Abstract — Brain Control Interface (BCI) is an evolving field of bio/electrical engineering. Cybernetic implants can enhance the quality of life for both disabled and non-disabled people. This paper aims to showcase a variety of methods to interface the human body with external applications. The bio-potential signals used include: EEG, EKG, EMG. An accelerometer will also be used based on head tilt. These signals will be obtained and digitally processed and sent wirelessly to control 3 external applications, which include: (1) Remote Control Car, (2) Water Fountain, and (3) Claw Machine.

Index Terms — BCI, Biomedical DSP, Wireless communications, MCU Control

I. INTRODUCTION

Biomedical signal processing is a quickly growing field encompassing everything from assistive devices for the disabled to cybernetic implants to enhance everyday life. This project intends to use biomedical DSP to showcase the technology by controlling various novel applications. The control flow of the project starts by taking bio-potential signals from the test subject using a BCI headset and wirelessly transmitting this signal via Bluetooth to a laptop computer in order to perform DSP. This will then send commands via Wi-Fi to an ESP8266 WiFi bridge, which will then send commands to a MCU over UART to control applications. Each application will have its own configuration of bio-potential signals and are as follows: (1) RC Car, (2) Water Fountain, and (3) Claw Grabber Machine. The bio-potential signals used include: EEG, EKG, and EMG. An accelerometer will also be used to measure head tilt.

II. BCI HARDWARE

In order to record signals from the brain, specialized BCI hardware must be used. This processes involves taking micro-volt signals and converting them from analog-to-digital. The OpenBCI is the hardware of choice for this project; it allows for a high channel resolution and bit-depth, as well as support for recording all 3 bio-potential signals. The OpenBCI is unmatched at its price point for both quality and flexibility. The OpenBCI communicates to the PC via Bluetooth 4.1 communication. This allows for the necessary bandwidth and latency to have seamless EEG capture. The OpenBCI is powered by a 3-6V DC Battery. This allows for portability and flexibility by allowing the user to move freely without having to be connected to a wall-outlet. One clear limitation of this is it adds the variable of battery life to the end product. An estimated battery life of the OpenBCI has been shown to be around 23 hours given a constant load of 62 mA. This can be seen in the figure below.

A 10-20 standard BCI headset will be used to both hold the BCI hardware, and to hold the EEG electrode in place. This headset will allow for the user to be untethered with free range of movement. The BCI hardware will be located in the rear of the headset and will be transmitting data to a computer within close proximity. This project will be targeting the Alpha Wave. The Alpha wave propagates the strongest when the user’s eyes are closed. This is directly related to the image processing center of the brain, which is located in the O1 and O2 Region on the brain. This information is reflected in the image below.
Electrodes will be needed to obtain signals from the human body. The two types of electrodes can be divided into two different categories: active and passive. Passive electrodes overcome the trace impedance of the skin by using a capacitive gel. Active electrodes overcome this impedance by using a built-in amplifier. This project uses a combination of both active and passive electrodes. The electrode placements are as follows: 2 ear clip electrodes used as a reference and a ground, 1 EKG electrode used to monitor pulse, 1 EEG electrode for monitoring alpha brainwave activity, and 2 EMG electrodes to measure muscle movement. Active electrodes were chosen for this project due to comparable SNR to passive electrodes, and ease of use when switching between different users of the test setup. A comparison between active and passive electrodes is detailed in the figure below.

III. DSP

Digital signal processing is one of the most important aspects of the project. Once the bio-potential signals are captured by the OpenBCI board, they need to be transferred into the workhorse computer in order to mathematically determine what the various signals actually mean. The signals that we will be using to control our various applications include muscle activity (electromyography), alpha wave detection (electroencephalogram), heart rate (electrocardiogram) and the tilt of the head which is measured using accelerometers built into the OpenBCI board.

A. Receiving Data

The first step is transferring the raw serial bits into the python interpreter. This is accomplished using a script provided by the OpenBCI team, in which the information is sent into ‘LabStreamingLayer’, or LSL. LSL is a low level transport library, which includes interfaces in several programming languages. The data now consists of the actual measurements recorded by the OpenBCI board, minus the start and stop bits. It is received as two arrays in python, named ‘eeg_stream’, and ‘aux_stream’. Within eeg_stream, each of the 8 indexes within this array contains one of the eight channels that the OpenBCI board can detect. The second array consists of 3 indexes, one for each axis measured by the accelerometers. One major shortcoming of this setup requires that a small delay be added after every ten samples. The OpenBCI board samples at 250 Hz, but LSL transfers significantly faster than that. Adding in a short delay (4 hundredths of a second) every ten samples stops LSL from outpacing the OpenBCI board.

B. Determining Head Tilt

After capturing the information, the process of determining what bio-signals are created by the user begins. The signal that the team began with was the accelerometer data. Each entry contains the acceleration experienced by the board, in terms of one earth’s gravity. The first testing began with determining which directions were output as positive, and which index in the sample array contained information from the respective axis. The following image shows the information from the x-axis, during a basic test. First the board is held flat, with the positive z axis upward. The board is then rotated so the positive x-axis is upward, then tilted back towards flat.
Based on this testing we decided to use .8G as the threshold. If an axis experiences more than eight tenths of a G, the board has been tilted and the corresponding output is transferred to the MCU.

C. Heart Rate

The first bio-potential signal is heart rate. The heart rate will be detected by looking for periodic minimums in the ECG signal, marked in the figure below.

Once two minimums have been found, the script will compute the difference in sample numbers, and use that to find the heart rate using the following formula:

\[
\text{Heart Rate} = \frac{255 \text{ samples}}{\text{samples between minimums}} \times \frac{60 \text{ seconds}}{\text{minute}} = \frac{\text{Rests}}{\text{Minute}}
\]

D. Muscle Activity

The next signal to be determined is muscle activity. The goal is to determine if a user has flexed a muscle. By attaching electrodes on top of the muscle, we can see strong peaks when a flex occurs. This can be seen in the image below, when a user flexes their jaw muscle (the masseter) at 5 and 10 seconds.

The image above was generated after taking the signal and filtering it using a Butterworth band pass filter. Initially created as a 9th order, .1-55Hz, after testing it has been changed significantly. A 9th order filter was unnecessary for this signal. The benefits gained from a steeper roll-off aren’t very beneficial for this project, and the increased computation difficulty increases the lag in the system. Changing the pass band improved the signal dramatically, and it has been updated to 5 Hz - 50 Hz. An output is then found if the absolute value of the signal is above 50 uV.

E. Alpha Wave Detection

The final signal that needs to be found is the EEG signal, checking for the presence of alpha waves. These waves are the first stage of relaxation, produced in the occipital region of the brain when a person closes their eyes. Alpha waves are always generated around 8-10 Hz. [1] A similar band pass filter to the one used in the EMG detection will be used, but this time the pass band will be significantly smaller. The band will be from 8-11 Hz, and the order of the filter will need to be higher than it was for the EMG script. We will be using a 6th order filter, which appears to be the point where the increased computation speed and the roll-off balance out. If significant activity is found within that band, an alpha wave has been detected. In the figure below, the signal has been filtered as described above. The right half of the image shows when the eyes have been closed, and the left half when the eyes are open.
Additionally, to avoid false positives, we will monitor activity in the 6-8 Hz and 11-15 Hz ranges. Unwanted signals that activate the 9-11 Hz band will also have activity in the frequency bands near the alpha band. If significant activity is detected in these two bands, an alpha wave has not been detected, and the output will reflect that.

IV. WIRELESS COMMUNICATION

When the discussion of control began, it was clearly established that the cornerstone of any modern project is wireless communication. There are many different wireless communication standards to choose from, as well as many sub-classifications. Initial thoughts were to use Bluetooth 3.0 as the project’s communication standard, even the lower power consumption version Bluetooth LE was considered. The available chips were competitively priced and relatively within the budget that was being considered. A flow chart of this process is provided in the figure below.

The main problem came in the form of what was required from the system’s communication, which was an invisible wireless Serial Bridge. As seen in the figure above, the ideal flow of our data is depicted. It was desired to build the communication network as close to this control flow as possible. This meant a chip that could be configured and then left alone. However, when looking at other projects utilizing Bluetooth and its other variants, most users were utilizing them to connect to smartphones, rather than desktop or laptop computers.

The ability to connect to a computer was desirable, but the problem of adding a second Bluetooth dongle meant dealing with drivers, and trying to drive serial data through the system. This approach seemed to overcomplicate the simple requirements that were put in place. It was also a desirable feature to obtain a way for the team to wirelessly flash our chosen microcontroller.

At the time of the selection for a wireless standard, a microcontroller had also not been selected so whichever standard was selected would require documented parts that were known to work with the chosen microcontroller. The chosen wireless standard would decide which part would be used, which would then lead to a decision of which microcontroller met these standards.

The ATMEL chip met all of the required standards for the project. In researching chips that were known to be compatible with ATMEL, and were well documented for problems. The optimal wireless capable chip was found to be the ESP8266. The ESP8266 is a cheap WiFi microcontroller that is commonly used as an Internet of Things device. This device is also normally used to connect to an ATMEL chip for connecting to the internet and allowing for interaction with the server. After further research it was found that there was a custom firmware written for the ESP8266 that did exactly as required for this project.

It also bypassed a lot of potential problems that were present with the initial choice of WiFi. There was no dongle to be installed because the workhorse computer was chosen to be a laptop which has the capability of WiFi. There were no drivers to install and no potential conflicts with other Bluetooth devices on the same computer. The custom firmware accepted ASCII characters over simple telnet which could then transmit them over its own UART into the chosen microcontroller.

To add even more functionality, it also added a much desired feature: wireless flashing of our ATMEL chips firmware. In choosing this chip, three issues were overcome. WiFi was chosen as the communication standard, while the ESP8266 ran a custom firmware, and utilized an ATMEL microcontroller.
The custom firmware also allowed the project a plethora of debugging tools. This information could be sent to the microcontroller via a locally hosted status page, as seen in the figure above.

By simply building a python script to send the data in a string format, over telnet, to the ESP8266 the goal of wireless communication is complete in a way that satisfies all the requirements.

V. APPLICATION INTRODUCTION - RC CAR

The challenge when dealing with a system such as this is getting a response time that is acceptable with a control scheme that is simple to use. If there are not enough reliable responses to read, then controlling some applications will require specific planning and well thought-out designs in order to achieve fluid control that is acceptable.

In order to control the applications, a flexible PCB that will house a microcontroller with many possibilities for control was desired. After deliberation between the main manufacturers, Texas Instruments, ATMEL, and PIC the chosen device was the ATMEL 2650. The ATMEL 2650 had many desirable features such as 4 Serial UARTS, which the project requires minimum of 2. It’s also capable of PWM and has enough digital outputs for this application. Another advantage to using ATMEL chips, is the large community and available documentation. Utilizing the ATMEL 2560 allows for a simpler programming environment. Another advantage of using ATMEL was the compatibility with the ESP8266 WiFi chip. This could be useful if a BCI application had a control module placed in an area that was not easy to access.

The first application to showcase in this project is control over an RC car. The RC Car in this project is from the 1990’s. The design which of the controller is simple, if the control scheme was complicated at the circuit level, there would be difficulty in controlling the RC car without swapping out the hardware. While designing an RC car is an option, this would have added extra cost and time.

The Car’s remote control unit works with metal contact sliders. When a control stick is pushed forward, the slider moves forward and sends the signal to the car to move forward. At the circuit level this design is convenient for many reasons. The sliders connect the 9 Volt battery to different traces on the control board, when no commands are being sent, there is no draw on the battery. This is an efficient design due to the fact that there is no need for a power switch which could be left on to drain the battery.

The controller uses this control method for all of its directions. The circuit behind it was then analyzed to ensure simple control could be established. While the full circuit was not relevant, the figure below shows a simple overarching view of how the RC car can be controlled. After analysis the possibility of using NPN transistors were not capable to control the system. Configuration and a new, high side switching circuit had to be developed as shown in the figure below.

The new high side switch combines a PNP transistor and an NPN in order to fully control the RC Car’s controller.

One problem which occurred was multiple commands which consisted of moving and Turing at the same time. During preliminary development it was unclear how much data was obtainable. How to acquire consistent signal that could be sent in order to keep a constant direction is an issue.

The simplest solution is to make every direction a toggle. The user will send a forward command, and the car moves forward until it is sent an additional forward comment again. The code will simply initialize this input to zero on startup, and then XOR every time it received the command, thereby toggling the command as needed. Depending on the amount of information that can be received, there is also the possibility of adding an emergency stop command. The emergency stop command can be activated via a jaw clench.

The debug log shows the most recent remote exceptions present in the module, which is important for finding issues.
In the event of an emergency stop, it there will be a LED triggered on the controller. This will provide valuable feedback the user. The PCB that was designed as the main control unit which accomplishes control even though it was not in the initial design.

VI. WATER FOUNTAIN

The second application is the EKG water fountain. Based on the user’s heart rate, different tiers of the fountain will be activated. The diagram below shows the different tiers of the fountain.

![Diagram of water fountain](image)

The bottom 14” reservoir will hold all three pumps, with hoses moving up into the first, second, and third levels. Each level will flow into the next, until dumping into the final reservoir on the bottom. While not present in the diagram, each level is a square of acrylic glass with different areas. The first is 8” x 8”, the second 5.5” x 5.5”, and the third 4” x 4”.

The fountain pumps will be controlled by using three relays. When the MCU outputs a high, the corresponding pump will activate. Each relay is controlled using an N-MOS connected to two resistors. One resistor limits the amount of current flow from the MCU output in order to ensure the MCU output does not short, due to the high capacitance of an N-MOS. The second resistor is connected at the gate to ground which acts as a pull down resistor to ensure the MCU output is logically low, the gate voltage is pulled to ground and the FET is turned off. The FETs were chosen based on the current draw of the relay selected, the source current was required to be higher than the turn on current draw of the relay. The relay selected for this application is the Parasonic 5V Relay: JS1A-B-5V-F which has a current draw of 72mA at 5V. The BS105A N-MOS was chosen based on this parameter, which allows for a maximum source current of 250mA. A layout of the water fountain’s electronic is detailed in the figure below.

VII. CLAW MACHINE

The idea for the claw machine is to allow for control utilizing accelerometer, EEG, and EMG inputs from the BCI headset. Using the accelerometer, left and right head tilts will move the claw left and right. Forwards and backwards head tilts will correlate to forwards and backwards motion. A jaw clench, EMG input, will be used to adjust the speed of the motor to allow for higher precision. An alpha wave, EEG input, will be used to drop and close the claw.

A. Structure and Linear Actuation

The claw machine application’s mechanical structure was based on a previous senior design project, which was used for an automated chess board. The previous project’s rails had to be modified to hold a third motor to incorporate a z-axis range of motion. Stepper motors were utilized for all three ranges of motion and were chosen for this application because of the high level of precision, fair amount of torque, relatively cheap costs, and long lifespan stepper motors provide. The downsides to using stepper motors are the amount of current drawn and in turn power consumed, a dedicated driver is required, and a lack of an internal feedback system. The power consumption can be minimized with proper use of the stepper motor driver.
enable pins and an internal feedback system can be implemented through the use of hardware and software. The claw machine subsystem is detailed in the figure below.

B. Stepper Motor Selection Process

The maximum torque required to move the x, y, and z axis were found to be .216 Nm, .061 Nm, and .0223 Nm respectively and was calculated with Leadshine’s statics equations found online. Once the maximum torque was known, the stepper motors were selected with the torque to be at least 1.5 times more than the maximum required for each axis. The parameters of weight, voltage required, current draw, and cost were also factors which were accounted for in the selection process and specs.

C. Feedback System

The feedback system for the x and y-axis is implemented using limiting switches on all four ends of the rails. When a collision occurs with a rail and the platform, the switch will close and send a logical high to the MCU input, which will be coded to reject any commands to step the motor in the direction of collision until the switch opens again. This implementation is simple, reliable, cheap, and effective compared to other alternatives such as a software implemented position tracking, which would not be reliable in that the case the timing belt slips or an encoder which would be expensive and heavy.

The feedback system for the z-axis is implemented through software using a counter system which will be initialized to zero when the claw is at its resting point at the top of the platform and will increment by one with each step taken until it reaches its bounding condition. The bounding condition will be the position where the claw has reached the bottom. The same counter system can be used to bring the claw back to its starting position, however the system will decrement instead of increment until counter reaches zero. The feedback system is detailed in the schematic below.

D. Stepper Motor Driver

Based on the specific voltages and currents needed by the stepper motors, the stepper motor driver needs to be able to supply a minimum of 12V to the stepper motors and be capable of sourcing up to 1.7A. For that reason the DRV8825 was chosen which allows a minimum of 8.2V to a maximum of 48V and a max load current of 2.5A which meets the needs of this project. The DRV8825 also has a built in thermal shutdown to prevent the chip from burning out in the case of overheating due to too much current draw. The amount of current capable of being drawn without overheating is 1.5A however with a proper heat sink the full 2.5A can be drawn. The DRV8825 also allows for micro stepping which allows for more precision in the case the user needs more control than the default setting offer.

E. Claw Relay

The circuit to close and open the claw was designed to support higher voltage values if needed with the use of a relay in the case the developer intended to change out the claw for one with a higher voltage setting. The relay is controlled using a NPN transistor connected to the MCU output. When the MCU outputs a logical high, the transistor will turn on and trigger the relay and close the claw. When the MCU outputs a logical low, the transistor will shut off and cut power to the relay and open the claw.
The schematic for the claw relay is detailed in the figure below.

![Schematic Diagram]

**F. MCU**

The MCU chosen for this application was the ATMEGA328. This chip was selected due to the amount of community support on both the hardware and software end, the use of the Arduino IDE which makes the coding easier, and the open source schematics made for easier reference materials to create PCB. The chip has one set of pins reserved for UART communication which is necessary for this application. The ATMEGA328 also allows for six PWM pins of which three are required for this project. Lastly this MCU has up to 20 digital input/output pins, of the 20 available this project requires 15.

**G. Power Supply**

The claw machine application will run off of the RX-300XT 300W ATX 12V computer power supply. The MCU is requires 5.5V to operate, the limit switches require 5.5V, the stepper motor driver requires 12V, and the claw requires 24V. The max current draw required is less than 500mA from the 5.5V supply, 3.4A from the 12V power supply, and 1A from the 24V supply. The RX-300XT 300W ATX supplies 5V at a max of 15 amps current draw and 12V at a max of 21A current draw which is more than enough for this application’s lower voltage needs. The RX-300XT 300W ATX also has a -12V source which could be tied across the claw with the 12V power supply to sum up to the required 24V. The problem with this set up is the -12V supply can only source 500mA while the required amount is 1A, therefore a 24V wall adapter with a 1.5A maximum current draw capacity will be used to substitute the required 24V.

**VII. CONCLUSION**

All BCI and bio-potential measurements for this project are meeting industry standards. Some public libraries were used, however all code is of original work. All RF and safety standards have been met and accounted for.

**ACKNOWLEDGEMENT**

The authors wish to acknowledge the assistance and support of Boeing and Leidos for their financial contributions for this project.

**BIOGRAPHY**

Daniel Warner is a senior at the University of Central Florida. He plans to graduate with a degree in Electrical Engineering. He plans to continue his education at UCF pursuing his Master’s degree. Daniel started as an intern at Motorola and will continue to work there as a graduate intern.

John Parker is a senior at the University of Central Florida. He plans to graduate with a degree in Electrical Engineering. After graduation John will start working at Kennedy Space Center as a NASA contractor with Jacobs.

Justin Dwyer is a senior at the University of Central Florida. He plans to graduate with a degree in Electrical Engineering. Justin has worked for Northrup Grumman as an intern and after graduation Justin will start working at Northrup Grumman full-time.

Duy “Bob” Nguyen is a senior at the University of Central Florida. He plans to graduate with Honors with a degree in Electrical Engineering. After graduation Bob will start working at Texas Instruments as an application engineer.

**REFERENCES**