentially organize that data into ‘packets’ that would be sent out to the external device. Each packet would contain the data received from one gesture. In our case the group can use Bluetooth or a hard-wired connection to send this data. For the external computer, the group can undoubtedly use a desktop computer. However, the group would try to use an Android device. In that case, the group would essentially use the same application that the group are using for the user interface (See “User Interface” section) and “piggyback” some back-end programming onto that application. At that point the group can implement the machine learning algorithm on the android device and interpret the hand gestures, and display it on the same application.

Perhaps a more straightforward way to interpret the data being received from the sensors would be to perform all of the programming on-board of the glove. If the group were to go this way, the group would probably shy away from the machine learning algorithm, and explore some other options to interpret the data. This would require significantly less memory and computing power because in this

situation, the group would likely have a static set of instruction which run indefinitely on a loop and can interpret the data.

The group believe that it would be preferable if the group can do all of our processing on the actual glove. If the group can help it, this is our preference. Firstly, in this way, the group can have our two software components (control system and user interface) working independently of each other. This would mean that if the control system is working, the group know that the data that the user interface is receiving is correct. Conversely, if the user interface is having problems displaying the correct character, then debugging can be a little bit simpler.

The group would be able to debug the two systems independently and find the error. However, if the group were to use the Android device to perform the computing of the data, it would be 'blended' together with the user interface program. In this scenario, if the group were to have an issue with the system, it would be a bit more complicated to debug. Also, processing the data on the glove provides us with the opportunity to have a stand-alone glove which can process data from the user itself. This is the ideal situation because anybody who owns the glove and would like to use another medium to view the letter can do so.

6.2.1.2 Functions

The control system for the SLIG will employ a few functions, which the group will write ourselves. This function will be used to 'map' the range of values retrieved from the user in the calibration phase. As described in the "Calibration" section above, the calibration process will involve the user closing and opening their hand, as to provide the system with information about how much bend will be present on that person's hand. The function will be provided with parameters for the minimum and maximum amount of bend that was measured from the user during the calibration process. The function will then take this data and it will "normalize" it through the use of some arithmetic, to make it represent the 'standard' maximum and minimum values. These standard values will have to be determined by us as the group go along with the implementation of the project, once the group see what are typical values that the group get from the sensors and how the system behaves when the group try to implement the hand-gesture recognition (without calibration). Once the group have an idea of how the system works without the use of calibration, the group can see the amount of error that the group experience due to the differences in the users' hands and the group can pick a range of values that can mitigate this error.

Another function that would have to be written by us would be the 'mapping' function that takes in the minimum and maximum values (after being normalized by the calibration process) and breaks up that range of values into a certain amount of "levels". In our preliminary planning, the group believe that about 4

different ‘levels’ of bend can be appropriate and enough to determine what position each finger is currently placed on. For example, after going through the calibration function and through this mapping function, a gesture performed by the user will produce an output such as “1, 2, 3, or 4”.

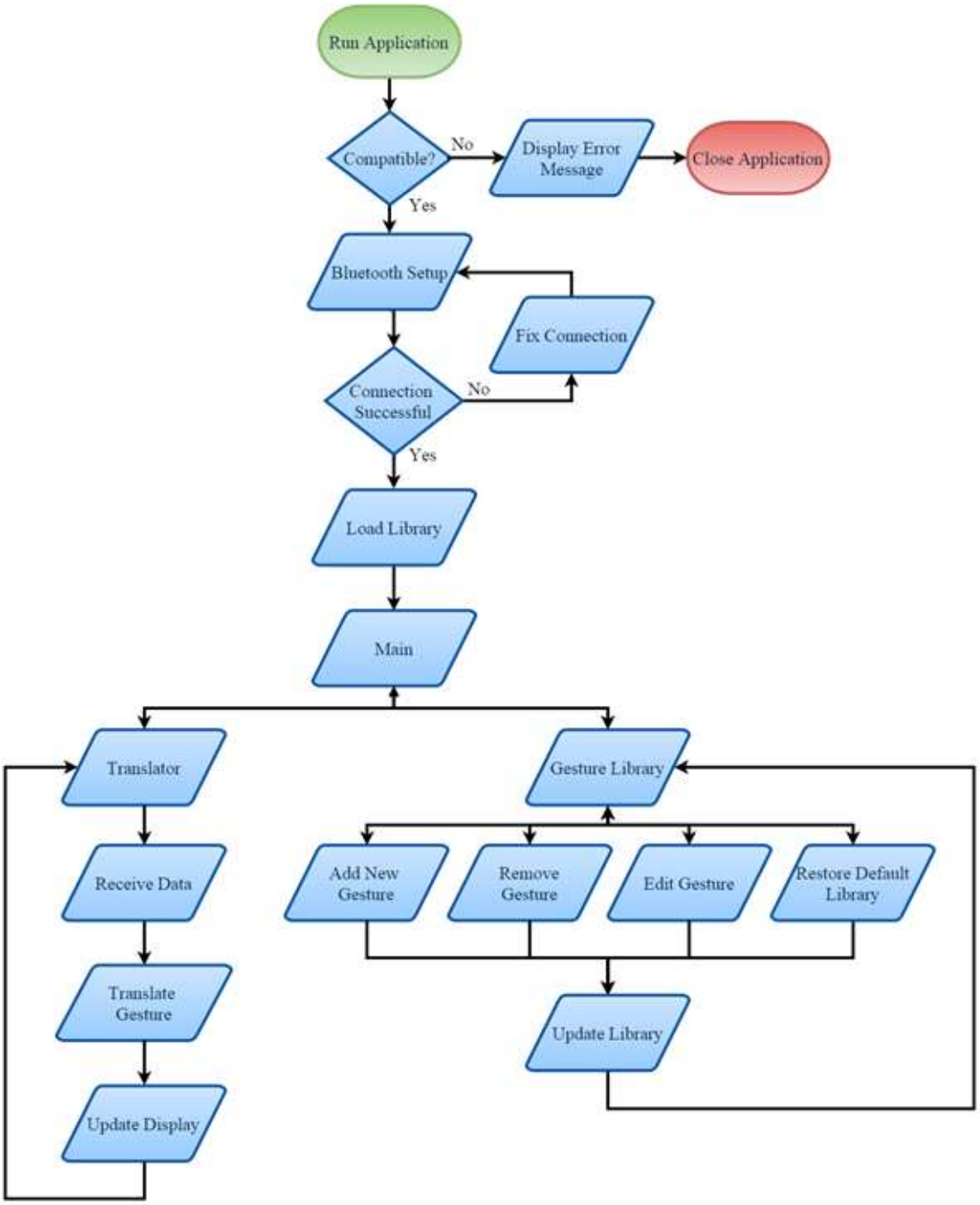
Even through the range in voltages being read from the sensors varies greatly within each one of these ‘levels’, each of the four levels are distinguishable enough so that a the system can easily tell if the finger is fully straight, slightly curved, halfway bent, or fully bent. This is all that is necessary for the determination of what hand gesture the user is trying to perform. If this can be accomplished successfully and accurately, then actually having the infinite loop make decisions on what hand gesture is being performed by the user becomes exponentially less difficult. At this point the infinite loop can be populated with a series of conditional statements that inquire about the state of each sensor, only using 1 of 4 possible outputs from the sensor. If the group did a good job of providing an accurate depiction of each position of bend as an integer between 1 and 4, then this part should go by fairly smoothly.

6.2.2 Mobile Application

A successful mobile application is essential to the project because it is the final piece of the puzzle that displays the output of the glove to the user. Learning how to create a mobile application is very complex and there are many things that a beginning developer will learn along the way such as how to approach the design from the start. Therefore, it is a good idea to create a preliminary mobile application before diving into the actual mobile app used for the project because of the steep learning curve that will cause errors and redesign in the long run. However, the preliminary mobile application is for learning purposes only and will not be discussed in this paper.

Now that the group has come to a consensus on using Android as the mobile platform, the design of the mobile application needs to be completed. The best approach for designing the mobile application will be to keep it simple and focus on the main objective, which is displaying the letters translated by the sign language glove. The first step will be to go through the process of what we want our mobile application to do and how the sign language glove uses it. Below is a flowchart that shows the mobile application’s processes and expected behavior.

Figure 6.20: Mobile Application Flowchart



When the user is performing sign language gestures with the Sign Language Interpreter Glove (SLIG), the flex sensors and accelerometer are sending data to the microcontroller. This data is then translated into a letter of the alphabet and sent wirelessly over Bluetooth to the mobile phone which will search the gesture library for the corresponding letter and display it in real time onto the screen as

text. In this process, the mobile application is responsible for receiving data via Bluetooth, searching an internal gesture library, and displaying text onto the screen and constantly updating the value. The main design features of the mobile application are the graphical user interface, Bluetooth communication, and gesture library which will all be discussed in detail in the following sections.

6.2.2.1 Bluetooth Communication

The research, design, and connection of the hardware for the Bluetooth module have already been discussed in other sections but the design of how the mobile application software will interact with the Bluetooth module needs to be covered. Establishing Bluetooth communication between the glove and the mobile application requires using the programming language Java in Android Studio, but this will be explained in more detail later. The first requirement for our mobile application to communicate with the Bluetooth module is to ensure that they are both compatible. After this has been established, the Bluetooth module on the glove must connect with the mobile device and maintain a stable connection so that the mobile application is ready to be used. Making a connection with the Bluetooth adapter on the glove requires calling methods in Java that use the Generic Attribute Profile (GATT) – explained in section 6.2.7.2 Bluetooth Generic Attribute Profile. Once the mobile device has made a connection with the glove, the mobile application is still not allowed to use this connection for sending or receiving information; the application must be given permission to use the Bluetooth connection within the source code to ensure security. The next couple sections will explain which devices are compatible with the Bluetooth module selected and how to give permission to the mobile application so that it can use this Bluetooth connection for its purpose.

6.2.2.1.1 Mobile Phone Compatibility

The Bluetooth module selected for the Sign Language Interpreter Glove (SLIG) will be using Bluetooth Low Energy (BLE), which was adopted into the main Bluetooth Standard in 2010 along with the adoption of the Bluetooth Core Specification Version 4.0. In order for the mobile phone to be compatible with BLE the version of Android on the device must be Android 4.3 or newer; Bluetooth Low Energy is not backwards compatible unfortunately. Most phones on the market today are compatible with BLE but there could be other software compatibility issues if the Android API level used for the application is newer than the API used by the device. The API level determines which versions of Android that an application is compatible with and can be used as a tradeoff between having more features or more compatible devices.

The list of requirements for Android mobile phones to use Bluetooth Low Energy is shown below in Table 6.2.

Table 6.2: Minimum Software Requirements for Bluetooth Low Energy

Minimum Requirements	
Device Type	Android
Mobile Platform	Smartphone
Bluetooth Version	Bluetooth Low Energy v4.0
Platform Version	Android 4.3
Codename	Jelly Bean-MR2
Android API Level	18

6.2.2.1.2 Finding BLE Devices

When the glove is ready to be paired with the mobile phone via Bluetooth, both the Bluetooth adapter and the mobile phone have to begin searching for each other. This section will be discussing how the mobile application will search for the Bluetooth Low Energy devices and pair with them. Using a method in Java called startLeScan(), the mobile application will scan for a peripheral device – the sign language glove – and return any devices using the supported GATT services. After the application has recognized the glove as a compatible BLE device, it needs to connect the mobile device to the GATT server hosted by the BLE device. This connection can be made by using the Java method connectGatt() that handles connecting the BLE device as soon as it becomes available. The connectGatt() method also passes back an object that contains information about the connection status and other GATT client operations that can be used to perform more GATT client operations. When the Android application has successfully connected to the glove it can begin to receive data wirelessly using the Bluetooth adapter.

6.3.2.1.3 Bluetooth Permissions

Android mobile applications do not automatically have permission to use Bluetooth features; the developer must declare the Bluetooth permission BLUETOOTH in the application manifest file. The mobile application needs this permission in order to use Bluetooth communication for actions such as requesting a connection, accepting a connection, and transferring data. There is another Bluetooth permission used for initiating device discovery, pairing devices, and managing Bluetooth settings called BLUETOOTH_ADMIN. An example of how both of these permissions would be declared in the application manifest file is shown on the top of the next page.

```
<uses-permission android:name="android.permission.BLUETOOTH"/>
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN"/>
```

6.2.2.1.4 Enabling Bluetooth Features

Once the mobile application has enabled the Bluetooth permissions, the application must perform a compatibility check for Bluetooth Low Energy (BLE). This can be done by adding the following code into your application manifest file, the same place that the Bluetooth permissions were added:

```
<uses-feature android:name="android.hardware.bluetooth_le"
android:required="true"/>
```

The application must also verify that BLE is enabled on the device if it is supported. There is a class called “BluetoothAdapter” that can ask the user for permission to enable BLE and it won’t leave the application. There are two steps required in this process, getting the “BluetoothAdapter” and enabling Bluetooth. Getting the “BluetoothAdapter” means that there needs to be a call made in the application that finds the adapter by using other Bluetooth specific classes. An object is made when the Bluetooth adapter is found, which is an instance of the class “BluetoothAdapter”, and can now be used to perform any Bluetooth actions. But first, the application needs to confirm that the Bluetooth is enabled by calling the method “isEnabled()”, which is provided in the Android SDK.

6.2.2.2 Graphical User Interface

The graphical user interface (GUI) is a type of interface for applications and programs that uses visual features to control the application/program instead of text-based instructions. The main purpose of a GUI is to offer users an interactive and user-friendly navigation system for an application that gives the user control of the available features. Some examples of a GUI are Windows and Mac OS for computers which both have a visual interface with icons, buttons, and much more that are used to navigate and perform tasks. The main design considerations when it comes to the graphical user interface are the menu layout schemes, appearance, and the usability of the application. The menu layout will be one of the first impressions of the application where users will have a choice to receive gesture inputs from the Sign Language Interpreter Glove, manage the gesture library, and more so it is important to make the menus easy to use and look good.

When it comes to designing the appearance of the GUI, it is critical to make the application proportionally fit the screen size of the phone or else everything else

will appear distorted. For example, an application UI that was designed for a mobile phone would not fit onto a tablet screen without being fit for a bigger screen size and vice versa. Having to adapt the screen size to every Android mobile device on the market would be a hassle because each cell phone carrier has their own line of smart phones and each smart phone has multiple generations of models with varying screen sizes and resolutions. Thankfully, Android provides APIs that support multiple screen sizes and will simplify this process by splitting the range of screen sizes and densities into four groups: small, normal, large, and extra-large. The range of these screen sizes are not exact, but to get a better idea of what they look like view Figure 6.21a which displays an approximate range of screen sizes that these four groups are used for.

These groups are based on the screen size of the device, measured as the screen's diagonal, and the screen density, the number of pixels within a given physical area (measured in dpi, or dots per inch). However, when defining a UI layout, there is another unit that is used instead of the size and screen density called a density-independent pixel (dp or dip). A "dp" is a physical unit of measurement that represents a virtual pixel unit used for expressing UI layout dimensions or positioning independent of the screen density. The main purpose of a "dp" is to allow the mobile application software to work in pixels as a unit of measurement and then convert from a virtual pixel to a physical pixel using different scaling factors based on the device's screen density. There are six generalized screen densities shown below in Figure 6.21 ranging from low (120 dpi) to extra-extra-extra-high (640 dpi), but the standard measurement for many calculations is 160 dpi corresponding to a medium density screen.

Figure 6.21: Illustration of how Android™ roughly maps actual sizes and densities to generalized sizes and densities (figures are not exact).

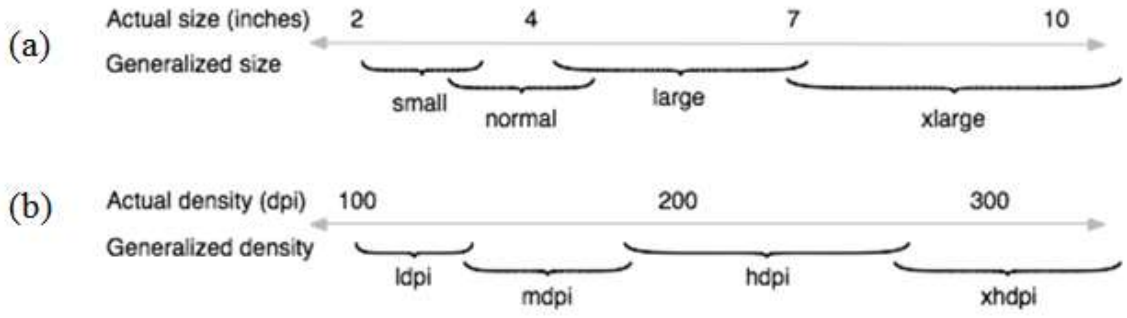


Figure 6.21 was reprinted with licensing permission from Creative Commons Attribution 2.5

The calculations required to render the UI layout for each device is executed by the system in the background, so there is no need to manually calculate the resolution. Using these four predefined layout sizes, the mobile application

should be able to automatically detect the screen size of the device and select the appropriate UI resolution. For most cases, Android will resize the application UI to fit the device screen fittingly but there are a couple other measures that should be taken to guarantee a proper screen configuration. The first measure is to explicitly declare the screen sizes that your application supports in the manifest file – a file required by Android applications that contains essential information needed by the device in order to run – which guarantees that only devices with supported screen sizes may download the application. For more specific UI layouts, configuration qualifiers can be used to adjust the size of the screen and the elements within it as well as other features such as changing the language and direction of the layout. The four screen size groups discussed earlier (small, normal, large, and extra-large) are actually one set of configuration qualifiers but there are many more that can be used to provide alternative resources for the application. One of these qualifiers that will not be used in this application is the landscape orientation, which is an alternative view of the application layout when the screen is held horizontally. The reason landscape orientation is not useful for this application is because the main functionality is to translate gestures into letters and having a landscape orientation will just add more complexity to the UI layout without any added benefits. Using these configuration qualifiers for alternative layouts is a nice addition to many applications, but the scope of the project does not require many of these features and therefore will only use the qualifiers that are necessary.

6.2.2.3 Menu Layout, Interface, and Usability

The mobile application's graphical user interface (GUI) design requires more than just configuring the UI layout to each device; the developer must also design the menu layout which involves creating a menu and the navigation through the application. Creating the menu can be done using Android Studio's built-in GUI workspace for drag and drop features and the programming language XML for more customizable interface design (found in section 3.3.3 Programming Languages). All of the available features of the app are accessed through the menu and can have an infinite number of unique designs thanks to the customization options provided by Android Studio. This section will go through the different menu layout styles and which one will fit the needs of the mobile application best.

The menu layout gives an application a certain look and feeling depending on the different shapes, colors, text, and positioning that are used. When creating a mobile application you don't want the interface to look like a scaled down website; instead, there needs to be a more mobile theme behind the design. There are a few of the features that websites and mobile apps share such as buttons, drop-down menus, scrolling, and more that need to be changed for mobile use. One example is that on websites it is common to use drop-down menus to access layers of information, but on mobile applications users generally don't want a drop-down menu at all because of the limited screen size and

difficulty accessing multi-leveled menus on a phone. The best approach for this situation is to keep the menus simple and only have one or two levels of information. There are many other examples like this one but first it is more important to talk about the different tools available to Android developers for creating a menu layout.

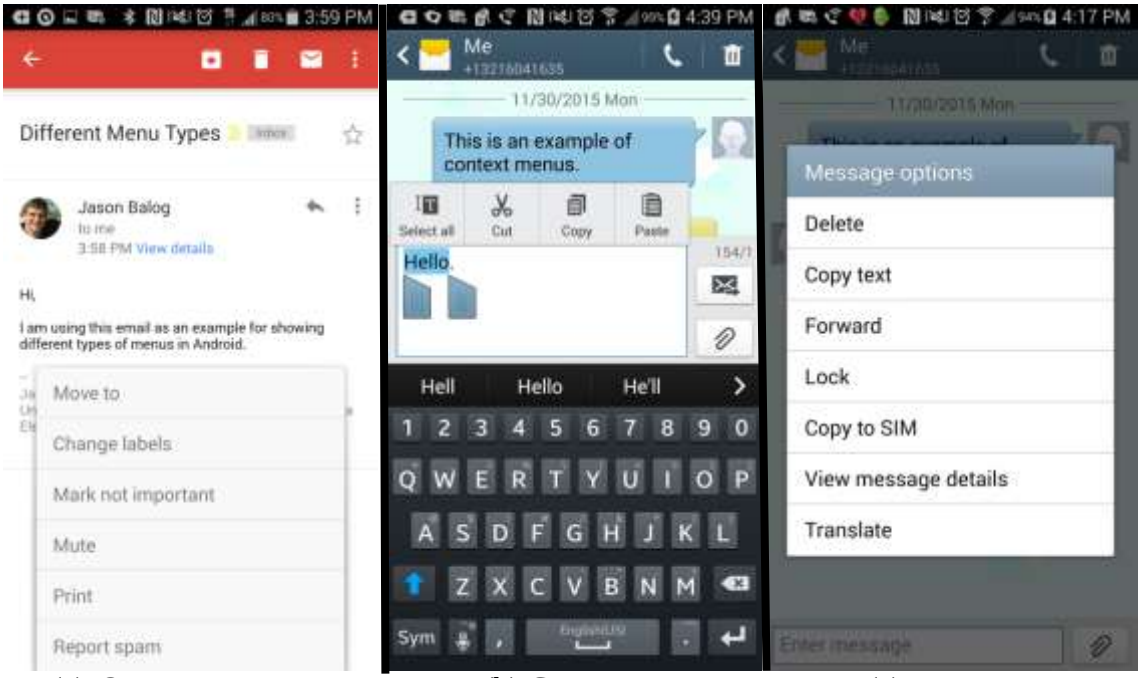
Since menus are common in many mobile applications, Android provides a list of menu APIs to use for actions and other functions to maintain a familiar and comfortable user experience. This means that there are a few different menu layout options available to developers depending on how he/she wants to design the application. The three main types of menus available through Android are an options menu, context menu, and popup menu (or submenus) which have different usage and workflow. An options menu is a very common type of menu which is displayed when a user selects the menu button on his/her phone. This is generally where users can access global items like the settings, search bar, help, and other additional information. An example of an options menu is shown below in Figure 6.22a when the user presses the menu button on an Android phone, which displays the different options available to users viewing an email using Gmail. The second type is the context menu which displays a floating menu with various options when a user long-clicks an element. The reason this type of menu is called a context menu is because the floating menu that appears is related to the context of the element and selecting an option will affect the content or context frame. In Figure 6.22b, a context menu is shown using the Android text messaging application and long-clicking on the selected word that is being typed which allows the user to edit the message. The last type of menu is a popup menu that provides a list of extended options from an element but do not actually change the context of the item selected. An example of a popup menu is the menu displayed when trying to manage a message shown in the thread of and Android messaging application as shown below in Figure 6.22c.

After going over the different menu types available in Android Studio, it is time to design the menu layout of the SLIG's mobile application. As the interface is being constructed, the developer always wants to look at the layout from the user's point of view so that the application is user-friendly. The audience base using the SLIG will be mostly be new people who are testing the glove for the first time, and thus creating a user-friendly environment for new users requires making the application intuitive to learn. This process of designing an application interface based on the ease of navigating through the menus and accessing the features defines the usability of the mobile application and is a big part of the interface design process.

As mentioned earlier, one of the most important aspects of usability for mobile platforms is implementing mobile-friendly features such as simple drop-down menus, buttons, etc. that will complement the user interface and take advantage of the mobile phone's capabilities. Hence, one of the simplest ways to design the application is to have the real-time gesture translator open first and serve as the

‘home’ screen since this is the main purpose of the app. Once the application is open, there needs to be a general menu where the user can switch between the different features like the translator, gesture library, and other additional features. As shown below in Figure 6.22, there are a few different methods for accessing menus depending on how the developer wants their application to look and feel. Out of the three choices, the ‘options menu’ appears to be the best choice for creating a general menu since this type of menu is designed to access global features. The next step is deciding how to access this general menu which can be implemented in many different ways.

Figure 6.22: Different Types of Menus on Android Version 4.4.4



(a) Options Menu (b) Context Menu (c) Popup Menu

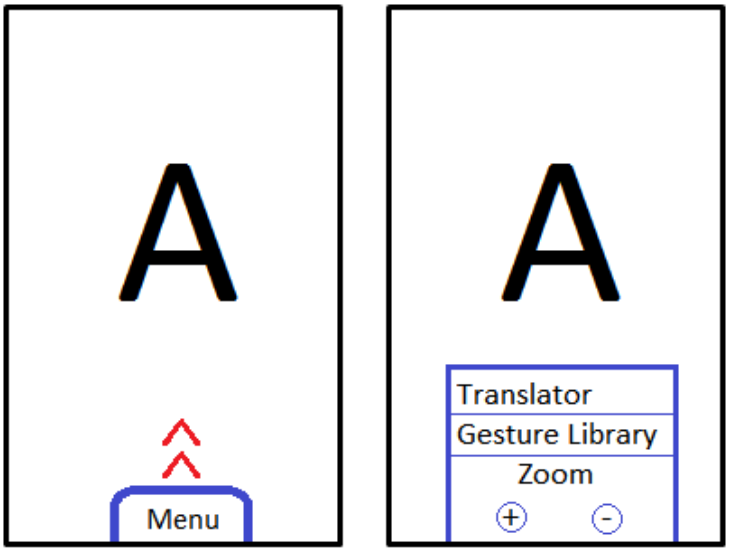
As a developer, opening a menu is a simple task and many applications use the same method for completing this task because they want users to be familiar with their menu navigation before they even use it. Seeing that the SLIG’s mobile application will open to the gesture translator screen first, the main menu will have to be accessed from this screen somehow and needs to be obvious to the user. This must be done using some kind of interaction between the user and the mobile application like touching a button, swiping, voice commands, etc. The simplest and most common methods among mobile applications for opening a menu are pressing a button or swiping to open a new screen/menu. Using one of these options is advantageous simply because most people using these applications already own a mobile phone and they are very familiar with the functionality of buttons and swiping already.

Generally, buttons and swiping can be used for the same purpose but they each have their own pros and cons. Buttons are good for developers and users

because they are quick, intuitive, and visual means of navigation that lets the user know that tapping the button will do something related to what is displayed on the icon (text and symbols). The bad parts about buttons are that having too many can clutter the screen and a misplaced tap can open an unwanted feature during use and cause an issue. The good news is that swiping features don't have these problems and can be used in replacement or in addition to a button. Swiping is a natural movement for users that allows developers to open features by swiping in any direction, however the four basic directions (up, down, left, and right) are the most popular. Some of the biggest benefits of swiping are that it doesn't take up as much screen space as a button and it can't be accidentally opened as easily as a button. The bad parts are that swiping can cause users to accidentally use other features such as scrolling or tapping a button instead of swiping. Also, if there is no visual indication that swiping is an available feature many users will overlook that feature without realizing it exists. However, if the visual icon of a button is combined with a swiping motion this will eliminate one of the issues with swiping and make it a better choice. Using this information, a visualization of what the application should look like is shown on the next page in Figure 6.23 where the menu button is at the bottom and can be swiped upwards to open the menu. As you can see, the menu button is easy to access, does not take up much screen space, and will not accidentally open which makes swiping the best option for the mobile application's menu. Also, the user can zoom in or out to fit the letter to their screen manually and view the changes in the background without closing the menu which is another big plus for usability.

There are other design requirements needed to completely design the menu and user interface, but the most important design features have already been discussed. The rest of these features are negligible in terms of functionality for this project but will be briefly mentioned so there are no misunderstandings about the design of the interface for the application. The most obvious design factor that was left out is making sure that the application is readable. Readability is an important factor in mobile design but as long as the developer uses appropriate font sizes/styles, colors, and contrast for the interface there should be no issues. Also, developers must keep in mind how much screen space their features require so that the application functions as intended. For example, in Figure 6.23, the letter displayed must be large enough for the user to read but not too large where it will overlap with the menu icon at the bottom of the screen. The default screen configuration should take care of this issue but if the user prefers to change the size of the letter they may do so with the zoom feature.

Figure 6.23: Sample of Menu Layout Interface



6.2.2.4 Gesture Library

The gesture library is the feature of the mobile application where all of the recognized sign language gestures are saved. This library is where the translator feature will search for matches when the sign language glove is transmitting gestures over Bluetooth. The user will be able to manage this library by adding, editing, or removing gestures from the gesture library. Also, there will be a "restore default gestures" option in the case of something going wrong with any changes done. A list of the features available in the gesture library is shown below in Table 6.3.

Table 6.3: Gesture Library Features

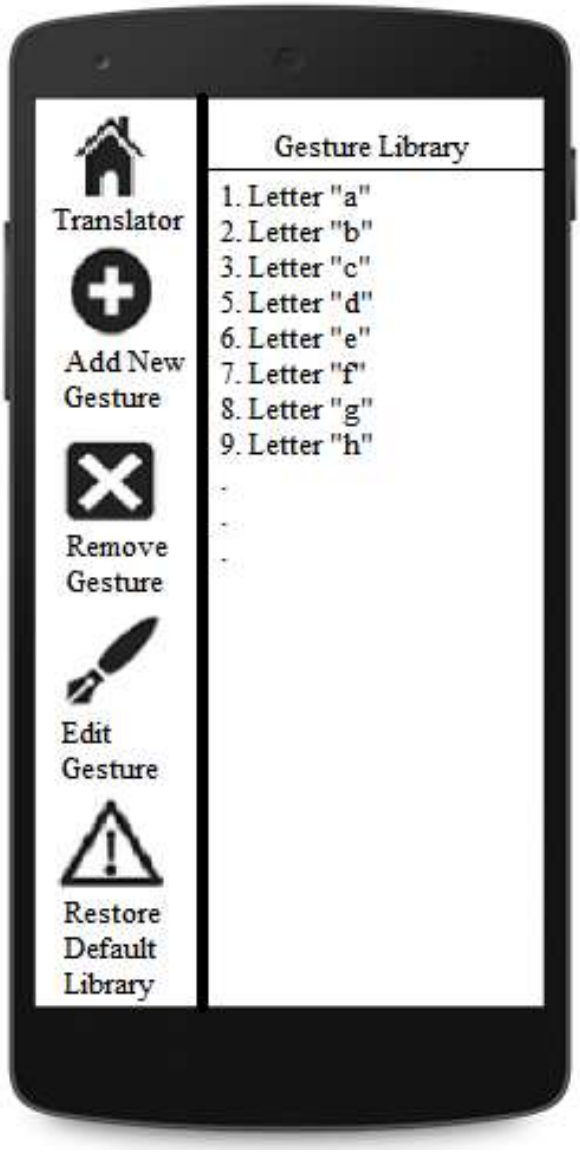
Function	Description
Add New Gesture	Add a new gesture to the library by performing the corresponding hand movements. Repeat multiple times for calibration.
Remove Gesture	Delete a gesture from the library.
Edit Gesture	Change an existing gesture by editing its information or recalibrating the hand movements.
Restore Default Gestures	Restore original supported gestures (a – z)

The supported gestures that will be included in the gesture library by default are the American alphabet letters, "a" through "z". Additional gestures may be added to the library like numbers, words, and more if desired; however, some of these sign language gestures may be too complicated to translate accurately. For this reason, gestures can be added, removed, or edited so that the gesture library

only contains supported gestures. These gestures will be stored in the memory with the application and include information about the values needed to perform the gesture with the Sign Language Interpreter Glove (SLIG). The mobile application is not responsible for translating the data sent directly from the glove sensors; this is the job of the microcontroller. However, the mobile application is responsible for interpreting the translation sent from the glove (sent as an ASCII character) and finding the corresponding gesture in the gesture library. Also, the mobile application is responsible for requesting permission from the microcontroller to add new gestures to the library by recording the sensor values and storing them as a new option for translation on the microcontroller. The process of adding/editing gestures requires repeating the hand gesture multiple times so that the machine learning algorithm can store multiple sets of data and create a range of accepted values. Calibrating the microcontroller for each gesture is necessary so that the SLIG will perform translations accurately even if the user performs them with slight variations.

In addition to the functionality of the gesture library, a menu layout will need to be created for this feature because it is the second most important aspect of the mobile application after the translator. In the Menu Layout section (6.2.2.3), a general menu for the translator screen was created which will be the main theme of the application. Therefore, a second menu must be created for the gesture library which has the same theme as the main menu. The menu for the gesture library will be opened when the user selects the “Gesture Library” link in the main menu, opening a new interface that does not lie over the translator feature like before. The interface for this new menu needs to be simple just like the main menu, so only necessary features will be added to keep the user focused on the task at hand. An example of what the user interface might look like for the Gesture Library feature is shown in Figure 6.24, on the next page.

Figure 6.24: Gesture Library Menu Layout



6.4 Bill of Materials

Below is a listing of the key components that will be used in creating the SLIG. The team chose to omit consumable such as resistors, wiring and the like for the sake of simplicity.

Table 6.4: Bill of Materials

Item	Description	Vendor	Part Number	Quantity	Unit Price
1	4.5" flex sensor	SparkFun	SEN-08606	5	$12.95
2	6 Degrees of Freedom IMU Digital Combo Board	SparkFun	SEN-10121	1	$39.95
3	Force Sensitive Resistor 0.5"	SparkFun	SEN-09375	5	$6.95
4	UA Strikeskin Tour glove	Under Armour	1275442	1	$21.99
4	3.7 Volts Polymer Lithium Ion Battery – 2000 mAh	SparkFun	PRT-08483	1	$12.95
5	Tenergy TLP-4000 Universal 1A Smart Charger	Amazon	01281	1	$21.95
6	TI LP2985 Regulators	Mouser Electronics	595-LP2985-33DBVR	5	$0.55
7	HM-10 4.0 BLE Module	Amazon	B00C2FIHKQ	1	$10.53
8	MSP430	Mouser Electronics	595-MSP-EXP430G2	1	$9.99
9	PCB	PCBzone.com	N/A	1	$74.64
10	Google Play Developer fee	Google Play	N/A	1	$25.00

7. Construction

7.1 Testing and Evaluation Plan

7.1.1 Hardware Testing

Most of the hardware can be tested independently and multiple times throughout the different stages of the project. This team will plan to test these hardware components at least once before integrating all the parts together. After some sort of functional prototype has been constructed, all the hardware can be (re)tested through similar examinations that will most likely result in a trial, error and adjustments process.

Pre-Prototype:

Flex Sensors – The flex sensors could very well be the most straightforward of all the hardware components to test. All that is required is to assemble the voltage divider circuit mentioned previously and apply some sort of power or voltage supply. A multimeter must be connected to this circuit (with a parallel connection to the flex sensor) so that the voltage across the flex sensor can be measured. Then as the sensor is slowly flexed from its un-bent form to its maximum angle, the measured voltage should be monitored to verify that it is varying accordingly. More precisely, the voltage measured should be decreasing steadily.

Force Sensitive Resistors – The force sensitive resistor can be checked in the same manner. The key difference would be is that instead of bending the sensor, a force must be applied to the sensing area. It is here that the testing becomes a bit more complicated than before because it is not as easy to apply a consistently increasing force. Nonetheless, assuming that the proper pressure is applied, the voltage across this resistor should be seen decreasing.

Post-Prototype:

Flex Sensors – Once a prototype has been assembled the output from the sensors will be left to the MCU to interpret. However, the idea will continue to be the same. The voltages signals sent to the MCU will need to be decreasing as the sensor is flexed. Once the mobile application interface has been established, the more logic manner of testing would be to form different letters that soley depend on flexing each finger and see whether the correct letter is displayed through the application. If there are ever any inconsistencies either the way the voltage variation are being interpreted needs to be the adjusted or the wiring connecting the flex sensors need to be adjusted. This process should be repeated until the right letter is read.

6 Degrees of Freedom IMU – There are two components to this combo board and the two will require slightly different test approaches. The accelerometer's most important function will be to identify the orientation or tilt to the user's hand. This is important distinguish between pairs of letters and even to ensure single letters are being formed with the proper tilt since this is characteristic of those letters just as much as the shape of the hand. So a letter such as "Q" can be used to check the accelerometer since this letter requires that the user's hand be pointed downwards. If both the physical connections and the programming are set up correctly, the glove should only read the letter "Q" when then the hand is tilted downwards. If the user forms the right shape with his hand abd flexes the right fingers but down not tilt his or her hand in the right direction, the glove should not interpret the letter "Q" at that moment.

The accelerometer portion of the combo board is tuned to identify the two letters that are distinguished by their motion. Thus, in order to check whether this portion is functioning properly, one of those letters such as "J" should be tested along with the letter "I" which is nearly identical to "J" in all other respects except that is lacks the swinging motion. If the glove recognized the letter to be "J" after every repetition of the experiment then no further corrections need to be made. Otherwise, some adjustments will have to be made and the testing will have to be repeated.

Force Sensitive Resistors – The central idea will remain the same as this final major component of the hardware is tested. The FSR's will be used to distinguish between certain pair of letters that differ by which, if any, of the fingers are touching one another thereby creating some sort of force as the fingers are held together. One such pair could be "U" and "V". Both letters should be tested and the glove should be able to tell them apart on different trials.

7.1.2 Software Testing

7.1.2.1 Control System Test Plan

The control system will have to be tested after it is implemented. This requires making sure that the input from the user’s hand gestures matches the output that is viewed at the user interface, whether it be the Android application, and the on-board LCD display if the group decides to go ahead with the on-board display. To do this, there will need to be a method to meticulously test every possible hand gesture to make sure that can be performed by the user produces the desired outcome on the user end.

To efficiently test the control system and truly make sure that the control system is actually designed to make correct decisions based on certain outputs, the system will need to be pushed to its limits. This means that the group will need to provide the system with controversial inputs where the system may be likely to make a mistake. For example, as mentioned in an above section, certain letters

like U, V, and R are very similar to each other. During the testing of the control system, the group will make hand gestures that are clearly a certain letter, but performed in ways that may be close to one of the other similar letters while trying to induce an error. This will put to test the implementation of the pressure sensors or contact sensors. Also, the software that interprets the data provided by these sensors will likely have to be tweaked because it is highly likely that the control system will produce some level of error when processing these very similar letters. Performing this for a number of times will find the errors that were made when initially writing the code for the control system and will allow the group to take any necessary mitigations to correct the errors that were found while performing these tests.

Similar to the U, V, and R, the letter J will also be a controversial letter that will need to be tested very carefully. It is highly likely to produce an error because this letter not only depends on the position of the fingers, but also on the motion of the hand. The hand gesture interpretation function will use the data collected from the accelerometer in order to determine when the hand was moved in a certain direction which corresponds with the motion required to make a J, and also that the data from the flex sensors concur that the fingers are actually positioned in the position that corresponds with the J.

As seen, it is evident that this letter is multiple degrees more complicated than the rest of the letters. This letter will require some ample testing and there will inevitably be issues. The group suspects that it will be necessary to perform this letter in a certain way every time so as to remove some of the uncertainty that comes with the letter J.

All of the testing described above will be checked by monitoring on the computer screen within the development environment (likely a HyperTerminal window). However, it is also necessary to check that the data is able to make it out to the Bluetooth module correctly as well. The testing of the Bluetooth module will be handled in that section, but the group needs to make sure that the output from the control system that is being sent out to the Bluetooth module is actually the correct data that is being seen on the HyperTerminal window when the control system is being tested on the computer screen.

7.1.2.2 Mobile Application

The mobile application is a big portion of the project that operates to display the gestures translated by the rest of the sign language glove. In order to test the mobile application, it is necessary to identify the inputs and outputs of the application because these will determine the overall functionality. Since these parameters cannot be tested until a working prototype of the glove is built, testing the mobile application will require using simulated gestures from the computer. Also, it is standard programming procedure while building any software application to test the program during the development process and make sure

each component works separately. On a good note, Android Studio, along with most IDEs, provides tools to help developers fix the errors in their code so this type of testing will come with the process of building the application.

7.2 Facilities and Equipment

There will be a number of facilities the team has used and plans on using in order to create a final working prototype.

The first of these was the eli2 Idea Lab located in the UCF Engineering II building. Known for its unique design; a glass enclosed space with sails flying above, LED lighting, eccentric chairs and stools and large projection screen to display concepts that will invite creativity, this lab was modeled after creativity spaces at GOOGLE and Pixar. It was here that the team first brainstormed ideas about what it wanted to attempt for Senior Design and the innovative environment helped the team refine their ideas until they came to a consensus of what it was they wished to do. That consensus was to attempt a redesign of the Sign Language Interpreter Glove.

Second, the team plans to use the Senior Design Lab extensively for further research, development and possibly testing of the final prototype. This laboratory which is located in Engineering I building of UCF is a facility that provides 24/7 access to a workspace with instrumentation, equipment and software for students specifically enrolled in Senior Design. It has a wide range of equipment that includes things such as an oscilloscope, function generator, multimeter and different software such as Multisim that will most likely prove invaluable in troubleshooting and refining our project.

Moreover, another possible laboratory that might prove itself useful is the Texas Instruments Innovation Lab. This was designed to allow students to bring their ideas to this space, strategically located next to the Idea Lab, to quickly build prototypes with 3D printers, laser cutters, TI components and equipment and other high-tech machines. Materials such as plastic, foam and metal are also available. These types of resources will come especially in handy as the team begins constructing the prototype.

Aside from the equipment said to be available in the different facilities above, this project will most likely require use of other tools. A soldering iron will be needed to bring different electronic components together at one point or another. Although it may seem trivial, a computer with the correct software will nonetheless be crucial when it comes to programming the MCU and building the user-interface application for the android system.

7.3 Suppliers

In order to create the SLIG the team will need various parts supplied by different vendors. As far as the different sensors are concerned, the team's preferred supplier will be SparkFun Electronics. SparkFun Electronics is an electronics retailer in Niwot, Colorado, United States. It manufactures and sells microcontroller development boards and breakout boards. All products designed and produced by SparkFun are released as open-source hardware.

For the glove, the team has chosen a model made by Under Armour because of it solid reputation and sturdy, high quality products. This way no more than glove will be needed throughout the entire prototype creation process. However, it is possible that the glove will not be purchased directly from the Under Armour website but from a third-party retailer in order to reduce costs if one is found.

One such third-party website could definitely be Amazon.com. The team was already chosen to order both a charger and a Bluetooth module from the retail giant. There is no doubt that Amazon will be a speedy, reliable supplier with a myriad of other parts to offer for future need.

Mouser Electronics is a global leading authorized distributor of semiconductors and electronic components for more than 500 industry leading suppliers. Their vast inventory of products include semiconductors, interconnects, passives, and electromechanical components. They have even won awards for their reputable performance in global customer service. The team has decided to order the SLIG's msp430 and voltage regulators from Mouser after seeing their competitive prices.

PCBzone was selected for the printed circuit board construction. They are known to specialize in small quantity PBC manufacturing and have one of the fastest lead times available. The team selected them especially for their competitive pricing is achieved as it turns out by combining several clients' designs on one manufacturing panel and sharing the tooling costs between them. They cater to both large-scale and small-scale customers; never sacrificing reliability or quality for their speedy delivery.

8. Project Operation

Working the SLIG- The SLIG will most likely be made as a right handed glove given that most users will have a dominant right hand. The glove will need to be placed over the user's hand and secured using the straps around the wrist. From this point, the user can begin to form any sign language letter from the 26 letters of the English alphabet he or she chooses. The user must make sure to form the signs as accurate as possible paying close attention to how much he or she is bending each finger, the overall direction his or her hand is pointing and to complete each additional motion that may be part of the letter.

As the user performs each letter carefully and with sufficient time in between distinct letters, someone will have to access the mobile application to verify the correct letters are being displayed on the screen. The mobile application will most likely have a very simple interface and will automatically synchronize itself with the SLIG once it has been initialized. Each letter the user wearing the SLIG signs should appear on the screen and remain there until the next letter is signed. For the user's benefit, the team recommends he or she attempt every single letter at least once to make sure they are making the correct sign. When done with his or her exercises, all the user needs to do is power off the glove and remove it from his or her hand.

9. Administrative Content

9.1 Project Budget

The proposed budget for the SLIG takes into consideration all of the parts required to make the product as well as extra/replacement parts along the way. The prices listed are estimates from online research and will be updated in the future once the final product is built. As of now, there is no sponsor for SLIG and all of the costs will be distributed among the group members.

Table 9.1: Project Budget

Part Description	Price ($)	Quantity	Cost ($)
Power source	$10	1	$10
Microcontroller	$50	1	$50
Flex sensors	$10	10	$100
Accelerometer	$30	1	$50
Glove	$20	1	$20
PCB	$150	1	$200
Bluetooth adaptor	$10	1	$10
Feedback LEDs	$1	10	$10
Miscellaneous parts	$100	?	$100
Total Cost			$550

9.2 Milestones

The project milestones show a tentative schedule for the entire Senior Design course that breaks the project down into a list of tasks to be completed. Each team member is responsible for their own tasks as well as participating in team tasks so that by the end of the timeline everything will be completed as planned.

Table 9.2: Milestones for Senior Design I & II

Number	Task	Start	End	Duration (weeks)	Responsible
Senior Design I					
1	**Brainstorming**	9/1/2015	9/8/2015	1	The Team
2	**Project Selection & Role Assignments**	9/8/2015	9/15/2015	1	The Team
	Project Report				
3	Initial Document - Divide & Conquer	9/8/2015	9/15/2015	1	The Team
4	First Draft	9/15/2015	11/3/2015	7	The Team
5	Final Document	11/3/2015	12/8/2015	5	The Team
	Research & Documentation				
6	Bluetooth	9/15/2015	10/5/2015	3	Ramon
7	Flex Sensors	9/15/2015	10/5/2015	3	Chris
8	Accelerometers	9/15/2015	10/5/2015	3	Chris
9	Software	9/15/2015	10/5/2015	3	Jason
10	Power Source	9/15/2015	10/5/2015	3	Jason
11	Microcontroller	9/15/2015	10/5/2015	3	Emanuel
	Design				
12	Bluetooth	10/6/2015	11/3/2015	4	Ramon
13	Flex Sensors	10/6/2015	11/3/2015	4	Chris
14	Accelerometers	10/6/2015	11/3/2015	4	Chris
15	Software	10/6/2015	11/3/2015	4	Jason
16	Power Source	10/6/2015	11/3/2015	4	Jason
17	Microcontroller	10/6/2015	11/3/2015	4	Emanuel
18	**Order & Test Parts**	11/3/2015	12/8/2015	5	The Team
Senior Design II					
19	**Build Prototype**	1/11/2016	3/1/2015	7	The Team
20	**Testing & Redesign**	3/1/2015	3/29/2015	4	The Team
21	**Finalize Prototype**	3/29/2015	4/15/2015	2	The Team
22	**Peer Presentation**	TBA	TBA		The Team
23	**Final Report**	TBA	TBA		The Team
24	**Final Presentation**	TBA	TBA		The Team

9.3 Division of Labor

Group 24 is composed of four electrical engineering students from the University of Central Florida. Every member of Group 24 had his responsibilities and roles throughout the design phase that will carry into the following school semester as the team begins creating and optimizing a prototype. In general, Emmanuel Hernandez was in charge of the control unit, both from a hardware perspective and from the software perspective. Christopher Delgado was in charge of the different sensors, which are the primary means of data collection, while Ramon Santana divided his efforts between the power supply system and the Bluetooth communication. Lastly, Jason Balog is creating and handling the android application that will be the output interface for the project. Each member is listed below followed by specifics on his responsibilities and by his contributions to the group dynamics.

Emmanuel Hernandez – Emmanuel has taken up one of the main roles in designing SLIG. He is in charge of the physical aspects and the programming for the SLIG's control unit. This includes but is not limited to selecting a proper MCU, working out all the analog-to-digital conversions, making the leading decisions regarding the PCB and writing and developing all the necessary code and algorithms to interpret the data the glove collects. When it comes to the PCB, he will be deciding which software program to use to design the PCB board and whom the team will use to create the board. Essentially, Emmanuel has played the main role in helping the team define how they would design the sign language glove acting as the lead engineer for this project. Moreover, he has participated in every document for this project thus far.

Christopher Delgado – Christopher was in charge of researching and selecting the best model for the three different types of sensors the glove will require. As mentioned before, these are the flex sensors, the accelerometer and gyroscope, which are typically sold as part of a single unit and the some sort of contact sensor. He was also responsible for determining how these sensors will work and to integrate them to the main circuit board. Moreover, he has participated in writing every document for this project thus far.

Ramon Santana – Ramon has two separate features of the project to research and design. First, he was in charge of the power system for the entire glove. This included researching what type of battery technology would best power the glove as well as developing voltage regulation circuits to ensure each component of the glove will receive the proper supply voltage. Secondly, Ramon was in charge of the researching how to add Bluetooth capabilities to the glove to allow it to communicate to any android cell phone device, which has the user interface application. Aside from his technical responsibilities Ramon took it upon himself to keep the team in order and on track. He played the biggest role in coordinating team meeting and recommending due dates for different parts of the project in a sense taking up the role of group supervisor.

Jason Balog – Jason's roles are on the software side of the project where he has to design the android application that will allow the team to use any android-based smart phone as an output interface for the glove. He began by researching different mobile platforms before deciding on Android, then, he looked into which programming language would be most suitable to develop the application and lastly planned out how to develop the application. His efforts were crucial to the team, which lacked a computer-engineering student who typically take up the software tasks in most other senior design projects. Moreover, he has participated in writing every document for this project thus far.

9.4 Personnel

Ramon Santana – Electrical Engineering
Ramon Santana is a first generation, electrical engineer student at the University of Central Florida. He has maintained leadership presence on campus in numerous ways. Santana was a Teaching Assistant for two engineering classes. Currently, Santana is a Peer Mentor and Peer Tutor for engineering students at the office of Prime STEM at UCF. Santana is the Mentoring Program Coordinator for the Society of Hispanic and Professional Engineers at UCF. Santana is also a brother of Lambda Theta Phi Latin Fraternity Incorporated. During the summer of 2015, Santana did an internship at Florida Power & Light (FPL) as a Protection and Control Engineer. His responsibilities at that time were to make sure that all the equipment inside transmission and distribution substations were working and functioning properly; by performing maintenance on feeder breakers, calibrating relays, completing trip by lockouts and much more. His hands on experience will be a significant contribution to the implementation of the Sign Language Interpreter Glove. Santana will be responsible for the wireless communication of the SLIG and also for providing the right amount of power to all the components in the design.

Christopher Delgado – Electrical Engineering
Christopher Delgado is a first generation electrical engineering undergraduate student at the University of Central Florida. He has been a student there since 2011 and has been part of the Burnett Honors College since the beginning. Since the summer semester of 2015, he has been working as a System Performance intern for Verizon Wireless and has gained valuable experience in networking. Prior to his internship with Verizon, Christopher worked as a tutor for the SDES TRIO Center in UCF where he helped other students improve their performance in their physics, calculus and elementary engineering courses. His years at UCF have given him the background knowledge and learning skills to contribute to the success of Group 24 and the creation of the SLIG. Christopher has been assigned the task of researching, selecting and designing for the different sensors the SLIG will require. His successful completion of both Electronics I and II and their accompanying laboratories have prepared Christopher to deal with the hardware components to the SLIG.

Emmanuel Hernandez – Electrical Engineering

Emmanuel Hernandez is an electrical engineering undergraduate student. He has been at the University of Central Florida since 2011, where has developed the skills necessary to participate in this project. He has been a mathematics tutor at the UCF Mathematics Assistance & Learning Lab (MALL) since 2012, and has completed 3 semesters of internships with two different companies. During the first internship at The Walt Disney Company, Emmanuel was given the opportunity to work with programmable logic controllers (PLC). In this time he gained his first experiences with control systems, although it doesn't have much to do with microcontrollers. About a year later, he was given the opportunity to participate in an internship at Florida Power & Light Company, where he was given the opportunity to work with microprocessor relays and get more of an experience with control logic, and this time a little more in-depth in the microprocessor side of control systems. This being said, Emmanuel was responsible for the control system aspect of the Sign Language Interpreter Glove. He designed the control system, and performed the research necessary to formulate the best plan of action when it came to having the SLIG determine which hand gesture is being performed by the user. He feels like his previous experience in the industry, specifically with control systems made it so that he had a little bit more of an intuition at the time of analyzing the best course of action for the SLIG control system. In the building and implementation part of the project, he will be responsible for programming the microcontroller to make decisions on what hand gestures are currently being performed.

Jason Balog – Electrical Engineering

Jason Balog is an undergraduate majoring in Electrical Engineering and minoring in Mathematics at the University of Central Florida. He has focused on learning about power systems and computer simulation in his technical elective courses which has prepared him to take on the software application portion of the project. There are two main software components to the project – the mobile application and gesture recognition – which need two different people to work on since each component is not related to the other. The mobile application will be more software intensive than the gesture recognition feature and will require a good background in computer programming. Since all four of the team members are majoring in Electrical Engineering, nobody on the team is well equipped with the required programming skills to write an Android application using the languages Java and XML. However, Jason has taken multiple elective courses that required programming in Matlab and other software along with some programming experience on his own which gives him the best opportunity to successfully complete the mobile application. Also, he has expressed personal interest in learning how to write a mobile application and all of the hurdles that come with this unfamiliar territory for most Electrical Engineers so that he can expand his skillset.

10. Conclusion

In conclusion, the Sign Language Interpreter glove is a lightweight, thin glove that can be worn by individuals who have a speech impediment which translates sign language hand gestures into text. This text will be displayed on a mobile application that will run on an Android smartphone device. The device will make the use of flex sensors, which are variable resistors that change their resistance in proportion to the amount of bend that is currently present on the sensors. This allows the control system of the glove to determine how much bend is present on each finger. Other hardware that will be used to determine what hand gesture is currently being performed by the user is an accelerometer. The accelerometer will determine the X, Y, Z position of the hand at all times, which is important in determining when certain hand gestures are being made that require a specific movement of the hand in addition to simply bending the fingers in certain ways.

In addition to the flex sensors and accelerometer, the group will employ the use of pressure sensors or contact sensors that determine when two fingers are close together or not. This is necessary because there are certain letters that have very similar amount of bend on each finger, and the only way to determine between them is the actual position of the fingers relative to each other. By using pressure sensors or contact sensors, the control system of the SLIG can determine which of these very similar hand gestures is currently being performed.

The control system of the SLIG will consist of an MSP430 microcontroller unit. This unit already comes equipped with the necessary analog to digital conversion hardware that is needed to convert the analog signals that is coming from the sensors into a digital signal that can be analyzed and processed by the microcontroller unit. The MSP430 will go through its program which is designed to determine which of the many possible hand gestures is currently being performed, and it will constantly be sending the output out to the user interface.

The user interface will be an Android application that will receive the data that is being transmitted from the microcontroller and it will display it on the screen for the receiver of the message from the user to read. This mobile application will be capable of wirelessly receiving the information that is being sent out from the microcontroller unit. The mobile application will be written using the Java language and it will be capable of interfacing with the microcontroller unit mounted on the glove via the use of wireless communication.

The wireless communication that will be used to transmit the data between the microcontroller unit on the glove and the user interface will be the Bluetooth communication technology. Bluetooth essentially uses low power radio waves to wirelessly transmit data between electronic devices. There will be a Bluetooth module mounted on the glove which will receive the output messages from the microcontroller unit. This module will transmit the message through the air, and it

will be received at the user interface device. The Android smartphones come Bluetooth enabled, so all that will be necessary is to pair the device with the Bluetooth module mounted on the glove, and communication will be established.

All of the aforementioned electronic components will mounted on a printed circuit board. Because of the small size of the SLIG, a circuit board no larger than one square foot is desired. This will require the group to employ the use of a printed circuit board with multiple layers. The Bluetooth module, microcontroller unit, DC to DC converters (or voltage regulators), pull-down resistors, charging circuit, as well as all of the other electronic components that go into the SLIG will be mounted on the printed circuit board. The printed circuit board will ideally be placed on the palm of the glove, and it will be in a location that will allow the user of the glove to seamlessly move their hand around without having to worry about possibly damaging the electronic components or injuring themselves by the way of an electrical shock.

Appendix A: Copyright Permission

Permission to use Figures from SparkFun Electronics

Re: Fw: Permission to use picture from website
SparkFun Customer Service <cservice@sparkfun.com>
Mon 12/7/2015 12:30 PM
To: christopher.delgado <christopher.delgado@knights.ucf.edu>;

Type your response ABOVE THIS LINE to reply

christopher.delgado
Subject: Permission to use picture from website

DEC 07, 2015 | 10:29AM MST
Nick M replied:

Hello Christopher-

As long as the pictures you do end up using, whether from the site or the datasheet, are properly credited, we have absolutely no problem with you using them!

Please let me know if there is anything further I can do to help.

Nick Miranda
SparkFun Electronics
Distributor and Customer Service
303-945-2984 x 607

DEC 07, 2015 | 09:33PM MST
christopher.delgado replied:
Hello,

My name is Christopher Delgado. I am an electrical engineering student at the University of Central Florida. Currently, I am working on a Senior Design project and I would like permission from you to use the circuit schematics from the following data sheets http://cdn.sparkfun.com/datasheets/Sensors/ForceFlex/2010-10-26-DataSheet-FSR400-Layout2.pdf and https://www.sparkfun.com/datasheets/Sensors/Pressure/fsrguide.pdf .<https://cdn.sparkfun.com/datasheets/Sensors/ForceFlex/FLEXSENSORREVA1.pdf> These schematics will not be modified or published; only used as reference material for my project report.

Thank you in advance for your time. Your permission will be greatly appreciated.

Christopher Delgado
University of Central Florida
Electrical Engineering Student

DEC 07, 2015 | 08:29PM MST
christopher.delgado replied:

Hello,

My name is Christopher Delgado. I am an electrical engineering student at the University of Central Florida. Currently, I am working on a Senior Design project and I would like permission from you to use the circuit schematics from the following data sheet http://cdn.sparkfun.com/datasheets/Sensors/IMU/IMU_Digital_Combo_Board%20_-_6_Degrees_of_Freedom_-_ITG3200_-_ADXL345.pdf.<https://cdn.sparkfun.com/datasheets/Sensors/ForceFlex/FLEXSENSORREVA1.pdf> These schematics will not be modified or published; only used as reference material for my project report.

Thank you in advance for your time. Your permission will be greatly appreciated.

Christopher Delgado
University of Central Florida
Electrical Engineering Student

DEC 07, 2015 | 07:36PM MST
Original message
christopher.delgado wrote:

Hello,

My name is Christopher Delgado. I am an electrical engineering student at the University of Central Florida. Currently, I am working on a Senior Design project and I would like permission from you to use the circuit schematics from the following data sheet https://cdn.sparkfun.com/datasheets/Sensors/ForceFlex/FLEXSENSORREVA1.pdf.<https://cdn.sparkfun.com/datasheets/Sensors/ForceFlex/FLEXSENSORREVA1.pdf> These schematics will not be modified or published; only used as reference material for my project report.

Thank you in advance for your time. Your permission will be greatly appreciated.

Christopher Delgado
University of Central Florida
Electrical Engineering Student

Permission to use Figures from Sensor Products INC.
RE: Senior Design Permission
Vadim Shalyt <vshalyt@sensorprod.com>
Tue 12/1/2015 3:39 PM
To: christopher.delgado <christopher.delgado@knights.ucf.edu>;

Yes you can. Did you actually need to order any film Chris?

Vadim Shalyt
Sr. Application Specialist
Sensor Products Inc. USA
300 Madison Ave.
Madison, NJ 07940
1.973.428.8985 **(phone)**
1.973.495.9800 (cell)
vadim@sensorprod.com

Please contact me DIRECTLY for BEST Prices, Delivery, Service & Expert Advice.

The finest compliment I can ever receive from doing business is a referral from my clients.

From: christopher.delgado [mailto:christopher.delgado@knights.ucf.edu]
Sent: Tuesday, December 01, 2015 3:02 PM
To: Vadim Shalyt
Subject: Re: Senior Design Permission

There is a research section to our paper where we have to discuss different technologies/ models available even ones we might not end up using. Having the specs table would make it easy to compare and contrast with other parts.

Sent from my iPhone
On Nov 30, 2015, at 6:39 PM, Vadim Shalyt <vshalyt@sensorprod.com> wrote:

Thank you Chris, but I am not understanding how you can use specs without the product. Can you elaborate?

Vadim Shalyt
Sr. Application Specialist
Sensor Products Inc. USA
300 Madison Ave.
Madison, NJ 07940
1.973.428.8985 **(phone)**
1.973.495.9800 (cell)
vadim@sensorprod.com

Please contact me DIRECTLY for BEST Prices, Delivery, Service & Expert Advice.

The finest compliment I can ever receive from doing business is a referral from my clients.

From: christopher.delgado [mailto:christopher.delgado@knights.ucf.edu]
Sent: Monday, November 30, 2015 5:22 PM
To: Vadim Shalyt
Subject: Re: Senior Design Permission

Vadim,

I found your website through Google.

Thanks,

Chris

Sent from my iPhone

On Nov 30, 2015, at 5:19 PM, Vadim Shalyt <vshalyt@sensorprod.com> wrote:

Chris,

May I ask how you heard of our products?

Vadim Shalyt
Sr. Application Specialist
Sensor Products Inc. USA
300 Madison Ave.
Madison, NJ 07940
1.973.428.8985 **(phone)**
1.973.495.9800 (cell)
vadim@sensorprod.com

Please contact me DIRECTLY for BEST Prices, Delivery, Service & Expert Advice.

The finest compliment I can ever receive from doing business is a referral from my clients.

From: Info [mailto:info@sensorprod.com]
Sent: Monday, November 30, 2015 4:22 PM
To: 'Vadim Shalyt'
Subject: FW: Senior Design Permission

See below

From: christopher.delgado [mailto:christopher.delgado@knights.ucf.edu]
Sent: Saturday, November 28, 2015 10:36 AM
To: sales@sensorprod.com
Subject: Senior Design Permission

Hello,

I am an electrical engineer student at the University of Central Florida. I would like to use the attached specifications table in my senior design paper. Do I have permission to use this figure?

Thank you,

Christopher Delgado

Permission to use picture from Under Amour

Permission to use picture from website [Reference: 151208-001341]
Under Armour Customer Service <customerservice@underarmour.com>
Tue 12/8/2015 1:57 AM
To: christopher.delgado <christopher.delgado@knights.ucf.edu>;

Under Armour is here to help!

Thanks for contacting Under Armour. An Under Armour teammate is researching your request, and you should hear back from us shortly. If you requested an order change or cancellation, please call us directly at 1.888.727.6687, so we can make sure to get that taken care of as soon as possible.

To start shopping for the best performance gear on the planet, visit us at Underarmour.com

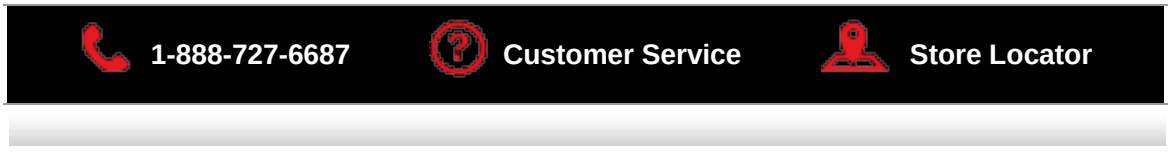
Or to keep you up-to-date on the latest incentives, insider info, and newest innovations, sign up for our emails here

And don't forget to follow us on Facebook and Twitter 

Thanks for choosing Under Armour.

Get better every day.

-UA Customer Service Team



Permission to use picture from website

Ramon Santana Alberto

To: BatteryU@cadex.com

Friday, December 04, 2015 9:57 AM

• Retention Policy: UCF Sent Items Default Policy Tag (7 Months) Expires: 7/1/2016

Hello,

My name is Ramon Santana. I am an electrical engineering student at the University of Central Florida. Currently, I am working on a Senior Design project and I would like permission from you to use "**Table 1: Characteristics of commonly used rechargeable batteries.**" in my document from the website: http://batteryuniversity.com/learn/article/secondary_batteries

This table will not be modified neither published, it will only be used for educational purpose.

Thank you in advance for your time. Your permission will be greatly appreciated.

Ramon Santana
University of Central Florida
Electrical Engineering Student

Re: Permission to use picture from website

BatteryU [BatteryU@cadex.com]

To: Ramon Santana Alberto

Friday, December 04, 2015 1:08 PM

• Retention Policy: UCF Inbox Default Policy Tag (7 Months) Expires: 7/1/2016

Hi Ramon,

Yes, you may use the material as requested. Please cite sources where appropriate.

Regards,

John Bradshaw - Marketing Communications Manager
Cadex Electronics Inc. | www.cadex.com
Vancouver | Minneapolis | Frankfurt
Tel: +1 604 231-7777 x319 | Toll Free: 1-800 565-5228

Follow us on Twitter: twitter.com/cadexelectronic
Join us on Facebook: facebook.com/cadexelectronics
Add us on Google+: plus.google.com/+Cadex

Message:
Hello,

I am an electrical engineering student at the University of Central Florida. Currently, I am working on a Senior Design project and I would like permission from you to use the picture of the "nRF8001 BLE pin diagram" in my document from the datasheet.

The picture will not be modified neither published, it will only be used for educational purpose.

Thank you in advance for your time. Your permission will be greatly appreciated.

Ramon Santana
University of Central Florida
Electrical Engineering Student

Hi Ramon,
Many thanks for sending your request for using one of our product photos.

You are of course free to use the picture in your document.
Wish you all the best on your Senior Design project!

Let me know if there is anything else I can do for you. Would be more than happy to provide a few dev kits for other future projects!

Best regards,
Anne Strand

Anne Strand
Marketing Communications Manager
P: +47 22 51 10 62 | M: +47 971 63 752
Skype: annestrand11

Nordic Semiconductor
Karenslyst Allé 5, 0278 Oslo, Norway
www.nordicsemi.com

Permission to use picture from your presentation

Ramon Santana Alberto

To: yeokm1@gmail.com

Friday, December 04, 2015 10:11 AM

• Retention Policy: UCF Sent Items Default Policy Tag (7 Months) Expires: 7/1/2016

Hello Meng,

My name is Ramon Santana. I am an electrical engineering student at the University of Central Florida. Currently, I am working on a Senior Design project and I would like permission from you to use the picture in slide 17 **"1d. BLE connection procedure"** in my document from your "Introduction to Bluetooth Low Energy" presentation.

The picture will not be modified neither published, it will only be used for educational purpose.

Thank you in advance for your time. Your permission will be greatly appreciated.

Ramon Santana
University of Central Florida
Electrical Engineering Student

Re: Permission to use picture from your presentation

yeokm1 . [yeokm1@gmail.com]

To: Ramon Santana Alberto

Friday, December 04, 2015 10:25 AM

• Retention Policy: UCF Inbox Default Policy Tag (7 Months) Expires: 7/1/2016

Sure. Go ahead. :)

On Fri, 4 Dec 2015 23:11 Ramon Santana Alberto <Ramon.Santana@ucf.edu> wrote:

> Hello Meng,
>
> My name is Ramon Santana. I am an electrical engineering student at the University of Central Florida. Currently, I am working on a Senior Design project and I would like permission from you to use the picture in slide 17 **"1d. BLE connection procedure"** in my document from your "Introduction to Bluetooth Low Energy" presentation.
>
> The picture will not be modified neither published, it will only be used for educational purpose.
>
> Thank you in advance for your time. Your permission will be greatly appreciated.
>
> Ramon Santana
> University of Central Florida
> Electrical Engineering Student

Appendix B: References

Similar Projects

"GloveSense » Electrical & Computer Engineering | Boston University." Electrical Computer Engineering RSS. Web. 4 Sept. 2015.

Troili, Brian, Laura Rubio-Perez, Ali Mizan, and Kirk Chan. "High Six." The Sign Language Glove - Home. Web. 4 Sep. 2015. <http://www.eecs.ucf.edu/seniordesign/fa2013sp2014/g06/>.

Wireless Communication

"A Guide to Selecting a Bluetooth Chipset." Argenox Technologies. Web. 17 Oct. 2015. <http://www.argenox.com/bluetooth-low-energy-ble-v4-0-development/library/a-guide-to-selecting-a-bluetooth-chipset/>.

"Bluetooth Low Energy." Bluetooth Low Energy. Web. 6 Dec. 2015. <http://developer.android.com/guide/topics/connectivity/bluetooth-le.html>.

Brain, Marshall, Tracy V. Wilson and Bernadette Johnson. "How WiFi Works" 30 April 2001. HowStuffWorks.com. <http://computer.howstuffworks.com/wireless-network.htm> 08 November 2015.

Franklin, Curt, and Julia Layton. "How Bluetooth Works" 28 June 2000. HowStuffWorks.com. <http://electronics.howstuffworks.com/bluetooth.htm> 07 November 2015.

Meng, Yeo. "Introduction to Bluetooth Low Energy." Introduction to Bluetooth Low Energy. Web. 28 Oct. 2015.

Power Source

"BU-107: Comparison Table of Secondary Batteries." Secondary (Rechargeable) Batteries – Battery University. Web. 1 Nov. 2015. <http://batteryuniversity.com/learn/article/secondary_batteries>.

"Learnabout Electronics." Regulated Power Supplies. Web. 18 Oct. 2015. <http://www.learnabout-electronics.org/PSU/psu22.php>.

"What’s the Best Battery?" Advantages and Limitations of the Different Types of Batteries. Web. 30 Nov. 2015. <http://batteryuniversity.com/learn/article/whats_the_best_battery>.

Flex Sensors

"Long Flex/Bend Sensor." *Adafruit*. Web. 1 Dec. 2015. <https://www.adafruit.com/products/182>.

"Two-Directional Bi-Flex Sensors™." *Flex Sensors*. Web. 8 Dec. 2015. <http://www.imagesco.com/sensors/flex-sensor.html>.

"Tactilus® Flex Sensor." *Tactilus*. Web. 1 Dec. 2015. <http://www.sensorprod.com/flex-sensor.php>.

"Bend Sensor." - *LittleBits Electronics*. Web. 1 Dec. 2015. <https://littlebits.cc/bits/bend-sensor>.

Accelerometer and Gyroscope

"SparkFun Triple Axis Accelerometer Breakout - ADXL335." - *SEN-09269*. Web. 2 Dec. 2015. <https://www.sparkfun.com/products/9269>.

"SparkFun Triple Axis Accelerometer Breakout - ADXL345." - *SEN-09836*. Web. 2 Dec. 2015. <https://www.sparkfun.com/products/9836>.

"SparkFun Triple Axis Accelerometer Breakout - MMA8452Q." - *SEN-12756*. Web. 2 Dec. 2015. <https://www.sparkfun.com/products/12756>.

"SparkFun Triple Axis Accelerometer Breakout - ADXL362." - *SEN-11446*. Web. 2 Dec. 2015. <https://www.sparkfun.com/products/11446>.

"Gyro Breakout Board - IDG500 Dual 500°/s." - *SEN-09094*. Web. 2 Dec. 2015. <https://www.sparkfun.com/products/retired/9094>.

"SparkFun Tri-Axis Gyro Breakout - L3G4200D." - *SEN-10612*. Web. 2 Dec. 2015. <https://www.sparkfun.com/products/10612>.

"SparkFun 6 Degrees of Freedom IMU Digital Combo Board - ITG3200/ADXL3." *45*. Web. 3 Dec. 2015. <https://www.sparkfun.com/products/10121>.

"9 Degrees of Freedom - Razor IMU." - *SEN-10736*. Web. 3 Dec. 2015. <https://www.sparkfun.com/products/10736>.

Contact, Force and Pressure Sensors

"Flexiforce Pressure Sensor - 25lbs." - *SEN-08712*. Web. 3 Dec. 2015. <https://www.sparkfun.com/products/8712>.

"Force Sensitive Resistor 0.5"" - *SEN-09375*. Web. 4 Dec. 2015. <https://www.sparkfun.com/products/9375>.

"Force Sensitive Resistor - Square." - *SEN-09376*. Web. 4 Dec. 2015. <https://www.sparkfun.com/products/9376>.

"Pressure Sensor." - *LittleBits Electronics*. Web. 4 Dec. 2015. <https://littlebits.cc/bits/pressure-sensor>.

"Phidgets Touch Sensor." *Phidgets Touch Sensor*. Web. 4 Dec. 2015. <http://www.trossenrobotics.com/p/phidgets-touch-sensor.aspx>.

"Phidgets Linear Touch Sensor." *Phidgets Linear Touch Sensor*. Web. 4 Dec. 2015. <http://www.trossenrobotics.com/p/phidgets-linear-touch-sensor.aspx>.

"Phidgets Circular Touch Sensor." *Phidgets Circular Touch Sensor*. Web. 4 Dec. 2015. <http://www.trossenrobotics.com/p/phidgets-circular-touch-sensor.aspx>.

"FlexiForce Adapter." *FlexiForce Adapter*. Web. 4 Dec. 2015. <http://www.trossenrobotics.com/store/p/6526-FlexiForce-Adapter.aspx>.
"Phidgets Force Sensor." *Phidgets Force Sensor*. Web. 4 Dec. 2015. <http://www.trossenrobotics.com/p/phidgets-force-sensor.aspx>.

Gloves

"Under Armour | Men's Gloves." *Under Armour*®. Web. 5 Dec. 2015. <https://www.underarmour.com/en-us/mens/accessories/gloves>.

"Men's Versaliner™." *OutdoorResearch.com*. Web. 5 Dec. 2015. <http://www.outdoorresearch.com/en/mens-versaliner.html>.

Miscellaneous

"Gathering Lab, IdeaLab, and Innovation Lab Concepts to Be Implemented." *Gathering Lab, IdeaLab, and Innovation Lab Concepts to Be Implemented*. Web. 6 Dec. 2015. <http://iems.ucf.edu/news/gathering-lab-idealab-and-innovation-lab-concepts-be-implemented>.

ECE Department. Web. 8 Dec. 2015. <http://www.ece.ucf.edu/labs/sdl.php>.

"T.I. Innovation Lab (ENG2)." *Center for Entrepreneurial Leadership RSS*. Web. 6 Dec. 2015. <http://cel.ucf.edu/portfolio/t-i-innovation-lab/>.

"Bluetooth." Web. 8 Dec. 2015. <http://arxiv.org/pdf/1102.4106.pdf>.

"Core App Quality." *Core App Quality*. Web. 8 Dec. 2015. <http://developer.android.com/distribute/essentials/quality/core.html>.

"Content License." *Content License*. Web. 8 Dec. 2015. <http://developer.android.com/license.html#attribution>.

"Lithium-Ion Batteries." Web. 8 Dec. 2015. <http://www.rechargebatteries.org/wp-content/uploads/2013/07/Li-ion-safety-July-9-2013-Recharge-.pdf>.

Brownlee, Jason. "A Tour of Machine Learning Algorithms." *Machine Learning Mastery*. N.p., 25 Nov. 2013. Web. Nov.-Dec. 2015.

"FPGA Architecture for the Challenge." *FPGA Architecture for the Challenge*. N.p., n.d. Web. 09 Dec. 2015.

Giovino, Bill. "Lowest Power MSP430 Microcontrollers from Texas Instruments." *Microcontroller.com*. N.p., 19 Mar. 2011. Web. 09 Dec. 2015.

"PCB Basics." *Learn.sparkfun.com*. N.p., n.d. Web. 09 Dec. 2015.

Shamim, Toxin. "Bluetooth Module Interfacing with Microcontroller." *Vshamu*. N.p., 19 Mar. 2011. Web. 09 Dec. 2015.

iOS

Wikipedia. Wikimedia Foundation. Web. 9 Dec. 2015. <https://en.wikipedia.org/wiki/Swift_(programming_language)>.

Grimes, Shawn. "Mobile Development Overview." *Mobile Development Overview*. 26 Mar. 2012. Web. 9 Dec. 2015. <http://www.slideshare.net/shawngrimes/mobile-development-overview>.

Oliver, Crhis. "Will Swift Apps Work on Older IPhones?" *Will Swift Apps Work on Older IPhones?* 16 Sept. 2014. Web. 9 Dec. 2015. <http://learn.onemonth.com/will-swift-apps-work-on-older-iphones>.

"The Swift Programming Language (Swift 2.1): About Swift." *The Swift Programming Language (Swift 2.1): About Swift*. 21 Oct. 2015. Web. 9 Dec. 2015. <https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift_Programming_Language/>.

"Xcode - IDE - Apple Developer." *Xcode - IDE - Apple Developer*. Web. 9 Dec. 2015. <https://developer.apple.com/xcode/ide/>.

"Android vs IOS: Which Platform to Build for First?" *Savvy Apps*. Web. 9 Dec. 2015. <http://savvyapps.com/blog/android-vs-ios-which-platform-to-build-for-first>.

"IOS Technology Overview." *About the IOS Technologies*. 17 Sept. 2014. Web. 9 Dec. 2015. <https://developer.apple.com/library/ios/documentation/Miscellaneous/Conceptual/iPhoneOSTechOverview/Introduction/Introduction.html>.

Android

"Android, the World's Most Popular Mobile Platform." *Android, the World's Most Popular Mobile Platform*. Web. 9 Dec. 2015. <https://developer.android.com/about/android.html>.

"Introduction to Android." *Introduction to Android*. Web. 9 Dec. 2015. <http://developer.android.com/guide/index.html>.

Agarwal, Tarun. "IOS vs Android Development." *Build Blog by ThinkApps Content on Entrepreneurship Mobile Apps Web Platforms and More IOS vs Android Development Comments*. 25 Aug. 2014. Web. 9 Dec. 2015. <http://thinkapps.com/blog/development/platform-build-first-ios-vs-android/>.

Henneke, Cameron. "Android vs. IOS: Comparing the Development Process of the GQueues Mobile Apps." *GQueues*. 29 July 2013. Web. 9 Dec. 2015. <http://blog.gqueues.com/2013/07/android-vs-ios-comparing-development.html>.

Thomas, Carter. "How Much Does It Cost to Develop an App?" *How Much Does It Cost To Develop an App*. 6 Mar. 2011. Web. 9 Dec. 2015. <http://www.bluecloudsolutions.com/blog/cost-develop-app/#>.

"The Android Source Code." *The Android Source Code*. Web. 9 Dec. 2015. <https://source.android.com/source/>.

"Android Compatibility." *Android Compatibility*. Web. 9 Dec. 2015. <https://source.android.com/compatibility/index.html>.

Noyes, Katherine. "10 Reasons Open Source Is Good for Business." *PCWorld*. Web. 9 Dec. 2015. <http://www.pcworld.com/article/209891/10_reasons_open_source_is_good_for_business.html>.

Windows Mobile

Yusuf, Sani. "Windows Phone 8: Platform Overview - Envato Tuts Code Tutorial." *Code Envato Tuts*. 20 Aug. 2014. Web. 9 Dec. 2015. <http://code.tutsplus.com/tutorials/windows-phone-8-platform-overview--mobile-20387>.

"What's a Universal Windows Platform (UWP) App?" - *Windows App Development*. Web. 9 Dec. 2015. <https://msdn.microsoft.com/library/windows/apps/dn726767.aspx>.

"Choosing a Programming Language for Windows Mobile Development." *Choosing a Programming Language for Windows Mobile Development*. 19 Apr. 2010. Web. 9 Dec. 2015. <https://msdn.microsoft.com/en-us/library/bb677133.aspx>.

User Interface

Cerejo, Lyndon. "The Elements Of The Mobile User Experience – Smashing Magazine." *Smashing Magazine*. 11 July 2012. Web. 9 Dec. 2015. <http://www.smashingmagazine.com/2012/07/elements-mobile-user-experience/>.

"Multi-Screen Resources – Google." *Multi-Screen Resources – Google*. 1 Apr. 2014. Web. 9 Dec. 2015. <http://www.google.com/think/multiscreen/whitepaper-sitedesign.html>.

Rocheleau, Jake. "Responsive Web Layouts for Mobile Screens: Intro, Tips and Examples." *Hongkiat*. Web. 9 Dec. 2015. <http://www.hongkiat.com/blog/responsive-for-mobile-screens/>.

Jain, Richa. "7 Best Practices for Designing a Mobile User Experience." *RSS*. 8 Apr. 2015. Web. 9 Dec. 2015. <http://www.sitepoint.com/7-best-practices-designing-mobile-user-experience/>.

Bhatnagar, Samir. "Importance of User Interface | GovernmentCIO Magazine." *Importance of User Interface | GovernmentCIO Magazine*. 1 Dec. 2011. Web. 9 Dec. 2015. <http://www.governmentciomagazine.com/2011/12/importance-user-interface>.

Singh, Manpreet. "Mobile App Development Considerations | Sourcefuse Technologies." *Sourcefuse Technologies*. 9 Sept. 2014. Web. 9 Dec. 2015. <http://www.sourcefuse.com/mobile-app-development-considerations/>.

Wikipedia. Wikimedia Foundation. Web. 9 Dec. 2015. <https://en.wikipedia.org/wiki/Mobile_application_development>.

Menu Layout

Prince, Darryl. "Tap Vs Swipe: The Good, The Bad, and The Ugly - User Insight." *User Insight*. 8 Apr. 2014. Web. 9 Dec. 2015. <http://www.userinsight.com/tap-vs-swipe-good-bad-ugly/>.

"Menus." *Menus*. Web. 9 Dec. 2015. <http://developer.android.com/guide/topics/ui/menus.html>.

Sun, Terrence. "How to Write an Android App with Activity Lifecycle Function Examples." *How to Write an Android App with Activity Lifecycle Function Examples*. 18 Nov. 2013. Web. 9 Dec. 2015. <http://www.thegeekstuff.com/2013/11/write-an-android-app/>.

"Menu." *Menu*. Web. 9 Dec. 2015. <http://developer.android.com/reference/android/view/Menu.html>.