|  |
| --- |
| University of central florida |
| Knight Bright |
| Group #1 |
| ­ |
| **Robin Adams, Shaun Sontos, Tyler Hemp-Hansen, Nathan Doran** |
| **3/26/2013** |

|  |
| --- |
|  |

Table of Contents

[1 Executive Summary 1](#_Toc354406362)

[2 Project Description 3](#_Toc354406363)

[2.1 General Description 3](#_Toc354406364)

[2.2 Objectives and Goals 7](#_Toc354406365)

[2.3 Specifications and Requirements 8](#_Toc354406366)

[2.3.1 Physical Characteristics and Operation 8](#_Toc354406367)

[2.3.2 Internal Software 9](#_Toc354406368)

[2.3.3 External Software 10](#_Toc354406369)

[2.3.4 Hardware 10](#_Toc354406370)

[3 Research 12](#_Toc354406371)

[3.1 Existing Projects and Products 12](#_Toc354406372)

[3.1.1 Commercial Designs 12](#_Toc354406373)

[3.1.2 Freelance Designs (Hobbyists and Students) 15](#_Toc354406374)

[3.2 Methods and Architecture 24](#_Toc354406375)

[3.2.1 Software 24](#_Toc354406376)

[3.2.2 Hardware 28](#_Toc354406377)

[3.2.3 Sensor Addressing 31](#_Toc354406378)

[3.2.4 Wireless Communication 33](#_Toc354406379)

[3.2.5 LED Color Control Methods 43](#_Toc354406380)

[3.2.6 Processor 44](#_Toc354406381)

[3.3 Relevant Technology 46](#_Toc354406382)

[3.3.1 IR Sensors 46](#_Toc354406383)

[3.3.2 Power Supply 50](#_Toc354406384)

[3.3.3 USB 52](#_Toc354406385)

[3.3.4 LED Drivers 58](#_Toc354406386)

[3.3.5 Bluetooth Module 62](#_Toc354406387)

[3.3.6 Microcontroller Selection 66](#_Toc354406388)

[4 Project Hardware and Software Design Details 70](#_Toc354406389)

[4.1 Overall Design 70](#_Toc354406390)

[4.1.1 Printed Circuit Board 72](#_Toc354406391)

[4.2 Hardware 74](#_Toc354406392)

[4.2.1 Microcontroller 74](#_Toc354406393)

[4.2.2 Power Supply 74](#_Toc354406394)

[4.2.3 Sensor Input 77](#_Toc354406395)

[4.2.4 LED Output 83](#_Toc354406396)

[4.2.5 USB-UART Interface 86](#_Toc354406397)

[4.2.6 Bluetooth 87](#_Toc354406398)

[4.2.7 RGB LEDs 88](#_Toc354406399)

[4.3 Software 88](#_Toc354406400)

[4.3.1 Microcontroller 88](#_Toc354406401)

[4.3.2 Android 96](#_Toc354406402)

[4.3.3 Desktop Application 100](#_Toc354406403)

[4.3.4 Assembly 103](#_Toc354406404)

[4.3.5 Programming LED Controllers 105](#_Toc354406405)

[4.3.6 Android Interfacing 105](#_Toc354406406)

[5 Design Summary of Hardware and Software 108](#_Toc354406407)

[5.1 Hardware 108](#_Toc354406408)

[5.2 Software 108](#_Toc354406409)

[6 Project Prototype Construction and Configuration 110](#_Toc354406410)

[6.1 Bluetooth 110](#_Toc354406411)

[6.2 Android Interfacing 110](#_Toc354406412)

[6.3 USB 111](#_Toc354406413)

[6.4 GUI Interface 111](#_Toc354406414)

[6.5 Microcontroller Communication 111](#_Toc354406415)

[6.6 LED Controllers 112](#_Toc354406416)

[6.7 Sensor and Addressing 113](#_Toc354406417)

[6.8 Infrared Sensor 114](#_Toc354406418)

[7 Project Testing 115](#_Toc354406419)

[7.1 Power System 115](#_Toc354406420)

[7.2 Bluetooth 115](#_Toc354406421)

[7.3 USB 115](#_Toc354406422)

[7.4 Microcontroller Communication 115](#_Toc354406423)

[7.5 LED Controllers 116](#_Toc354406424)

[7.6 Sensor and Addressing 116](#_Toc354406425)

[7.7 GUI Interface 116](#_Toc354406426)

[7.8 Android Interfacing 116](#_Toc354406427)

[8 Administrative Content 117](#_Toc354406428)

[8.1 Finances & Budget 117](#_Toc354406429)

[8.2 Milestones 118](#_Toc354406430)

# Executive Summary

After much consideration, the group has chosen to design and develop a tabletop interactive LED system. The primary motivation behind this project is to develop a fun, easy to use, user-programmable interactive tabletop. In doing this the group also hopes to develop a prototype that takes the typical hobbyist LED project to a higher level, showing the functional capabilities of an interactive user interface system as well as a fully integrated programming environment. Along with this primary motivation, the group hopes to display a solid understanding of the fundamental principles which have been studied by all of its members in the past several years. This will be done by drawing on the strengths of each of the group members and offering a collaborative and cooperative learning experience. The project will also serve as an opportunity, within reason of the overall design specification, to introduce or expand on exposure to unfamiliar materials which any of the group members seeks.

The general functionality of the system is an interactive LED experience with gaming, text, and other visually stunning modes. The experience will begin with the opportunity to choose either a simple and easy to use programming environment, or the selection of one of several preprogrammed modes. This will be achieved by use of a scrolling text menu on the device. For the programming mode, the group intends to generate an easy to use pc based programming environment which will give the user access to a list of predefined commands. These commands can be used to create a user generated program. When the device is placed in programming mode the user will be able to load a program from the pc based development environment to the device. For gaming mode, the user will be able to select one of many preprogrammed games to be played against the device or against one or multiple other players. This mode will allow interaction directly with the device through built in sensors or interaction through the user’s Android based mobile device. For the built in sensors the group has chosen to use infrared emitters and detectors which will provide the feeling of a capacitive touch screen experience to the users at a greatly reduced cost. Possible game modes discussed are a ‘Battleship’ style game, ‘Reversi’ style game, ‘Connect 4’ style game, ‘Missile Command’ style game, a general paint pallet, and many others. Based on the memory and I/O limitations of the system the group has also considered a simple game mode expansion feature which would allow the ‘purchase’ of new games as they are developed.

Due to the financial and spatial limitations of the project the group will have to develop addressing schemes to allow selection of any of the available coordinates (addresses) on the device as well as for the purposes of displaying the devices output. It will also be necessary to allow selection and display of multiple colors which the system will have to recognize and process. This will involve the use of encoding and decoding between the user interface and the processor and the use of LED drivers. This will also be dependent on the development and use of a pulse width modulation, which will generate an array of colors from basic RGB (red, green, blue) LEDs.

Other key components of the project which will be outlined in this document are the implementation of a Wi-Fi device to allow communication between the project and the user’s mobile Android device. The group will also have to develop the Android application itself which should be simple, yet completely functional. Also, the project will require integration of USB or some other communication tactic to the developers of the device to program it and to allow the user to program the device through their personal computer. Finally, the group will be required to develop a power distribution plan to not only provide power to the device components, but also for the sake of communication signals between the components.

# Project Description

## General Description

The project will be an interactive LED gaming and display board. The board will consist of 100 individual cells enclosed within an attractive case which will allow light to pass through the top surface of the board. This surface will be either a translucent white or lightly tinted black thin plastic. This will act as the gaming surface for all game modes and the display surface for all additional features. The device will have three primary modes of operation. The gaming mode will allow the user to select from one of many games to be played on the device. Display mode will allow the display of several visually stunning programs. Finally, programming mode will allow the user to program the device themselves through the use of a personal computer loaded with a simple, programming environment which will be created by the group.

The device hardware will consist of three different layers which will each need to work together to ensure proper functionality of the device. These layers are the individual unit cells, the communication layer, and the central control system. Figure 2.1.A below is a hierarchal view of these systems.

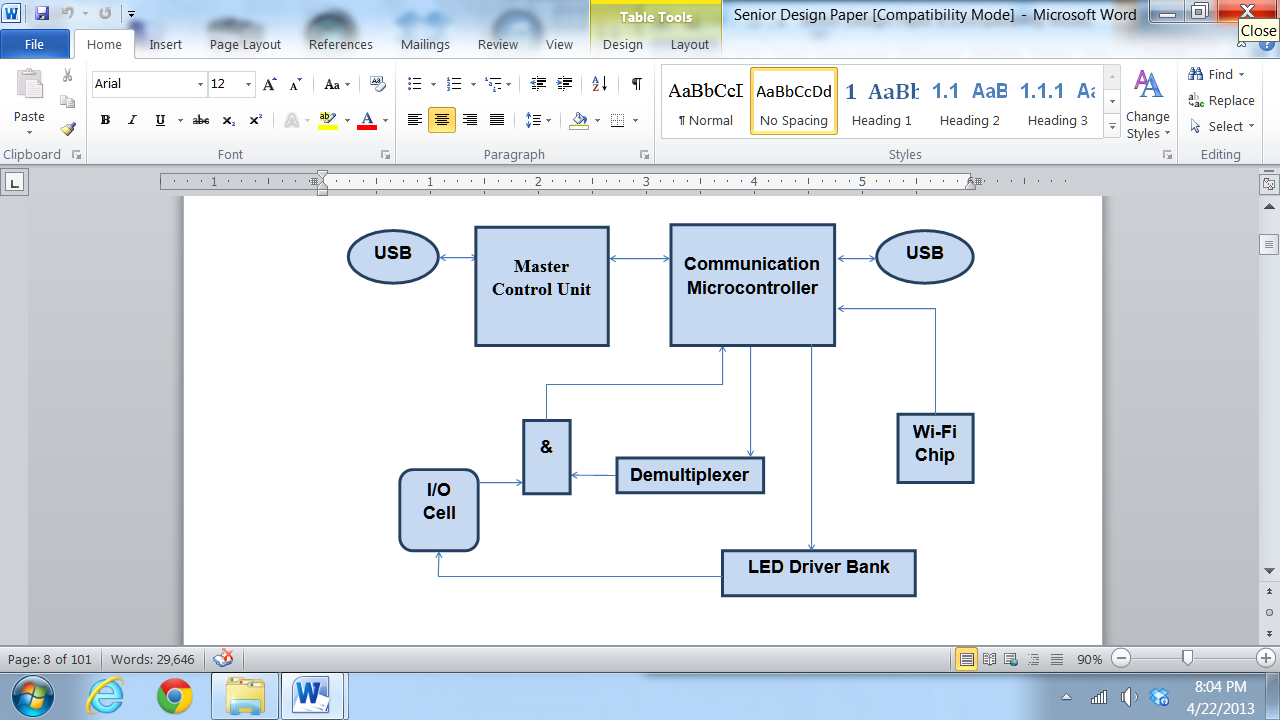


Figure 2.1.A - Hardware Hierarchy

Each cell on the device will be approximately two inches square and will contain the input and output interfaces of the device. To allow input from the user an infrared emitter and detector circuit will be used which will act as a proximity sensor to detect the users hand or any other object placed atop the cell. As an additional feature, the device will also have a dedicated Android based program which will allow input from the user’s Android device. This will allow simple and clear control of certain games and features wirelessly through the device. Additional means of input will be a USB connection to allow communication for the devices programming mode. For output, each cell will contain one red, green, blue (RGB) LED. The LED will generate a multitude of different colors and flash sequences to communicate with the user.

Beyond this interface layer will be a communication layer which will transmit data from the user to the master control unit. For the input, the group intends on using an innovative addressing technique which will allow rapid reading of all 100 inputs through the use of demultiplexers and logical ‘and’ gates controlled by a dedicated communication microcontroller. For control of the LEDs, drivers will be used in conjunction with the communication microcontroller. These drivers will allow the use of pulse width modulation to extract an array of colors and pulses from each LED. To allow communication between the board and an Android device a Wi-Fi chip will be included in the board. This chip will also be tied to the communication microcontroller. Additionally, a USB interface will be tied to this controller for programming the microcontroller.

At the heart of the device will be a master control unit. This unit will contain the primary code necessary to start up the device and run any features or preprogrammed code. This device will be directly connected to the communication microcontroller to directly communicate with it. This master control unit will have two USB interfaces, one for programming the microcontroller and the other to use by the user to upload programs. The development USB will be enclosed within the device as it is intended to only be used by the group for the sake of development and design. The other will be easily accessible by the user to directly connect to a PC running the group’s GUI programmer software.

To drive this hardware system the device will have an extensive software package that will not only control the device, but also allow the addition of several impressive features. The software package will include the master control code which will allow the hardware to be implemented and include games, visual displays, and programmable memory. In addition to this the software package will include an Android based control system, and a programming environment which will allow the user to program the device from their personal computer.

The main code in the device will be responsible for allowing the integration and interfacing of all hardware systems. This will include processing the inputs and generating the outputs to drive the LED controllers. This will also include allowing communication through the Wi-Fi device and access to the user programmable memory through USB. This main code will also be responsible for calling and executing the games and visual display included with the device from the device memory.

For the gaming and display modes the group hopes to include several preprogrammed games and visual displays. This will begin with a home screen list allowing the selection of game mode or display mode. From here the user will be directed to the appropriate list of either games or displays. In game mode the user the choice of games similar to “Battleship”, “Reversi”, “Snake”, “Connect Four”, air hockey, tic-tac-toe, and many others. The user will also have the choice of entering into a paint mode which will allow the selection and placement of different colors on the board. In display mode the board will act as an interactive display unit which will perform displays to run color streams, display moving objects, ripple, display text, and many others. The displays will not only run, but also respond to user interaction through the input sensors. For both the gaming mode and the display mode the user will have the option of using a wireless remote controller. To do this there will be an Android application developed by the group which will allow the use of an Android device as a controller. From the application the user will have control of basic directional motion (up, down, left, and right) as well as one or two select buttons. The user will also have an available text keyboard in the application which will allow user to display text on the board.

The final item in the software package for the board will be a programming environment which will allow the user to program the board. To do this the group will develop a programming environment which will allow the use of a predetermined set of commands. To generate this list the group will generate its own syntax and semantics. The programming environment will be loaded and run on the user’s personal computer, and will load the final program via USB connection to the device. Once loaded the micro control unit will store the users program in a segment of programmable memory and then execute the program written by the user. This function will be completely separate from the primary code in the microcontroller to prevent user interruption with the overall operation of the device. Figure 2.1.B below is a hierarchy of the software functions.

Figure . - Software Hierarchy

**Programing Environment**

**Primary Operation Code**

**Android Application**

**Programming Mode**

**Game Mode**

**Display Mode**

**Game Menu**

**Display Menu**

Diagram 2.1.B Software Hierarchy

## Objectives and Goals

Prior to the publishing of this paper, the group has conducted extensive research on all elements of the project. This research is included in the research portion of this document (Section 3). The purpose of conducting this research is so that the group members could understand the requirements of meeting the specifications outlined above. By conducting this research the group has been able draft an initial design for the project. Also, the group was able to make well informed selections of the many of the components needed to complete the project. The initial design and selected components are discussed in detail in the design portion of this document (Section 4).

In addition to this the group hopes to meet the following technical goals as a means of progressively achieving successful completion of the overall design. These goals and objectives are based on the specifications outlined above. This will serve as a template for monitoring progress toward overall completion of the project.

* Gain real world experience in design and implementation of a complex electrical device
* Gain real world experience in the selection and integration of multiple commercially available parts
* Obtain a firm understanding of the selected microcontroller and how to program it
* Obtain a firm understand of the selected LED driver and how to program them
* Use the microcontroller to control an array of LEDs through the selected LED driver
* Read the output of a sensor circuit into the microprocessor
* Successfully integrate Wi-Fi capabilities into the device
* Successfully integrate USB communication on both the user and developer levels
* Construct a functioning prototype of an infrared sensor circuit
* Gain real world application experience with multiple addressing schemes and related hardware and software tactics
* Generate code that uses the inputs and outputs in an intelligent fashion
* Develop a unique programming environment
* Develop an Android based application
* Design an application specific power distribution system
* Learn to use a printed circuit board design tool
* Design a printed circuit board which will house the main control circuit for the device

## Specifications and Requirements

The following specifications list will serve as a general outline for successful completion of the project. These items may be modified or removed as deemed necessary by the group, and additional specifications may be added if time permits.

### Physical Characteristics and Operation

The first set of specifications deals with physical limitations of the mechanical aspects of the device.

* Overall dimensions should not far exceed 20” x 20” x 2”
* The device will be relatively lightweight and easy to transport
* The outer package will be attractive and durable
* The top surface of the device will be removable allowing access to the input/output circuits within
* The top surface must allow visible light and infrared light to pass through it
* The top surface should diffuse the visible light generating the appearance of a square cell
* All other control circuits will be easily accessible
* Each cell will be approximately two cubic inches and contain one input/output circuit
* User interaction will be through sensor circuits contained within each cell or through a wireless Android Device
* Device display will be through RGB LEDs located on the each of the input/output circuits
* All input and output will be read or displayed in an acceptably timed manner, the device will not appear to possess excessive operational lag
* There will be one USB connection made easily available to the user for connection during user programming
* Wiring and connections will be made in a neat and professional manner
* The device will receive power from a wall outlet via a plug which will be safely installed and firmly secure to the device in order to prevent possible electrocution

### Internal Software

These specifications define the software used on the Knight Bright itself.

* General
  + An operating system will not be required for this project
  + The device will be preprogrammed with all software necessary to support functional operation and interfacing with all included hardware
  + The device will have three modes of operation Display Mode, Game Mode, and Programming Mode
  + Selection of these modes will be through a text menu displayed on the surface of the board
  + All subsequent menus will be displayed via scrolling text menus displayed on the board
* Display mode
  + Display mode will contain at least three visual displays
  + One of these displays will allow the user to input text which will then scroll across the board
  + The others will be visually stunning displays which will respond to input via the input circuit in each cell
* Game Mode
  + The device will be preprogrammed with at least three games
  + One will be a canvas board style game which will allow the user selection of a palate of colors displayed on the board which can then be placed on any of the remaining cells
  + Another game will be a puzzle style game with device generated puzzles which must be solved by the user
  + At least one of the remaining games will be playable against the device’s microprocessor with preprogrammed game logic, or against another player
* Programming mode
  + Enables the user to interface and upload programs from the GUI programmer.

### External Software

The next list will define the specifications for the mobile application as well as the user programmable environment.

* Android Application
  + An android application will be developed which will allow the user to control the unit from their wireless Android based device
  + The display will be easy to understand and use, and will attractive and professional in appearance
  + One feature of the application will allow the user to input text which will be displayed when the device is in text display mode
  + Another feature will be a game controller which will, at a minimum, contain up, down, left, right, and one select button
* User Programming Environment
  + The device will have available EEPROM memory which can be used to allow the user to upload their own programs and execute their code on the device
  + The group will develop an easy to use programming environment
  + The environment will be attractive and professional in appearance
  + The software will include a predetermined list of commands available to the user
  + The programming environment will be run on the user’s PC
  + The program will be loaded onto the device via a USB connection
  + Once loaded the program will be runnable on the device
  + This mode will not allow interference with or destruction of the primary device operation

### Hardware

Finally these specifications below deal with the computer and electrical hardware aspects of the device.

* The device will be primarily controlled by a master control microprocessor which will store and execute the primary operation software
* A second microcontroller will be used to interface with the master controller with all input/output devices
* The output will be displayed via 100 individually controllable RGB LEDs
* A minimum 16 scale color wheel will be used
* The LEDs will be driven by dedicated controllers to limit strain on the processor and improve overall processing speed
* The input will be taken through 100 individual infrared sensing circuits
* Multiplexers and demultiplexers will be used accordingly on inputs and outputs to limit the number of connections to the microprocessor
* The device will be Bluetooth enabled through the use of a Bluetooth chip and affiliated circuit to allow connection to the users Android based device
* A USB connector will be available on the PCB for programming each of the microcontrollers
* A USB connector will be available on the surface of the device with a proper circuit allowing the user to communicate data during programming mode
* A commercially available power supply will be used which meets the power consumption requirements of the device
* A power control circuit will be generated which will prevent damage to the device in the event of overload or other fault, and include appropriate power distribution to the device components

# Research

## Existing Projects and Products

The existing project designs that are referenced in this chapter of the report will appear in the following sequence: First, the projects will be partitioned between commercial and hobbyist/freelance designs. Second, project designs that are based on a 3D design will be referred to first, and 2D designs will appear after them. Third, projects that have no interactive features (animation-only) will be mentioned first. Projects that have interactive features will appear last.

Originally, the group was split between 2 design schemes. Half the group wanted an interactive 3D LED cube, and the other half wanted an interactive 2D LED array panel. To determine whether the group wanted to go with a 3D cube or 2D panel design, the following commercial and hobbyist designs were sought out and cataloged. Most of the projects referred to in this section will be non-interactive. However, these animation-only projects still provide some unique design features that will be put into consideration by the group.   
  
There is an abundance of dynamic and/or interactive LED animation design projects on the internet. Most projects utilize a 3D cube design (with an (8x8x8) or (4x4x4) 3D LED layout being the most popular). Another popular design scheme is the 2D LED panel array layouts. Also, there is a project that utilizes a POV DC motor design with an onboard integrated LED card. A few of these projects are mass-produced commercial designs. However, such commercial designs tend to have fewer features, and are of a lower magnitude scale in design, when compared to hobbyist/freelance designs. The overwhelming majority of LED animation projects are designed by hobbyists and university students.

### Commercial Designs

These commercial designs have been chosen because they have at least 1 essential feature that is of interest to the group’s project design. Unfortunately, most of these projects will only have a portion of their technical specifications listed.

#### Cube– Hypnocube LLC

The 4Cube (http://hypnocube.com/product/4cube/) is the flagship product of Hypnocube LLC. The 4Cube is a 3D cube composed 64 RGB LEDs arranged into a 4x4x4 cube layout. There are 2 configurations of the 4Cube that are available on Hypnocube’s website. The first configuration allows the user to program their own custom animations from a USB-enabled personal computer. Furthermore, you have the option of either buying a finished pre-assembled design, or of buying a build-your-own kit. The USB-enabled 4Cube is shown in Figure 3.1.A.The USB-enabled 4Cube costs $275 pre-assembled, and $150 for the build-your-own kit. The second configuration of the 4Cube is the same design as the first configuration, but without user customization. There is no USB-port, no user-generated animation capabilities, and it does not come as a build-your-own kit. This scaled down 4Cube costs $100. This cube exemplifies the most basic features of a 3D LED that the group is interested in. However, if the group were to implement a 3D design, it would either be an (8x8x8) or (10x10x10) configuration. Furthermore, the group’s design would be interactive, and not animation-only.

Technical Specifications:

* 16 (24) levels per color channel.
* PIC18F4620 Microcontroller unit (32 MHz clock)
* 64 kB ROM + 3.8 kB RAM memory
* C programmable

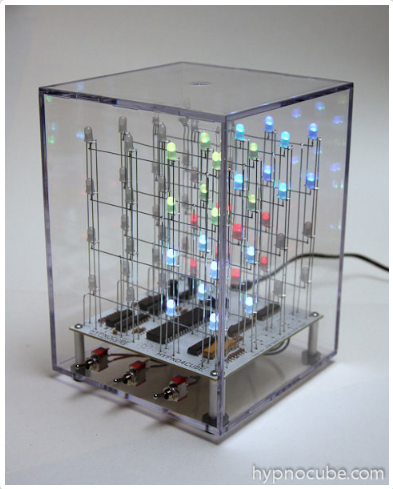


Figure 3.1.A - USB-enabled 4Cube – Hypnocube LLC

#### Interactive LED Panel – Evil Mad Scientist

Evil Mad Scientist (http://shop.evilmadscientist.com/directory) is the designer of the Interactive LED Panel (ILP). The ILP is a (1 ft x 1 ft) square panel with 80 white LEDs (arranged in a uniformly distributed array) and 4 motion sensor nodes attached. Each motion sensor node controls 40 LEDs. An ILP can be connected in series with any *n* number of ILPs. However, these panel permutations are limited by a 24 VDC, 2.5 Amp power supply. This PSU can supply up to 12 ILPs. When a person’s hand or an object is waved over the panel, it will cause LEDS that are immediately adjacent to the triggered motion sensor to turn on. Furthermore, adjacent LED-motion sensor panels will start to flicker on and off, generating a wave-like light motion along the entire panel.

The ILP comes in 3 different design configurations. First, there is the “Soldering Kit” ILP. The “Soldering Kit” is a build-your-own kit for the ILP, and comes included with circuit boards, components and instructions on how to build the ILP. The 24 VDC power supply must be bought separately from the “Soldering Kit”. The ILP “Soldering Kit” costs $85 for 1 panel all the way up to $580 for 12 panels, with the cost-per-panel decreasing as you buy in bulk. The second design option is the “Ready-to-Use” ILP. These ILPs come pre-assembled, and cost $130 for the first panel. The “Ready-to-Use” ILP is shown in Figure 3.1.B. Evil Mad Scientist sells 24 VDC, 2.5 Amp power supplies for $45 each.

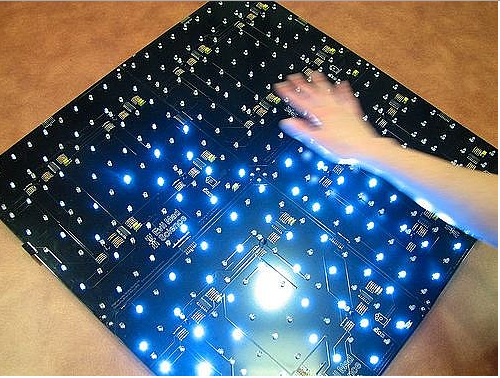


Figure 3.1.B - Four ILPs connected into a (2 x 2) module

Finally, there is the “Game of Life” ILP. The “Game of Life” ILP consists of a (4 in x 8 in) panel board with 8 white LEDs arranged in a (2x4) array, and each LED is connected to an IR proximity sensor. Like the “Soldering Kit” and “Ready-to-Use” ILPs, the “Game of Life” ILP can be interconnected to form (n x n) modules. The “Game of Life” ILP requires a 5 VDC, 1 Amp power supply unit. The “Game of Life” ILP, as per it’s namesake, is programmed to run John Horton Conway’s famous “Game of Life”. “Game of Life” is a zero-player game, and the objective of the game is to display the evolution of motion of LEDs after an initial condition(s) has been inputted. The “Game of Life” ILP costs $34.95 per unit. Evil Mad Scientist sells a 5DC, 1 Amp power supply for $7.50 each. The “Game of Life” ILP is shown in Figure 3.1.C. All 3 ILP designs and configurations are the most basic examples of LED control interaction that the group wishes to utilize, whether or not the final project turns out to be a 3D cube or a 2D panel.

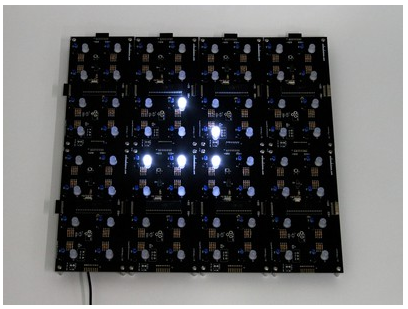


Figure 3.1.C - “Game of Life” ILPs arranged into a (2x4) module

### Freelance Designs (Hobbyists and Students)

Freelance designers tend to be more open about the technical specifications and features of their design projects. Commercial projects were also (intentionally) restricted in scale and limited in features, in order to be mass-produced and profitable. Freelance designs, on the overhand, have more features, and tend to be much greater in scale. Freelance projects provide the most reliable template for the group to base the senior design project on.

#### Dynamic Animation Cube – UCF Senior Design Group 1

The Dynamic Animation Cube (DAC) is a (16x16x16) RGB LED Cube designed by Group 1 (Joseph Clark, Michael Alberts, Isaiah Walker, and Arnold Li) of the Spring-Summer 2012 UCF Senior Design Class. All DAC related documentation can be found at (http://www.eecs.ucf.edu/seniordesign/sp2012su2012/g01/). Although the magnitude of the DAC’s design is greater than most, it is nevertheless a good design template that the vast majority of 3D LED cube hobbyist designs adhere to. The Dynamic Animation Cube is shown in Figure 3.1.D.

Technical Specifications:

* Stellaris LM3S8962
  + 32-bit ARM Cortex-M3 50-MHz processor core
  + 256 KB flash and 64 KB SRAM
  + 42 GPIOs
  + Bit-Banding
  + UART
  + Synchronous serial interface (SSI)
* TLC5941
  + 12-bit PWM
* SD Card
* Stellaris LM3S8962 GPIO layer selection
* 24 fps

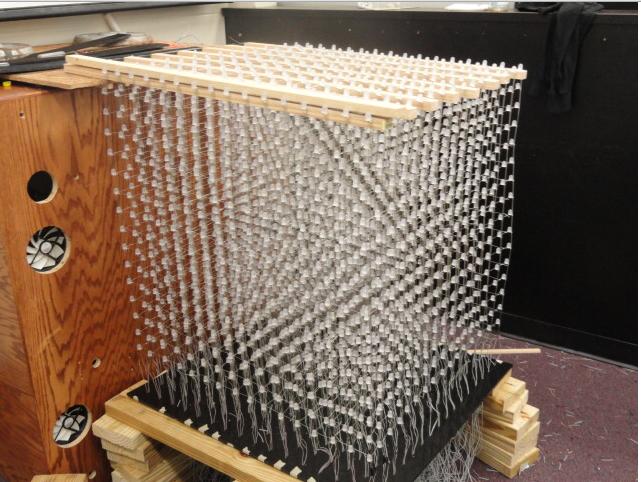


Figure 3.1.D - 16x16x16 RGB LED Cube – Group 1 (Senior Design Summer 2012)

The DAC can be thought of as a superposition of 16 layers of (16x16) RGB LEDs. Furthermore, because a single RGB LED is really a superposition of 1 red, 1 blue, and 1 green LED, the DAC can be thought of as a superposition of 3 (16 x16) LED arrays, 1 for red, 1 for green, and 1 for blue. These 3 arrays are then multiplexed 16 times, each, for the added 3rd dimension. This gives a total of 4096 RGB LEDs, which are then really 12288 individual LEDs. To control all these LEDS, the DAC group used 16-channel, 12-bit grayscale, PWM LED drivers. Each individual LED (red, green, or blue) is terminated to each channel of the LED driver. The LED drivers are then connected in cascade to allow a single row of drivers to receive 1 control line. All the LED drivers were connected onto 1 PCB board.

Using the pervious LED driver scheme allows a single (16x16) array of LEDs to be controlled at a time. Since there are 16 arrays stacked on top of each other, the DAC group used the MCU to control the 3rd dimension of the LED cube. 16 GPIOs on the MCU were used to multiplex the (16x16) array from the LED driver PCB, and the output would be sent to another PCB consisting of 16 transistors. The sprite or function animation files which hold the animation programs would be stored on an SD card. Once the animation program is loaded onto the SD card, the ISR calls interrupts to the MCU, which then serially transmits the grayscale of each LED onto the LED driver, and the layer selection to the GPIO of the MCU. This process is repeated throughout the length of the animation cycle. If the group decides to implement a 3D cube design, many of the features found in the DAC would also be used in the 2013 group’s interactive cube.All of the UCF DAC Group’s LED Cube animations can be found on their senior design website: (http://eecs.ucf.edu/seniordesign/sp2012su2012/g01/photo.html).

#### 8x8x8 RGB LED Cube – How Not To Engineer

Nick Schulze of “How Not To Engineer” (www.hownottoengineer.com) designed this popular RGB LED Cube. The Cube is made up of 512 RGB LEDs arranged in 8 layers of (8x8) arrays. The vast majority of LED animation design projects use PWM to control the dimming of an LED array. PWM can be achieved by terminating the LEDs with a driver chip, or by terminating the LED with an MCU. What’s special about Schulze’s design is that it does not utilize PWM. Instead, Schulze uses an 8-bit “Bit Angle Modulation” technique. The theory beyond BAM will be elaborated upon in this report, but for the purposes of Schulze’s design, the group can say that BAM significantly reduces the processing power on the microcontroller when compared to 8-bit PWM. The (8x8x8) HNTE RGB LED Cube is shown in Figure 3.1.2.B.

Technical Specifications:

* + RGB LED, diffused x512
  + STP16 LED driver x12
  + ChipKIT UNO32 x1
  + 5V, 60W switched mode PSU x1
  + 5v-3.3v DC-DC converter x1

Schulze wrote all his source code from scratch in C and C++. The multiplexing and BAM fading of the LED cube are executed in the ISR of the MCU. Schulze also wrote a Color Array program that generates opaque (less washed out) colors for the LEDs. As it stands right now, the group has shown preference for PWM using LED drivers. However, the group can instead opt to control their LEDs using BAM if there were to be any issues in the design project such as LED skipping, dimming, or MCU processing issues with PWM drivers. All of Nick Schulze’s LED Cube animations are on his YouTube channel: (http://www.youtube.com/watch?feature=player\_embedded&v=5Ip2\_RDZ5jE).

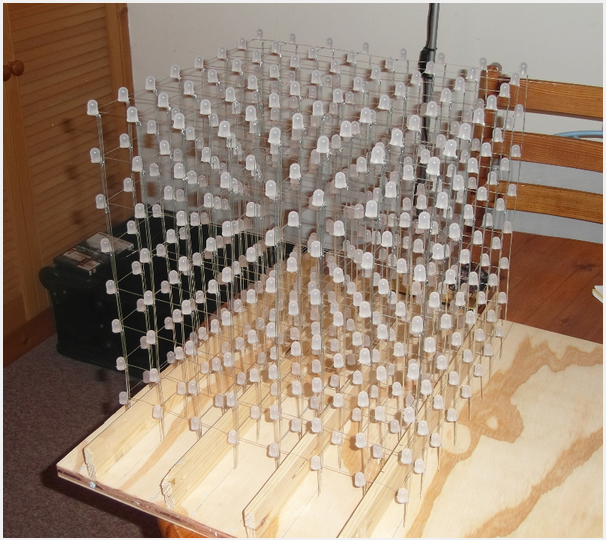


Figure 3.1.E - 8x8x8 RGB LED Cube – How Not To Engineer

#### 8x8x8 RGB LED Cube (Interactive) – Ohm’s Projects

Greg Brault of Ohm’s Projects (http://ohmprojects.com) designed this (8x8x8) RGB LED Cube in early 2012. This cube’s LED display is controlled by 8-bit PWM, 16-channel, LED drivers. Like the DAC group’s LED cube, and the HNTE LED cube, all the PWM drivers are integrated onto 1 PCB board, and the MCU and transistors multiplex the cube’s layers from a 2nd PCB. However, unlike the DAC or the HNTE cube, Greg Brault’s cube is interactive. The cube has a game control peripheral composed of an NES controller with ports reconfigured with USB.

Unfortunately, the components and specific technical specifications are not provided on the Ohm’s Projects website (http://ohmsprojects.com). The design of Greg Brault’s 8x8x8 is shown in Figure 3.1.2.C. This NES controller has 8 outputs that are terminated to an 8-bit USB MCU (SiLabs c8051F320.) These 8 outputs are controlled using the Human Interface Device (HID) class feature of USB. 3 Bytes are used to control all 8 outputs; 1 Byte for the x-axis(2 ouputs, 1 for left and 1 for right) of the D-pad control, 1 Byte for the y-axis(2 outputs, 1 for up and 1 for down) of the D-pad control, and the last Byte is used to denote the state of the remaining 4 outputs (“Select”, “Start”, “A”, and “B”.) The USB MCU then outputs the data commands serially over a 3-wire interface. If the group decided to implement a 3D LED Cube model for senior design, Greg Brault’s cube design exemplifies some of the interactive features that would be implemented. A control peripheral that is programmed and terminated by means of USB is one possible outlet of command for such an interactive 3D cube. What the group is most interested in, however, is something more akin to the touch-sensor activated LED controlling in Evil Mad Scientist’s ILP (see section 3.1.1.2). Greg Brault’s interactive 8x8x8 LED Cube can be viewed on his YouTube channel: (http://www.youtube.com/watch?v=rqknc9Df7Lw).



Figure 3.1.F - 8x8x8 RGB LED Cube (Interactive) – Ohm’s Projects

#### Propeller Display (RGB) - Ohm’s Projects

Greg Brault’s (8x8x8) LED Cube is the last 3D design that will be covered in this chapter. But he is also the designer of this 2D propeller LED design. While weighing the options of a 3 dimensional vs. 2 dimensional design, the group came across this dynamic POV propeller display. The propeller display is composed of an integrated circuit board (with onboard LEDs) that’s mounted onto a DC motor, which then rotates the board. The board is encased in a glass display box.

Technical Specifications:

* 32 RGB LEDs (3 \* 32 = 96 total LEDs )
* 50 Hz with a DC motor
* The propeller is controlled by a SiLab c8051F350 8-bit MCU
* Image data files are stored on a 125 kB EE-PROM (25LC1024).
* The LEDs are controlled with twelve 8-channel constant current LED drivers (PCA9922).

The propeller display suffers from some apparent design restrictions (only a finite number of LEDs can be mounted onto the DC motor at a time.) Despite this setback, a surprisingly vibrant spectrum of color and animation can be achieved. However, it would be impossible to incorporate a touch-sensor interactive feature into this propeller display. Although the POV display is a novel approach to LED display, it most likely will not be implemented by the group due to the display’s restrictive interactive features. The propeller display is shown in Figure 3.1.G. Greg Brault’s RGB POV Propeller Display can be viewed on his YouTube page: (http://www.youtube.com/watch?v=QAGnpKz7zvY)



Figure 3.1.G - Propeller Display (RGB) – Ohm’s Projects

#### RGB LED Coffee Table – Texas Instruments Microcontroller Project

Ian Cathey, Mark Labbato, and Max Thrun designed this animation-only RGB LED coffee table for the 2011 Texas Instruments Co-Op Design Challenge. The user plays music on their PC, and a Python program takes the FFT of the audio files beats, and sends the data through the USB onto the coffee table, where it is converted into LED displays. All references and technical documentation related to their project can be found at the following site: (http://e2e.ti.com/group/microcontrollerprojects/)   
  
Technical Specifications:

* TI LaunchPad MSP430G2553 x1
  + 16 MHz
  + 512 B SRAM and 16 kB Flash Memory
  + UART enabled
* TSP7333 5-to-3.3 V LDO Regulator x1
* TLC5940 16-Channel 12-bit LED Driver x8
* CD74HCT126 three-state buffer(4 on-chip) x1

The coffee table is comprised of an (8x16) RGB LED array that is controlled by 16-channel, 12-bit PWM drivers. The group designed a Python module (“pytable.py”) to allow for PC-to-table communication via the UART. The team developed C programs to take the FFT of music and to query audio beats. The PCB for the RGB LED Coffee Table is shown in Figure 3.1.H. The RGB LED Coffee Table itself is shown in Figure 3.1.I.

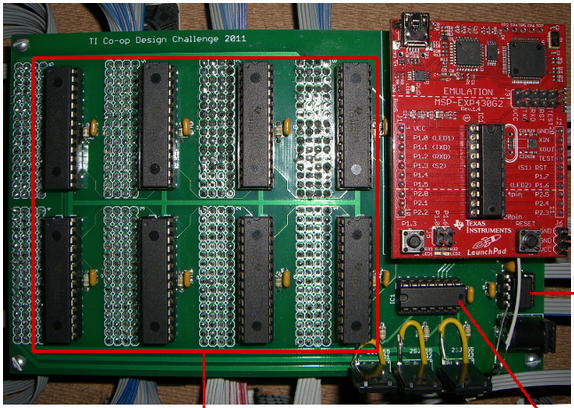


Figure 3.1.H - RGB LED Coffee Table PCB – TI MCU Project



Figure 3.1.I - RGB LED Coffee Table – TI MCU Project

#### 100 Pixel RGB LED Table – IT-Gecko (Leipzig, Germany)

Julius Fischer of IT-Gecko designed a 2D interactive (10x10) RGB LED coffee table. The summary and related documentation for Julius Fischer’s project can be found at his website: (http://www.it-gecko.de/100pixel-rgb-led-tisch-interaktiv-touch.html). This coffee table is comprised of 100 individual IR touch sensor LED circuit “cells”. Each “cell” consists of an IR touch sensor, an IR LED, and an RGB LED. All 100 “cells” will then be terminated onto a common PCB. The LEDs are controlled using 12-bit PWM LED drivers. The LED Table can be programmed with many various interactive games, which are stored onto the MCU. The board peripherals are arbitrarily designated LED touch cells that run along the perimeter of the 10x10 array. The LED Table also has a BTM-222 Bluetooth module. Julius has programmed an Android app that interacts with the LED Table’s Bluetooth module, allowing for Android devices to act as a control peripheral. Finally, Julius has also designed a Java app on his PC that allows for programmable animation/audio design with the LED Table (similar to Microsoft Paint.)

Technial Specifications:

* ATxmega256A3 x1
  + 32MHz clock
  + 16 kB RAM, 256 kB Flash memory
  + USART port
* TLC5940 (12-bit PWM Driver) x19
* Sharp IS471F (IR Sensor) x100
* BTM-222 Bluetooth Interface x1
* 60W PSU x1

There are five games that the interactive coffee table can perform. These games are Tetris, Snake, Tic Tac Toe, HSV color model, and Draw. The games are written in C, and saved directly on the MCU. The animation design app on the PC and the peripheral control app on the Android smartphone are written in Java. The 100 Pixel LED Table (playing a game of Tetris using an Android smartphone as a controller) is shown in Figure 3.1.J. The IR touch sensor, and LED “cells” are shown in Figure 3.1.K. Julius Fischer’s 100 Pixel RGB LED Coffee Table and it’s interactive gaming features, such as a GUI animation program, touch-sensor control and Bluetooth enabled control peripherals were what best reflected what the group was seeking to implement in the senior design project. All of Julius Fischer’s LED Table game/animation videos are on his YouTube channel: (http://www.youtube.com/watch?v=DTb0k\_P1wlY).

#### 

Figure 3.1.J - 100 Pixel RGB LED Table

**(with the Android phone peripheral enabled) – IT-Gecko**

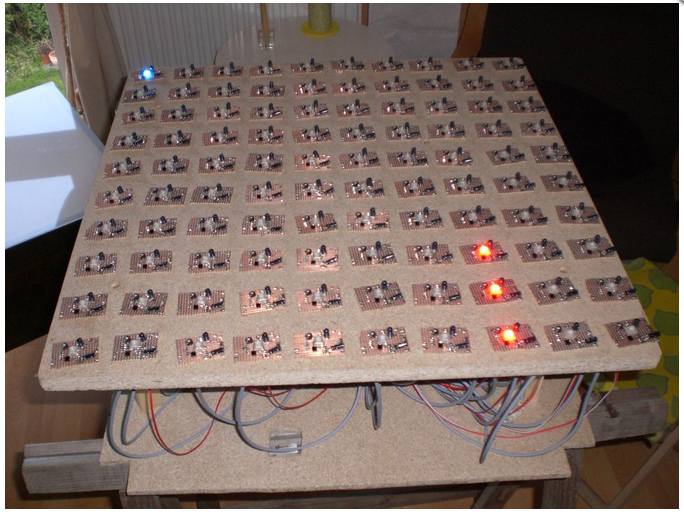


Figure 3.1.K - 100 Pixel RGB LED Table (all unit sensor cicuits shown)

## Methods and Architecture

This section compares and describes various control methods and architecture schemes available to meet the design specifications. Many of these method comparisons were used to help the group decide which relevant technologies could be considered for the project and ultimately which devices will be chosen for the design. This section focuses on high-level design methods as applicable to specific components which will be discussed in the following section.

### Software

#### Microcontroller Programming

There are several options to program the group’s Atmel microcontrollers. The two most common are the Arduino integrated development environment (IDE) and Atmel Studios. Both environments have the ability to program in an ANSI-C based code. Both environments have built-in support for this particular microcontroller, and avoid 3rd party software that isn’t directly aimed at this microcontroller.

The Arduino IDE comes with several libraries that have a lot of support, due to the abundant availability of Arduino development support. It comes with a way to flash the bootloader on an Atmel chip, and can be easily used for the group’s purposes. A screenshot of this development environment can be seen in Figure 3.2.A.

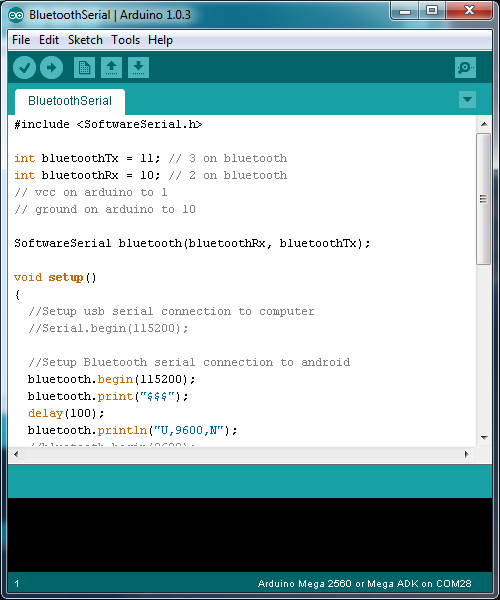


Figure 3.2.A - Arduino Integrated Development Environment (IDE)

An alternative microcontroller programming environment is Atmel Studio (shown in Figure 3.2.B). This software, provided by Atmel, gives several similar features to that of the Arduino IDE, including a built-in boot-loader and open-source libraries. This IDE is based in a Visual Studio shell, so it gives the appearance that is more familiar to those who have used Visual Studio in the past.

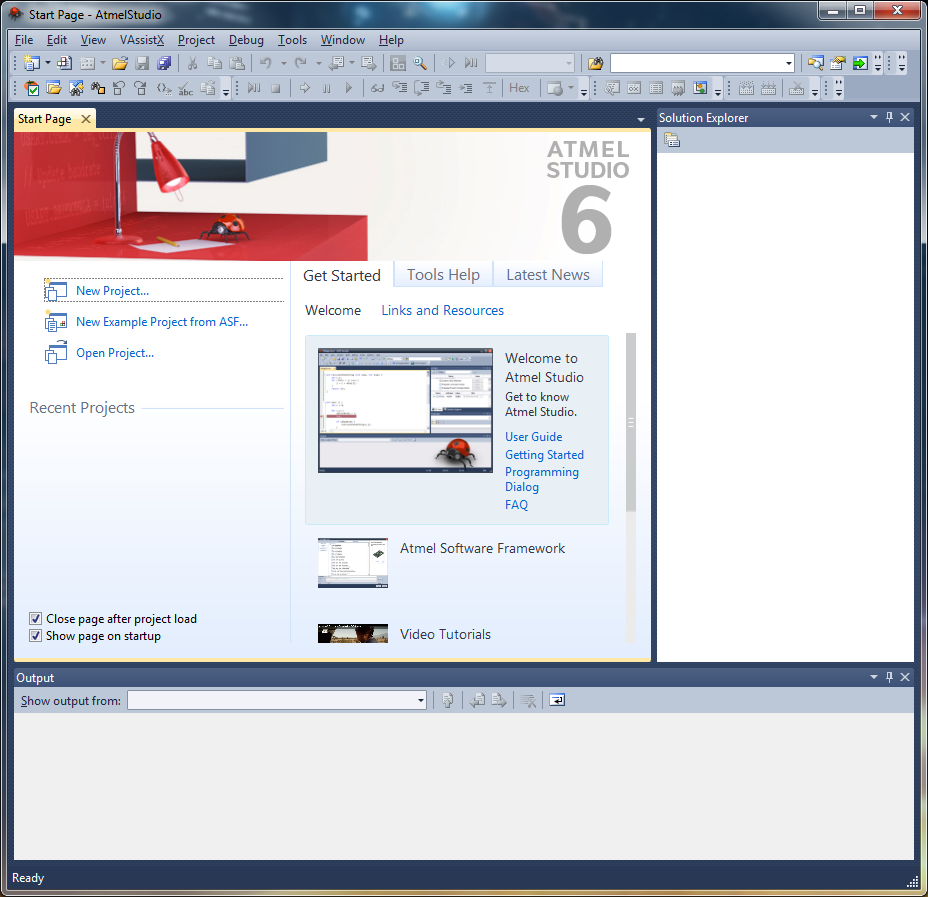


Figure 3.2.B - Atmel Studio

While both options can be used in a relatively similar fashion, the group have decided to use the Arduino IDE because it is more familiar to the group from programming Atmel microcontrollers on Arduino boards with the boot-loader already installed. This makes an easier transition to the group going from prototyping to having standalone microcontrollers on the group’s printed circuit board (PCB).

#### Android

There are two major IDEs that are used for android development, Netbeans and Eclipse. The main difference between these two IDEs is their backing. Eclipse is backed by IBM while Netbeans is backed with Sun/Oracle environment. This means that each IDE has different support. In the late 1990s and early 2000s, IBM developed and released a better product more quickly than Sun could, making it so their support grew before Netbeans could get off the ground as a pure open source project. By 2003, Eclipse had a large following in the IBM community. Therefore, Eclipse has much more plugin support, making it more versatile than Netbeans. Until recently, the Netbeans interface was more intuitive and pleasing than that of Eclipse but has since become equal or even worse. Eclipse now features a slick GUI for designing each window which makes creating the visuals of the application seamless. It is as easy as dragging the components of your application right onto the phone screen; all the code is implemented automatically. This feature alone caused one of the group members who was using Netbeans to switch instantaneously.

There is an all-in-one package available for Android developers from Google Android Developers, ADT bundled with Eclipse. Below in Figure 3.2.C is an example of the interface. Along with the normal coding scheme of XML, it gives users the ability to program with a drag-and-drop style interface which makes it much simpler than typing code. This gives the group more time to focus on the Bluetooth functionality and less time worrying about what it looks like.

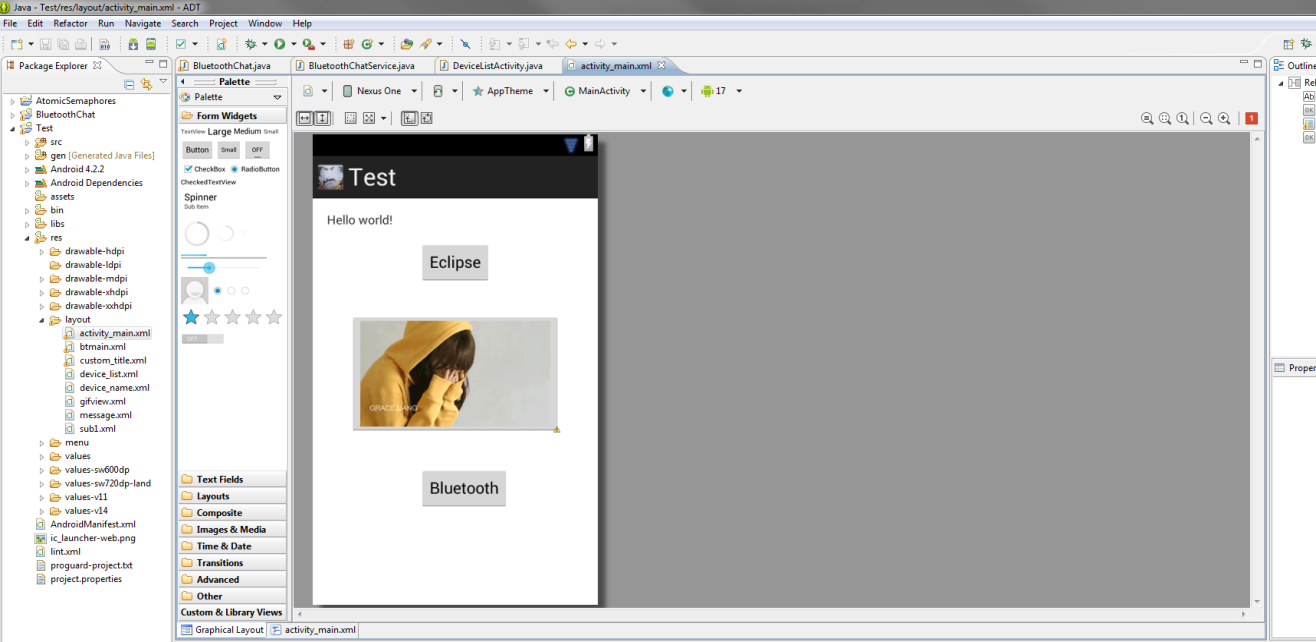


Figure 3.2.C - ADT plugin for Eclipse

#### Desktop Application

The group will need to create a GUI board programmer which can be used by a user to program the group’s Knight Bright board. The group could use any one of several GUI development environments to create this program. Both of the mentioned IDEs have visual GUI design, where controls can be laid out visually and then handled events created as functions.

Since this will be a Windows based GUI, the group could use Visual Studio. It has many libraries built in for creating a GUI event-driven application on a Windows platform. This can be programmed in several languages including C++ and C#.

An alternate development environment is Labwindows/CVI. This provides an incredibly simple interface which the group’s lead programmer has worked with before. It is an ANSI-C based environment. The group have decided to use this to design the group’s GUI board programmer desktop application in.

While not as powerful, the group has decided to use Labwindows/CVI. It is more familiar to the dedicated program for this project, as well as typically more simplified for the group’s purposes.

### Hardware

#### LED Addressing and Architectures

The group needed to come up with an architecture scheme for the LED array. There are 3 general methods that can be used to set up and control LEDs. These 3 methods are Charlieplexing, Decoding, and the use of LED drivers. Because the project will feature touch-sensor-like interactive capabilities, it is important to consider that the LED array must be able to respond to the IR-sensor logic.

##### Charlieplexing

Information and reference for the theory of Charlieplexing was found on (http://www.instructables.com/id/Charlieplexing-LEDs--The-theory/). Charlieplexing is a method of LED control that was invented by Charles Allen, who was an engineer for Maxim Integrated. Charlieplexing allows an y number of LEDs to be controlled by terminating the anode and cathode sides of the LEDs on alternating switch pins. This allows for the high voltage and grounding of the LED to be set dynamically, instead of having just one common ground and one common voltage supply for all LEDs. This allows for the minimum amount of pins need to control any number of LEDs. Since 2 pins can control 2 LEDs at a time, any higher order number of pins can be thought of as a superposition of any nth-number of 2-pin/2-LED pairs. The number of pins needed by Group 1 to address all 300 RGB LED cathodes is characterized by the following equation:

Equation - Charlieplexing Pin to LED Ratio

It was determined then, that 18 pins would be needed to control 300 LEDs, or 100 RGB LEDs. To demonstrate these higher order pairings, Figure 3.2.D represents a 3-pin Charlieplexed network. Let’s say that Pin 1 is high, and Pin 2 is low. This would result in LED1 becoming active, and every other LED remaining off (Pin 3 is considered to be in an high impedance state.) All 3 pins, alternating between the high, low, and high impedance states, results in LEDs being controlled with minimal pins.

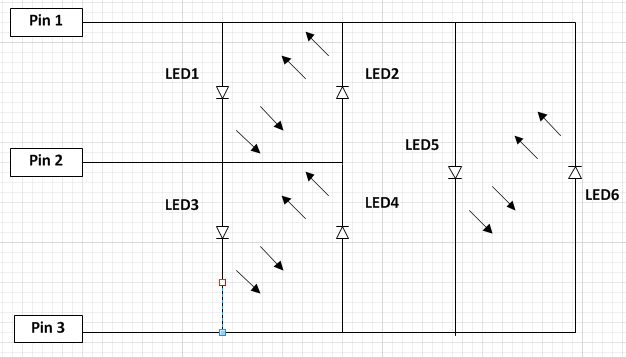


Figure 3.2.D - 3-pin Charlieplexed LED matrix

##### 4-to-16 LED Decoder

Another popular method of LED addressing is through the use of integrated decoders. The LEDs could be arrayed into an NxN matrix, with a specific value given to each row and column of the LED matrix. One decoder would sink up with all the rows of the LED array, and the other decoder would sink up with all the columns. If a specific LED were to be called, the LED’s row and column number would be selected, and the appropriate data outputs of the decoder would select that LED. The MCU can be programmed such that one port (it will be called Port A in this example) could handle the input commands for the Row Decoder, and another port (assume Port B) could handle input commands for the Column Decoder. For example, say that a (3x3) LED array has the addressing scheme depicted in Figure 3.2.E.

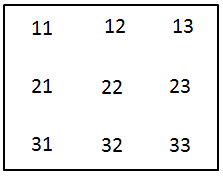


Figure 3.2.E - Addressing scheme of a (3x3) LED matrix

If the MCU were to select the LED at address 23, then the hexadecimal value 02 would be uploaded into the Port A register, and the hexadecimal value 03 would be uploaded into the Port B register. Once both commands are outputted by the MCU, a current will be allowed to flow at the specified LED. This would require that LEDs turn on to an active-low logic scheme. This would be achieved by sinking the 3 RGB cathodes to the output of a 2-to-1 NAND gate. The NAND gate’s inputs would be the row and column decoder outputs. To control all 300 RGB LED cathodes would require a superposition of 3 of these 4-to-16 LED Decoder Row/Column arrays. This would result in the MCU requiring only 12 pins to address the (10x10) 300 RGB cathode array (12 data selection pins for rows, and 12 data selection pins for columns.) Figure 3.2.F gives an example of a (3x3) LED matrix with a row decoder, column decoder, and NAND gates set up at every cathode pin of the LED.

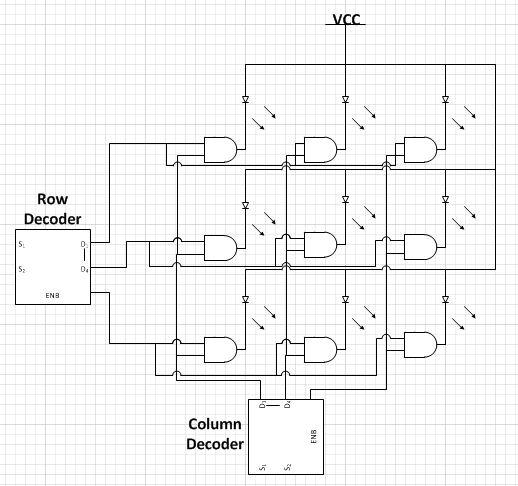


Figure 3.2.F - (3x3) LED matrix decoder (Each LED is terminated to a NAND gate output.)

##### LED Drivers

The final method of LED addressing to be considered, is the use of LED drivers. LED Drivers are the most widespread method of addressing LEDs available on the market. The vast majority of prior hobbyist projects that will be referenced in the group’s project design, used LED drivers of one form or another for their respective designs. An (NxN) LED matrix would be arrayed across a grid, with every LED anode terminated to a VCC supply. The LED driver is comprised of multiple channels (usually 8 or 16) which the LED cathodes could be terminated to. Assuming 16 channels per LED driver, and a 10x10 RGB LED array, the group would need 300/16 LED drivers to address all the LED cathodes (100 LEDs in total, multiplied by 3 for every RGB bulb cathode.) A depiction of the LED driver layout is shown in Figure 3.2.G. 99 of the red LED bulbs would sink to one row of drivers, 99 of the blue LED bulbs would sink to the second row of drivers, and 99 of the green LED bulbs would sink to the bottom row of LED drivers. The final RGB LED would sink to 3 channels of the last LED driver in the top right corner.

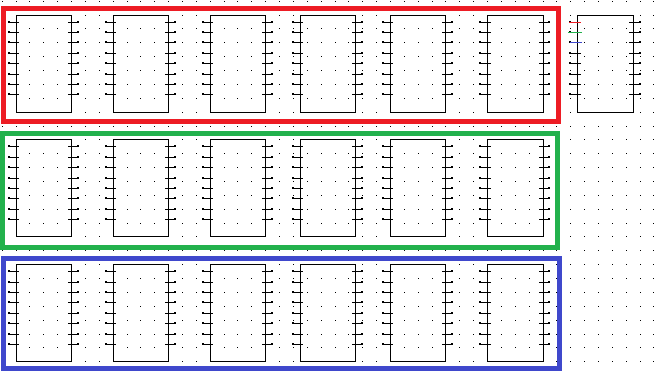


Figure 3.2.G - LED addressing using 19 LED drivers

### Sensor Addressing

Due to the fact that the project has one hundred individual user inputs, 100 output LEDs, multiple USB inputs, plus the availability of Bluetooth communication, any of which may all be active at one time depending on the program, it is necessary to consider addressing schemes available to the group which would help limit the input communication channels between the user and the device controllers. Before, reviewing the possible modes it is beneficial to first consider a few key specifications of the direct user input. The group would like the following:

* The device should be capable of accepting one or more inputs at a time depending on the needs of the program
* The inputs will be used to allow the selection of a cell or control of the device depending of the specific needs of the program being executed
* In some instances a group of cells or multiple cells may need to be selected, read into the program, and responded to accordingly.
* Hardware expenses should be minimized where possible due to the limited resources of the group

With these considerations in mind the following addressing modes were considered to allow communication between the input sensors and the microprocessor.

#### Direct Connection

The simplest possible means of communicating the data to the microprocessor is through direct connection. The projects one hundred individual would each have their connection tied directly to a dedicated pin on the microprocessor which could then be read into or transmitted out of an array and analyzed accordingly. This method would probably allow the quickest response to an input and corresponding output but would come at massive costs. First, the processor would need a minimum of one hundred inputs for the sensors. While not unheard of, these devices are quite costly. Also, the need of an individual communication line run between each pin and sensor or LED would drastically increase the cost of communication channels (wire).

#### Linear Addressing

A simple step forward from direct connection is a linear binary addressing scheme. This scheme would allow a limited number of pins on the microcontroller to be used based on simple binary principles. The appropriate number of pins on the microprocessor would be allocated and each cell would be linearly numbered and then have its number value expressed in binary terms. As can be determined by simple binary mathematics this method would limit the number of inputs to seven since. There would be twenty eight unused binary values for the inputs alone, but this is inefficiency is quickly negated by the drastic reduction in microcontroller input pins needed. Another issue with this addressing scheme is that it would most likely involve the use of a 7:128 demultiplexer which would not only add to the hardware cost, but also potentially limit the inputs to one at a time.

#### Row-Column Addressing

A final addressing scheme considered is row-column addressing. In this mode each row and column is designated a sequential binary value and an individual cell is chosen by using a combination of its row and column number. An example of this method for a four by four grid is shown below in Table 3.2.A.

Table 3.2.A - Example of addresses for Row-Column addressing

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Row/Column | 00 | 01 | 10 | 11 |
| 00 | 00 00 | 00 01 | 00 10 | 00 11 |
| 01 | 01 00 | 01 01 | 01 10 | 01 11 |
| 10 | 10 00 | 10 01 | 10 10 | 10 11 |
| 11 | 11 00 | 11 01 | 11 10 | 11 11 |

For a one ten by ten grid, which the project will consist of, there will need to be four digital input lines dedicated to the row (), and four digital inputs dedicated to the column designation. While this does waste 6 possible addresses for each row and column it does offer the distinct advantage of isolating each input, and with the proper use of inexpensive additional hardware discussed later, and some simple software features, this scheme can allow multiple inputs to be selected seemingly at the same time.

Additional advantages to this scheme are the ability to hardwire each individual row and column together. This reduces the number, and ultimately cost, of communication channels (wire). This scheme also allows, through the use innovative practices discussed next, for the output of all of the cells to be tied together. This allows the number of return value pins to be reduced to its optimal value of one.

The group hopes to use this row-column addressing scheme described above in conjunction with demultiplexers and triple input positive-and gates to reduce the number of pins used for sensor input on the microcontroller. The intended hardware configuration and corresponding software features are described in the following sections.

### Wireless Communication

The LED table will require wireless communication to talk between the microcontroller and the android device. The following research investigates a multitude of wireless communication methods that the group can take. Among the many technologies available, the group will focus on only the prominent ones of today such as: Bluetooth, ZigBee, Wi-Fi, and RF communication. The key features for the wireless communication method that would be looked at are the range, the data rate, and connectivity to the android device (cell phone) that the project would use. If the group’s senior design project is going to be able to communicate wirelessly with mobile devices, the group has to decide between two prevailing wireless technologies: Bluetooth, and Wi-Fi. The differences between Wi-Fi and Bluetooth are described by Table 3.2.B below.

Before deciding on what wireless communication method to choose from, first the group needed to establish what was needed from the wireless communication from the device. For the most part, because of the simplicity task the group needs, the project does not need strenuous requirements. The transmission range was the first key aspect to look into. Since the user will only need wireless connection if he is near, or at least in the same room as, the device, it is not necessary to have a range larger than 10m. In addition, the fact that the range will be so close, means that the group should not have to worry about interference.

Data rate is another key aspect when choosing a wireless communication method. Once again, the project doesn't require too much data to be transmitted, depending on how the group decides it wants to store the individual programs. In the case where the programs are loaded onto the LED table with the android device, a higher data rate would simply mean a faster load time and would not affect the use of the table otherwise. Even the minimal data rate of all the mentioned wireless communication methods would be sufficient enough for the project.

Table 3.2.B - Bluetooth Properties vs. Wi-Fi Properties

|  |  |  |
| --- | --- | --- |
| Properties | Bluetooth | Wi-Fi |
| Frequency | 2.4 GHz | 2.4 – 2.5 GHz |
| Cost | Cheaper | More Expensive |
| HW Requirements | BT adaptors on all devices | Wireless adaptors on all devices, and a wireless router |
| Range | 10 meters | 100 meters |
| Power Consumption | Low | High |
| Latency | 200 ms | 150 ms |
| Bit-rate | ~2.1 Mbps (Bluetooth 2.X) | 600 Mbps |

For commercial type products, the price always comes into play. The range of power the project requires is small, so the price is a big determining factor. Originally, the budget for wireless communication was around 30 dollars. After researching, the group is confident that it can easily stay below that budget. Finally, compatibility with the microcontroller and android device is also an issue because if it does not work with them then there is no way for the group to utilize it. The compatibility issue does limit the choices for microcontrollers that the group could choose from, if it was a newer technology or not as widely used. For anyone to use the android application, it would be best to use something that most phones have standard.

Clearly, Wi-Fi has more powerful and robust wireless capabilities than Bluetooth. However, because of the low-power, low-data rate nature of the project, Bluetooth 2.X will be used (more background on wireless communication will be provided in another chapter of this research paper.)

#### Wi-Fi

Wi-Fi is the most prominent wireless technology for computers and operates in the unlicensed 2.4GHz or 5GHz radio bands. Wi-Fi is a networking solution to connect multiple computers to a wireless network, such as the router that most people have in their homes. Specifications given by the Wi-Fi alliance provide for well-established connections that compensate for congestion in the network as well as built-in error correction. The most used applications that Wi-Fi is used in are video game consoles, smart-phones, tablets and laptops.

Although Wi-Fi’s range is larger in comparison to other technologies, performance decreases rather significantly as the range of transmission increase. The range of the transmission also causes the power consumption of the Wi-Fi transmitter to increase. The average power consumption for Wi-Fi is the highest compared to the other wireless methods and when combined with transmitting over long distance, it becomes even more so. If the group decides to make the group’s table battery powered, this must be considered.

Wi-Fi does have the highest data rate transfer out of all other technologies, which provides real world performance similar to a wired connection. Until recently, Wi-Fi has required the use of some kind of hub to connect two devices together such as a router. Wi-Fi Direct, previously known as Wi-Fi P2P, is a new standard that allows two to connect directly to each other. This works by embedding a limited access point into the devices. Another feature of Wi-Fi is security; however the group will not likely be in need of this.

Pros:

* High data rate (most likely unnecessary for the group’s project)
* Works in the unlicensed 2.4 and/or 5 GHz range
* Real world performance similar to wire connection
* Easy to integrate with tablet

Cons:

* Higher power consumption
* Range of about 1-100 meters

#### Bluetooth

Bluetooth technology is another viable, wireless option for sending and receiving data. Bluetooth is most popularly used with cell phones. When Bluetooth is mentioned, the average person will immediately think of a hands free headset for their cell phone. In addition, it can also be used to transmit files or data between two devices. So long as a device has a Bluetooth module attached, it will be able to communicate to others.

Bluetooth is has a wireless communication protocol primarily designed for low power consumption within a short range. This range can vary depending on what the power-class type is. The devices use a radio communications system, which means that they do not have to be in line of sight communication of each other, however, they still must have a viable optical wireless path to communicate. The ranges of Bluetooth classes are shown below in the following table, Table 3.2.C.

Table 3.2.C – Bluetooth Classes

|  |  |  |
| --- | --- | --- |
|  | Power Consumption | Range |
| Class 1 | 100 mW | 100 m |
| Class 2 | 2.5 mW | 20 m |
| Class 3 | 1 mW | 5 m |

According to Table 3.2.C, the class 1 power type can travel roughly 100 meters while consuming a maximum of 100 mW of power. For the group's application, this range will most likely not be necessary. The class 2 power type uses significantly less power than class 1, using a maximum of 2.5 mW of power. Consequently, class 2, can only travel about 10 meters. If the group decides to use Bluetooth, it will go with this power-class. Finally, class 3 can only travel about 5 meters, but uses a mind blowing, small amount of power that maxes out at 1 mW while transmitting. It may also be possible to use this class as well, however slightly less convenient it may be. While these classes all have different ranges and power requirements, it is still possible for modules of different classes to communicate with each other.

Pros:

* Does not require straight Line-of-Sight
* Low battery consumption
* Many robust profiles

Cons (don't apply too much to the group’s application)

* On cluttered 2.4 GHz ISM band
* Low range
* Low penetration qualities

#### ZigBee

ZigBee is a high level communication protocol using small, low-power digital radios based on the IEEE 802.15.4 standard for personal area networks. ZigBee operates within the industrial, scientific and medical radio band of 915MHz in the United States of America. Applications that utilize ZigBee are home control, medical monitoring and building automation. Much like Bluetooth, ZigBee networks are formed ad-hoc, with no centralized network point.

A great aspect of ZigBee is that it is generally low cost and power-efficient, which are features that any wireless module would hope to achieve. ZigBee allows for mesh networking, which provides high reliability and more extensive range. The low power is caused by the fact that ZigBee has a sleep mode which awakens and can be fully active in as little as 15 milliseconds. This response time allows ZigBee to remain inactive for the most of the time. Should the group decide on battery power, ZigBee won't be much of a power hog.

In addition to being very power efficient, ZigBee also has an astounding range that even compares to Wi-Fi. ZigBee offers a wide range that should be effective for the device, although the group doesn't actually require it. ZigBee has a defined rate of 250 kbit/s, best suited for periodic or intermittent data or a single signal transmission from a sensor or input device. The group wants to use it for an input device; therefore, this would be a good choice.

Pros:

* Easily implemented
* Transmission range between 10 and 75 meters
* High battery life
* Low cost
* Average supply voltage of 2.8 to 3.5 VDC

Cons:

* Low data rate up to 720kbs
* On cluttered 2.4 GHz

#### 6loWPAN

6loWPAN is an acronym for the IPv6 over Low Power Wireless Personal Area Networks. 6LoWPAN operators in the radio band range of 1GHz and 2.4GHz. 6loWPAN allows the ability to obtain a wireless connection not only between individual nodes but also to the internet. 6loWPAN has application most in large scale networks that require connection to an IP backbone network, which the group's project will not have.

A major problem with this is the fact that it is a much newer technology than the other wireless communications previously mentioned. Therefore, it is a little more untested, lacks example coding and does not have the compatibility with very many devices. This type of wireless communication would cause a limitation on the microcontroller because there are a very few of them that have compatible chips. This wireless technology will not be looked into anymore.

Pros:

* Low power
* Operates in the range 1GHz and 2.4 GHz

Cons:

* New technology
* Few compatible microcontroller
* IP backbone network

#### RF Communication

Radio frequency (RF) is a rate of oscillation in the range of about 3 kHz to 300 GHz. The previously mentioned wireless communications all use a kind of radio frequency. The radio frequency that the project would be using is basic civilian entertainment applications, where the common radio frequencies used are 915MHz, 2.45GHz and 5GHz. These bands are not supposed to be used without special licensing or ownership granted by the Federal Communication Commission (FCC). Some of the applications that RF communication is used in are health monitoring, home security, and energy harvest monitors.

The main difference between RF communication and the other wireless communication methods is that there is no specific protocol associated with general RF. This allows the group to create its own protocol. Creating the protocol will allow the group to develop and use a protocol that will only work with this project specifically and will only work for the project for which it is intended. Whether or not the group is willing to undertake this task is a big factor in making a decision.

Pros:

* Low power consumption
* Low cost
* Wide range selection
* Built RF transceiver into most microcontrollers
* Operates in the unlicensed range
* A lot of example code online

Cons:

* No helpful protocols and need to all protocols need for the device
* Unsecure

#### Wireless Comparison

The more likely options were looked into further in this section. The result of this section would yield what method of communication would be used for the wireless communication from the microcontroller to the android device. 6loWPAN will not be compared since it was determined, based on initial research, that this method will not be used in this design, mainly due to the newness of the technology. Figure 3.2.H shows the typical range of each of the wireless communication devices that the group researched.

Figure 3.2.H - Wireless Ranges

As seen, Bluetooth has the lowest typical range of the four communication devices at a range of about 10 meters. Wi-Fi range is a little better with a typical range of 32 meter and can become as much as 95 meter but the greater the distance the performance also decreases. ZigBee has an average range of about 75m but can go much longer ranges by sacrificing battery life. RF communication can go much farther distances on average than any of the former mentioned communication methods. Despite all of these facts, the groups requirement for range is low and so all of the methods are acceptable in this aspect.

As the range increases so does the power consumption of the device. For the most part, the group has decided that it would be plugged into an outlet. With minimal cost of electric aside, should the group decide it make the table battery powered, this will come into account more so for battery life. Wi-Fi, mainly due to its’ high bit data rate, has the highest average power consumption out of all the methods. Bluetooth, ZigBee and RF all use low power in there typical range. ZigBee was built to use low power and have long distance and RF communication uses around the same amount of power but can achieve a far greater distance in comparison. Bluetooth uses the least amount of power of all the methods and has a comparatively high data rate but has an extremely short transmission distance, which is perfect for the group’s application since the distance doesn't matter.

The compatibility of the android device with the microcontroller is extremely important because unless the group wants to hook it up via USB, the group need wireless. The main compatibility issues arise with the android device. The microcontroller the group will get is much more versatile as you can attach a module of any type with little problems. However, with the android application, it would be wise to use mainstream technology found in most phones so that there are little issues with the users. Unfortunately, that probably limits the group’s choices to either Wi-Fi or Bluetooth which come standard on most phones today.

Figure 3.2.I - Data Rates

The groups LED-table will never need to process wireless data from the android device in real time. That is, interaction between the two are not in real time. In fact, unless the group wants the table to send back a confirmation that the program or message was received correctly, the table does not have to transmit anything at all. Looking at Figure 3.2.I shown above, the typical data rate for each of the wireless communications is shown. The data rate would save power because the device would only send in short bursts, when the user sends a message or changes programs. Since the information that would be sent to the table is not much, Wi-Fi’s high data rate average of 65Mbps is not needed and would only be wasted on this type of project. Bluetooth, Zigbee and RF data rate, as seen in Figure 3.2.I are far better for the amount of information that would be sent in burst to the table.

#### Antenna

Depending on if the wireless device comes with an antenna or not, the group may also need to buy an antenna for the receiver or transceiver. The antenna for the wireless device would increase the performance of the device. Although the max distance is something the group does not particularly need to achieve, an antenna is still good for consistent reception. In choosing the antenna the group would look at Friss equation, which is the primary math model to predicting Line of Sight communication links. Friss equation can be seen in Equation 2 seen below.

Equation 2 - Friss' Equation

The Equation 2 will now be broken down into its variables for explaining. Pr is the power received in dBm and Pt is the transmit power in dBm. Gt is the transmit antenna gain in dBi and Gr is the receiving antenna gain in dBi. R is the distance between the microcontroller and the android device and λ is the wavelength. As mentioned before this equation is used to determine the range of the antenna. The transmit power could vary by phone, but remain within reasonable specifications.

There are many different types of antenna to choose from and each has benefits and disadvantages that come with the type. PCB antennas are low cost and a standard design that is widely available but PCB antenna have the potential to be large in size at low frequencies. The chip antenna is usually built-in to most wireless devices that are designed for short range and the device will need to communicate in long range. The whip antenna has good performance but usually at a high cost and difficult to fit into most applications.

Most wireless devices have a built-in antenna, but if the device the group chooses does not then the proper antenna would be chosen based on the Friss’ Transmission Equation and the cost.

#### FCC Regulations

For this project the group took into consideration the rules of the Federal Communications Commission. Some of the rules to take into consideration when using radio frequency were as follows:

* Under section 15.23 paragraph (a), home-built devices authorization is not required for devices that are not marketed, and built in quantities of five or less for personal use.
* Under section 15.23 paragraphs (b) Since the FCC recognizes that an individual builder may not have the means to perform measurements required to determine compliance with regulations, the builder is expected to design using good engineering practices to conform to regulations, as stated “to the greatest extent practicable.” Provisions in section 15.5 of the FCC code still apply
* Under section 15.103 paragraph( c) an exemption from specific technical standards in part 15 is given to “a digital device used exclusively as industrial, commercial, or medical test equipment.” As the LED table could technically be a commercial test, although the group has no current plans to market, it qualifies as exempt from regulation, except as required under Sections 15.5 and 15.29.
* Under section 15.15 states, “the operation of an intentional or unintentional radiator is subject to the conditions that no harmful interference should occur to authorized users of the radio frequency spectrum; else the party responsible is required to cease operation.” The group would make sure to comply with all such requirements.
* Section 15.29 sets forth the requirement that all certificates, registrations and technical data must be kept readily available for inspection by the FCC. Since under section 15.23 no registration or authorization is required, due to low quantity, the device was exempt from this requirement.

#### Wireless Decision

The key aspects of the wireless communication module were range, data rate, power consumption and compatibility. In terms of range, RF or Zigbee were the better choices but they do not have the data rate that Wi-Fi offered. Despite this, RF and ZigBee both have far lower power consumption than Wi-Fi. Wi-Fi’s data rate was overkill for the amount of data that would be sent so using the extra power would be useless. Bluetooth has a good data rate for the project and the short range being its “downfall” was not a significant loss for the group. ZigBee was very similar to RF communication in terms of range and power consumption. A difference betweenRF and Zigbee was that RF communication had a higher range and data rate on average than ZigBee does. ZigBee does have many protocols and helpful features when it came to programming. In the end, the group had to choose either Bluetooth or Wi-Fi, simply because the android device would most likely be compatible with only these, or rather it meant more money to buy a module for android to become compatible. The group ruled out Wi-Fi because it was overkill, as well as more costly and more complex to program. In the end the group decided on Bluetooth.

### LED Color Control Methods

The 2 predominant forms of digital LED fading control is Pulse Width Modulation(PWM) and Bit Angle Modulation(BAM). All information on PWM and BAM was referenced from Batsocks’s website: (http://www.batsocks.co.uk/).

#### Pulse Width Modulation (Using MCU peripherals)

There are two types of PWM that can be implemented on LEDs that are connected directly to MCU ports. They are hardware PWM, and software PWM. Built in to all (except a very few) MCUs is at least one hardware PWM generator. If only a couple of LEDs are being controlled, than the built-in PWM channels are ideal for controlling the LEDs. Hardware PWM channels are well understood, easy to set-up, and have little or no processor overhead while they're running: a base frequency and duty cycle are the only parameters needed. No software intervention is needed. To change the duty cycle, you adjust the values of a register. Software PWM can be implemented for *n* channels of outputs. For *n* channels, *n + 1* interrupts per cycle will be needed; one interrupt to turn all the channels on, and one interrupt for each channel in turn to switch them off. This overhead is fixed, regardless of the resolution of the output. MCU-channel PWM is the most basic architecture scheme for controlling the LED array. However, it is also the most cumbersome and feature-restrictive method. Furthermore, the Software PWM technique requires a significant amount of processing power.

#### Pulse Width Modulation (Using LED drivers)

The most common method for controlling LEDs is, without a doubt, PWM LED drivers. LED Drivers are integrated circuits that control the light dimming and current regulation of an LED. Most LED drivers are designed with internal shift registers, where the Grayscale of the LED dimming is stored (most of these Grayscale registers have 12-bits of scaling.) To control the Grayscale of an LED, the driver will have a Serial input pin, which is then connected to the MCU. MCU can be programmed with interrupts, and the ISR will send serial commands to the LED driver. The LED driver will receive these serial commands, and will store the data onto the 12-bit Grayscale register. To change the dimming of the LED, the MCU simply sends a new serial command with an updated register value.

#### Bit Angle Modulation

BAM is similar in a lot of ways to PWM. Both modulation techniques rely on a stream of repeating pulses, whose duty cycle is arbitrarily determined by the designer for whatever design goals the project requires of it. In the case of Group 1’s senior design project, the duty cycle and frequency of modulation determines the rate of flicker in the LED. But there is just 1 subtle difference between PWM and BAM. In a 12-bit PWM cycle, the duty cycle of each bit per pulse is constant In BAM, the magnitude of each bit of pulse is binary logarithmic in nature. The binary logarithmic nature of BAM can be exemplified by the following 8-bit BAM response in Figure 3.2.J.

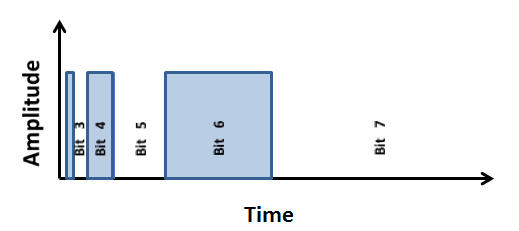


Figure 3.2.J ­- An 8-bit BAM pulse

In the preceding figure, bits 2, 4, and 6 are “on” while bits 0, 1, 3, 5, and 7 are “off”. The bit values are actually the magnitudes of an exponential function with a base of 2. The total “on” cycle of the pulse is the summation of all those exponential terms, subtracted by 1, and divided by . This is shown in the following equation, Equation 3.

Equation - "On" Cycle for Bit Angle Modulation

This is equivalent to a 33% duty cycle on an 8-bit PWM signal. However, only 4 interrupts are required, whereas an 8-bit PWM signal would need many times as many interrupts per cycle. BAM provides a novel, ingenious method of LED dimming with minimal overhead. However, the group will most likely use PWM with LED drivers as a means of multiplexing and controlling the LED array.

### Processor

Before the group can select from the processing hardware, it must first decide on the processing unit scheme that fits the projects best. The basic specifications of the project requires that the group use at least two processing units. The primary processor will be used to run the applications that the group specifically designs for the LED table. The secondary processor will relay commands from the input and output of the table. This means receiving input from the sensors and lighting the LEDs though the drivers. The two prevailing methods the group will look into are the use of microcontrollers and FGPAs. The group can choose to use two microcontrollers, two FPGAs with a CPU embedded in one or a combination of these things.

#### Microcontrollers

Microcontrollers are small computers contained within a single integrated circuit. At the very least they contain a processor core, programmable input and output, and memory. MCUs are used in a vast variety of applications from digital signal processing to a remote for a television. The difference of these two applications lies mainly in the clock speed of the processors. A low clock speed remote controller application might use as little as 4 bit words to operate a clock frequency as low as 4 kHz. This low clock also means it has lower power consumption, as low as single digit mW or even µW ranges. While the device is unused, it consumes nanowatts of power, making battery applications long lasting and efficient.

For the group’s application, however, it is going to take a bit more power than mentioned above. Most microcontrollers on the market today, almost 50% of them, are 8-bit. The group feels that this is an adequate amount, so long as the clock rate is high. Another feature is the amount of general purpose input/output pins that a microcontroller can have. The more pins the microcontroller has, the higher the control of the LEDs can be.

Programming a microcontroller was originally done in only assembly language, but many high-level languages are commonly used. These high-level languages can be designed for a specific purpose or be a general purpose language such as C. Compilers are made uniquely even for general purpose languages that may have restrictions, but also better support the characteristics of the microcontroller. The programming part of the group feels more comfortable using a language like C, rather than learning a new specific language.

#### FPGAs

A field-programmable gate array or FPGA is a programmable digital logic chip, which means that you can program them to do almost any digital function imaginable. FPGAs today have very large resources of logic gates (in the millions) and RAM blocks that make implementing complex digital computations possible. FPGA designs have very fast IOs and bidirectional buses which makes it difficult to verify correct timing of valid data with setup and hold times. Within an FPGA there are many programmable logic components termed “logic blocks”, as well as a hierarchy of reconfigurable interconnects which allow blocks to be “wired together” in a logical way rather than physical. The general workflow and properties of FPGAs are as follows:

* Program a “logic function” that you want with your computer. You can do this with a schematic or text design
* Compile your design to created something like a binary file that can be loaded on to you FPGA with a cable
* After it’s loaded the FPGA will behave according to the logic that is set. There is no limit to the amount of times that beside the standard that the RAM is on. There is no need to change any soldering or components like a normal PCB.
* Designs run much faster than a bored with discrete components because it is run in software on the board.
* Final note is that because the logic is saved on RAM, it is cleared after it is powered down. This problem would have to be addressed with extra implementation with flash memory or something like it.

The implementation of an FPGA is mainly used for digital signal processing, medical imaging, computer vision, speech recognition, computer hardware emulation, radio astronomy and much more. These applications require a tremendous amount of processing which isn’t necessarily required by the group’s application. Their applications also benefit by the re-configurability of the FPGA, something the group should not have to worry about. Powerful means that the price is also a lot higher; even the smallest FPGA goes for at least two hundred dollars, which frankly the group cannot afford. From a marketing perspective, it would raise the cost of the device unnecessarily.

#### Microprocessor Decision

For this project, the group has one hundred input circuits and three hundred LEDs to control with the group’s processing scheme. Although this will require more processing power than a common remote control for a television, the FPGA seems to be overkill. Even if a minimum spec FPGA was used, the cost would not maximize the group’s use of it. Furthermore, the programming language is important to the individuals in charge. The group thinks it is best to go with two microcontrollers, with as many general purpose pins and highest clock rate processor while maintaining a lower budget.

## Relevant Technology

This section covers the available technology and devices which can be used to achieve the methods and architectures described in the previous section. The components can be used to ultimately achieve the device design. This section goes into detail comparing various devices needed for each category.

### IR Sensors

For the purposes of this project the group has chosen to use an infrared detection circuit to provide input to the device. Each pixel of the device will contain an independent sensing circuit which will detect the presence of the operator’s hand, or another object, and respond accordingly. The goal is to allow the selection of individual or multiple cells to allow operator interfacing with the device. These inputs will be used at strategic times depending on the current mode of the device to allow selection and control where necessary.

Ultimately, it is hoped by the group that the interface method will provide a convenient, easy, and fun method of interaction. The use of infrared detection was chosen by the group in an effort to provide a similar experience to modern technologies such as capacitive touch screens, but at a much lower cost and technical expense. Infrared detection was also chosen due the very nature of infrared devices and the over design of the project.

In making this selection it is necessary to fully understand many aspects of infrared detection. This is due to the vast array of commercially available infrared detection devices. The level understanding necessary to make a wise selection begins with a fundamental understanding of the principles of infrared radiation. Then it is necessary to understand any complications which could arise while using infrared devices. Finally, one must consider all of the different types of devices commercially available and measure the benefits and risks of each.

The electromagnetic spectrum is a full range of frequencies which are classified as waves emitted from a source of electric or magnetic fields. This span ranges from what are classified as radio waves starting at a wavelength of meters long and ends at gamma rays which aremeters in length. The visible light spectrum lies between 390 and 700 nm (solarwiki.ucdavis.edu).

Just below the visible light spectrum is the infrared spectrum which lies between visible light and microwave radiation. At the low end infrared has a wavelength of 700 nm which is known as near-infrared. This puts infrared just below visible red light on the electromagnetic spectrum giving it its name. At the high end the wavelength is 350000nm for what is far-infrared. The range from 5000 to anywhere from 25000 to 40000 nm is mid-infrared (coolcosmos.ipac.caltech.edu).

Infrared radiation is primarily generated through heat or thermal energy. Therefore, any object which has a temperature above absolute zero (-459.67 K) will generate some amount of infrared radiation. In fact, before an object gets hot enough to emit photons, most of its energy is emitted through the release of infrared radiation (coolcosmos.ipac.caltech.edu).

Infrared emitter LEDs are offered in a variety of wavelengths but most commercially available models range from 830nm to 940nm in wavelength (futureelectronics.com). In addition to wavelength, the devices are also classified by angle of half intensity, drive current, forward voltage, and intensity. These devices operate on the same principles as visible light LEDs; however the materials used in their construction emit Infrared radiation instead of or in addition to visible light (futureelectronics.com).

Infrared detectors are any device which senses infrared radiation. These devices can detect the heat of an object, its location, or its motion. At the broadest range infrared sensors can be used in everything from the most advanced cosmological imaging equipment, which use near, mid and far-infrared to map the cosmos; to operating a hands free soap or paper towel dispenser in a bathroom.

For the sake of determining the best type of device to use the devices were divided into several categories, some with sub categories. The first general classification of an infrared device is either active or passive. This depends on the functional capabilities of the device and typically helps determine the next classification. Infrared devices can also be classified as either a sensor or a switch. Switches are then usually divided into sub-categories based on their physical package configuration. Another often used means of classification is by detector type. The primary classifications of detector types that will be considered are either photodiode or phototransistor. This also usually depends on the materials used in the sensor’s construction and determines the type of response or output the sensor will generate.

Active infrared detectors are devices which not only detect but also emit infrared radiation (www.hkvstar.com). Due to the nature of the device these types of detectors are often used in switches as discussed below. This is typically done with a two part device. The emitter is often an infrared light emitting diode which provides radiation at a specific frequency or wavelength. The detector is usually also a diode device which detects the radiation (www.wisegeek.org). These devices are typically sold in matching pairs with the detector having been calibrated to primarily respond to the frequency of the infrared radiation being given off by the emitter. This helps reduce the possibility of the detector giving off a false signal due to ambient radiation. Since no detector is immune to all infrared radiation, another tactic can also be used to limit the effects of ambient radiation. In this process the emitter is switch off and on or “pulsed” at a certain frequency or sequence of frequencies. Through the use of a microcontroller the signal coming off of the detector can then be analyzed to determine if it matches the input given to the emitter. While this offers some reduction to ambient interference, there is still a risk of the detector being saturated by an outside source. If the frequency of the infrared interference falls within the range of the detector the detector would then constantly be on, and never pass the signal from the emitter this creating a constant false output (www.wisegeek.org). This process of pulsing an infrared signal is very often used in common electronic devices such as television remote controls and wireless video game controller. Unlike active devices, passive infrared devices do not emit their own infrared radiation. These devices only consist of a sensor, typically a diode, which detects infrared radiation (www.wisegeek.org). Because of this, passive devices are typically used in an array when reaction to a broad range of infrared frequencies is desired. This means that passive infrared devices are more often used in sensors such as motion detectors or imaging equipment (www.wisegeek.org).

Another classification used to sort infrared devices is as either a sensor or a switch. As mentioned above most active infrared devices are classified as switches. These devices respond to the presence of a particular range of frequencies of infrared radiation by passing a current signal. These devices are usually photodiodes or phototransistors which act as a typical diode or transistor except that current is only passed when the frequency of radiation is detected which the diode was constructed to respond to (www.wisegeek.org). Infrared switches have several sub-categories based on their operation. Reflective switches use isolated emitters and detectors, usually mounted sided by side, which only respond when a reflective surface passes. This allows the infrared radiation being transmitted by the emitter to be bounced back to the detector. This is often used in applications where the presence of an object needs to be detected at close range as in manufacturing and automation (www.wisegeek.org). Another type of infrared switch is transmissive or “pass through” switch. In this configuration the emitter and detector are placed across from one another allowing a direct path for signal transmission. If the signal interrupted by an opaque object the detector no longer sees the signal from the emitter and responds accordingly. The most common application of this type of switch is the safety mechanism installed at the bottom of a garage door. This configuration often allows much greater range than reflective switches. Unlike switches, infrared sensors are used most often to detect a wide range of infrared radiation instead of a specific control signal. Because of this infrared sensors are almost always passive devices. Infrared sensors can be used as standalone devices or in an array (coolcosmos.ipac.caltech.edu). As a standalone device one of the most common applications of infrared sensors is in home security devices. In this application the detector is often fitted with a Fresnel lens or a small parabolic mirror to allow the detection of infrared radiation over a wide area (wisegeek.org). These devices then respond to the infrared radiation emitted by the thermal energy of any warm body.

The final and perhaps most influential classification of that must be considered in the selection of a detector is the choice between a photodiode and a phototransistor. Photo diodes function just like traditional diodes except that light is allowed to enter the depletion region of the diode thereby increasing the amount of current passing through the device (www.radio-electronics.com, photodiode). These devices have zero gain so the signal is not amplified. While there are several types of photodiodes the two most common are PN and PIN. Others include the Avalanche photodiode and Schottky photodiode, to name a few, which have characteristics that make them far from ideal for consideration. The primary difference between the two types of devices considered is that the PIN diode has an intrinsic layer between the PN junction making the device more sensitive (www.radio-electronics.com, photodiode) than the PN diode. While useful in some applications this feature increases the devices signal to noise ration making it not ideal for the project application. PIN diodes also need reverse bias to prevent excessive dark current from the intrinsic layer. This reverse bias makes the device ideal for higher bandwidth applications and high dynamic range applications (www.radio-electronics.com, photodiode) which the project is not. In general for a photodiode the response of the device to different wavelengths of light depends primarily on two factors. The first is the width of the top p layer, and the second is the semiconductor materials used in the construction of the device (www.radio-electronics.com, photodiode). Table 3.3.A below lists the material and wavelength sensitivity of some common sensor semiconductor materials.

Table 3.3.A - List of common sensor semiconductor materials and their wavelength sensitivity (www.radio-electronics.com, photodiode)

|  |  |
| --- | --- |
| Material | Wavelength Sensitivity (nm) |
| Germanium | 800 – 1700 |
| Indium Gallium Arsenide | 800 - 2600 |
| Lead Sulfide | 1000 - 3500 |
| Silicon | 190 - 1100 |

It can be seen from the discussion above on infrared radiation and the infrared emitter that any of these materials could be used in the infrared range if properly constructed to do so. While the wavelength sensitivity is very important due the application of the sensor in this project it is also very important to consider the signal to noise ratio. As a primary example, because of their larger bandgap silicon diodes tend to allow less interference from noise the germanium diodes (www.radio-electronics.com, photodiode). However the tradeoff is that silicon diodes can also operate in the visible spectrum making noise from the visible spectrum more of a concern.

### Power Supply

It has been decided by the group that the device should be powered through a wall outlet connection. Since the standard voltage in the United States is 120 Volts, 60 Hz AC and the components in the device all operate between 3 and 6 volts DC the group must chose a power supply for the device. To supply power to the device components, the group has decided to use a commercially available power supply. In making this selection it is important to consider the options and make a selection based on the design criteria of the project and the pros and cons of the choices presented.

There are three main types of commercially available power supplies. They are unregulated supplies, linear power supplies, and switching power supplies. Table 3.3.B shows several of the key characteristics of these devices. Unregulated power supplies are basically like transformers. The output is directly proportional to the input. Due to the sensitivity of the components used in the device and the expected load variance of the device this alone makes an unregulated power supply an inappropriate choice for the project.

In a linear supply the first process is that the incoming AC voltage is stepped down to a lower AC voltage. This lower voltage is then passed through a full-wave rectifier. In this configuration a capacitor is also used to maintain a constant DC voltage and to reduce rippling (http://urgentcomm.com). For linear power supplies a power transistor is used in its active mode as a resistor to control the output voltage. This transistor receives its control signal from a circuit which monitors the output voltage. This control line changes to transistor bias to regulate the output voltage (http://urgentcomm.com).

As can be seen in Table 3.3.B this leads to a moderately complex circuit with a medium number of components. However, the device is large and heavy when compared to the others. Another disadvantage of the linear power supply is a relatively low efficiency when compared to the others. One key advantage to using commercially available power supplies is that they are commonly used in a variety of applications which leads to a wide range of devices at reasonable prices.

Switching power supplies start by filtering and rectifying the input AC voltage to produce a high voltage DC signal. Then a MOSFET is used as a power transistor in series with a transformer to switch at a preset frequency. This turns the signal off and on which energizes and de-energizes the magnetic field in the transformer creating transformer action and stepping the voltage down on the secondary side of the transformer. This stepped down voltage is then passed through a full wave rectifier which sets it at the designed DC level. Samples of certain circuit parameters such as the output voltage and load current are then sent back to a pulse width modification circuit which sets the switching frequency of the device and ultimately the output parameters (http://urgentcomm.com).

While these supplies have a much greater efficiency then linear supplies, and are often lighter and smaller, they do carry some disadvantages. First, the circuits are generally much more complex and contain many more components than the linear power supply. Also, due to the use of MOSFETs the devices provide uneven reliability and often have issues with EMI or electronic noise. While these issues have been recently improved upon they are still present in many models. Also, even though production costs have come down recently the devices are still not widely in use. This means that they are on average more expensive to a comparably rated linear power supply (www.acopian.com).

Table 3.3.B - Comparison of linear, switching, and non-regulated power supplies (www.acopian.com)

|  |  |  |  |
| --- | --- | --- | --- |
|  | Switching  Power Supplies | Linear  Power Supplies | Unregulated Power Supplies |
| Circuit Design | complex | moderately complex | simple |
| Number of parts | high | medium | low |
| Efficiency | 70-85% | 40-60% | 90-95% |
| EMI (Noise) | high | very low | very low |
| Leakage | high | low | very low |
| Size | small size | large size | medium size |
| Power density | high | low | medium |
| Weight | light | heaviest | heavy |
| Power to Weight Ratio | high | low | low |
| Power Factor | 0.6 - 0.7 without PFC  >0.95 with PFC | 0.6 - 0.7 | 0.6 - 0.7 |
| Cooling | convection or fan | convection or fan | convection |
| Isolation | yes | yes | yes |
| Input Voltage  Range | 90 - 132 VAC  (without PFC) 90 - 264 VAC  (with PFC) | 105 - 125 VAC | 0 - 125 VAC Output directly  proportional to input |

### USB

The functionality of USB is pivotal to the functionality of the group’s design. The group wishes to use its own USB design to be able to program the microcontrollers directly on the projects device, instead of programming the chip on separate board and moving the chip around. This is also important for the consumer PC application that will allow them to make their own simple programs. If USB does not exist, this feature is completely thrown off the table.

#### Integrated Circuit Serial Communication

The group’s project design will be programmed and controlled from 2 sources: the user PC, and the mobile phone app. A GUI app program will be designed on the PC that will allow the user to program 100 pixel animations onto the LED table array from the desktop. The mobile phone app will act as a peripheral control to interact with the LED table array’s game programs. The GUI app will communicate with the MCU through a USB port. The mobile phone app will communicate to the LED table’s MCU through a Bluetooth module. Both the USB port and Bluetooth module will need to communicate with the MCU through the MCU’s onboard UART port channels.

#### Desktop PC to MCU Communication

General summary information that explain serial protocols of UART, RS-232, and USB technologies were researched with information from Microchip Technology Inc. (http://ww1.microchip.com/downloads/en/appnotes/00956b.pdf). There are 2 popular methods of serial data communication: UART and USB. Although the vast majority of PCs no longer come equipped with UART ports (such as RS-232), UART is still popular among microprocessor and embedded system board designs. Desktop and laptop PCs, on the other hand, now come equipped with standard USB ports for serial data communication. Group 1 will have to find a way to mediate serial data communication between the UART port in the MCU with the USB port of desktop PCS and laptops. There are 2 possible solutions that could be utilized to overcome this interface incompatibility, a software approach, and a hardware approach. The software approach would be to connect the PC’s USB port to a PCB with a dedicated microcontroller that acts as a virtual USB port. This dedicated microcontroller converts the USB data into UART data for the MCU. The hardware approach would be to connect the MCU’s UART port to a TTL-to-serial data dedicated IC. The IC chip provides the appropriate voltage regulation levels between the TTL data and the serial data for the MCU and PC desktop.

#### Software solution – Virtual USB Port

It is possible to program a microprocessor to emulate the USB data protocol. A popular virtual USB port software package is V-USB. V-USB is a firmware pack designed by Objective Development.V-USB is offered under the GNU General Public License, so the firmware is open to developers. There is an abundance of projects on Objective Development’s website that utilize both V-USB, and the USB device classes. Once V-USB has been downloaded onto a host device, the microprocessor can transfer data with specifications that adhere to USB 1.1. A depiction of a USB-UART interface is given in Figure 3.3.A. Information on V-USB was referenced from Objective Development’s website: (www.obdev.at/). There are some hardware requirements for V-USB to run on the host microprocessor. These requirements are:

* The host device must be an AVR microprocessor
* The device must have 2 kB of Flash memory, and 128 bytes RAM
* The device must be capable of being clocked at 12 MHz

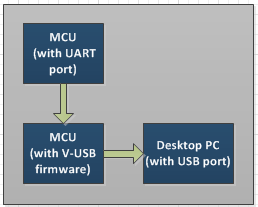


Figure 3.3.A - UART-to-USB using V-USB enabled MCU

#### Hardware solution – TTL-to-serial data dedicated IC

General information on interfacing UART with RS-232 was referenced from (http://praveendb.wordpress.com/2009/05/18/interface-the-atmega-1632-with-the-pc/). The UART port of any MCU adheres to low-voltage TTL. That is to say, the voltage requirements of the UART port will be +/- 3.3 V – 5 V when denoting serial data communication logic. For an RS-232, the voltage requirements are variable, but can be as high as +/- 15 V when denoting serial data communication logic. These significant voltage discrepancies can be overcome by interfacing the UART port of the MCU with a dedicated IC.

There are 2 generic approaches for interfacing UART to USB. The first approach would involve converting the UART port of the MCU into an RS-232 serial data line. This serial data line could be terminated into a DB-9 connector, which could be connected with a USB-to-serial adapter cable. Once connected to the PC desktop/laptop, the PC operating system would then utilize a device driver to convert the RS-232 data into USB. This is a user-end PC approach to interfacing. This is the most basic approach for interfacing the MCU with the PC, but it is dependent on the quality of the adapter cable’s firmware. The benefit of this design approach is that the user now has the option of connecting with the group’s interactive LED table through an RS-232 port.

A second hardware approach would be similar to the first one, but instead of terminating the RS-232 data with an adapter cable, instead it would sink to a dedicated IC. This dedicated IC would then convert the RS-232 data into USB, and relay data streams between the MCU and desktop PC. The benefit of this approach is that no programming or firmware would be required to synchronous the serial data communication lines.

General information on calculating and controlling serial data flow was referenced from (http://www.swharden.com/blog/2009-05-14-simple-case-avrpc-serial-communication-via-max232/), and authored by Scott W Harden. If the hardware approach is to be implemented, then an oscillator frequency will be required to clock the data rate of the TTL/RS-232 converter IC. Most TTL/RS-232 converters come equipped with their own internal RC oscillators, but their error rates make them impractical for clocking the data flow. The external oscillator frequency must be selected such that the clock frequency of the MCU will synchronize with the TTL/RS-232 converter. Once the external oscillator frequency has been selected, next the intended bit transfer rate (Baud) must be selected. Once these two parameters, the clock frequency and the baud rate, have been determined, the MCU is ready to be programmed with an appropriate USART Baud Rate Register (UBRR) value. The UBRR value can be calculated from the following equation:

Equation - UBRR Value

As a general rule of thumb, the base oscillator frequency value is set as 1.8432 MHz. Any whole integer multiple of 1.8432 MHz can be set as the external oscillator frequency clock, and when calculated a desired Baud rate (which generally falls into a range between 2.4 mbps – 115.2 mbps), the UBRR value of the MCU can be programmed. The first hardware approach is depicted in Figure 3.3.B. The second hardware approach is depicted in Figure 3.3.C.

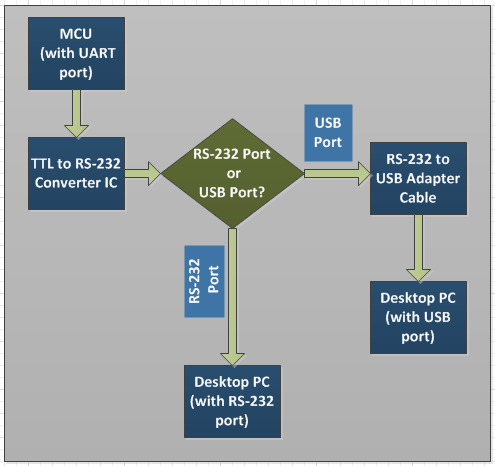


Figure 3.3.B - UART-to-USB 1st Configuration (optional RS-232 PC port)

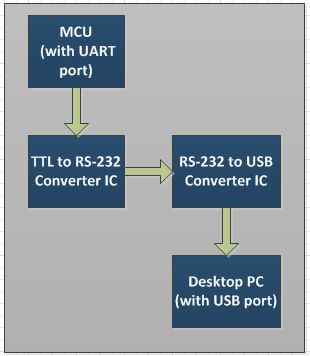


Figure 3.3.C - UART-to-USB 2nd Configuration (USB port only)

#### USB Software

Atmel ATmega2560 microcontrollers have the ability to be reprogrammed via I/O ports. It can be set up with a USB chip to be programmed via USB. This gives a simple solution for updating driver firmware, or program functionality on the two microcontrollers. Additionally, a USB port can be set up to specific digital I/O ports to programmatically write to the EEPROM and “install” a program. From the PC side, a driver to communicate with the board will have to be created. This would work in conjunction with the GUI programming environment (Section 4.3.3). The group plans to use the Windows Driver Foundation (WDF) class to create the group’s USB driver. Windows API functions can then be used to interface with the driver from the group’s GUI program.

Specifically, the group has no plans to create a USB 3.0 interface. The group will be sticking with USB 2.0 due to the abundance of resources, as well as pricing, availability of parts, and overall simplicity.

#### Serial Communication Interface Components

There are three (2 software and 1 hardware) design approaches that the group can utilize for the serial communication interface. The first option would be to utilize the V-USB firmware software developed by Objective Development. V-USB could be downloaded onto an ATtiny2313, which would then follow the CDC protocol and convert the USB 1.1 data stream into UART. A second option would be to use the ATmega16u2 as a USB-UART interface between the PC desktop and MCU. The ATmega16u2 would allow for USB 2.0 data stream to flow to/from the MCU and PC desktop. The final option would be to use the 234XD USB-to-UART interface chip developed by Future Technology Devices International Ltd. (http://www.ftdichip.com/Products/ICs/FT234XD.html).

##### ATtiny2313 (Atmel) + V-USB (Objective Development)

An extensive tutorial for configuring the ATtiny2313’s hardware and software requirements to implement V-USB was prepared by Joonas Pihlajamaa on the website Code And Life (http://codeandlife.com/2012/01/22/avr-attiny-usb-tutorial-part-1/). One notable consideration for designing the interface are the 3.3 V requirements of the USB port. Furthermore, V-USB will only work for precise clock frequencies of 12 MHz, 12.8 MHz, 15 MHz, 16 MHz, 16.5 MHz, 18 MHz, and 20 MHz. Although the V-USB enabled ATtiny2313 provides USB 1.1 support and is available under the GNU General Public License, it would be time consuming to troubleshoot the CDC protocol of the IC. Although V-USB has numerous uses for peripheral and data storage devices, it is overkill for the minimal serial data communication requirements of the group’s project.

##### ATmega16u2 (Atmel)

The group is planning on using Atmel microcontrollers to control the decoding of the IR sensors and RGB LEDs. In particular, the group is planning on testing the subassemblies of the Knight Bright using the ATmega2560 MCU. The ATmega2560 board comes built in with a USB-UART interface, and the group could emulate this USB-UART interface on the ATmega2560. This interface is comprised of a USB port which is terminated to an ATmega16u2 microcontroller, and the ATmega16u2 terminates directly to a UART port on the ATmega2560 chip. The group could opt for the ATmega16u2 USB-UART interface because it provides USB 2.0 serial data transfer. Because the ATmega16u2 and ATmega2560 are both produced by Atmel, this makes it easier to troubleshoot software problems that were to occur in interfacing both the ATmega16u2and the ATmega2560.

##### FT234XD (Future Technology Devices International Ltd.)

If the software approaches prove to be too cumbersome for the group to debug, another option for USB-UART interfacing would be to use the FT234XD IC, produced by FTDI Ltd. One of the strongest features of the FT234XD is that the chip requires no firmware programming to handle the USB protocol; the chip already comes equipped for USB interfacing. The chip can also allow up to 3 Mbps data transfer rates (equivalent to RS-232). USB drivers are not required on the user-PC side of the serial interface, because of the use of Virtual Com Port (a device protocol licensed by FTDI). The VCP allows the user PC to read the USB port as just a standard COM port. The FT234XD is a hardware approach to serial data transfer, so the only parameters that need to be controlled are the oscillator clock rate of the MCU UART port and the FT234XD, and the UBRR setting of the MCU. All information on the FT234XD was referenced from FTDI’s website: (http://www.ftdichip.com/Support/Documents/DataSheets/ICs/DS\_FT234XD.pdf).

### LED Drivers

The group came to the conclusion that PWM through LED drivers was the best control method for the LED array in the project design. Since there is an overwhelming abundance of LED driver designs available on the market, the group will use the pre-existing projects and products mentioned in section 3.1 as a starting point for selecting the right LED driver model. There were 4 different LED driver models used in the pre-existing projects mentioned in section 3.1; the PCA9922 by NXP Semiconductors, the STP16CP05MTR by STMicroelectronics , and the TLC5941 and TLC5940 by Texas Instruments. Low power consumption was an important feature of Group 1’s design. Most RGB LEDs on the market have a peak current of ~20-30 mA per RGB bulb (this is equivalent to ~60-90 mA peak current per LED.) Each of the 3 bulbs on the RGB LED will sink to an individual channel on the LED driver (Example: the “R” bulb would sink to channel 0, the “B” bulb to channel 1, etc.) The LED driver to be considered must be able to drive ~30 mA per channel. Low cost was also an important consideration in selecting an LED driver. The properties of each of the 4 LED driver models (current supply, voltage supply, features, cost, etc.) are tabulated in Table 3.3.C. Furthermore, the TLC5941 and TLC5940 (both manufactured by Texas Instruments) come built in with two features which the group may find beneficial for troubleshooting the final project design. These two features are the Grayscale fading, and the Dot Correction. Grayscale is a one-dimensional method of controlling intensity of a particular color scheme. Given the feature’s namesake, the color intensity can be converted into a series of discrete values with a range from total absence of color (or, white) to a total saturation of color (or black). The TLC5941 and TLC5940 are both capable of 4096 discrete values along the grayscale spectrum. The other feature of interest, the Dot Correction, is a current divider that can finesse the luminosity of a particular driver’s channel. Each channel is capable of 64 discrete levels of Dot Correction, with each discrete corresponding to a incremental division of the original saturation current (the Dot Correction can output 64 discrete levels of the original current, ranging from 0 to 1.)

Table 3.3.C - Properties of LED Driver Models

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Model Name | TLC5941 | TLC5940 | STP16CP05MTR | PCA9922 |
| Current supply(max) | 80 mA | 60 mA (< 3.6 V)  120 mA (> 3.6 V) | 100 mA | 60 mA |
| Voltage supply (input) | 3 to 5.5 V | 3 to 5.5 V | 3 to 5.5 V | 3.3 to 5.5 V |
| Voltage supply(output) | 17 V | 17 V | 20 V | 6 V |
| Interface | Serial | Serial | Serial | Serial |
| Data Transfer rate | 30 MHz | 30 MHz | 30 MHz | 25 MHz |
| Channels | 16 | 16 | 16 | 8 |
| Cost | $1.80/per unit | $2.21/per unit | $4.32/per unit | $0.49/per unit |
| Additional  Features | 12-bit Grayscaling  6-bit Dot Correction  LED Open Detection  Thermal Error Flag | 12-bit Grayscaling  6-bit Dot Correction (EEPROM storable)  LED  Open Detection  Thermal Error Flag |  |  |

Information on the PCA9922 was referenced from NXP Semiconductor’s website: (http://www.nxp.com/products/). Information on the STP16CP05MTR was provided by STMicroelectronic’s website: (www.mouser.com). Also, information on the TLC5941 and TLC5940 were provided by Texas Instrument’s websites: (http://www.ti.com/product/).

##### PCA9922 (NXP Semiconductors)

The PCA9922 RGB LED driver was used in Greg Brault’s (8x8x8) RGB LED cube. The PCA9922 has 8 constant current LED driver channels, each with an adjustable output current (controlled by an external resistor.) The output current can be controlled from 15 mA, to a maximum rating of 60 mA. The outputs are controlled via a serial interface with a maximum clock frequency of 25 MHz. The driver is capable of LED open-circuit, and short-circuit error correction, and the error data is then serially outputted to the user. The error correction feature is useful if LEDs are connected in series to the driver channels. Of the 4 LED driver models being researched, the PCA9922 is the most basic design. The PCA9922 has the smallest output supply voltage, the smallest maximum supply current, and it also has the smallest amount of channels when compared to the other 3 LED driver models. Furthermore, at $0.49/per unit, it is by far the cheapest of the LED driver models. The PCA9922 does not have on-board PWM; PWM would be executed by the MCU ISR.

PCA9922 Technical Specifications:

* 60 mA current supply(max)
* 6 V voltage supply(output)
* 25 MHz data transfer rate
* 8 Channels

##### STP16CP05MTR (ST Microelectronics)

The STP16CP05MTR LED driver was used by Nick Schulze (HNTE) in his (8x8x8) RGB LED cube. The STP16CP05MTR takes in a 16-bit word serial input, and sends out sixteen 1-bit parallel channel outputs. The STP16CP05MTR is capable of an adjustable output current supply range from 5 mA to 100 mA (which is controlled by an external resistor) from each of the 16 output channels. The STP16CP05MTR has a maximum serial data rate of 30 MHz. The STP16CP05MTR has no on-board PWM, so PWM would need to be executed by the MCU ISR. At $4.32/per unit, the STP16CP05MTR is the most expensive LED driver model out of the 4. Although its features and capabilities are superior to the PCA9922, the STP16CP05MTR’s cost and lack of PWM makes it unsuitable for the group’s project.

STP16CP05MTR Technical Specifications:

* 100 mA current supply(max)
* 20 V voltage supply(output)
* 30 MHz data transfer rate
* 16 Channels

##### TLC5940 (Texas Instruments)

The TLC5940 was used in the “RGB Coffee Table – Texas Instruments” project, and the “100 Pixel RGB LED Table – IT Gecko” project. The TLC5940 is a 16-channel, 12-bit Grayscale dimming, 6-bit Dot correction, PWM-enabled RGB LED driver. The TLC5940 takes in a serial input, and latches 12-bit words onto an internal register, one for each channel. This 12-bit internal register stores the Grayscale dimming value of the channel. This Grayscale dimming value is cycled endlessly in the internal register, until the MCU changes the Grayscale value to a new register value. Furthermore, the TLC5940 also comes with a 6-bit Dot correction feature. If all LEDs are given the same constant-current value, and the same Grayscale value, chances are that not all the LEDs will have exactly the same dimming. If an open (burned out) LED is detected on a channel, this can be detected on the TLC5940 using the XERR and BLANK pins. If the TLC5940 begins to overheat (> 160 C), the XERR and BLANK pins can notify the MCU. The TLC5940 has 2 maximum currents; 60 mA if VCC < 3.6 V, and 120 mA if VCC > 3.6 V. The 6-bit Dot correction was designed to help even out this discrepancy. The Dot correction takes a ratio of the maximum current, from 0 to1 (at 64 different increments), and this new value is what gets outputted to the LED channel. What separates the TLC5940 from the TLC5941 is that the TLC5940 can save the Dot correction values of each output channel onto an EEPROM.

TLC5940 Technical Specifications:

* 60 mA (VCC < 3.6 V) or 120 mA (VCC > 3.6 V) current supply(max)
* 17 V voltage supply(output)
* 30 MHz data transfer rate
* 16 channels
* 16-bit Grayscale dimming (4096 brightness levels)
* 6-bit Dot Correction (stored on EEPROM)
* LED Open Detection
* Thermal Error Flag

##### TLC 5941 (Texas Instruments)

The TLC5941 was used in the “Dynamic Animation Cube” project from Group 1 of the Spring-Summer 2012 semesters at UCF. The TLC59401is a 16-channel, 12-bit Grayscale dimming, 6-bit Dot correction, PWM-enabled RGB LED driver. The TLC5941 takes in a serial input, and latches 12-bit words onto an internal register, one for each channel. This 12-bit internal register stores the Grayscale dimming value of the channel. This Grayscale dimming value is cycled endlessly in the internal register, until the MCU changes the Grayscale value to a new register value. Furthermore, the TLC5941 also comes with a 6-bit Dot correction feature. If all LEDs are given the same constant-current value, and the same Grayscale value, chances are that not all the LEDs will have exactly the same dimming. If an open (burned out) LED is detected on a channel, this can be detected on the TLC5941 using the XERR and BLANK pins. If the TLC5941 begins to overheat (> 160 C), the XERR and BLANK pins can notify the MCU. The TLC5941 has a maximum current of 80 mA per channel. The 6-bit Dot correction was designed to help even out this discrepancy. The Dot Correction takes a ratio of the maximum current, from 0 to1 (at 64 different increments), and this new value is what gets outputted to the LED channel. The Dot Correction is not saved on the TLC5941.

TLC5941 Technical Specifications:

* 80 mA current supply(max)
* 17 V voltage supply(output)
* 30 MHz data transfer rate
* 16 channels
* 16-bit Grayscale dimming (4096 brightness levels)
* 6-bit Dot Correction
* LED Open Detection
* Thermal Error Flag

### Bluetooth Module

After selecting Bluetooth as the group’s primary means of communicating wirelessly, the next step is selecting a physical module that will be attached to the group’s board. Since all the modules are the same technology at its core, it should be simpler to choose between devices. Along with buying the device itself, the group decided to make it easier on themselves by acquiring some kind of break out device so that the pins would not have to be soldered to work. This is especially helpful when testing the functionality of the device, as it can be moved and connected with a breadboard. Three modules were explored after looking at previous successful projects. The RN-42, KC-21 and WT12 were further examined.

#### WT12

The WT12 is a fully integrated Bluetooth solution from Bluegiga, an impressive company that has been in business for over twelve years. This module provides an ideal solution for developers who wish to quickly integrate Bluetooth without having to worry about investing time into developing their own stack or radio. Bluegiga’s custom Bluetooth stack iWRAP, is embedded on to the WT12, implementing thirteen Bluetooth profiles including Apple iAP. With this and Bluegiga’s technical support designers guarantee fast track to mark with low costs and risks.

The WT12 has an awesome list of features that increase its value to the group’s project. So much so that the group could be paying for features that it doesn’t even need. This chip contains six software programmable I/O pins for more complex control of the chip, which for the most part, the group should not have to deal with in its application. Due to its amount of profile support, this single chip is capable of dealing with almost every possible wireless application you can think of which is impressive, mainly emphasizing its audio capability, however meaningless to the group’s needs. The hardware even has a metal shield to prevent RF interference in dense metropolitan areas.

One of the only problems with the WT12 is the lack of breakout support. One of the only breakout boards was custom built and still required soldering to be implemented for the group’s testing purposes. If possible and cost effect, the group is aiming to purchase a breakout board that can be plugged right into a breadboard and be easily manipulated. Table 3.3.D - WT12 Basic Spec List below shows specifications for comparison:

Table 3.3.D - WT12 Basic Spec List

|  |  |
| --- | --- |
| Property | Value |
| Bluetooth Versions | 2.1 + EDR, Most likely backwards compatible |
| Data rate | With onboard stack 250Kbps;  HCI mode: 1.3Mbps sustained, 3Mbps burst |
| Profiles | SPP, DUN, HID, iAP, HCI, RFCOM, L2CAP, SDP, a vast amount of other audio and msc profiles. |
| Supply voltage | 3.2V – 3.4V regulated |
| Power consumption | Standby 25mA; Connected 2.5mA; Deep sleep < 1mA |
| Operating temperature range | -40C to +85C |
| Antenna | Integrated chip antenna |
| Size | 14mm x 25.6 mm x 2.4 mm |
| Weight | 0.039 oz. |
| Price | $28.36 |

#### RN-42

The RN-42 is a powerful Class 2 device that is assembled by Roving Networks, one of the leading companies that sets itself apart in the wireless industry. It is a perfect solution for a short range, battery powered application and by default uses Serial Port Profile (SPP) configuration. However, it is also capable of using most standard Bluetooth profiles, combined with a simple UART hardware interface, it is easy to integrate into an embedded system or connect to an already existing device. The group also found a breakout device which surprisingly enough wasn’t any more expensive than buying the module stand-alone. The stand-alone device can be mounted easily on to a PCB should the group choose it.

This device has an impressive set of features which help the group’s job and reduce the amount of programming required for the hardware to be compatible. This is because substantial amount of software is already programmed on the device. This software includes auto-discovery/pairing which is as seamless as replacing a cable. It also has an onboard embedded Bluetooth stack that does not require a host processor, much like the iWRAP for the WT21, which will conserve some processing power of the group’s own microcontrollers. Not to mention the time required to design a custom Bluetooth stack.

On top of all of these features the RN-42’s power consumption is very efficient. When the module is inactive and in sleep mode, it only draws 26 micro Amps. This module is capable of being programmed for low power use and can be adjusted very simply to match any application. This is also the cheapest solution the group has found yet. Below is a list of relevant specifications to easily compare modules, Table 3.3.E:

|  |  |
| --- | --- |
| Property | Value |
| Bluetooth Versions | 2.1 + EDR, 2.0, 1.2, 1.1 |
| Data rate | With onboard stack 300Kbps;  HCI mode: 1.5Mbps sustained, 3Mbps burst |
| Profiles | SPP, DUN, HID, iAP, HCI, RFCOM, L2CAP, SDP |
| Supply voltage | 3.3V ± 10% |
| Power consumption | Standby 25mA; Connected 3mA; Deep sleep 26 µA |
| Operating temperature range | -40C to +85C |
| Antenna | PCB trace |
| Size | 13.4mm x 25.8 mm x 2 mm |
| Weight | 0.045 oz. |



#### KC-21

This device is a light weight, low power option that comes from KC Wirefree, a small company out of Arizona who leads the industry of OEM Bluetooth Modules. What’s most impressive about their device is the custom firmware they have developed for it, kcSerial. This firmware is designed to be easy to configure and operate by providing an easy to use, character based AT command language. Configurations can be saved to the onboard flash memory, providing an easy way to customize your chips default startup settings and features. Since the other modules are flashed with the profile that the group requires, it is unlikely that this module would be picked for their firmware alone. If needed, the other modules mentioned above are also programmable. Table 3.3.F shows comparison data below:

Table 3.3.F - KC-21 Basic Spec List

|  |  |
| --- | --- |
| Property | Value |
| Bluetooth Versions | 2.1 + EDR |
| Data rate | With onboard stack 300Kbps;  HCI mode: 1.5Mbps sustained, 3Mbps burst |
| Profiles | SPP, DUN, HID, L2CAP, SDP |
| Supply voltage | 1.8V ± 10% |
| Power consumption | Low power connection modes < 500 µA; Peak current 90mA |
| Operating temperature range | -40C to +85C |
| Antenna | Integrated Chip antenna |
| Size | 15.2mm x 26.9 mm x 2.5 mm |
| Weight | 0.047 oz. |
| Price | $36.00 |

#### Bluetooth Module Decision

After looking at even more modules than discussed above, it is clear that the technology is more or less equivalent and many modules meet the needs of the project. The problem in researching different company’s modules came when they didn’t always fully highlight each feature. When a particular company focused on a feature the other companies also had, but did not mention, it made it seem as though the feature in question was unique when it really wasn’t.

Most of the examples the group researched led to the RN-42 which seems to have ample Arduino support. The auto-discovery and pairing sounded like a nice feature as well. It was also the only module which breakout board did not require any soldering, to make testing it the easiest. It also has iAP profile support in case the group wanted to port the application to iOS. On top of all this, the price for the RN-42 is the cheapest by almost 30%. The next step is to delve into the RN-42 and figure out exactly how it will be integrated into the project.

### Microcontroller Selection

After deciding the general architecture of the projects processing block, the next step is choosing microcontrollers that fit the needs of the project. There are two strict requirements for the microcontrollers set by the lead programmer. One is that the microcontroller should have as many general purpose I/O pins as possible, the more the better. The second is that the high-level language the group should be C or some variant of C. This is the most comfortable language for the group’s programmers and will help them focus on getting code to work instead of dealing with the language itself. Furthermore, it is preferred if the microcontroller the group selects is compatible with the Arduino IDE. Some other general specifications including having a high clock rate and memory. These specs should be maximized while minimizing the price.

#### PIC18

The first company the group looked into was Microchip, a leading provider of microcontroller and analog semiconductors. Of their hundreds of options, the group specifically at an 8-bit MCU with as many IO pins they could offer. The group managed to come across the very powerful, efficient PIC18, specifically the PIC18F97J94.

This chip is extremely cheap for the amount of power and number of features that it contains. With features like Vbat mode, this chip can sleep and draw on average roughly 60 nA. It has multiple, switchable clock modes for power management and optimum performance for a given application. Can drive 480 segments of LCD display which was one feature the group was thinking about adding in at one point. It even has built in USB adaptable ports, which would take implementing USB to a simpler level. There is also a good amount of serial ports for all the needs of the group’s peripherals.

Despite all of these awesome features, there’s one fact about this microprocessor that could be a deal breaker. The default programmable language used is MIPS assembly code which is something the group’s programmers are not thrilled about. It is possible to obtain a C compiler for this chip, but it is not recommended due to the fact that C will use more memory. Once again, the programmers prefer a chip that can utilize Arduino and its libraries. Below is a list of specs for comparing to the other microcontrollers: Table 3.3.G:

Table 3.3.G - PIC18F97J94 Basic Spec List

|  |  |
| --- | --- |
| Property | Value |
| Program Memory (Flash) | 128KB |
| CPU Speed | 64MHz (16 MIPS) |
| RAM | 3,862 Bytes |
| Digital Communications Peripherals | 4 – A/E/USART  2- MSSP(SPI/I2C) |
| USB | Yes 2.0 |
| Operating temperature range | -40C to +85C |
| Pin Count | 100 (87 IO) |
| Operating Voltage | 2V to 3.6V |
| Timers | 4 |
| Comparators | 3 |
| Price | $3.75 |

#### ATmega2560

Another company the group explored was the ever famous Atmel Corporation, world-class supplier of microcontrollers and much more. Keeping close the specs, the ATmega2560 appears to be a great option. It contains higher memory while keeping the other specs in close contention. It also has 4 UART ports which can be used for Bluetooth and USB interfacing, something this project’s extra components need to be able to function. In addition to these quality specifications, this chip can be programmed in C language, specifically using Ardruino and its helpful libraries. Below is a list of comparative specifications, Table Table 3.3.H.

Table 3.3.H - ATmega2560 Basic Spec List

|  |  |
| --- | --- |
| Property | Value |
| Program Memory (Flash) | 256KB |
| CPU Speed | 16MHz |
| RAM | 8 KB (SRAM)  4KB (EEPROM) |
| Digital Communications Peripherals | 4- UART  5- SPI  1- TWI(I2C) |
| USB | No |
| Operating temperature range | -40C to +85C |
| Pin Count | 100 (86 IO) |
| Operating Voltage | 1.8V to 5.5V |
| Timers | 6 |
| Comparators | 1 |
| Price | $18.75 |

#### ATmega6450

Also from Atmel, there is the ATmega6450 which is similar to the 2560 discussed above. Despite being a higher number than the previously discussed ATmega, it is actually on the lower end for specs. The CPU speed remains the same but there is less RAM and programmable flash. Although there are the same number of max ports, fewer are able to be used for IO functions. Digital communication peripherals, preferably UART, are also reduced which could hinder the use of a Bluetooth module. Table 3.3.I below contains spec data to compare to the rest of the devices:

Table 3.3.I - ATmega6450 Basic Spec List

|  |  |
| --- | --- |
| Property | Value |
| Program Memory (Flash) | 64KB |
| CPU Speed | 16MHz |
| RAM | 4 KB (SRAM)  2KB (EEPROM) |
| Digital Communications Peripherals | 1- UART  2- SPI  1- TWI(I2C) |
| USB | No |
| Operating temperature range | -40C to +85C |
| Pin Count | 100 (69 IO) |
| Operating Voltage | 1.8V to 5.5V |
| Timers | 3 |
| Comparators | 1 |
| Price | $10.36 |

#### Microcontroller Decision

After all is said and done, the group requires the chip be able to take on the Arduino boot loader. The libraries from Ardruino will make the group’s lives simpler and it will be able to focus more on functionality of the device. This fact takes the PIC18 out of contention. Between the two ATmega chips, there is one which is more powerful and more expensive. The group feels that worth the extra cost for the amount of power it is increasing. Therefore the group selects the ATmega2560, at least two of them, for the final product. Even though this is the more powerful and more costly ATmega, the group already has a demo board of this chip for immediate testing.

# Project Hardware and Software Design Details

## Overall Design

The overall design of the device needs to be attractive and professionally presented within reason of the fact that the device is a prototype. In presenting overall design choices it is first helpful to revisit some of the physical characteristics mentioned in the Specifications portion of this paper.

* Overall dimensions should not far exceed 20” x 20” x 4”
* The device will be lightweight and easy to transport
* The outer package will be attractive and durable
* The top surface of the device will be removable allowing access to the input/output circuits within
* The top surface must allow visible light and infrared light to pass through it
* The top surface should diffuse the visible light generating the appearance of a square cell
* All other control circuits will be easily accessible
* Each cell will be approximately three cubic inches and contain one input/output circuit

In addition to this the group hopes to build the device at as little cost possible. It is also important that little time be spent on the structural components of the device. With all of this in mind, the group decided that it should pursue some commercially available vessel for the overall construction of the device. Considering all of this, the group chose to use plastic divided containers to construct the device. A photograph of a container similar to the ones the group hopes to use are seen below in Figure 4.1.A



Figure 4.1.A - Photograph of closed storage container

As can be seen from the photographs these containers fit all of the required specifications. They are light weight and inexpensive. They are durable and provide protection from outside elements. The top cover is clear and removable. The containers are already divided allowing isolation between the sensor cells. It is also important to note that the containers are commercially available in a number of different overall sizes and inter-dimensional cell division sizes. The group also intends to use a completely undivided container mounted below the divided containers to house the primary control circuit.

While these containers are almost a perfect solution to the overall design needs for the project there are a few primary complications that they create, which the group has addressed. The first is that there may be need to modify the containers to get the ten by ten grid needed for the project. Secondly, the cells are completely sealed so the group will have to make openings to allow for the passage of wires. Also, the bottom of the containers is clear which makes the control and other circuits visible from the outside. This is coupled with the issue that the clear dividers between the cells will allow light and infrared waves to pass between cells causing distortion and interference. To address this, the group intends on painting the bottom portion of the containers a solid non-transparent color. Finally, the top surface of the containers is clear, but too translucent which again makes the sensor cells visible. To deal with this issue the group will attempt to ‘tint’ the top surface of the device with polarized window tint. This tint is often one sided allowing visible light to pass out of the device but not back in. This should make the sensor and display cells less visible and also reduce the possibility of ambient light interference with the infrared sensor.

### Printed Circuit Board

It is required that the group use, at a minimum, one professionally printed circuit board (PCB). To accomplish this, the group will learn to use free design software offered from PCB123. This design will be used to generate the board which will be purchased from the supplier. Depending on price the group hopes to have two separate boards constructed. One board will be for the primary control circuit for the device and the other will be for the individual sensor cells.

The primary circuit will hold all of the main components necessary for the operation of the device and its features. This includes the primary microcontroller, the secondary microcontroller, all of the LED drivers, the Wi-Fi chip, the two USB interfaces to connect to the microcontrollers, the decoders used for the sensor circuits, power for the board, and all input and output connectors necessary to interface with the rest of the device. A simple diagram of this board can be seen below in Figure 4.1.C. The second circuit the group would like to mount on a PCB is the input/output circuit for each of the devices cells. This would contain the sensor circuit components as well as the RGB LED. Again a simple diagram of this circuit is given in Figure 4.1.D for illustrative and initial stage planning purposes.

In each of these cases it is vital that the following criteria are met for the design of the printed circuit board for the sake of cost and ease of troubleshooting:

* Each board should be designed in such a way that only one power input and ground are needed, any additional voltage regulation should be conducted on the board
* Each board should be designed to have a minimum number of layers
* The circuits should be designed to reduce the size of the PCB to a minimum
* Each board should contain all necessary components to perform its designed function

Figure . - Primary control circuit initial PCB layout

Master

MCU

Secondary

MCU

MUX

Bluetooth

LED Out

Blue

LED Drivers

Green

LED Drivers

MUX

Sensor Request

Sensor Return

LED Out

Power In

LED

Out

Red

LED Drivers

USB

USB

Figure . - Input/Output PCB initial layout

& Gate

Vcc

Ground

Resistors

RGB

Resistors

Return

Selection lines from microcontroller

IR LED

Photodiode

## Hardware

### Microcontroller

This project will be using Atmel ATmega2560 microcontrollers for the group’s LED Board. Specifically, one will be used as a primary controller to actually run the current program logic, and the second to serve as a low-level peripherals monitor and hardware driver.

An ATmega2560 microcontroller is an 8-bit, low-power microcontroller. It has 128 Kbytes of programmable flash, with a write and erase cycle limit of 10,000. It also has 4 Kbytes of Electrically Erasable Programmable Read Only Memory (EEPROM), with a write and erase cycle limit of 100,000. One of the major features of this microcontroller is the high count of input and output pins. Altogether, the Atmega2560 has 86 programmable I/O lines, with 54 specific digital pins. It also has 32 general-purpose registers.

### Power Supply

Before choosing a power supply the group must first produce a design for a general power distribution scheme. This scheme should meet the following parameters:

* The input voltage will be 120VAC, 60Hz
* Overcurrent protection should be placed appropriately
* All power wiring should be safe and professional in appearance
* The input will be passed through a commercial supply which must
  + Supply the appropriate voltages to all components
  + Be capable of supplying the necessary current for full device operation
  + Be capable of adjusting its output based on the anticipated varying power needs of the device

A general power distribution scheme can be seen in Figure 4.2.A. This figure shows the basic elements which need power supplied, as well as the operating voltage of the devices. This table also displays the need for a 5VDC to 3.3VDC voltage regulator. This device will be incorporated in the final PCB design.

120VAC

Sensor Circuit (x100)

RGB LED:

1.7VDC-3.7VDC

IR LED:

1.3VDC-1.7VDC

AND GATE:

4.5VDC-5VDC

Primary control circuit (x1)

Microcontroller (x2):

1.8VDC-5.5VDC

Bluetooth Device:

3.3VDC

Demultiplexers (x2):

2VDC-6VDC

LED Drivers (x19):

3VDC-5.5VDC

5VDC

5VDC

3.3VDC Regulator

Power Supply

Figure . - General power distribution chart

To select the appropriate power supply it is also necessary to know the total power consumption of the device. Table 4.2.A lists the power consumption characteristics of each component in the device, the number of those components in the device, the total power consumption for those devices, and the total device power consumption.

Table 4.2.A - Estimated device power consumption

|  |  |  |  |
| --- | --- | --- | --- |
| Component | Component Max Power Consumption | Number present | Total Consumption for Device |
| Microcontroller | 14 mA | 2 | 28 mA |
| Led Driver | 80 mA | 19 | 1.520 A |
| Bluetooth Device | 25 mA | 1 | 25 mA |
| Decoders | 50 mA | 2 | 100 mA |
| RGB LED | 100 mA | 100 | 10.000 A |
| IR Emitter | 150 mA | 100 | 15.000 A |
| IR Detector | 50 mA | 100 | 5.000 A |
| And Gate | 24 mA | 100 | 2.400 A |
| Total Estimated Power Consumption |  |  | **34.073 A** |

This shows that the device requires approximately 34.1 amps of current at full draw. Adding a twenty percent cushion takes the current up to 40 amps. Using this and the fact that the primary circuit voltage for the device is 5VDC the group has chosen the TDK-Lambda LS200-5 Series power supply. Table 4.2.B shows the important characteristics of the device. As can be seen by the table, not only does the power supply meet the requirements of the device it is also compact and contains a fan for self-cooling. The device also has two separate outputs so the line to the primary circuit can be run independent from the input/output circuits in the sensor cells.

Table 4.2.B - Power Supply characteristics (TDK-Lambda)

|  |  |
| --- | --- |
| Input voltage | 85 – 263 VAC |
| Input Frequency | 47-63 Hz |
| Output voltage | 5 VDC |
| Max current | 40 A |
| Typical Efficiency | 72- 75 % |
| Enclosed fan | Yes |
| Overvoltage protection | 5.75 – 6.75 VDC |
| Overcurrent protection | 105% nominal peak |
| Over Temperature Protection | Yes |
| Size | 7.8 x 3.9 x 1.6” |

### Sensor Input

At several points during operation, especially in game mode, the device will require input from the user to control direction or selection of game components. One method of achieving this input will be to allow the user to place their hand over one of the device’s cells. This means that the device must be capable of detecting the presence of the user’s hand over one or multiple cells at a time. It has been decided that the detection of objects will be achieved through the use of an infrared sensor. To aid in the design of this sensor circuit the group has compiled a list of specifications which the sensor must abide.

* The sensor must return a digital signal capable of being read by the microcontroller
* The range of the sensor should be between two and three inches to allow the detection of an object at or just above the devices surface
* The circuit should consume little power
* The circuit should employ any practical tactics available to eliminate interference
* The return signal from a particular cell will only be available when requested by the microcontroller
* All sensor return lines will be tied together to reduce the number of pins needed on the microcontroller
* All sensor cells will be addressed, and an intelligent addressing scheme will be used to reduce the number of pins needed on the microcontroller

#### Infrared Emitter-Detector

To transmit and receive the infrared signal the group will use an LED infrared emitter and a photodiode. These devices are typically sold in pairs with the detector optimized to sense the same frequency of light being given off by the emitter. One example of this is the Infrared LED Emitter and Detector package sold by RadioShack. This package contains an infrared LED and a paired detector with the following specifications listed in Table 4.2.C and Table 4.2.D (www.radioshack.com).

Table 4.2.C - Emitter Specifications (www.radioshack.com)

|  |  |
| --- | --- |
| Reverse voltage | 5V |
| Continuous forward current | 150mA |
| Forward voltage | 1.3V typical, 1.7V max |
| Wavelength (peak emission) | 950nm |

Table 4.2.D - Detector Specifications (www.radioshack.com)

|  |  |
| --- | --- |
| VCEO | 70V |
| VECO | 5V |
| IC | 50mA |
| Total power dissipation | 150mW |
| Peak sensitivity wavelength | 850nm |
| Spectral bandwidth range | 620-980nm |
| Angle of half sensitivity | +/- 20° |

A diagram of the typical circuit using these types of components as a sensor is seen below in Figure 4.2.B - Common emitter detector diode configuration. As can be seen in the figure there are two possibilities for output voltage available. The output V will give low voltage when the sensor does not detect an object and a high output when the sensor detects an object. The output V’ will provide a high voltage when the sensor does not detect an object and a low voltage when something is in range. The value of the output voltage in either case can be adjusted by varying the resistance of the resistor in series with the sensing diode. This level of control is one aspect of the diode circuit that makes it attractive for this application.

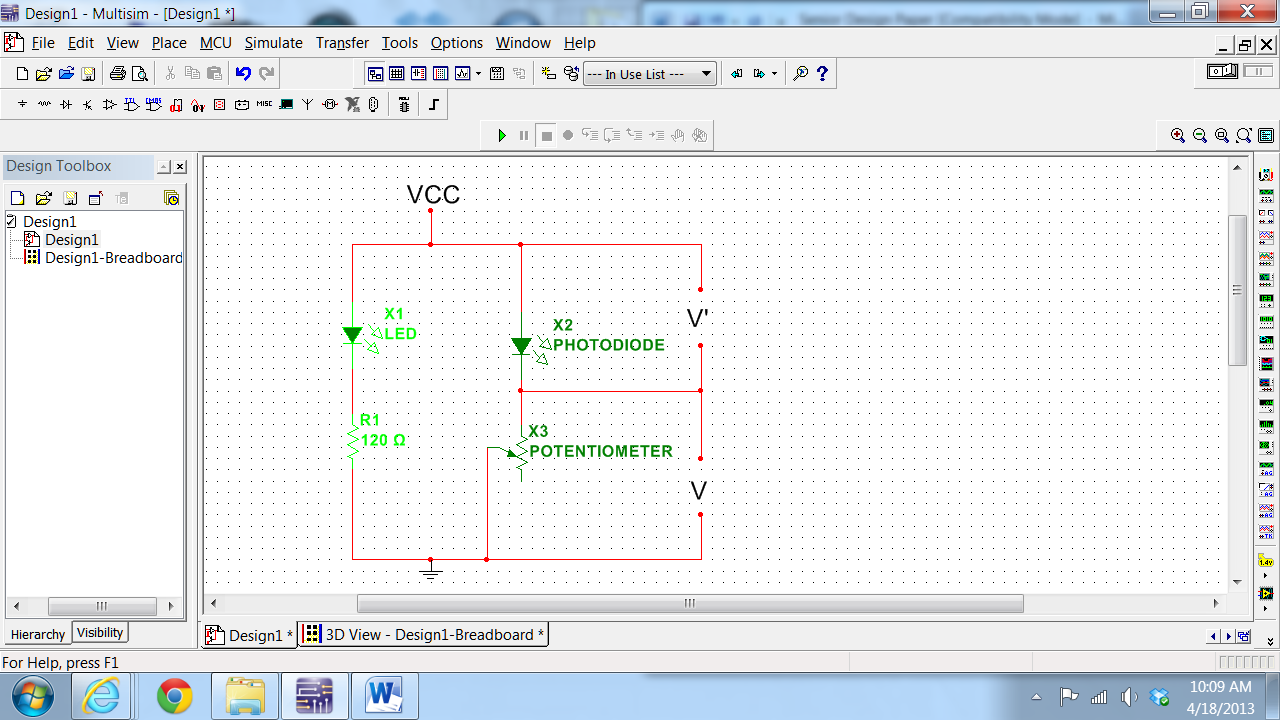


Figure 4.2.B - Common emitter detector diode configuration

#### Input Multiplexing and Positive-And Gate

On each of the individual user input/output cells the group will attach positive logic ‘and’ gate to aid in the selection of a return value. For this the group will use a triple input ‘and’ gate with each pin assigned as follows. One pin on the ‘and’ gate will be tied to the sensor circuit output. This circuit will then be required to output a high signal to the gate when an object is in the detection zone. The second input to the gate will be tied to the row selection signal from the controller, and the third input to the ‘and’ gate will be tied to the column select signal from the microcontroller. This is illustrated below in Figure 4.2.C.

Figure . - Use of ‘And’ gates in sensor circuit

Sensor circuit from

Figure 4.2.3.A

&

V

Sensor circuit from

Figure 4.2.3.A

&

V

Sensor circuit from

Figure 4.2.3.A

&

V

Sensor circuit from

Figure 4.2.3.A

&

V

MCU Demultiplexer

Row Select

Column Select

Return to MCU

The row and column selection signals will originate from the microcontroller, but as later discussed will directly come from a demultiplexer. These signals will be sent out in a sequential order to each of the cells. This will allow the device to scan through each of the inputs rapidly one at a time. The only time a positive signal will be returned is when the sensor signal is high in addition to the row and column being “pinged” by the controller at that time. This cycle will be completed as often as possible by the program to deliver fluid input to the device.

This configuration will allow all of the return lines from the ‘and’ logic gates to be tied together. The program will then store a value of 1 or 0, as received from the device, into a two dimensional array mimicking the construction of the device. The values of this array will be used to manipulate the program as necessary. In modes where the user only has limited controls such as up, down, left, right, only the cells available for selection can be scanned lending to increased response time from the device.

Given the parameters of the addressing scheme, as well as other requirements such as power usage, the group has chosen the Texas Instruments SN54ACT11 Triple 3-Input Positive-And Gate. While the group only intends on using one of the three ‘and’ gates on this chip for each of the sensor circuits, the integrated circuit’s common use lends to an ease of availably and low cost that is attractive to the group.

On the device pins Vcc and GND are the operating voltage and ground respectively. #A, #B, and #C are the ‘and’ gate inputs for an individual logic gate where # represents gate 1, 2, or 3. The #Y pin is the output for the corresponding inputs 1, 2, or3 (Texas Instruments, SN54ACT11). The chip has the following applicable parameters listed in Table 4.2.E:

Table 4.2.E - Operational Parameters of SN54ACT11 (Texas Instruments, SN54ACT11 Technical Data Sheet)

|  |  |
| --- | --- |
| Operating free-air temperature | −55 to 125 °C |
| VCC Supply voltage | 4.5V to 5.5V |
| High-level input voltage | 2V |
| Low-level input voltage | 0.8V |
| Input voltage 0 VCC | 0 VCC |
| Output voltage | 0 VCC |
| High-level output current | −24mA |
| Low-level output current | 24mA |
| Input transition rise or fall rate | 8 ns/V |

#### Demultiplexers

In addition to using ‘and’ gates to reduce the number of pins needed to provide information to the microcontroller the group also plans on using demultiplexers to aid in the selection of rows and columns. Demultiplexers will allow the implementation of the row-column addressing scheme mentioned in earlier sections of this document. Each row and column will be addressed through a demultiplexer allowing the use of only four pins for row selection and four for column selection. These eight pins are a drastic reduction from the twenty needed for a direct row-column connection or the 100 needed for direct cell connection. This scheme is visualized below in Figure 4.2.D.

**Microcontroller**

**Column Select 4:16 Demultiplexer**

4:16

**Row Select 4:16 Demultiplexer**

Sensor circuit from Figure 4.2.D

Figure . - Demultiplexer connection design for sensor circuit

Given the addressing needs of the project in addition to other design requirements the group has chosen to use the Texas Instruments CD54HC154F3A High-Speed CMOS Logic 4- to 16-Line Decoder/Demultiplexer. This integrated circuit has the operational specifications listed below in Table 4.2.F which the group has used to aid in its selection, and will consider in the design on the device.

Table 4.2.F - Operational Parameters of CD54HC154F3A (Texas Instruments, CD74HTC154 Technical Data Sheet)

|  |  |
| --- | --- |
| Temperature Range | -55oC to 125oC |
| Supply Voltage Range, VCC | 2V to 6V |
| DC Input or Output Voltage, VI, VO | 0V to VCC |
| High Level Input Voltage  Vcc:  2V  4.5V  6V | 2V  4.5V  6V |
| Input Rise and Fall Time (Max)  Vcc:  2V  4.5V  6V | 1000ns  500ns  400ns |

### LED Output

The secondary microcontroller will be able to set LED light strength and color for each individual board cell. This will be done through a combination of decoding used with LED controllers. Each TLC5941 LED controller will be able to support 5 RGB LEDs. It will have Pulse-Width Modulation (PWM) support to allow variations in color brightness. Essentially, this gives the group several different light levels for each color in RGB LEDs.

Depending on the number of possible PWM levels, the number of total colors is a power of three. For example, having 16 possible levels per color gives 163 = 4096 possible colors. The larger the number of levels, the smoother transitioning between colors will be in scaled mode.

The group will have 100 RGB LEDs arrayed into a (10x10) matrix. Each RGB LED anode will be terminated to a 5 V voltage supply, and every LED cathode will be terminated to a channel pin on the TLC5941. Because every RGB LED consists of 4 pins (3 cathodes and an anode), this means that an LED driver channel will be needed for every LED cathode to terminate to. A total of 19 LED drivers will be used to control the grayscale fading of every LED. The Grayscale fading of all 16 LED channels on the driver will be controlled by Microcontroller. There are 4 pins that must be terminated between the microcontroller and a single LED driver in order to control the grayscale of the LEDs. This pixel-level LED control design circuit is depicted in Figure 4.2.E.

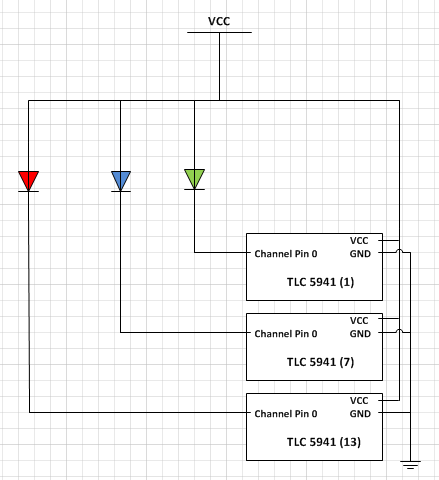


Figure 4.2.E - LED unit pixel design circuit

#### TLC5941 Grayscale Control

In order for the TLC5941 to operate in Grayscale mode, the “MODE” pin must be set into a low voltage (GND) state. The “SCLK”, and “GSCLK” pins control the serial and grayscale clocks of each LED driver, and all 19 LED drivers will share the same “SCLK” and “GSCLK” pins on the microcontroller. Every “SIN” and “XLAT” pin will be used to input and latch the 12-bit grayscale PWM value onto each unique LED driver’s channel registers. This means that a total of 40 pins(19 for the “SIN” pins, 19 for the “XLAT” pins, 1 for the “SCLK” pin , and 1 for the “GSCLK” pin) must be allocated from the microcontroller to the LED drivers. When the grayscale serial data is inputted into an LED driver, 192 bits of data is required to control all the channel ports. This 192 bit word is inputted with the “SCLK”, and the 192 bit word is then broken up into 16x12 bits of data, which are latched onto the grayscale registers of every channel pin. This means that anytime 1 LED must undergo a color transition, 16 channels are accessed in 1 command from the MCU. A depiction of an LED Grayscale cycle is depicted in Figure 4.2.F.

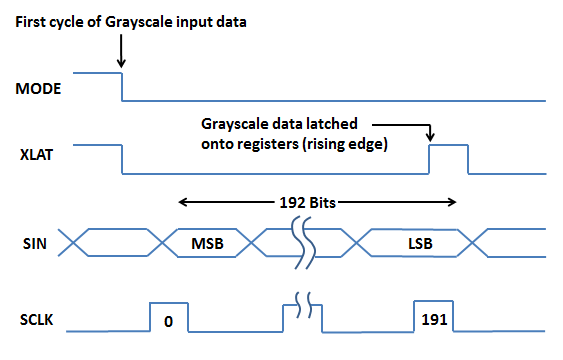


Figure 4.2.F - Grayscale cycle of TLC 5941

#### TLC5941 Dot Correction

Because the project will require 100 RGB LEDs, and each RGB LED consists of 3 distinct bulbs, it will be unlikely that all of the LED bulbs will respond with a uniform luminosity to the same input current pulse. The TLC5941 accounts for this inconsistency between the driver’s 16 output channels by fine-tuning the output current pulse with the built in Dot-Correction feature. In order for the TLC5941 to operate in Dot Correction mode, the “MODE” pin must bet set to a high voltage state (VCC). The “SIN” pin will input 96-bits of data at a time, which is broken up into 16x6 bits of dot correction data for each channel. The “XLAT” pin will set the internal registers of the channels to 96-bit word lengths. A depiction of a Dot Correction mode cycle is depicted in Figure 4.2.G.

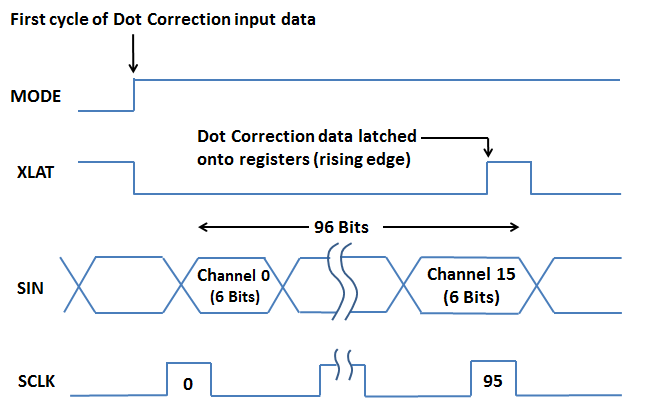


Figure 4.2.G - Dot Correction cycle of TLC 5941

### USB-UART Interface

The group opted to have a direct USB-UART interface between the MCU and the user PC. The ATmega2560 comes packaged with an integrated test board, and this board is already USB enabled. The group is conscientious of software troubleshooting problems that might come up when creating the USB-UART interface and the group decided to emulate the ATmega2560’s USB port interface. The board interface uses the ATmega16u2 to interface serial data between the PC and MCU, so the group would have to emulate this interface design, which is depicted in Figure 4.2.H. This approach was chosen because both microcontroller units are produced by the same manufacturer, so both chips would be programmable using the same software development set provided by Arduino.

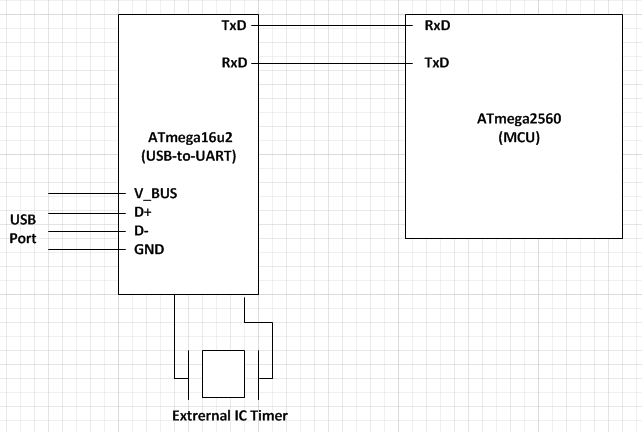


Figure 4.2.H - USB interface design using the ATmega16u2

### Bluetooth

With the RN-42 decided upon, the group should now integrate the module with its system PCB. Pins 1, 12, 28 and 29 should all be grounded. The only pins that the group must utilize are the UART-TX, UART-RX and VDD. VDD requires an input voltage of 3.3V ± 10%. UART-TX will go to one of the RX pins of the group’s microcontroller, while UART-RX will go to one of the TX pins. Figure 4.2.I shows the connections to the secondary microcontroller.

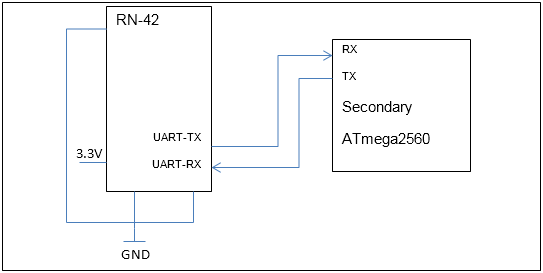


Figure 4.2.I - Pin Connections for RN-42 to microcontroller

### RGB LEDs

The group chose the M500RGB4D as its RGB LED selection. The group had an extensive selection to choose from because for the most part there is little discrepancy in the design of these devices between companies. The two major types of LEDs differ in the direction that the current flows through them. One design is for there to be a common anode, with three cathode pins, one for each color. The other is the reverse, one common cathode with three anode pins.

In order for the LED drivers to function properly, the group has to select the f type with three ground pins and one voltage in pin. The LEDs are required to put the ground side going to the channel in the LED driver. If they all have the same ground, it will provide zero control to the driver selecting colors. Table 4.2.G below shows the basic specs of the LED:

Table 4.2.G - Specs for RGB LED M500RGB4D-A data sheet (lc-led.com)

|  |  |  |
| --- | --- | --- |
| Parameter | Minimum Value | Maximum Value |
| DC Forward Current | 30 mA | 100 mA |
| Forward Voltage (Red) | 1.7 V | 2.6 V |
| Forward Voltage (Green) | 3.0 V | 3.7 V |
| Forward Voltage (Blue) | 3.0 V | 3.7 V |
| Luminous Intensity (Red) | 3000 mcd | 5000 mcd |
| Luminous Intensity (Red) | 4000 mcd | 6000 mcd |
| Luminous Intensity (Red) | 1800 mcd | 3200 mcd |
| Operating Tempuature | -40C | 80C |

## Software

### Microcontroller

This project uses two Atmel ATmega1280 microcontrollers. The primary microcontroller will be in charge of the main logic, and dividing up tasks. The secondary microcontroller will act as a hardware driver and peripherals monitor. In order for these two microcontrollers to communicate and work in parallel, a software-based communications scheme must be developed.

The primary microcontroller will be in charge of storing and running installed programs. It acts as both the secondary storage via EEPROM, as well as the primary processor for the programs. It will be able to interface with the secondary microcontroller for hardware driving and peripheral support. It will be connected to a USB port for an easy way to install programs. Specifically, it will interface with the group’s LED Board GUI programming environment discussed in Section 4.3.3. While the primary microcontroller can be reprogrammed via EEPROM with different user programs, the secondary microcontroller is meant to have the same “firmware” for driving the hardware.

The secondary microcontroller will be used for low-lever peripheral and hardware driving. It acts as the hardware driver, and will spend its time processing sensor input and handling output to the LEDs. The second microcontroller can monitor input from two primary sources: an android phone connected via Bluetooth, and the touch sensors. It will have the ability to monitor activity and provide an output buffer for interrupts. It can also receive commands from the primary microcontroller to enable or disable monitoring activity for particular sensors or Bluetooth control. It can receive a simple signal to be sent to the android phone via Bluetooth from the microcontroller, and it will handle the low-level processing for that command. It can provide functions to the primary microcontroller for setting specific LED colors, or scaling LED colors over a period of time. Figure 4.3.A shows the connections to the dual microcontrollers from a software perspective.

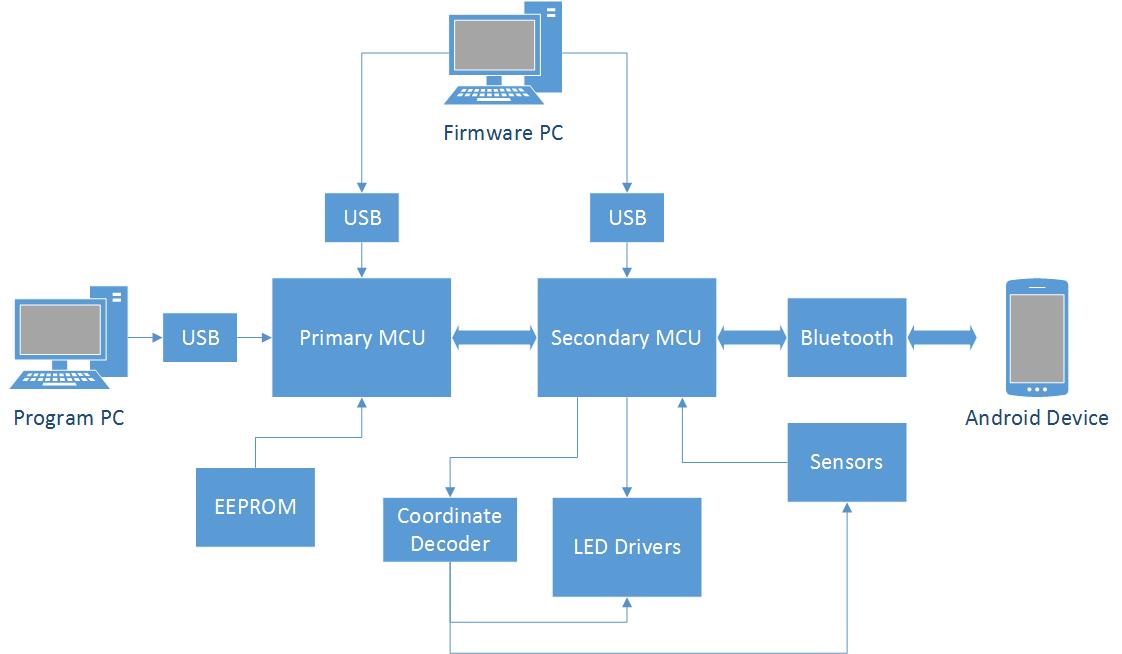


Figure 4.3.A – Dual Microcontroller Connections

Having this setup allows less processor demand per processor, than if only one was used. It also allows an interrupt system if needed, where the secondary microcontroller will constantly monitor any group of particular sensors and provide an interrupt signal to the primary microcontroller. Meanwhile, the primary microcontroller can continue with other logic instead of being bogged down with both tasks.

To communicate between the two microcontrollers, a minimum of 5 Digital Input/Output (DIO) pins is needed. This allows up to 32 unique command messages. Additionally, any command parameters (such as color values) are limited to a maximum value of 32. However, in most cases this could be broken up in to separate Red, Green, and Blue (RGB) values. Again, this is a minimum. For each additional pin connection, it doubles the value maximum. Having even 64 would allow both upper-case and lower-case values, as well as various control message and punctuation characters, as far as Bluetooth transmissions go.

Table 4.3.A - Driver Function Descriptions

|  |  |
| --- | --- |
| Function Name | Description |
| Initialize Driver | Will display a splash screen, and then initialize all variables. By default LED Updating, Sensor Monitoring, and Bluetooth Monitoring will be disabled. All LEDs will be set to “black” color, or off. |
| Reset Driver | Will clear any allocated memory, and then initialize the driver. In effect, this is a Disable Driver function which calls the Initialize function. It is like a “reset” button on a PC. |
| Connect Bluetooth | Will change the Bluetooth status to discoverable, and wait up to 1 minute to connect to an android device. A “Discoverable” message will be displayed on screen. Upon timeout, a “Connection Failed” message will be displayed. If the connection succeeds, a “Connected” message will be displayed. |
| Disconnect Bluetooth | If there is a device connected, the connection will be terminated. Bluetooth monitoring will be disabled. |
| Enable Bluetooth | If there is a Bluetooth device connected, monitoring will be enabled for interrupts and message sending. |
| Disable Bluetooth | If Bluetooth is enabled, it will be disabled. |
| Send Bluetooth Message | If Bluetooth is enabled, a message will be sent to the controller. This is done through the control message by first sending the message size (maximum of 32), and then sequentially the value of each array element. Finally, an ID is sent that will be received if that option is clicked on the Android device. |
| Check Bluetooth Interrupt | Will send a bit signal status if there is a Bluetooth message interrupt. |
| Get Bluetooth Interrupt | If there is a Bluetooth interrupt, the first message will be sent and removed from the buffer. |
| Clear Bluetooth Interrupt | The Bluetooth incoming message buffer will be cleared and the interrupt status will be set to false. |
| Enable Sensor | Will allow interrupts from a particular sensor. Interrupts for a single sensor can only be click. For consecutive sensors, dragging events will be enabled. |
| Enable Sensor Line | Will enable sensors within two specified endpoints of a certain specified thickness. |
| Enable Sensor Empty Circle | Will enable sensors on the circumference of a circle, specified by a center point, radius, and thickness. |
| Enable Sensor Full Circle | Will enable sensors with in a circle, specified by a center point and radius. |
| Enable All Sensors | Will allow interrupts from all sensors. |
| Disable Sensor | Disables interrupts for a particular sensor. |
| Disable All Sensors | Disables interrupts for all sensors. |
| Check Sensor Interrupt | Will send a bit signal if there is a sensor interrupt. |
| Get Sensor Interrupt | If there is a sensor interrupt, the signal ID will be sent and removed from the buffer. |
| Clear Sensor Interrupt | The sensor signal buffer will be cleared and the interrupt status will be set to false. |
| Enable LED Update | Will enable LED color updating to the LED Drivers. |
| Disable LED Update | Will freeze all communications to the LED Drivers. |
| Set LED Color | Will set a particular LED to the specified color. |
| Set All LEDs Color | Will set the color for all LEDs to the specified color. |
| Scale LED Color | Will scale a particular LED from its current color to the specified color over a specified period of time. |
| Scale All LEDS Color | Will scale all LEDs from their current colors to the specified color over a specified period of time. |
| Draw Line | Will set all LEDs between two specified coordinates and for a specified thickness to a specified color. |
| Draw Empty Circle | Will set all LEDs along the perimeter of a specified circle center, radius, and thickness to a specified color. |
| Draw Full Circle | Will set all LEDs within a specified circle center and radius to a specified color. |
| Display Text | Will scroll a text message across the screen. A specified text, shadow, and background color will be specified. |

Table 4.3.A describes the message needed to be sent between the two microcontrollers. Table 4.3.B describes the parameters and return values that correspond to each of these message. This is important because the secondary microcontroller must recognize each of these messages and be able to expect additional parameters necessary for each function. It must also return the correct return value based on the commanding function.

Table 4.3.B - Driver Function Message Enumeration, Parameters, and Return Values

|  |  |  |  |
| --- | --- | --- | --- |
| Function Name | Control Enumeration | Parameters | Return Value |
| No Operation | 0b00000 (0) |  | boolean |
| Initialize Driver | 0b00001 (1) |  | boolean |
| Reset Driver | 0b00010 (2) |  | boolean |
| Connect Bluetooth | 0b00011 (3) |  | boolean |
| Disconnect Bluetooth | 0b00100 (4) |  | boolean |
| Enable Bluetooth | 0b00101 (5) |  | boolean |
| Disable Bluetooth | 0b00110 (6) |  | boolean |
| Send Bluetooth Message | 0b00111 (7) | uInt size  char[size]  uInt ID | boolean |
| Check Bluetooth Interrupt | 0b01000 (8) |  | boolean |
| Get Bluetooth Interrupt | 0b01001 (9) |  | uInt ID |
| Clear Bluetooth Interrupt | 0b01010 (10) |  | boolean |
| Enable Sensor | 0b01011 (11) | uInt x  uInt y | boolean |
| Enable Sensor Line | 0b01100 (12) | uInt x1  uInt y1  uInt x2  uInt y2  uInt weight | boolean |
| Enable Sensor Empty Circle | 0b01101 (13) | uInt x  uInt y  uInt radius  uInt weight | boolean |
| Enable Sensor Full Circle | 0b01110 (14) | uInt x  uInt y  uInt radius  uInt weight | boolean |
| Enable All Sensors | 0b01111 (15) |  | boolean |
| Disable Sensor | 0b10000 (16) | uInt x  uInt y | boolean |
| Disable All Sensors | 0b10001 (17) |  | boolean |
| Check Sensor Interrupt | 0b10010 (18) |  | boolean |
| Get Sensor Interrupt | 0b10011 (19) |  | uInt x  uInt y  uInt type |
| Clear Sensor Interrupt | 0b10100 (20) |  | boolean |
| Enable LED Update | 0b10101 (21) |  | boolean |
| Disable LED Update | 0b10110 (22) |  | boolean |
| Set LED Color | 0b10111 (23) | uInt x  uInt y  uInt red  uInt green  uInt blue | boolean |
| Set All LEDs Color | 0b11000 (24) | uInt red  uInt green  uInt blue | boolean |
| Scale LED Color | 0b11001 (25) | uInt x  uInt y  uInt red  uInt green  uInt blue | boolean |
| Scale All LEDS Color | 0b11010 (26) | uInt red  uInt green  uInt blue | boolean |
| Draw Line | 0b11011 (27) | uInt x1  uInt y1  uInt x2  uInt y2  uInt weight  uInt red  uInt green  uInt blue | boolean |
| Draw Empty Circle | 0b11100 (28) | uInt x  uInt y  uInt radius  uInt weight  uInt red  uInt green  uInt blue | boolean |
| Draw Full Circle | 0b11101 (29) | uInt x  uInt y  uInt radius  uInt weight  uInt red  uInt green  uInt blue | boolean |
| Display Text | 0b11110 (30) | uInt size  char [size]  uInt red\_bkgnd  uInt green\_bkgnd  uInt blue\_bkgnd  uInt red\_shadow  uInt green\_shadow  uInt blue\_shadow  uInt red\_text  uInt green\_text  uInt blue\_text | boolean |
| SPARE | 0b11111 (31) |  |  |

For two-way communication, there will need to be dedicated digital lines for each microcontroller to signal when data is valid to be read. For all communications, the primary microcontroller will always signal the secondary microcontroller first to avoid messaging conflicts. Simplified, when microcontroller 1 wants to send a control signal to microcontroller 2, it first sets the appropriate output lines and then sets the signal valid bit. For microcontroller 2, it will see the valid bit go from 0 to 1 and will read the signal and act accordingly. Until microcontroller 2 sees the valid bit go from 1 to 0, and then back to 1 it will not read the signal.

Both sides will have a timeout, in the event a valid bit toggle is not seen by the other microcontroller, where it will reset the valid bit to 0 and then back to 1. The sending microcontroller will wait for the other microcontroller’s valid bit signaling either message acknowledgement or in the case of a return message the valid return signal status.

For all boolean return signals, this will be signified by setting the least-significant bit (LSB), which in this case is 0b00001. For true, this bit will be set along with the signal valid bit for that microcontroller. Similarly, 0b00000 will be used to signify false along with the signal valid bit.

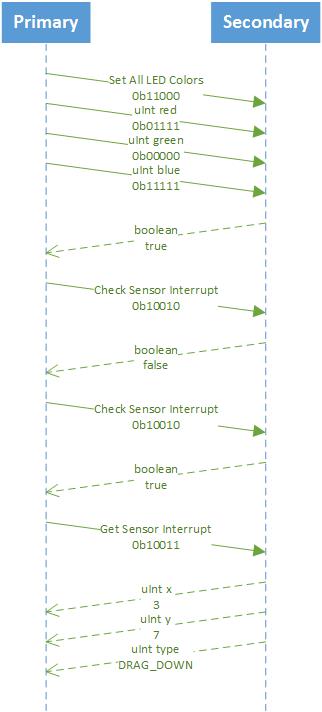


Figure 4.3.B - Microcontroller Control Message Protocol

In the previous example (Figure 4.3.B), the primary microcontroller first wants to set all LEDs to a specified color. It first sends the signal 0b11000 to signify the corresponding control message. Upon receiving this, the secondary microcontroller will expect three consecutive messages for the red, green, and blue values. Upon successful receipt, it will return a true signal. If any error occurs, a false signal will be returned. The primary microcontroller could then go and attempt to resend the message.

Next, say the program wants to get sensor input for its next procedure. In the second stage, it sends the “Check Sensor Interrupt” signal 0b10010. By the secondary microcontroller returning false, it signifies there is no input. A short while later, this stage is repeated but the secondary microcontroller returns true, signifying a signal is in the buffer.

By calling “Get Sensor Input” with 0b10011, the secondary microcontroller sends three consecutive messages following a similar procedure to step one. This signal returns the x and y coordinate of the last sensor with interrupt input. Finally, the type signifies what sort of input was performed by the user, such as clicking or dragging in a certain direction.

There are a few possible signal types that could be sent in a sensor interrupt message. These types include the following: click, double-click, drag\_up, drag\_up\_right, drag\_right, drag\_down\_right, drag\_down, drag\_down\_left, drag\_left, drag\_top\_left. These will be defined in an enumeration in software.

### Android

To make it easy to switch between the functionality of the group’s device, the group will implement a simple android application to select the different modes of the group’s LED table. The design of the application is simple and straight forward. At the very least it will have a page of buttons that when pressed, programs or selects the code to run for that particular function. The group would like to expand upon the application if it has time to. A simple function to implement would be a text input that displays on the LED board from the phone.

This project will be using any Bluetooth enabled Android device for interfacing with the group’s LED Board as a controller. Upon connection, it can request a list of applications currently installed on the board, and will provide a list to the user. The user can select a program, and it will send that program ID to the LED Board which upon receipt should start the program. A program will have the ability to use the Android phone as a controller. The android application will be able to send commands such as “Start Program”, “End Program”, “Restart Program”, as well as basic control commands necessary for games and programs.

The Android device will be able to connect to the LED Board and display one of the pre-programmed layouts for controls. This layout is specified and received from the LED Board. Such layouts will include the following: menu, 4-way direction arrows, 8-way direction arrows, text entry, etc.

#### Using JAVA for Bluetooth in Android

The Android program will be done entirely in JAVA with specific android packages. Using Eclipse IDE, designing the layout of each frame makes “programming” seamless, because of the GUI drag and drop style interface. The real programming will be done on the Bluetooth classes. One class will handle the connecting of the module and transmitting data between the group’s board and the android device. Another class is made to handle the list of available devices which will interact directly with the first. Finally, a class which handles the view for the application and, depending on the interface, also handles the user input. This class will most likely be the one which selects the program to be run on the LED table. A multitude of other classes can be made to handle specific functions, such as a simple controller interface or a virtual grid for touch. Show below in Figure 4.3.C and Figure 4.3.D are two such schemes the group has contemplated using.

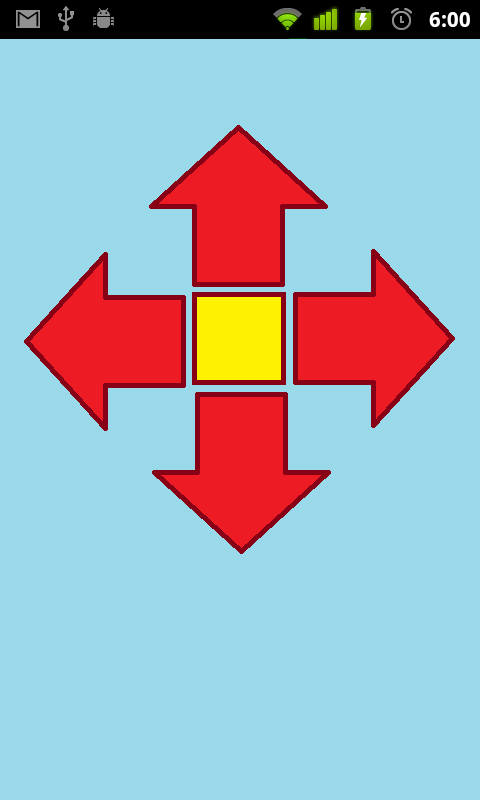


Figure 4.3.C - Example of envisioned Android interface

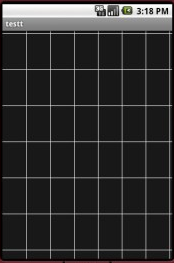


Figure 4.3.D - Grid Layout for touch capability from phone

The first view user will see handles all of the communication between devices, whether it will be a list of programs to start, a controller interface or a touch grid. This activity will function normally so long as there is connecting between the application and the Bluetooth module. While connected, a status bar will show that the device is connected. If there is no connection, and the user tries to initiate a normal action, a message will display and prompt the user to connect. The menu button in this view will make the android discoverable by the RN-42 module on the board; however, it is likely that the Android will be the device that initiates the connection. In the upper right corner will be a search icon to initiate a connection to any device, which will bring up another view. A sample is seen below in Figure 4.3.E, note: The current placeholder application is a simple text input/output scheme.

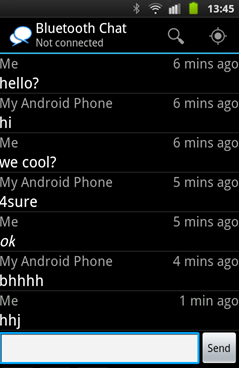


Figure 4.3.E - Basic UI for Android Application

Between the content activity and the device list activity is a class that handles the work for setting up the connections between the devices as well as data transmission between the devices. This class is broken up into multiple threads. One thread will listen for incoming connections. Although it may not be utilized, its implementation is relatively simple and will remain in the code. Another thread will work on actually getting the devices connected to one another. Lastly, a thread will be running while devices are connected, which performs data transmission.

The view that handles scanning for devices, and displays their name and information is invoked by the search icon. This activity will display all previously paired devices as well as ones that scanned for in the area. The interface is a simple window with paired devices in one section and a button to touch to scan and view new devices, shown below Figure 4.3.F. When the user selects a device to connect to, the application will return the MAC address to the middle class.

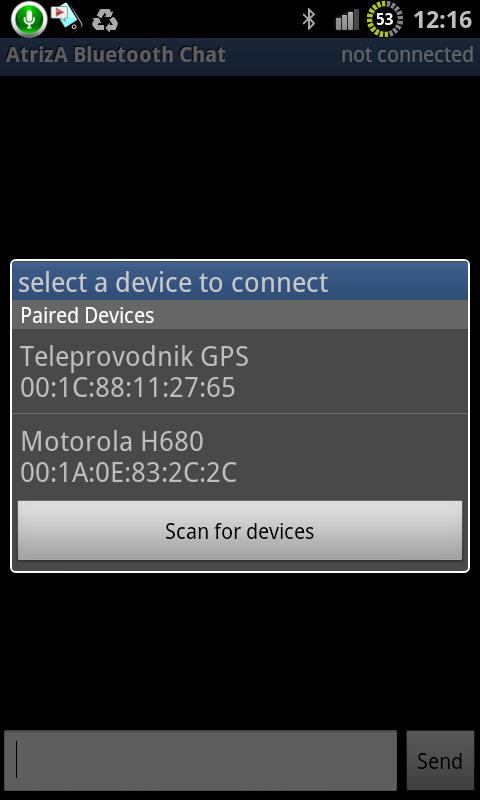
[](http://www.google.com/url?sa=i&rct=j&q=&esrc=s&frm=1&source=images&cd=&cad=rja&docid=WNofEkGzYDlnQM&tbnid=DgkjgchpmkF-oM:&ved=0CAUQjRw&url=http://www.appszoom.com/android_applications/social/atriza-bluetooth-chat_bijaz.html&ei=0ll0UemOGILi8gT1zYGgAQ&bvm=bv.45512109,d.eWU&psig=AFQjCNFYf61KeJ1_KuDPgr7WdqBE6gG5qA&ust=1366665997163914)

Figure 4.3.F - Connection interface for Android Application

All of these activities and views utilize an expansive android library specifically for Bluetooth use. These special classes contain many functions necessary for Bluetooth implementation. Table 4.3.C is an expansive list of functions; the bold is which class these functions belong to:

Table 4.3.C - Functions of the Bluetooth package for Android (developer.android.com)

|  |  |
| --- | --- |
| Function | Description |
| BluetoothAdapter |  |
| startDiscovery() | Begins the discovery of mobile device for other devices to find it |
| cancelDiscovery() | Cancel the current discovery of mobile device so connection isn’t slowed down |
| getDefaultAdapter() | Gets handle to local Bluetooth adapter for startup parameter |
| enable() | Enables the Bluetooth functionality on the device |
| isEnabled() | Returns whether the Bluetooth is currently functioning |
| listenUsingInsecureRfcommWithServiceRecord(String name, UUID uuid) | Creates a listening socket with name and UUID parameters |
| getBondedDevices() | Returns the set of paired devices |
| getRemoteDevice(String address) | Returns Device with address name |
| BluetoothDevice |  |
| createInsecureRfcommSocketToServiceRecord(UUID uuid) | Creates BluetoothSocket socket ready to initiate an outgoing connection to the remote device with UUID lookup |
| BluetoothServerSocket |  |
| accept() | Blocks until a connection with device is established |
| close() | Closes the socket, releasing all associated resources |
| BluetoothSocket |  |
| connect() | Attempts to connect to remote device, returns exception otherwise |
| close() | Closes object and returns and system resources it has |

### Desktop Application

A GUI Application will be developed to provide a graphical user interface specific to programming the group’s LED Board. It will be able to interface to the board via USB (discussed further in the USB section). It will provide a simple way to store multiple programs on to the LED Board, which will be stored in the primary microcontroller’s EEPROM. It provides a graphical environment that almost anyone can use to create a basic program.

The GUI application will have buttons which reflect the different LED and sensor coordinates on the board. A user can select a coordinate, and be provided with a list of the driver functions applicable (discussed further in the microcontroller section, Section 4.3.1). These functions include setting colors, enabling sensor input, enabling or disabling LEDs and sensors, drawing simple shapes, etc.

There will be a simulation mode that will provide a preview of what the program will look like on the board. Buttons will change colors to reflect the LED statuses, and button presses will simulate clicks and dragging. While this reflects the logic behind how the program will run on the LED Board, things such as dragging glitches and delays may not be well reflected.

In addition to the graphical programming provided by the buttons, it will give options for loop, if-then-else logic, and interrupt waiting. A user will also be able to program in a simplified version of ANSI-c code, which directly translates to functions provided in the graphical environment.

Any programs installed to EEPROM will mimic a version of assembly code unique to this project. Any commands will be either logic for loops and jumps, or commands that directly translate to driver control messages. These will be handled in the primary microcontroller, and driver functions will be sent and handled as necessary to the secondary

Another feature for the GUI interface will provide a way to use variables and arrays for program variable storage. This makes the graphical side of the GUI just as robust as the c-coding way. This will however be limited to static sizing for arrays and other restrictions. Overall, all programs “installed” to the microcontroller must not exceed 4 Kbytes of memory.

The GUI Board Programmer has a grid layout that represents each of the board cells. These can be used to select any particular combination of LEDs or Sensors on the board for which the user wishes to program. After selecting what the user wants to use, the user is then presented with a list of possible commands can be used to create a program. Examples of these functions are: monitor this set of sensors and run a corresponding function if set, enable Bluetooth and report if a particular command is sent, change led color directly to this, scale LED color to this over a certain period of time, etc.

And finally, there is an area to the right where the user can see the auto-generated code and enter in their own functions which will be executed if an event occurs. By default, there are three functions: Initialize, State Loop, and Uninitialize. They are fairly straightforward in that Initialize runs first, and can be used to set up the program, such as Bluetooth communication, setting certain LED colors or sensors to watch. State Loop will run forever until an exit function is hit. When that exit function hits, the Uninitialize function will run and the program will terminate. Any functions the user has entered to act as callbacks will be run after the main state loop has finished and if that event was triggered. The main state loop will resume after the function is completed.

When compiling a program, this acts as a compiler in that it will check syntax and produce any errors and warnings it encounters. After syntax and semantics are checked, it will be parsed and a type of assembly code will be generated. This generated file should not be modified by the user. Below is a table listing some of the most important errors and warnings.

Table 4.3.D - GUI Board Programmer Compiler Warnings and Errors

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Syntax Error | Error | Indicated when the code has a syntax error, and there was no way to continue compiling the program. |
| Device Not Connected | Error | Indicated when the user attempts to upload programs to the board, but it is not connected. |
| No Initialize Function | Warning | No Initialize function exists in the program. |
| No State Loop Function | Error | No State Loop function exists in the program. |
| No Uninitialize Function | Warning | No Uninitialize function exists in the program. |
| No Exit Command | Warning | No Exit command was detected in the State Loop. This prevents the program from ever exiting the State Loop programmatically. The Uninitialize function will never be called. |
| No Bluetooth Disconnect Function | Warning | This will only be displayed if Bluetooth functionality is used in the program. The error indicates there is no way to handle a Bluetooth disconnect in the program. If this function is not handled correctly, a command expected only from the Android device could cause the program to freeze and be caught in an endless loop. |

When the user wants to upload programs to the LED Board, they are allowed to select any number of programs at once. The GUI Board Programmer will compile all of the programs together into a type of assembly program and send the final program to the LED Board via USB. While compiling, it will create a menu program that runs at startup, which simply displays a menu of all programs that the user can select. This is a scrollable menu, that will be replicated on and Android device if it is connected.

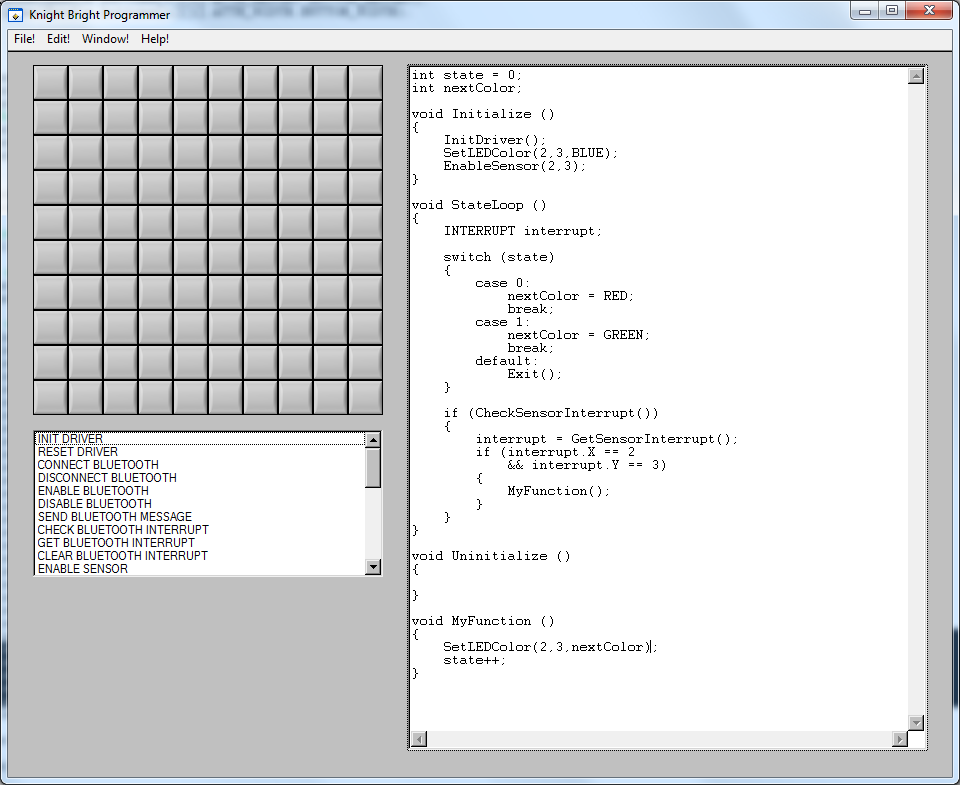


Figure 4.3.G - Prototype GUI Board Programmer

Once a user creates and compiles a program, they will need to send the program to the LED Board. Assuming a USB connection can be made, it is very important to define how the signals will be sent to the board, and how the board will interpret them. The communication will be converted from USB to serial so the RX and TX lines on the microcontroller can be used. For communication, the following messages will be used.

### Assembly

Once a program is uploaded to the LED Board, the microcontroller will be responsible for running the program. Since this program stored in EEPROM is a type of project-specific assembly code, there is a custom definition for each type of assembly function. Many such as jumps and skips are used to mimic basic C logic, but not all functionality is present and can be converted. The following flags will be used in this implementation:

* Overflow (O) – set if the resulting value for an operation overflows an address value size.
* Zero (Z) – set by the COMPARE instruction if the two values are equal.
* Greater (G) – set by the COMPARE instruction if the register value is greater than the specified address value.
* Less (L) – set by the COMPARE instruction if the register value is less than the specified address value.

The microcontroller runs this assembly code in a type of virtual machine environment. The firmware on the microcontroller will take each function in assembly and perform the corresponding high-level function from C. For example, if ADD is encountered, the next two values will be loaded from the following EEPROM addresses, and the microcontroller will add these two values together.

Table 4.3.E - Assembly Instructions

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Value | Parameters | Flags | Description |
| RETURN | 0 |  |  | If this statement occurs during a function, it will return to its calling function. |
| CALL | 1 | ADDRESS |  | Executes the assembly at the defined address. The current position will be saved on the stack and will resume upon function return. |
| ADD | 2 | ADDRESS | O, Z | Adds the value currently in the register with the value at the address and stores the result in the register. |
| SUBTRACT | 3 | ADDRESS | O, Z | Subtracts the value currently in the register from the value at the address and stores the result in the register. |
| MULTIPLY | 4 | ADDRESS | O, Z | Multiplies the value currently in the register with the value at the address and stores the result in the register |
| DIVIDE | 5 | ADDRESS | O, Z | Divides the value currently in the register with the value at the address and stores the result in the register. |
| PUSH | 6 |  |  | Pushes the value currently in the register onto the stack. The stack pointer is incremented. |
| POP | 7 |  |  | Retrieves the value at the top of the stack and stores the value in the register. The stack pointer is decremented. |
| LOAD | 8 | ADDRESS |  | Loads the value at the address and stores into the register. |
| STORE | 9 | ADDRESS |  | Stores the value in the register at the specified address. |
| COMPARE | 10 | ADDRESS | Z, L, G | Compares the value of the register with the value at the address. Will set the corresponding flags. |
| JUMP | 11 | ADDRESS |  | Unconditionally changes the program counter (PC) to the specified address. |
| JUMP Z | 12 | ADDRESS |  | Will jump to the address if the Z flag is set. |
| JUMP L | 13 | ADDRESS |  | Will jump to the address if the L flag is set. |
| JUMP G | 14 | ADDRESS |  | Will jump to the address if the G flag is set. |

### Programming LED Controllers

To interface with the group’s TLC5941 LED drivers, the group will be using the free and open-source Arduino TLC5940 Library. These .cpp and .h files give the functionality to easily send commands to the LED controllers.

* Tlc.init() – Sets up TLC. A value can be passed to default all PWM values to.
* Tlc.clear() – Sets all the grayscale values to zero, but does not send them to the TLCs.
* Tlc.set(channel,value) – Sets the specific channel (0-15) to the specific PWM value.
* Tlc.update() – Sends all set grayscale values to the LED driver.

### Android Interfacing

To interface with the group’s Bluetooth module requires only a serial RX/TX connection. The device is readily available for any Android device to connect to. Once connected, simple messages can be sent between the two. It will be handled in software how those messages are interpreted. From preliminary testing, the group has determined that there is no reliable way to send a whole string message, as it is often broken up. The group must allow the software to wait for an end-of-transmission character before it considers the message finished. Additionally, checksums will be used to eliminate static or transmission failures.

For the microcontroller, the group will use the Software Serial library built in to the Arduino development environment. The group will simply open up a connection with the intended clock-rate on the necessary ports, and the devices can easily send and receive messages.

The group will need to have a protocol that is used to communicate with the Android device to and from the microcontroller on the LED Board. It has already been determined from preliminary prototyping that the group must use some sort of end-of-transmission (EOT) character to indicate the transmission finish. For single byte commands, this would not be an issue. However, for multibyte messages, the receiving end has no way to determine the finish of the message without some other indicator. Additionally, the group will need to include a checksum for each transmission to verify the correct message was received. All messages received should be acknowledged, preferably with a unique value to that message.

These needs force the group to create a higher-level network protocol that gives message integrity and allows message acknowledgement. This protocol will be similar in fashion to UDP messages in Ethernet. All transmissions are sent without any lower-level message handling, so any message integrity must be managed by higher-level programming.

This will be a serial protocol, opposed to parallel where multiple messages will be sent at the same time before acknowledgement receipt. Each message must be acknowledged before the next message is sent. Timeouts will be needed in the event an acknowledgement is not received. The group will use the message count to acknowledge packet receipt. The message structure for each “packet” is described below.

A setup packet will be used to set the message count for both senders. This message count must be acknowledged before any data can be sent.

Table 4.3.F - Packet Field Contents

|  |  |  |
| --- | --- | --- |
| Field | Size | Description |
| Header | 1 Byte | Describes what the contents of the message contain. |
| Message Count | 1 Byte | Incremented separately for both sender and receiver. |
| Message Size | 1 Byte (optional) | Describes the size of the message contents. |
| Contents | Determined by message size field. | Will contain a value relevant to the message type. |
| Checksum | 1 Byte | An overflowed sum of all message contents. This has 256 possible values. |
| EOT | 1 Byte | End of transmission. Contains a value of 0. |

For each message, the group will have a header which is a one byte field which gives an enumerated value for the type of message. The message size field will only be used in the case where there is not already a defined byte size. A list of possible header values is given below:

Table 4.3.G - List of possible Header Values

|  |  |  |
| --- | --- | --- |
| Name | Value | Description |
| SETUP | 1 | Used when first connected to set/reset the message count to some random number for both the sender and receiver. |
| ACKNOWLEDGE | 2 | Used only in response to a received packet. This acknowledgement does not need a response. |
| LAYOUT | 3 | Command determined by the layout desired from the program. Following messages will be based on the layout selected. |
| COMMAND | 4 | Only used from the Android device to the microcontroller. |
| BYTE | 5 | Always a one-byte message that contains an integer between 0 and 255. This can also be used for a single character message. |
| SHORT | 6 | Always a two-byte message that contains an integer between 0 and 65535. |
| INTEGER | 7 | Always a 4-byte message that contains an integer between 0 and 2^32. |
| STRING | 8 | Will contain the number of bytes in the message size field. This will typically be character-based. |

In the event the acknowledgement is not received, the sender will resend the message after a timeout. There are two possible situations that can happen. The first is that the receiver really didn’t receive the message, and the message should be resent. The second is that the receiver actually received it and the acknowledgement never made it back. In this case, the receiver will have to watch out for the same message count number. It will acknowledge it again if the same message is received twice.

# Design Summary of Hardware and Software

## Hardware

### Overall

The following list defines the overall specifications for designing Knight Bright.

* Overall dimensions should not far exceed 20” x 20” x 4”
* The device will be lightweight and easy to transport
* The outer package will be attractive and durable
* The top surface of the device will be removable allowing access to the input/output circuits within
* The top surface must allow visible light and infrared light to pass through it
* The top surface should diffuse the visible light generating the appearance of a square cell
* All other control circuits will be easily accessible
* Each cell will be approximately two cubic inches and contain one input/output circuit

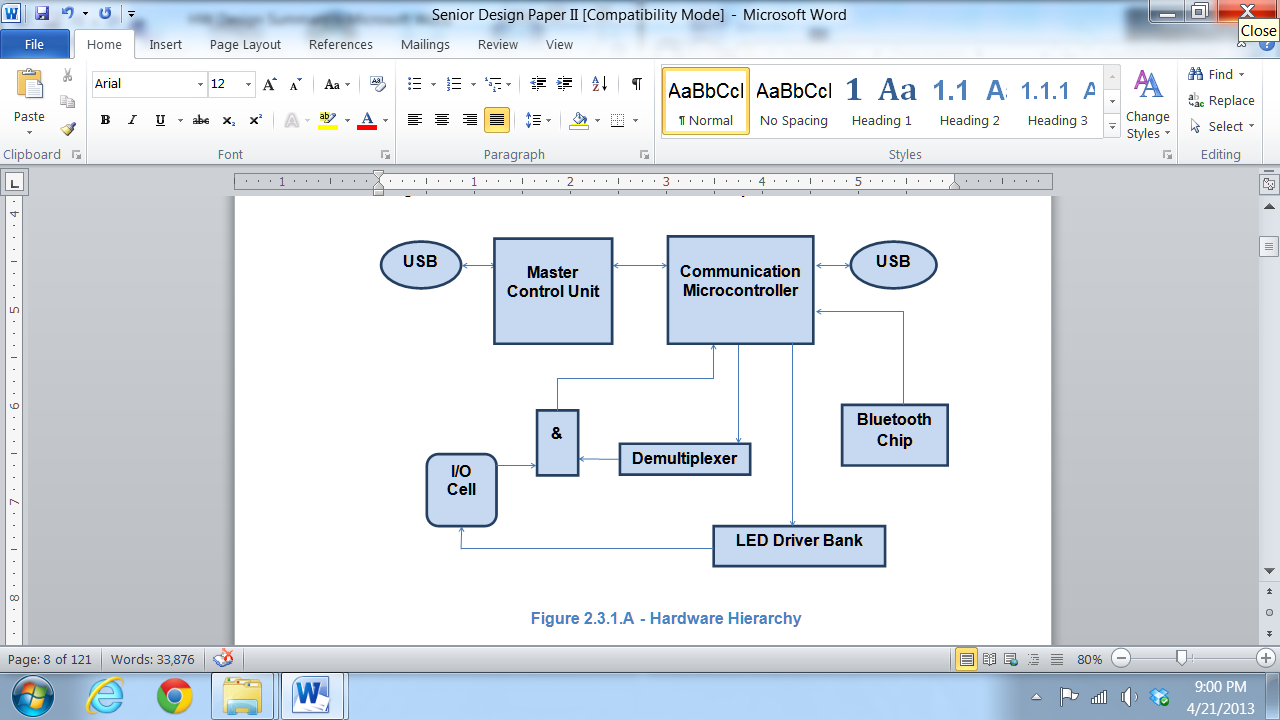


Figure 5.1.A - Hardware Overview

### Power Supply

The figure below shows the projected design of the power system. Save for the Bluetooth Device, everything requires 5 V logic and does not need to be regulated.

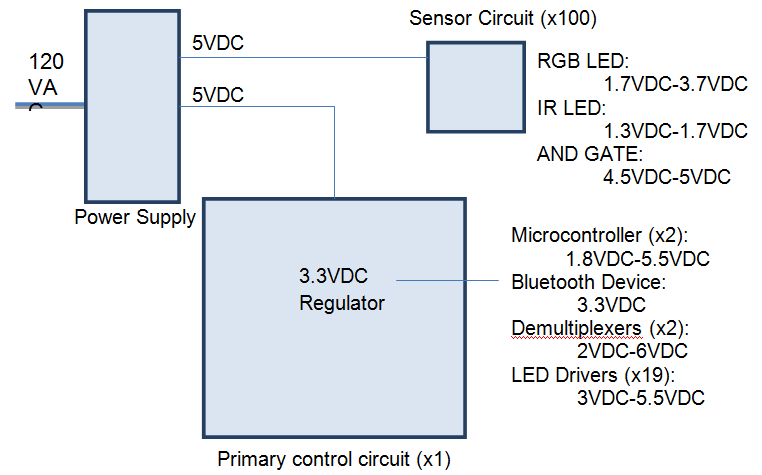


Figure . - Power Distribution Chart

Table 5.1.A - Power Supply characteristics of TDK-Lambda LS200-5 (TDK-Lambda)

|  |  |
| --- | --- |
| Input voltage | 85 – 263 VAC |
| Input Frequency | 47-63 Hz |
| Output voltage | 5 VDC |
| Max current | 40 A |
| Typical Efficiency | 72- 75 % |
| Enclosed fan | Yes |
| Overvoltage protection | 5.75 – 6.75 VDC |
| Overcurrent protection | 105% nominal peak |
| Over Temperature Protection | Yes |
| Size | 7.8 x 3.9 x 1.6” |

### Microcontroller

The table below defines the selected microcontroller of the group. In the current design there will be three of these.

Table 5.1.B - ATmega2560 Basic Spec List

|  |  |
| --- | --- |
| Property | Value |
| Program Memory (Flash) | 256KB |
| CPU Speed | 16MHz |
| RAM | 8 KB (SRAM)  4KB (EEPROM) |
| Digital Communications Peripherals | 4- UART  5- SPI  1- TWI(I2C) |
| USB | No |
| Operating temperature range | -40C to +85C |
| Pin Count | 100 (86 IO) |
| Operating Voltage | 1.8V to 5.5V |
| Timers | 6 |
| Comparators | 1 |
| Price | $18.75 |

### Input/Output circuit

There will be multiple parts in the circuits that make up each cell on the board. Below are tables that describe their characteristics. The circuits will include an IR emitter, detector, ‘and’ gate, and a RGB LED.

Table 5.1.C - Emitter Specifications (www.radioshack.com)

|  |  |
| --- | --- |
| Reverse voltage | 5V |
| Continuous forward current | 150mA |
| Forward voltage | 1.3V typical, 1.7V max |
| Wavelength (peak emission) | 950nm |

Table 5.1.D - Detector Specifications (www.radioshack.com)

|  |  |
| --- | --- |
| VCEO | 70V |
| VECO | 5V |
| IC | 50mA |
| Total power dissipation | 150mW |
| Peak sensitivity wavelength | 850nm |
| Spectral bandwidth range | 620-980nm |
| Angle of half sensitivity | +/- 20° |

Table 5.1.E - Operational Parameters of SN54ACT11 (Texas Instruments, SN54ACT11 Technical Data Sheet)

|  |  |
| --- | --- |
| Operating free-air temperature | −55 to 125 °C |
| VCC Supply voltage | 4.5V to 5.5V |
| High-level input voltage | 2V |
| Low-level input voltage | 0.8V |
| Input voltage 0 VCC | 0 VCC |
| Output voltage | 0 VCC |
| High-level output current | −24mA |
| Low-level output current | 24mA |
| Input transition rise or fall rate | 8 ns/V |

Table 5.1.F - Specs for RGB LED

|  |  |  |
| --- | --- | --- |
| Parameter | Minimum Value | Maximum Value |
| DC Forward Current | 30 mA | 100 mA |
| Forward Voltage (Red) | 1.7 V | 2.6 V |
| Forward Voltage (Green) | 3.0 V | 3.7 V |
| Forward Voltage (Blue) | 3.0 V | 3.7 V |
| Luminous Intensity (Red) | 3000 mcd | 5000 mcd |
| Luminous Intensity (Red) | 4000 mcd | 6000 mcd |
| Luminous Intensity (Red) | 1800 mcd | 3200 mcd |
| Operating Tempuature | -40C | 80C |

### LED Control

The LED’s will be controlled by the TLC5940 and the specifications are as follows:

* 80 mA current supply(max)
* 17 V voltage supply(output)
* 30 MHz data transfer rate
* 16 channels
* 16-bit Grayscale dimming (4096 brightness levels)
* 6-bit Dot Correction
* LED Open Detection
* Thermal Error Flag

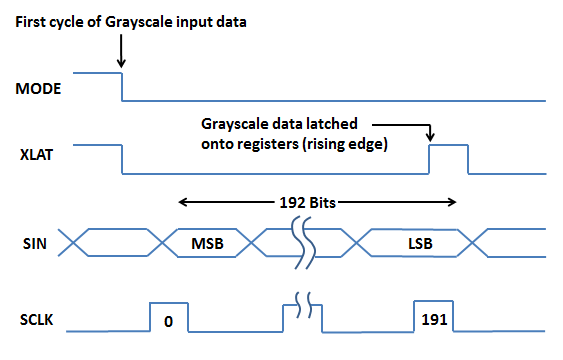


Figure 5.1.C - Grayscale cycle of TLC 5941

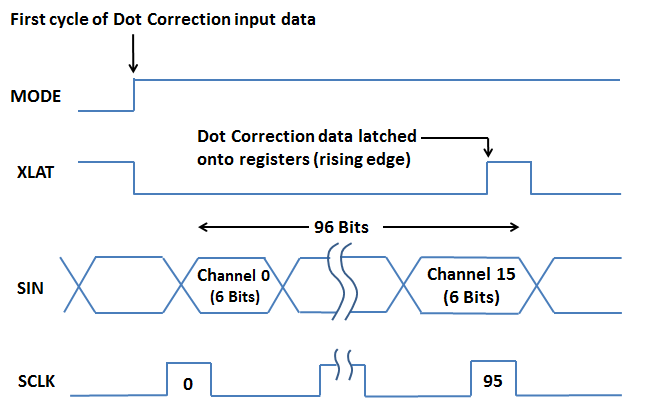


Figure 5.1.D - Dot Correction cycle of TLC 5941

### USB/UART

There will be a USB device for easy user programming.

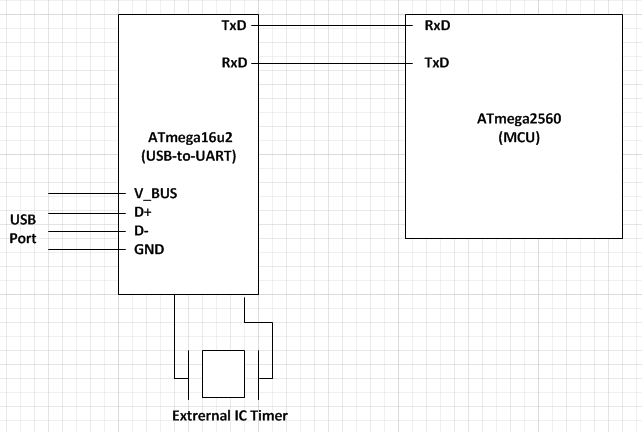


Figure 5.1.E - USB interface design using the ATmega16u2

### Bluetooth

The Bluetooth device which connects to the group android application will be used as a peripheral to the board with these specs:

Table 5.1.G - RN-42 Basic Spec List

|  |  |
| --- | --- |
| Property | Value |
| Bluetooth Versions | 2.1 + EDR, 2.0, 1.2, 1.1 |
| Data rate | With onboard stack 300Kbps;  HCI mode: 1.5Mbps sustained, 3Mbps burst |
| Profiles | SPP, DUN, HID, iAP, HCI, RFCOM, L2CAP, SDP |
| Supply voltage | 3.3V ± 10% |
| Power consumption | Standby 25mA; Connected 3mA; Deep sleep 26 µA |
| Operating temperature range | -40C to +85C |
| Antenna | PCB trace |
| Size | 13.4mm x 25.8 mm x 2 mm |
| Weight | 0.045 oz. |

## Software

**GUI Programmer Program** – This application will be used by the user in order to make various games and applications for the group’s LED board in a simplified coding environment. It provides various user controls applicable to the situation, such as programming various LEDs to change to a certain color. It allows the user to use specific driver functions, such as Bluetooth control and sensor interrupt handling. The expected outcome of this program is to provide a simplified interface to program the LED board. It will connect to the LED board via USB and will store any selection of programs to the Primary Microcontroller’s EEPROM.

**Microcontroller Configuration** – From a software perspective, there will be two microcontrollers that have different functions. The primary microcontroller serves as the main controller for program logic. It initiates the LED Board bootup sequence, and proceeds to run loaded programs at user command. The secondary microcontroller takes certain commands from the primary microcontroller, such as handling sensor interrupts and LED driver controls. For a more detailed list of commands that the primary microcontroller can send to the secondary microcontroller, see Section 4.3.3. This makes it faster and simpler for the primary microcontroller to continue with program logic without being blocked by constantly having to check sensor output and program LEDs manually. The secondary microcontroller will spend its time checking through all monitored sensors, as well as carrying out any LED scaling commanded by the primary microcontroller. Any sensors that are detected as ‘on’ will be put in to a buffer until the primary microcontroller requests the next sensor interrupt.

**Microcontroller Programming** – The microcontrollers will be programmed in the Arduino IDE. This gives access to libraries for various I/O including built-in serial communication. It allows the microcontrollers to be programmed in an ANSI-C based language.

**USB Interfacing** – The group will be using three USB to serial converter to program the Atmel microcontrollers. Two will be for the primary and secondary microcontroller firmware, which is built in to the Arduino IDE. For uploading the user programs from the group’s Board Programmer, the group will need to include a similar functionality. This involves creating a USB driver specific to the group’s purposes. Once this driver is made, the group will be using the USB connection to communicate with, and send applications to the primary microcontroller to store in EEPROM.

**LED Drivers** – The group will be using a set of TLC5941 LED drivers to control the group’s LEDs. These modules control the PWM signals and can keep them at a constant level, so this task is not specifically burdened on one of the microcontrollers. The secondary microcontroller will be directly interfacing with each of these. It will need a total of LEDs. These modules control the PWM signals and can keep them at a constant level, so this task is not specifically burdened on one of the microcontrollers. The secondary microcontroller will be directly interfacing with each of these. It will need a total of 5 to control each LED driver (Not including power and ground). The secondary microcontroller will be using the Software Serial library provided as open-source by Arduino.

**Android Interfacing** – The group’s microcontrollers will be communicating through a Bluetooth module to the group’s android device. The group will have a preset of control layouts that can be selected by the LED Board by this means of communication. By default, the android device starting the group’s app will display a screen indicating the attempt to connect. Once connected, it will await a command for what control layout to display. Upon receiving the command, it will display that layout with any additional information needed. Similarly, when a command is made from the Android device, it will send the appropriate command back to the microcontroller via Bluetooth. The microcontroller will handle the interrupt accordingly. Using an Android device is completely optional and may not be used for all programs.

**Program Assembly** – Once programs are stored on the primary microcontroller’s EEPROM, these programs will be displayed in a list to the user at board startup. Once selected, these programs will be run in a type of virtual machine. The data stored in EEPROM will be treated as a custom assembly and will be run on the primary microcontroller. These assembly instructions include basic operations, as well as more complicated commands for controlling LEDs, communicating via Bluetooth, reading sensor interrupts, etc.

**Sensor Addressing** – Sensors will be read from the secondary microcontroller in a sweep type fashion. Each sensor will need to be individually read, which can mean up to 100 separate reads per cycle. This is in addition to any other LED commands the secondary microcontroller will need to carry out. For addressing any particular sensor, a row and column will be selected such that the decoders will map to the correct sensor circuit. Eight lines will be used for addressing, 4 for X and 4 for Y. One return line will be used for reading if the sensor is active or not.

**Bluetooth** – The group will be using a Bluetooth module that can be used with serial RX and TX connections. Using the Software Serial library provided by Arduino, the group can easily send and receive messages to an Android device. All communications will need to end with terminating characters, such as end-of-transmission since there will be little or no reliability sending whole messages. Checksums will also need to be used in communication to prevent errors.

# Project Prototype Construction and Configuration

In order for the group to have a fully functioning device, it is necessary to test each aspect of the project individually first. As the group decided upon parts to use, it found the cheapest price and ordered a couple for immediate testing purposes. As tests for the individual parts are completed successfully, the group plans to combine parts together and test them as a more complete device until a smaller version of the final project is inevitably constructed. Some components of the device are easily tested, such as the LEDs themselves and creating simple programs with an Arduino test board to light up an LED on its own. Other components, however, will require much more time to get working on their own. This includes such devices as the LED driver, USB interface and Bluetooth module.

## Bluetooth

The group will need to create a system which connects to the group’s Bluetooth module in the ways specified in the data sheet. Using the group’s microcontroller, the group will attempt to communicate with the module to the extent to send a Bluetooth signal to any other Bluetooth compatible device, such as a laptop. Eventually, the group will need to use this functionality to prototype and test the group’s Android interface.

The main signals needed for Bluetooth communication for the group’s application are the following: connection setup, data send, data receive, and connection termination. As far as connections, it is currently known that the project requires the need to connect the following: power, ground, TXD, and RXD. Other pins more specific to options will need to be determined at the time of prototyping.

The purpose of prototyping the group’s Bluetooth module is mainly proof of concept. It is known that the device will work, but the group will need to prototype the setup when connecting to the microcontroller. The group will attempt to find a library made for interfacing with this device to test its Bluetooth setup. The result of prototyping this portion of the project will give the group a known circuit layout which the group can apply when designing and purchasing its PCB board.

## Android Interfacing

Following the previous prototype, the group will attempt to make a connection to the group’s android program and verify correct send/receive transmissions. The group will have a setup phase, where the application will attempt to connect. If successful, it will send a series of random values. The microcontroller will be programmed to receive these values via Bluetooth and echo them back. The group will confirm that the correct values are received on an Android device.

## USB

The group will have three different USB interfaces, so this prototype is critical to the completion of this project. Two will be used to directly program the microcontrollers, while the third will be used in conjunction with the GUI Board Programmer to save programs on the system.

For programming the two microcontrollers, the group will need to either buy or make a USB/Serial TTL converter. Under the assumption this is bought, the group will need to connect the following pins on the microcontroller: VCC to USB +5V supply voltage, ground to USB ground, RX from USB to RX1 on microcontroller, TX from USB to TX0 on microcontroller. This configuration will allow programming of the microcontroller using the Arduino programming environment with built-in USB compatibility.

For the third USB connection, a similar setup will be used but for a different application. A RX port will be used to receive the values to move into EEPROM memory for actual programs. A TX port will be used to send verify messages for if the transaction was successful or had an error. For testing, the group will not need to perform the final function of moving anything into EEPROM, but the group will verify it can send messages and have their values echoed back through USB.

## GUI Interface

This prototype will be a software interface on any PC. The goal of this section is to confirm that it can create a USB connection and send the necessary values to receive on the primary microcontroller. To test this, the group will use the previous prototype for USB interfacing and send random values to the microcontroller. These values should be received and echoed back through USB to the PC. The group will confirm that these values are received and determine the PC to microcontroller connection working at this point.

## Microcontroller Communication

For prototyping this, the group will simply need to connect the dedicated communication lines from both the primary and secondary microcontrollers together. For prototyping, the group will be using any 5 digital I/O pins from each microcontroller. Additionally, a clock pin will be connected for timing purposes in communication. The details of communication protocols are defined in Section 4.3.1.

Each of the connection lines will need to use a pull-down resistor connected to ground. Resistor values will be fairly low, and will be determined upon prototyping primarily based off of availability. These resistors will be used in order to prevent static on the lines that might give incorrect values.

To test this prototype, the group will be sending values from one microcontroller to the other, and verifying the correct values are received. Figure 6.5.A shows our planned layout to test this communication between the primary and secondary microcontrollers.



Figure 6.5.A - Connections between Primary and Secondary Microcontroller

## LED Controllers

For prototyping the TLC5941 LED controllers, the group will be using Arduino TLC5940 library. Note that while the number in the library is slightly different from the group’s LED controllers, the functionality for the group’s purposes is the same. For the Atmega2560, the group will be using the default pins defined in this library. For later implementation, these pins may change. The pin connections will be as follows:

* +5V from Arduino 🡪 TLC pin 21 and 19 (VCC and DCPRG)
* GND from Arduino 🡪 TLC pin 22 and 27 (GND and VPRG)
* Digital 9 🡪 TLC pin 18 (GSCLK)
* Digital 11 🡪 TLC pin 24 (XLAT)
* Digital 12 🡪 TLC pin 23 (BLANK)
* Digital 51 🡪 TLC pin 26 (SIN)
* Digital 52 🡪 TLC pin 25 (SCLK)
* TLC GND pin 22 🡪 TLC pin 20 (IREF)

Any LEDs the group will be testing will have their anode connected to the +5V pin on the LED driver, and their cathode in series with a resistor connected to any of the output pins (0 – 15). Additionally, a resistor (approx. 10kohm) should be connected between TLC pins 21 (VCC) and 23 (BLANK) to reset the drivers upon Arduino reset. The Figure 6.6.A below shows the connected circuit on a test board:

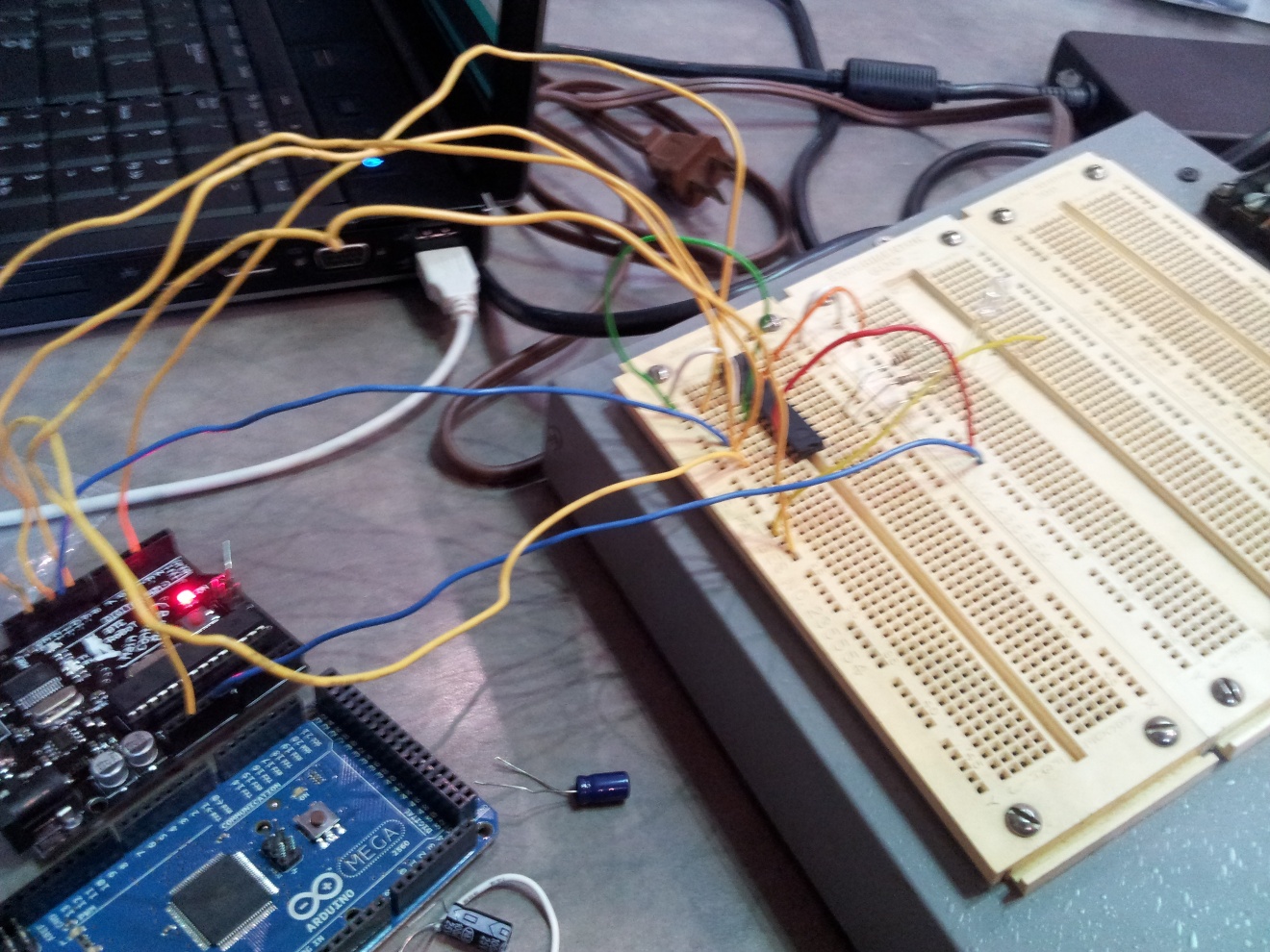


Figure 6.6.A: Microcontroller connected to LED driver

Pulse-Width Modulation or PWM is a feature of the LED driver that the group selected. With PWM, it is possible for the LEDs to achieve a larger spectrum of colors and shades. The first test will be to get PWM implemented in the code in any kind of form, whether it be getting a dim version of the basic colors RGB or making new colors and keeping them static. Just like the driver test above, it will first be conducted with a single MCU and then conducted with the group’s original two MCU setup.

## Sensor and Addressing

The group will be using a couple of decoders for addressing the IR sensors. One decoder will be used for row, and the other for column. This allows the group to address all cells of the board in a coordinate system fashion, using (x,y). This simplifies both connections, as well as software to read these addresses.

The purpose of this prototype is more for proof of concept, as the group will create a more simplified X/Y sensor 2D array for the microcontroller to interface with. Assuming a 2x2 grid, the group will need two 1:2 decoders. This allows the group to use 2 lines from the microcontroller, one for X and one for Y. The group will set up each of the 4 sensor circuits and connect them to their respective X and Y lines from the decoders. All sensor circuit outputs will be connected to the return line, which will have a pull-down resistor. This return line will connect to a third digital pin on the microcontroller.

The expected outcome of prototyping this section of the group’s project is to confirm that the addressing scheme works for the group’s sensor sub-circuits.

## Infrared Sensor

The group will need to test the sub-circuit for the IR sensors. To test out this prototype, the group will experiment with the distances necessary to trigger the sensors and vary the components in its circuit accordingly to match the desired sensing distance. This may vary by ambient light in the room, and the group will need to ensure a good medium between both brightly-lit rooms and dark rooms.

The group will verify the different resistors and other components are the correct values so as to achieve the desired sensitivity. This circuit will be used in the following prototype to test out the sensor addressing scheme.

# Project Testing

After all of the parts have come together, it is important that it be tested for synchronicity. After the group has confirmed that all the parts of the project are functional separately as well as in small sub units of the overall project there are a few things the group will be looking for:

## Power System

Before connecting any outputs, 120 VAC will be applied to the power supply and the output pins will be tested to ensure they match their nominal rating within the tolerances of the device.

## Bluetooth

1. The board is visible upon power-up, and other devices can find it.
2. If no connection is made, the board should continue with its default programming to use without Bluetooth.
3. At any time during a program, the Bluetooth connection can be made and the android device can be used as a controller.
4. If communication is lost for Bluetooth, the board will continue with its default programming to use without the android controller. The android program will automatically attempt to reconnect.
5. The android controller can be used successfully as a controller for several different layouts of controls.

## USB

1. Each microcontroller can have the firmware upgraded via their respective USB connectors.
2. A program, or series of programs, can be uploaded to the primary microcontroller from the GUI Board Programmer via the third USB connector.
3. If the connection is invalid between the GUI Board Programmer and the microcontroller, such as the cable isn’t connected or power is off, it will be displayed that it cannot connect to the microcontroller.
4. If during transmission connection is lost, the GUI Board Programmer will display that an error occurred. The programs on the board will be corrupt until a new attempt at writing data is made.

## Microcontroller Communication

1. The primary microcontroller should be able to command the secondary microcontroller to change LED color values.
2. The primary microcontroller should be able to command the secondary microcontroller to read certain sensor signals and report interrupts.
3. The primary microcontroller should be able to command the secondary microcontroller to ignore certain sensor signals and not report interrupts.
4. The primary microcontroller should be able to command the secondary microcontroller to start, maintain, and stop connections for Bluetooth.

## LED Controllers

1. Values should be able to change directly to any available color for any LED.
2. LED colors should be able to scale from any available color to any other available color.

## Sensor and Addressing

1. Any sensor can be monitored, and should give an ON value when sensing within range.

Values should not be given as false-positives.

## GUI Interface

1. Should be able to fully produce a program and interface via USB.

## Android Interfacing

1. The android controller can display the correct control layout. It should be able to send the corresponding commands via Bluetooth.

# Administrative Content

This section is partitioned for the purpose of managing time and money for the project. Time management is one of the most important factors and can improve effectiveness, efficiency and productivity. It is also important that the group tracks its budget so that financial contributions are equally divided upon completion of the project.

## Finances & Budget

The group is financing this project with its own money. With this in mind it was decided that the project cost be kept as low as possible. Upon initial evaluation, the group decided to try and keep the project under $800. Should anyone want to keep certain parts after the project is compete, that member could pay off the others to keep it. Below is a compilation of all the parts of the project, Table 8.1.A. If a part price is listed as $0.00, it is most likely already in one of the member’s possession and will be returned to that member upon completion of the project.

Table 8.1.A - Finance Table

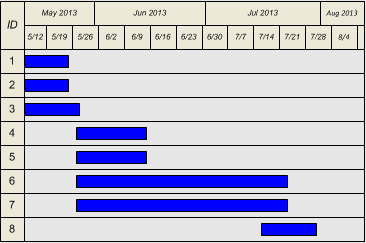
|  |  |  |  |
| --- | --- | --- | --- |
| Part | Price | Quantity | Total |
| Microcontroller | $18.00 | 2 | $36.00 |
| TLC5941 | $1.80 | 19 | $34.20 |
| RN-42(one with breakout) | $21.50 | 2 | $43.00 |
| Power Supply | $56.66 | 1 | $56.66 |
| Resistors | $0.00 | ~ | $0.00 |
| Transistors | $0.00 | ~ | $0.00 |
| Capacitors | $0.00 | ~ | $0.00 |
| Decoders | $.75 | ~10 | $7.50 |
| IR Detectors | $1.73 | 100 | $173.00 |
| IR Emitters | $0.95 | 100 | $95.00 |
| RBG Light Emitting Diodes | $1.95 | 100 | $195.00 |
| USB | $5.34 | 2 | $10.68 |
| PCB | $30.00 | 1 | $30.00 |
| Total |  |  | $748.54 |

## Milestones

The following Table 8.2.A displays our groups estimated milestones and completion time periods. To date, the previous milestones and completion dates have been fairly accurate to the table.

Table 8.2.A - Milestone Estimate





**Appendix**

Appendix A: Works Cited

<http://owl.english.purdue.edu/owl/section/2/>

4Cube. Hypnocube, 2013. March 2013.

<http://hypnocube.com/product/4cube/>

A Primer on Photodiode Technology. University of SanDiego. n.d. March 2013. <http://home.sandiego.edu/~ekim/photodiode/pdtech.html>

Batsocks - LED dimming using Binary Code Modulation. Batsocks, 2009. March 2013. <http://www.batsocks.co.uk/readme/art\_bcm\_1.htm>

Bluetooth vs Wifi - Difference and Comparison. Diffen, 2012. March 2013. <http://www.diffen.com/difference/Bluetooth\_vs\_Wifi>

Difference between PIR Motion Sensor and Infrared Beam Motion Sensor. Shenzhen Meidasi Technologh. 2013. March 2013. <http://www.hkvstar.com/technology-news/difference-between-pir-motion-sensor-and-infrared-beam-motion-sensor.html>

FT234XD USB To Basic UART IC. Version 1.1. FTDI Inc, 2013. April 2013. <http://www.ftdichip.com/Support/Documents/DataSheets/ICs/DS\_FT234XD.pdf>

Infrared Astronomy; What is Infrared?. NASA. n.d. March 2013. <http://coolcosmos.ipac.caltech.edu/cosmic\_classroom/ir\_tutorial/what\_is\_ir.html>

Interactive Game of Life Kits. Evil Mad Scientist, 2013. March 2013

<http://shop.evilmadscientist.com/productsmenu/tinykitlist/477>

Interactive LED Panels Kits. Evil Mad Scientist, 2013. March 2013.

<http://shop.evilmadscientist.com/productsmenu/majors/46>

LS200 Series. TDK-Lambda, December 2011. April 2013. <http://www.alliedelec.com/images/products/datasheets/bm/TDK-LAMBDA/70177525.pdf>

PCA9922. NXP, 2011. April 2013. <http://www.nxp.com/documents/data\_sheet/PCA9922.pdf>

Phototransistor Tutorial. Radio-Electronics.com. n.d. March 2013. <http://www.radio-electronics.com/info/data/semicond/phototransistor/photo\_transistor.php>

RadioShack Infrared Emitter and Detector. RadioShack. n.d. March 2013. <http://www.radioshack.com/product/index.jsp?productId=2049723>

Schematics-Infrared Emitter Detector. Society of Robots. 2013. March 2013. <http://www.societyofrobots.com/schematics\_infraredemitdet.shtml>

SCHS152D. Texas Instruments Incorporated, June 2004. April 2013. <http://www.ti.com/lit/ds/schs152d/schs152d.pdf>

Sn74act11. Texas Instruments Incorporated, October 2003. April 2013. <http://www.ti.com/lit/ds/symlink/sn74act11.pdf>

STP16CP05. ST, 2010. April 2013.

<http://www.mouser.com/ds/2/389/CD00126634-114406.pdf>

The Electromagnetic Spectrum. Solarwiki. n.d. March 2013. <http://solarwiki.ucdavis.edu/The\_Science\_of\_Solar/1.\_Basics/B.\_Basics\_of\_the\_Sun/I.\_Properties\_of\_Light/1.\_The\_Electromagnetic\_Spectrum>

TLC5940. Texas Instruments, 2007. March 2013. <http://www.ti.com/lit/ds/slvs515c/slvs515c.pdf>

TLC5941. Texas Instruments, 2008. March 2013. <http://www.ti.com/lit/ds/slvs589d/slvs589d.pdf>

V-USB - A Firmware-Only USB Driver for Atmel AVR Microcontrollers.

Objective Development, 2013. April 2013. <http://www.obdev.at/products/vusb/index.html>

What is an Infrared Emitter?. Future Electronics. n.d. March 2013. <http://www.futureelectronics.com/en/optoelectronics/infrared-emitters.aspx>

What is an Infrared Sensor?. Wisegeek. Conjecture Corporation, 2013. March 2013. <http://www.wisegeek.org/what-is-an-infrared-sensor.htm>

What are Proximity Switches?. Wisegeek. Conjecture Corporation, 2013. March 2013. <http://www.wisegeek.org/what-are-proximity-switches.htm>

Alberts, Mike ; Clark, Joseph ; Li, Arnold ; Walker, Isaiah. Group 1 Dynamic Animation Cube. EECS UCF, 2012. March 2013. <http://www.eecs.ucf.edu/seniordesign/sp2012su2012/g01/>

Brault, Greg. 8x8x8 LED Cube. Ohm's Projects, 2012. March 2013.

<http://ohmprojects.com/cube/cube.html>

Brault, Greg. Propeller Display (RGB). Ohm's Projects, 2012. March 2013.

<http://ohmprojects.com/prop-rgb/prop-rgb.html>

Cathey, Ian ; Labbato, Mark ; Thrun, Max. RGB LED Coffee Table. Texas Instruments Microcontroller

Projects, 2011. March 2013.

<http://e2e.ti.com/group/microcontrollerprojects/m/msp430microcontrollerprojects/>

Fischer, Julius. 100Pixel RGB LED Table. Gecko IT, 2012. March 2013.

<http://www.it-gecko.de/100pixel-rgb-led-tisch-interaktiv-touch.html>

Harden, Scott W. Simple Case AVR/PC Serial Communication via MAX232. SWHarden.com, 2009. April 2013.

<http://www.swharden.com/blog/2009-05-14-simple-case-avrpc-serial-communication-via-max232/>

Pihlajamaa, Joonas. Code and Life - AVR ATtiny USB Tutorial Part 1. Code and Life, 2012. April 2013.

<http://codeandlife.com/2012/01/22/avr-attiny-usb-tutorial-part-1/>

Praveendb. Interface the Atmega 16/32 with the PC. Wordpress, 2009. April 2013.

<http://praveendb.wordpress.com/2009/05/18/interface-the-atmega-1632-with-the-pc/>

rgbphil. Charlieplexing LEDs. Instructables, 2007. March 2013.

<http://www.instructables.com/id/Charlieplexing-LEDs--The-theory/>

Rojvanit, Rawin. Migrating Applications to USB from RS-232 UART with Minimal Impact on PC Software. Microchip, 2004. April 2013.

<http://ww1.microchip.com/downloads/en/appnotes/00956b.pdf>

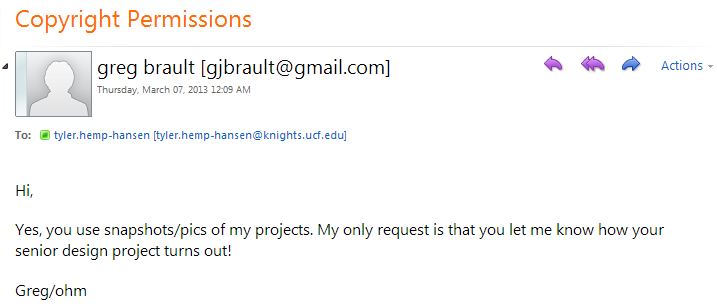
Schulze, Nick. HNTE - RGB LED Cube. HowNotToEngineer.com, 2012. March 2013.

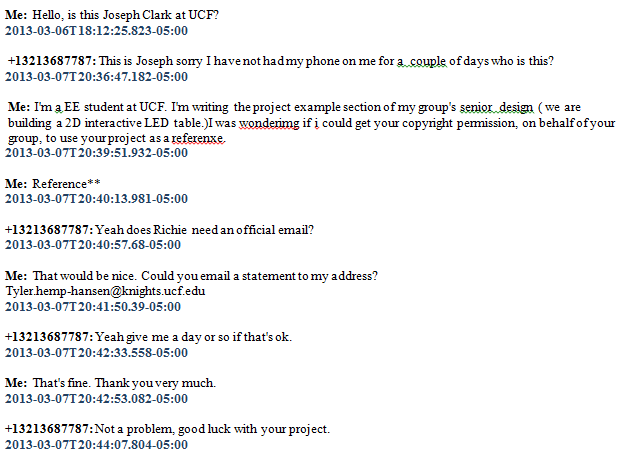
<http://www.hownottoengineer.com/projects/rgb-led-cube.html>

Appendix B: Copyright Permissions

All copyright permissions for figures are listed in descending alphabetical order based on author name.

Copyright Permission: Figure 3.1.2.C and Figure 3.1.2.D.

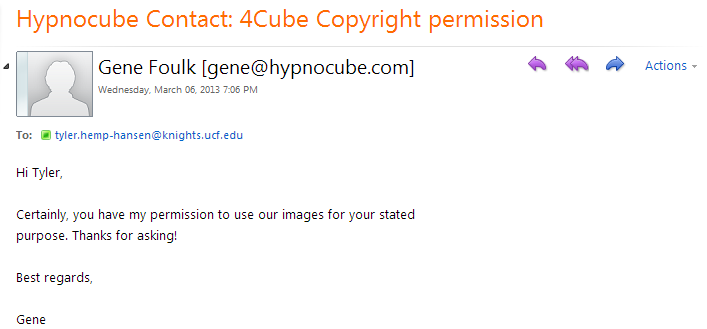


Copyright Permission: Figure 3.1.2.A 

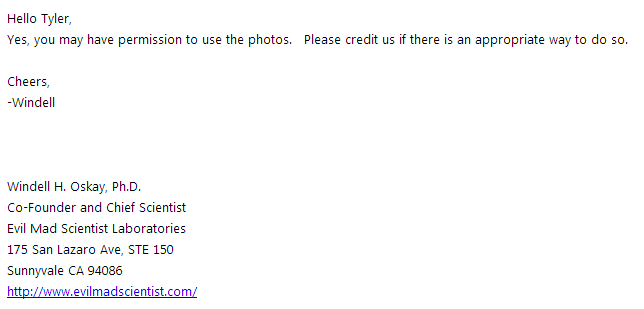
Copyright Permission: Figure 3.1.2.G and Figure 3.1.2.H



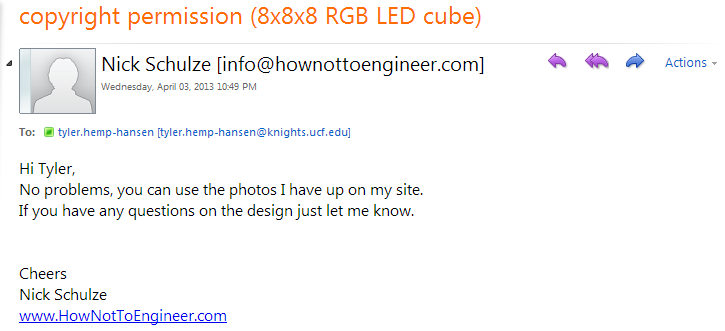
Copyright Permission: Figure 3.1.1.A



Copyright Permission: Figure 3.1.1.B and Figure 3.1.1.C



Copyright Permission: Figure 3.1.2.B



Copyright Permission: Figure 3.1.2.E and Figure 3.1.2.F

