Brain-controlled Portable Programmable Embedded System

Final Report

Yujie Chen | Shiyang Liu | Xuanyu Zhong

TA: Lydia Majure

April 29, 2013

Abstract

This project is about building an embedded system that can be controlled by human brains. It utilizes the EEG technology. The system has groups of LED lights blinking at different frequencies. The user controls the system by staring at one group of the LED lights. The EEG technology will detect which LED the user is staring at and the system will react accordingly. In that way, user can interact with the system using only “brain” instead of hands or voices.

The system contains 5 modules, communication, micro-controller, display, power and detection. The detection module is sponsored by James Norton’s lab. The display module has 5 LED lights in 4 groups and a LCD. Each group of LED light corresponds to a control option to interact with the screen. The detection module can tell which LED light the user is staring at by classifying the neuro-signals generated through the scalp and send out that information through Bluetooth. The communication module has a Bluetooth transceiver to receive detection results from detection module and passes it to the microcontroller. The microcontroller is based on an Arduino Mega 2560 programmable board. It analyses the information it gets and update the content on LCD accordingly. Power module consists of three 9V batteries which provide power to microcontroller and the LED lights.

The project was completed with the expected functionalities working. However future improvements can be made in various aspects to make the system more user-friendly, more energy-saving and more reliable.

Table of Contents

Introduction

1.1 Statement of Purpose

The technology of detecting and analyzing electroencephalographic (EEG) signals already exists in labs. For instance, when we stare at a light source which blinks at a specific frequency, our brain will generate a signal at the same frequency along the scalp. The EEG technology is able to detect, amplify and recognize this particular neural signal, which gives us a new way to interact with machines: directly from brain. However, the equipment used for EEG signal detection and analysis is not portable and currently there are no devices integrated with a software system on the EEG technology. Both drawbacks prevent this new technology from prevailing in market. In this project, we aim to create a product which is specially designed for EEG technology in order to provide better user control experience in general. It has the wireless communication component to enable portability and an embedded system which takes EEG signal as input.

1.2 Functionality and Features

The device contains groups of LED lights which blink at different frequencies around the LCD so as to visually stimulate the generation of EEG signals along the scalp, which can be captured by the analyzing equipment. The LCD aims to provide interactive displays. The device wirelessly obtains analyzed signals from the detection module through Bluetooth, and then classifies and computes the signals and updates the LCD based on the software system integrated in the microcontroller. Brain-control-friendly embedded software system is implemented. The software of detection module is also modified to better integrate into this project. All these objectives of the project up from the design stage are met eventually.

1.3 Blocks and Modules

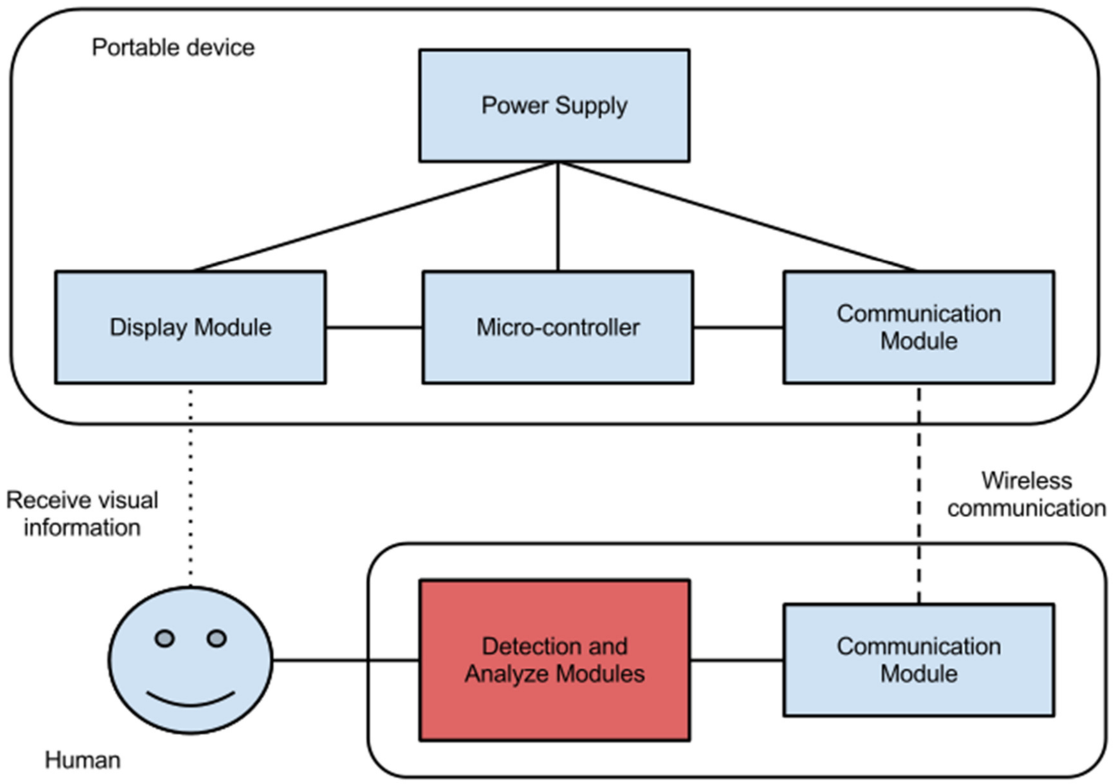


Fig.1.3 Overall Block Diagram

Power Supply:

Three 9V alkaline batteries are used. One is used to provide power to the microcontroller unit and the other two are used to drive the LED flashing circuit. The embedded 5V/3.3V modulator on the Arduino board is also utilized to provide converted power source to the Bluetooth module since a 5V logic source is required for the Bluetooth transceiver chip.

Microcontroller:

We use the Arduino Mega 2560 microcontroller board to analyze wireless signals and drive a LCD with digital I/O pins. It is programmed to react to different commands (selection of different LED arrays) analyzed from input signals from the user's brain, execute the command accordingly or provide the next list of commands for the user to select from by updating the LCD. The board is powered by the 9V external power supply.

Display module:

A SainSmart 3.2" TFT LCD touch screen is used. It is surrounded by arrays of LEDs flashing at varying frequencies. By looking at one of these regions, the brain will generate signal that can be distinguished by the detection module through EEG. In this way, the regions act as menus that can be selected by brain. The LCD has integrated power and a 40 pins interface and SD card reader design. It is relatively convenient to program and provides vivid graphics, which serve as the primary reason for choosing this device.

Communication module:

The communication interface implements wireless transmission of signal data from detection module to micro-controller. This allows synchronization between the generation of EEG signals along the scalp and the computation of microcontroller in order to update displays as well as next set of displays based on user's intention. The chip we are using is the HC-05 Bluetooth transceiver module. The Bluetooth module operates at 3.3V and is wired to the Arduino board through RX and TX ports. Analyzed signals are sent wirelessly from the computer in the detection module, picked up by the Bluetooth receiver, inputted to the microcontroller and will finally take action on the current screen.

Detection module:

The detecting and analyzing modules capture the EEG signals from user by recording brain's spontaneous electrical activity over a short period of time with EEG sensors attached along the scalp. The signals detected are then amplified so that they can be applied to the algorithms of classification and wirelessly transmitted to the transceiver and microcontroller. This module is provided by James Norton and his group.

Design

2.1 Power Supply

We used three 9V alkaline batteries as the power supply. Two of them are used to drive the LED flashing circuit. Another one is used to provide energy for Arduino Mega 2560 board.

For the flashing circuit, LED lights driven by the batteries with supply voltage below 9V are not bright enough and cannot stimulate the user brain effectively. We have also tried 12V power supply. It actually gives better performance. The LED lights are brighter and it is easier for user to react to the flashing of LED lights. However, power supply is too large and it has to be connected to an external power source. Its lack of portability made us choose the 9V batteries in our final design. It is the best combination of portability and effectiveness.

The battery connected to Arduino board not only drives the board itself, but also drives the LCD and the Bluetooth receiver. LCD is connected to the Arduino board through a shield. And Bluetooth is driven by the 5V DC output pin from the Arduino board.

The basic concept is illustrated in Fig. 2.1 below.

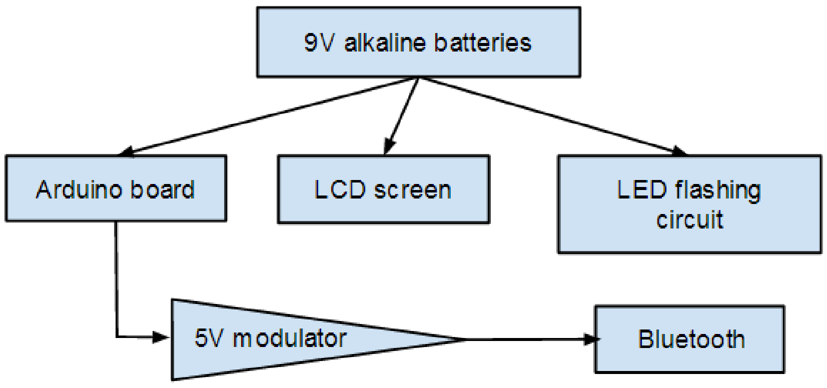


Fig.2.1 Illustration of Power Supply

2.2 Microcontroller

2.2.1 Arduino Mega 2560 R3

Arduino Mega 2560 is used as the main microcontroller in this project. Our design requires an LCD. Normally the LCD on the market cannot be directly connected to Arduino board. We have to connect the pins on the LCD with the pins on the Arduino board one by one using wires. Luckily, we have found a shield that provides us with an interface between Arduino Mega 2560 board and the LCD. One side of the shield has rabbets for pins from LCD and the other side has pins to fit into the rabbets on the Arduino board. With that shield, we don't have to manually connect the board with the screen, and the design is more concise and stable. The pin connections between Arduino board and the LCD are illustrated in Fig. 2.2.1 below. Shield is left out here since it only acts like wires and doesn't affect the actual pin connection. The Arduino board is also connected to a Bluetooth transceiver. Only 4 pins from Arduino board are used. VCC pin and GND pin (pin 10 and pin 11) supply energy to the transceiver. Pin 2 and pin 3 are used for data exchange between board and the Bluetooth receiver.

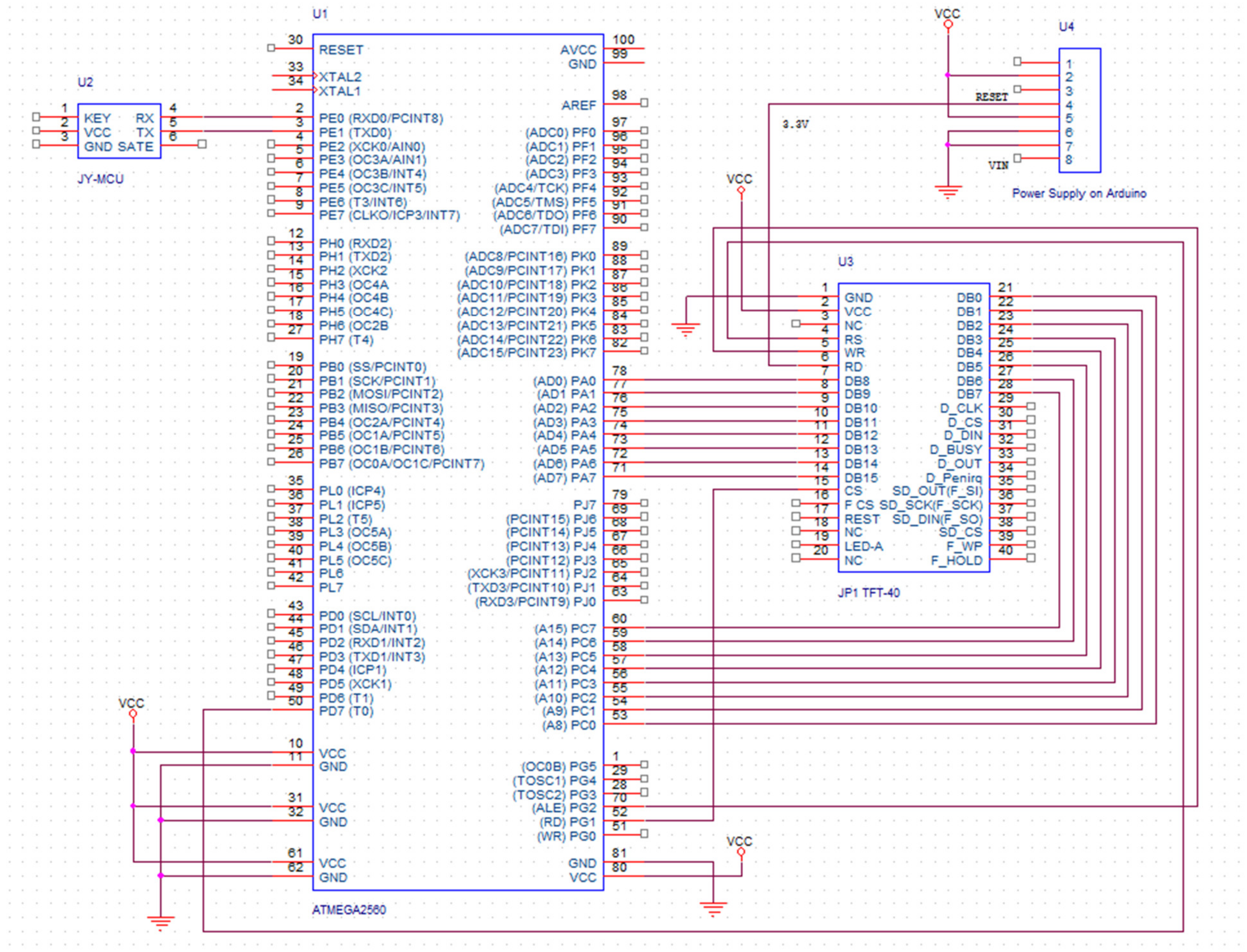


Fig.2.2.1 Schematics of Arduino and LCD and their connection

2.2.2 Arduino-based embedded programming

The Arduino board is programmed with a software system using the Arduino programming environment. The software system has 2 levels for demonstration purpose. The first level is the menu screen. It has 5 items at this stage: Setting, Picture, Music, Text and Video. The second level is the content screen for each item. The software logistic flow chart of the system is illustrated in Fig. 2.2.2.1 below.

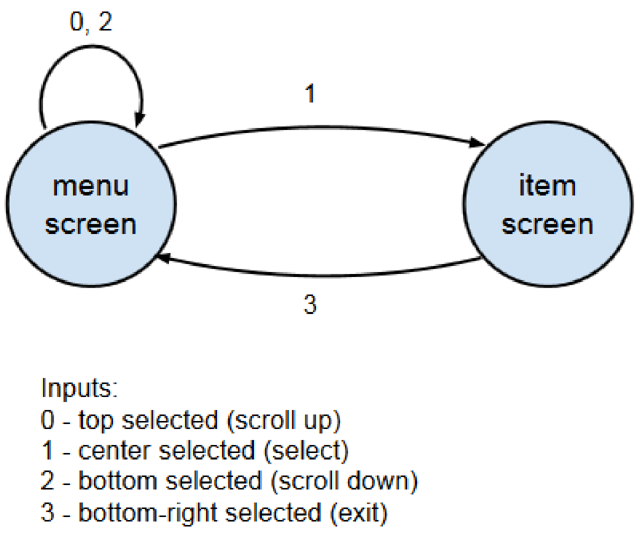

0, 2
1
3
menu screen
item screen
Inputs:
0 - top selected (scroll up)
1 - center selected (select)
2 - bottom selected (scroll down)
3 - bottom-right selected (exit)

Fig.2.2.2.1 Software Logistic Flow Chart

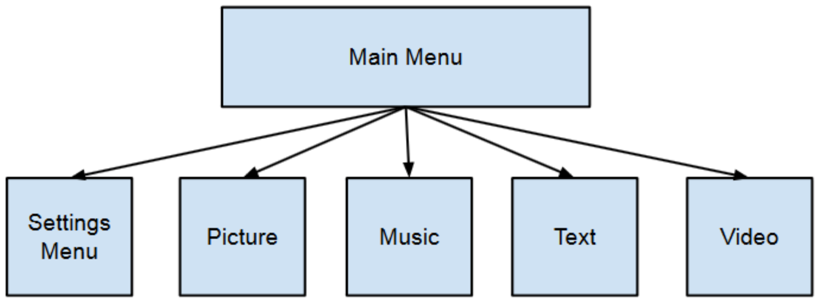


Fig.2.2.2.2 Software Logistic Hierarchy

The software system also applies UTFT library support for graphical design and programming. This library provides us with various functions we can use to deal with the LCD. The typical functions that we use widely are clearing the screen, drawing bitmap images on the screen and printing text on the screen.

2.3 Display Module

2.3.1 LED Flashing Circuit (based on LM348 Op Amp)

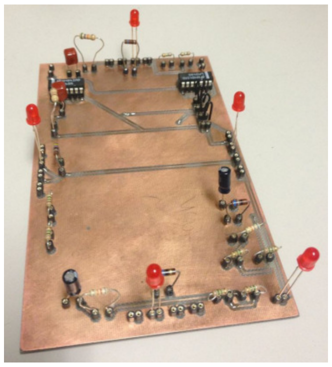


Fig.2.3.1.1 LED Flashing Circuit

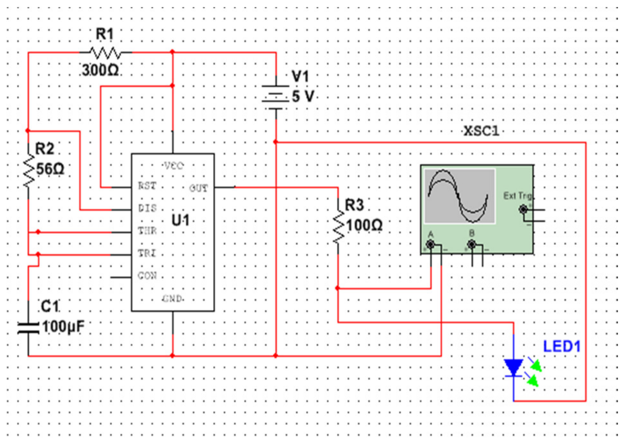
Although a variety of logic chips can be used to accomplish the tasks, the design based on LM348 Op Amp can be concluded from results of experiments as the most stable and reliable among all the others, for example, Timer 555 IC.

The main purpose of the LED flashing circuit is to stimulate EEG signals long the scalp which can be eventually collected and classified by the detection module. According to James Norton, our sponsor on the EEG detecting technology, the stimulation of detectable EEG signals requires extremely high performance of the flashing circuit. After conducting several experimentations on a variety of square-wave-generating logic chips and by observing the resulted waveforms as well as testing on the detection performance, we are able to conclude that the waveforms of the square wave have to strictly follow the requirements below:

- The duty cycle of the waveform has to be 50% on/off
- Frequencies have to be extremely stable without significant bandwidths
- Clean square waveforms without shifting

Once the above requirements are satisfied, the circuit is then capable to stimulate signals that could be detected by the detection module.

Our original plan was to utilize the Timer 555 IC, considering that it is a popular choice for electrical engineers when working on square wave generation circuitries. The basic connection is illustrated in the figure below. However, we found that the resulted waveforms do not meet the requirements, as mentioned above. Even after trying on different combinations of capacitors and resistors, we were still not able to improve the performance, as illustrated in Fig. 2.3.1.2 below.



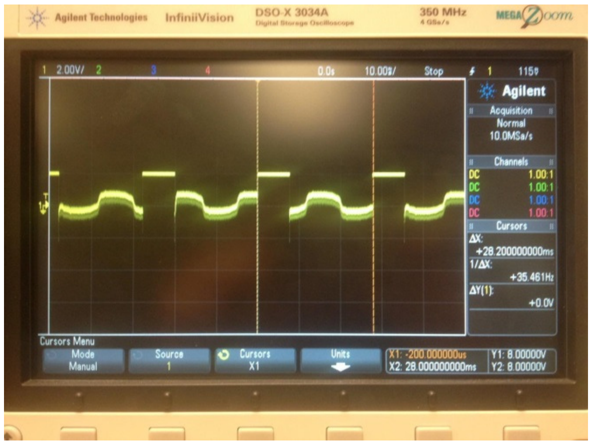


Fig.2.3.1.2 Schematic of Timer 555 IC design and screenshot of resulted waveform

To improve the performance so as to generate better square waves that are capable to stimulate detectable signals, we switched to instead use LM348 Op Amp chip. The schematic is illustrated in Fig. 2.3.1.3 below.

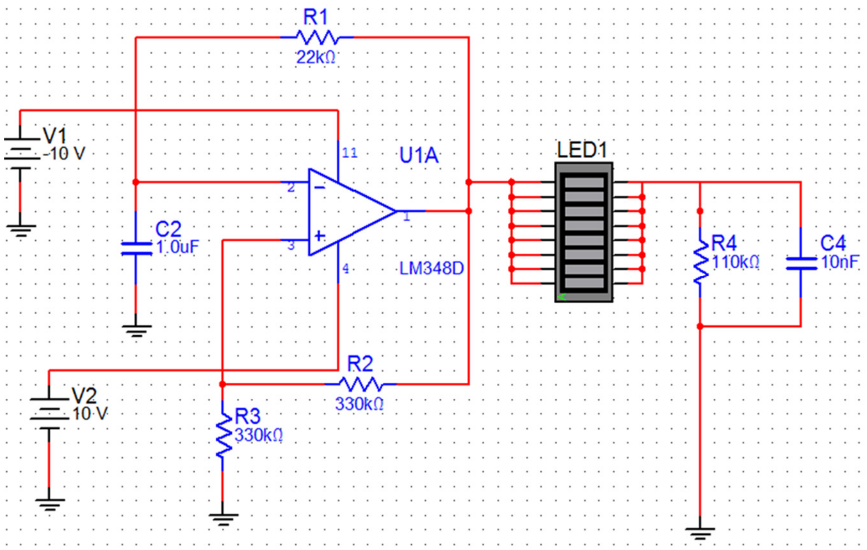


Fig.2.3.1.3 Schematic of LM348 design

This circuit adopts a quad general-purpose operational amplifier as an astable multi-vibrator. The output of the op amp is connected to the LED lights. Let

$$\mathbf{k} = \frac{\boldsymbol{R_3}}{\boldsymbol{R_2} + \boldsymbol{R_3}}$$

Assume V_1, V_2, V_3 are the voltage level at pin number 1 (output), 2 and 3 (inputs) of the LM348 chip. Therefore:

$$V_3 = kV_1$$

When the output voltage V_1 is at +Vcc, the capacitor C_2 should have a voltage level of less than +kVcc and will be charged until +kVcc which suddenly flips the output voltage V_1 to -Vcc. The same story applies when the same capacitor discharges itself until -kVcc which flips the output voltage V_1 back to +Vcc. This gives us a continuous square wave oscillating between +Vcc and -Vcc.

The generation process is described by the following general charging equation, where q_{C2} is the charge stored on capacitor C_2:

$$q_{C2} = C_2V_{cc}\left(1 - e^{-\frac{t}{R_1C_2}}\right) + kC_2V_{cc}e^{-\frac{t}{R_1C_2}}$$

It charges from -kVcc to +kVcc, taking half the period of a square wave. The full square wave consists of a charging as well as a discharging phase.

$$\frac{T}{2} = R_1C_2ln\left(\frac{1+k}{1-k}\right)$$

$$T = 2\ R_1C_2ln\left(\frac{1+k}{1-k}\right)$$

We have also implemented simulation of the circuit above using Multi-sim. Based on the combination of capacitors and resisters and their values, we are able to first calculate the theoretical frequency:

$$T = 2\ R1 * C2 * ln\left(\frac{1+k}{1-k}\right) = 0.04834\ second$$

$$k = \frac{R3}{R2 + R3} = \frac{330}{330 + 330} = 0.5$$

$$f = \frac{1}{T} = 20.69\ Hz$$

According to the waveform and the simulation result given by Multi-sim, as illustrated in Fig. 2.3.1.4 below, we are able to calculate the simulated frequency:

$$T = 0.492\ second$$

$$f = \frac{1}{T} = 20.32\ Hz$$

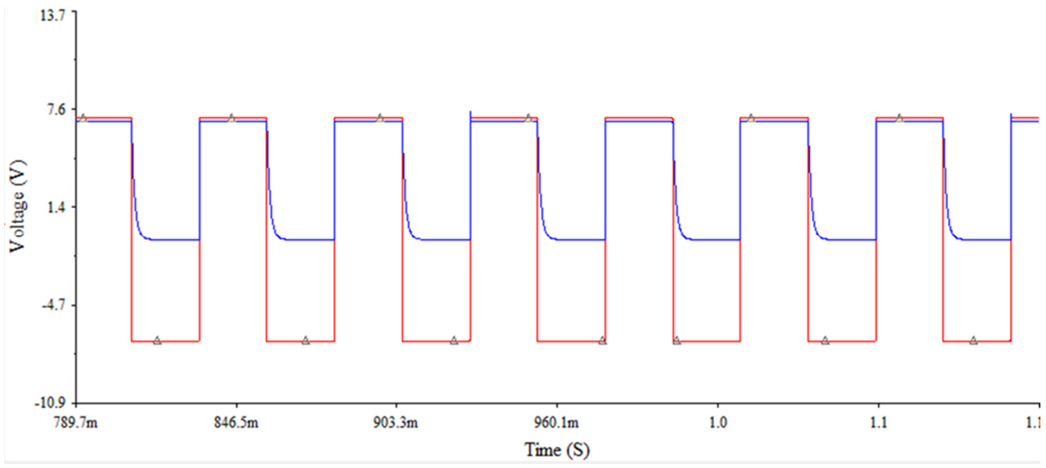


Fig.2.3.1.4 Simulation result of LM348 design

We also observe that the frequency obtained according to the simulation result is fairly close to the theoretical frequency calculated above.

2.3.2 SainSmart 3.2" TFT LCD with touch panel

The 3.2" LCD fits perfectly into our project considering its screen size as well as the connectivity to an Arduino Mega board, which are both important requirements of this project. The LCD also has variety of colors which can be taken advantage of to provide users vivid and interactive displays. The LCD comes with the functionality of touch panel, and we were able to utilize this feature to test the software system flow without the need of detection and communication modules. A screen shield is also used to hook up the LCD with the Arduino board and to save additional wires otherwise would be needed for connection. The basic schematic of the LCD is illustrated in Fig. 2.3.2 below.

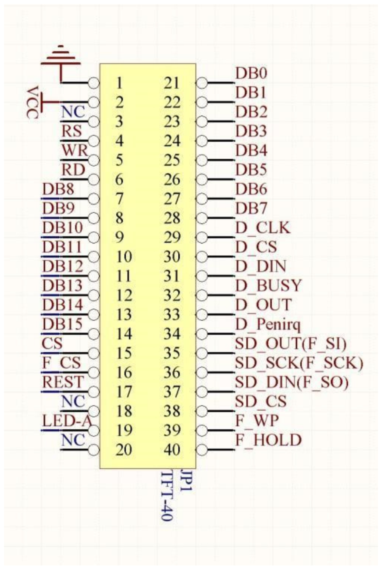


Fig.2.3.2 Schematic of LCD

2.4 Communication Module

2.4.1 HC-05 Bluetooth transceivers (slave)

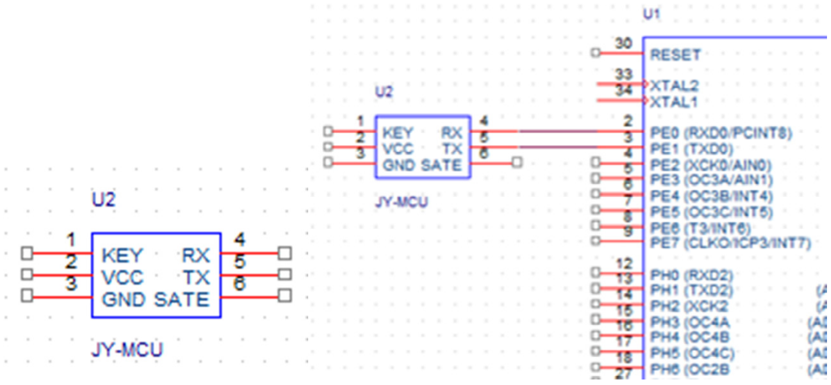
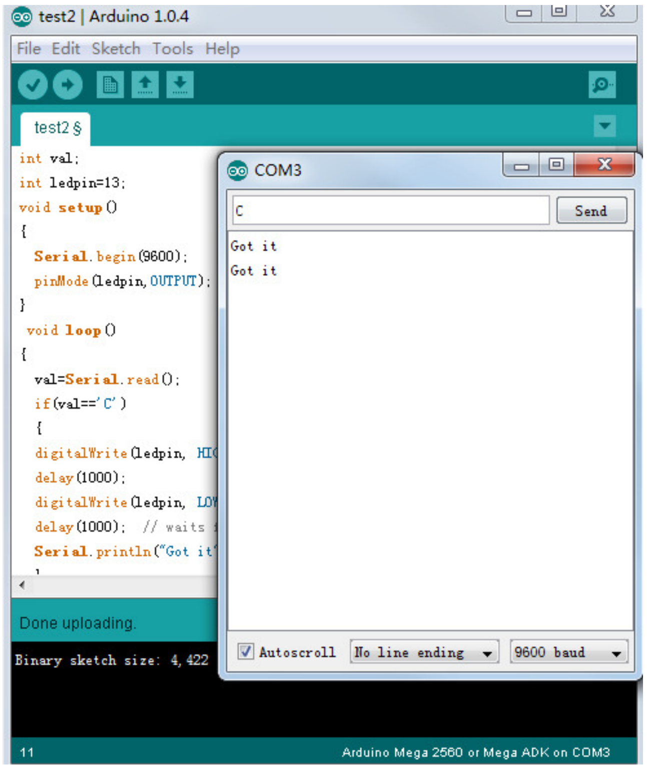


Fig.2.4.1.1 Schematic of HC-05

The left schematic view shows the Bluetooth transceiver itself, the right schematic view shows how the transceiver is connected to the Arduino board. The Bluetooth transceiver here is used in slave mode. It receives Bluetooth signals and passes them through asynchronous serial connection (RX/TX lines) to the Arduino board. When HC-05 device sends a character through its TX line, the RX line of Arduino which is connected to HC-05's TX line, will receive the character.


test2 | Arduino 1.0.4

```
int val;
int ledpin=13;
void setup()
{
  Serial.begin(9600);
  pinMode(ledpin,OUTPUT);
}
 void loop()
{
  val=Serial.read();
  if(val=='C')
  {
  digitalWrite(ledpin, HI
  delay(1000);
  digitalWrite(ledpin, LOW
  delay(1000);  // waits
  Serial.println("Got it
```

Done uploading.

```
Binary sketch size: 4,422
```

COM3

```
C
Got it
Got it
```

Autoscroll [✓] · No line ending · 9600 baud

11 Arduino Mega 2560 or Mega ADK on COM3

Fig.2.4.1.2 Simple test on the Bluetooth connection on HC-05

We wrote a simple program for Arduino that makes it constantly listening to a serial connection on 9600 BAUD. It will turn on LED "L" for 1000 milliseconds and respond a "Got it" when hearing a "C". When we were sending Bluetooth signals on 9600 BAUD from a smartphone to a paired HC-05 connected to Arduino, as we expect, the LED got turned on and a response "Got it" was generated.

2.4.2 Bluetooth module for PC (master)

The PC that runs the MATLAB code on EEG detection and classification should be able to generate specific Bluetooth signals accordingly based on the classified results. The Bluetooth connection here is set up through MATLAB. The code is included in Appendix D. The basic idea is that the Bluetooth receiver HC-05 is treated as a file and file operations such as fopen(), fwrite() and fclose() are used to set up the Bluetooth connection.

2.5 Detection Module

The detection module is provided by James Norton and his lab. The whole module consists of an EEG cap which measures the wearer's brain activity, an amplifier which amplifies the feeble EEG signals, a filter which filters out EEG signals of irrelevant frequencies and a MATLAB program on PC which detects and classifies EEG signals. The first three hardware components are configured and maintained by James Norton while we modified and optimized the MATLAB code for better response time and lower misclassification rate.

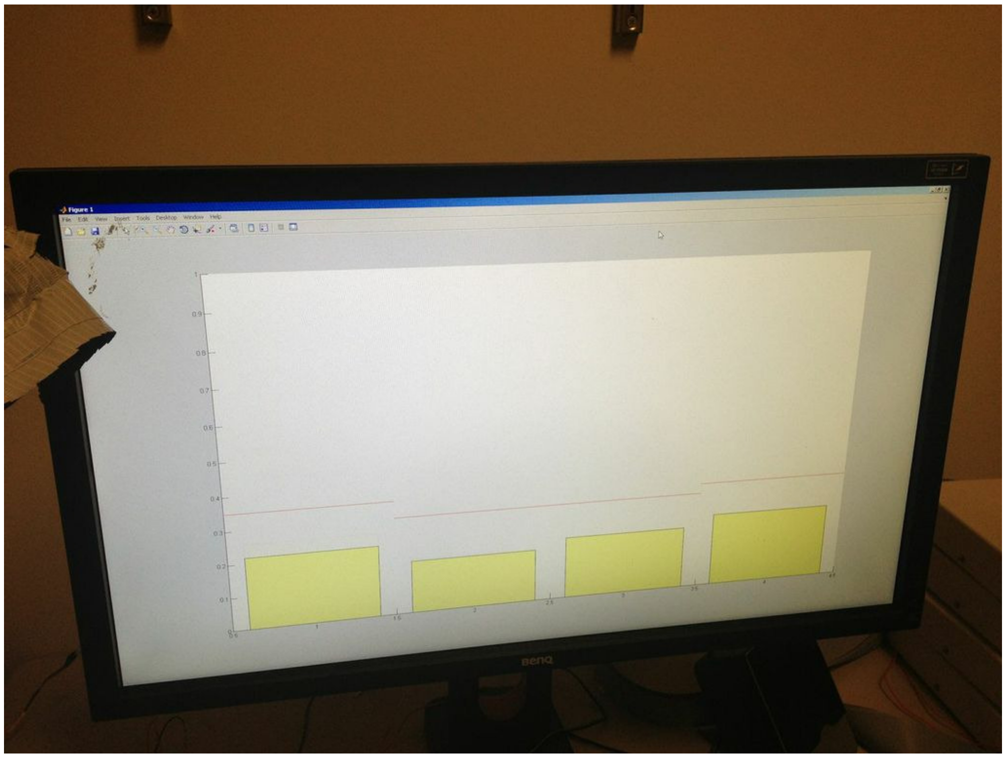


Fig.2.5. MATLAB interface showing EEG signals triggered by four frequencies

As shown in Fig. 2.5, the MATLAB interface has four fluctuating yellow columns. They stand for strengths of EEG signals triggered by four LED flashing frequencies. The four frequencies here are set to 5.7Hz, 7.1Hz, 7.9Hz, and 9.4Hz. The tolerance on each frequency is set individually to boost the corresponding column without affecting the other unrelated columns.

The four red lines on top of each column are the thresholds on signal classification. As soon as a certain threshold is met (exceeded), the corresponding Bluetooth command gets sent. For a frequency of signals that gets misclassified frequently, we raised its threshold to lower misclassification rate. For a frequency that gets classified correctly each time, we reduced its threshold to get a faster response. In our case, 7.1Hz and 7.9Hz have a lower threshold because it is relatively easy to classify EEG signals triggered by these two frequencies.

2.6 PCB

The PCB design involves a few considerations. It is relatively easier to build the board for Bluetooth chip as the design is very simple and the size and position are not concerns. On the LED flashing circuit side, we want to leave out a space with no chips or sockets on it in the center of the board where we are placing the Arduino board along with the LCD. Therefore, we have to place the other chips and sockets as on the boarder as we can. However, with the number of these components we have in our design and the consideration of having wider traces and enough spacing for easier fabrication and soldering, it took us some time to improve the spacing before we were able to finalize the design. Besides, since we have a separate board for Bluetooth, we placed several jumpers on both boards for later connection. Detailed information about PCB designs can be found in the Appendix.

Requirements and Verifications

3.1 Power Supply

The power source being used in this project are three 9V batteries. And the concern is that whether the output voltage from the battery is steady at about 9V enough to be used to support the Arduino board. As we mentioned in the Requirements and Verification Table (see Appendix), we need to test whether the input voltage consistently fall in the range of 8.5V to 9.5V, which is required by the Arduino board. In order to perform the test, the following circuit is to be built:

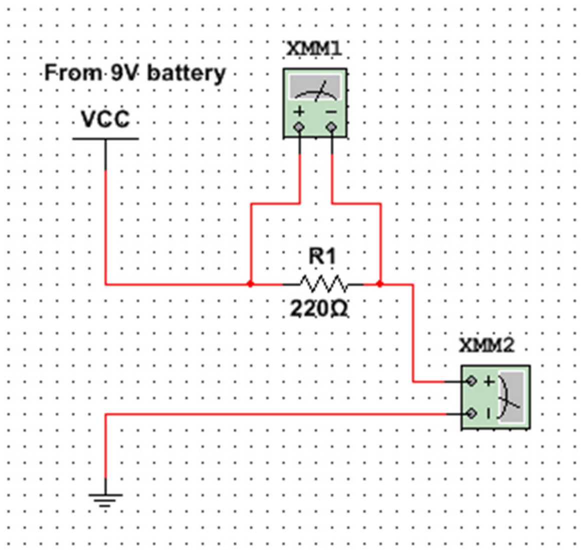


Fig.3.1.1 Circuit for power 9V verification

The circuit simply places the output voltage from the battery across the resistor. We use a set of different configurations (resistors of different values) to repeat the test 5 times. For each configuration, we can obtain the result (voltage) by measuring the current across the each resistor, and according to the following formula:

$$R = \frac{V}{I}$$

And the testing result can be found in Table:

Resistance (Ω)	Current (mA)	Voltage (V)
33	266	8.78
45	196	8.82
56	157	8.79
78	113	8.81
88	100	8.80

Table 3.1.1 Resulted voltages from 9V battery source

We are able to observe from the above table that the voltages across the resistors are consistently close enough to 9V, which meets the requirements indicated in the Requirements and Verification Table.

Similarly, the power used in display module or communication module is either 5V or 3.3V. We have to test whether the 9V power source gets modulated to 5V and 3.3V in a reliable and steady manner. In order to perform this test, the following configuration is set up:

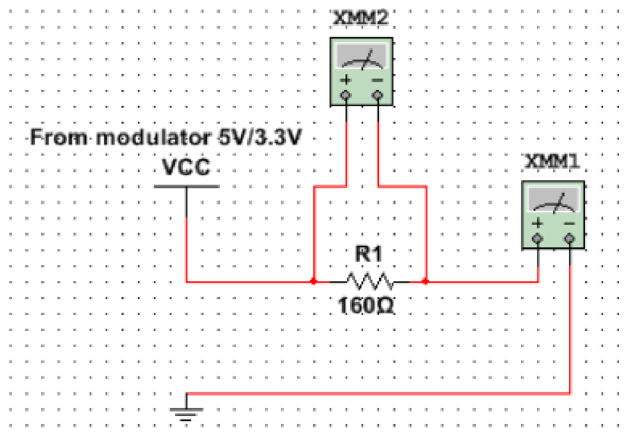


Fig.3.1.2 Circuit for power 5V/3V3 verification

The procedure of verification is very similar to that of the previous verification (of 9V batteries). We use a set of different configurations (resistors of different values) to repeat the test 5 times. For each configuration, we obtain the result (voltage) by measuring the current across the each resistor. The testing results can be found in the following tables:
And the testing result can be found in the following table:

Resistance (Ω)	Current (mA)	Voltage (V)
33	146	4.83
45	107	4.83
56	87	4.86
78	62	4.84
88	55	4.86

Table 3.1.2 Resulted voltages from 5V output from Arduino built-in modulator

Resistance (Ω)	Current (mA)	Voltage (V)
33	95	3.12
45	69	3.12
56	56	3.12
78	40	3.13
88	36	3.13

Table 3.1.3 Resulted voltages from 3.3V output from Arduino built-in modulator

3.2 Microcontroller Unit

The major issue regarding the software logic of the LCD display flow driven by microcontroller is that if it is able to guarantee the correctness of the display shown in each level of the state diagram. In order to perform this test, we first have to make sure that the touch functionality is enabled and properly programmed so that it can be used to control the screen in this test without the need of flashing LEDs and the whole detection and feedback modules. Secondly, once we are done configuring the touch panel, connect the LCD screen with the microcontroller and the power source to build the testing circuit, which can be found below:

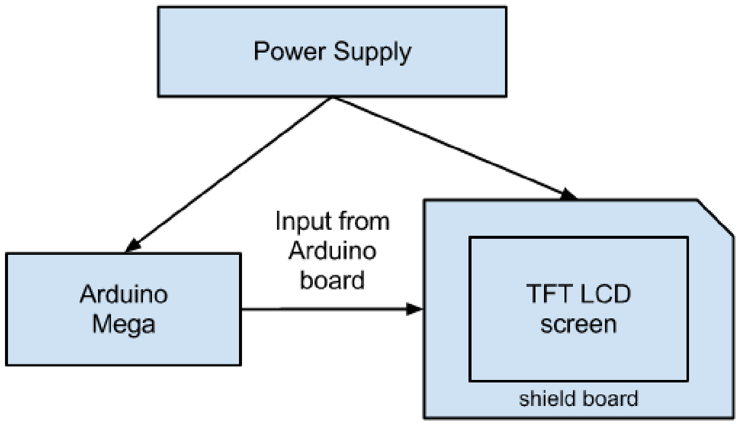


Fig.3.2 Testing circuit for verification of the microcontroller

As we indicated in the *Requirement and Verification table* in the Appendix, test the basic software level screen transitions between levels (main screen -> music screen -> main screen) for 20 times and record result (successful or not) and transition time of each transitions in Table 3.2.

Test #	Result (Y/N)	Transition time (second)
1-20	Y	< 0.2

Table 3.2 Result of software level screen transitions

As shown in Table 3.2, the screen reacts to all the control signals as designed in the software flow chart within a reasonable transition time.

3.3 Display Module

LED lights are used to stimulate user brain. Every LED triggers a corresponding signal with same frequency in user brain when user stares at it. And different LED lights blink at different frequencies because different LED lights correspond to different user inputs.

To achieve that, there are 2 things that need to be verified. The first is that LED lights should be driven by input signals at frequencies between 5Hz and 15Hz with 50% duty cycle, and stable waveforms. In that way, the LED lights will blink with half time on and half time off, and the frequencies of flashing will be quite stable. The LED lights blinking in this pattern are considered to be the ideal stimulations.

We use the oscilloscope to observe the waveforms and check duty cycles and stability. Fig. 3.3 is a typical waveform.

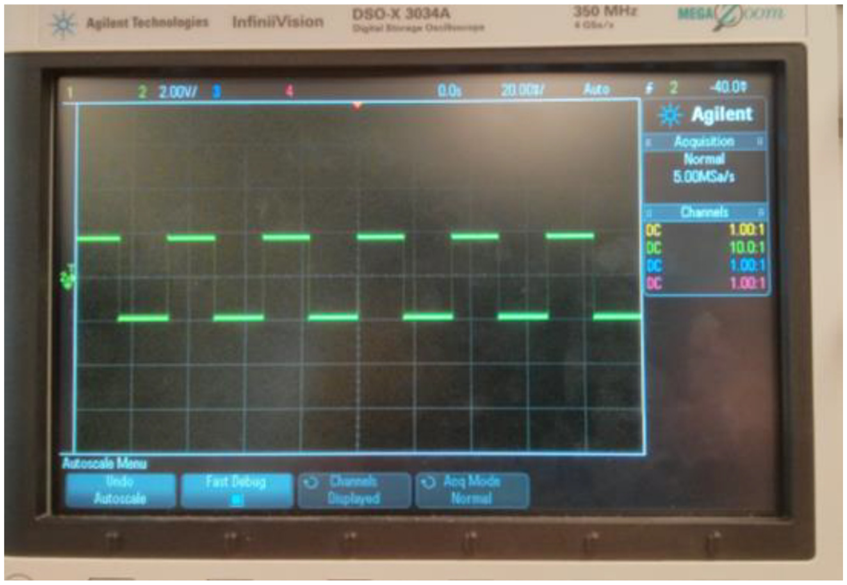


Fig.3.3 Screenshot of resulted waveform of LM348 design

As we can see, the duty cycle is clearly 50%, and the waveform is at a stable shape.

The second thing that needs verification is that LEDs blinking at different frequencies are not interference to each other so that the detection module can recognize the correct LED light the user is staring at. For example, we have 4 frequencies for LED lights, 5.7Hz, 7.1Hz, 7.9Hz, and 9.4Hz. When the user is staring at LED blinking at 5.7Hz, we don't want detection module to think it blinks at 7.1Hz, 7.9Hz or 9.4Hz.

We set up detection module and 4 groups of LEDs with frequencies 5.7Hz, 7.1Hz, 7.9Hz and 9.4Hz. The user then looks at a random LED and we check if the correct frequency is detected. We repeated 20 times and the result is shown in Table 3.3 below.

Frequency intended (Hz)	Frequency detected (Hz)
5.7	5.7
7.1	7.1
7.9	7.9
9.4	5.7
5.7	5.7
7.1	7.1
7.9	7.9
9.4	9.4
5.7	5.7
7.1	7.1
7.9	7.9
9.4	9.4
5.7	5.7
7.1	7.1
7.9	7.9
9.4	9.4
5.7	5.7
7.1	7.1
7.9	7.9
9.4	9.4

Table 3.3 Results of verification of LED frequencies

As we can see, 19 out of 20 times the detection module gives the correct result. The accuracy is 95%, which meets the requirements in the design review. That means our LED lights work as we expected.

3.4 Communication Module

The communication module consists of two parts, the master and slave. The distance of transmission is set at 10 feet. The HC-05 receiver as the slave which is connected to Arduino through UART, should be able to receive Bluetooth signals correctly. We use a Bluetooth app on a smartphone to send commands to HC-05 module and check if the indicator on Arduino responds. This test is repeated for 10 times.

On the other hand, the master, which is the PC with a Bluetooth adapter, should be able to send Bluetooth signals correctly to HC-05. We use MATLAB on the PC to pair and send commands to the HC-05 receiver and check if the indicator on Arduino responds. This test is repeated for 10 times.

Master/Slave	Test #	Result (Y/N)	Transition time (s)
Slave	1-10	Y	< 0.5
Master	1-10	Y	< 0.5

Table 3.4.1 Results of verification on communication module

As shown in Table 3.4, both tests yield no communication loss or latency violation.

After passing the basic functionality test on master and slave module, we incorporate the response time in the detection phase into total communication cost and check if every response from brain to display is instantaneous (< 2.5 seconds) for each triggering frequency.

LED frequency (Hz)	Test #	Transition time (s)
5.7	1	2.9
	2	3.2
	3	6.2
	4	1.3
7.1	1	2.8
	2	2.3
	3	1.2
	4	2.2
7.9	1	1.9
	2	1.7
	3	1.6
	4	2.0
9.4	1	12.2
	2	2.3
	3	8.2
	4	6.7

Table 3.4.2 Total communication cost for each frequencies

As we see, the response time for certain frequencies is far from instantaneous. As shown in Table 3.4.3, 5.7 Hz and 9.4 Hz have extremely large response time. Considering that we increase the threshold for these two frequencies to get lower misclassification rates, a slower response is expected. However, for 7.1 Hz and 7.9 Hz, a signal from brain to display mostly takes less than 2.5 seconds.

3.5 Modifications

As indicated in the Requirements and Verification Table (see Appendix), we have originally chose to implement the four groups of LED lights of the display module to blink at frequencies of 31Hz, 33Hz, 35Hz and 37Hz. However, after several experiments on the LED flashing circuits and by observing the classification results of detection module, we have decided to instead set the frequencies within the range of 5Hz to 15Hz). According to James Norton, the latter frequencies have better performance for EEG detection, which is also the main reason we made such change.

Costs

4.1 Labor

Name	Hourly Rate	Total Hours	Total = Hourly Rate x 2.5 x Total Hours
Yujie Chen	\$40.00	120	\$12,000
Shiyang Liu	\$40.00	120	\$12,000
Xuanyu Zhong	\$40.00	120	\$12,000
Total			\$36,000

Table 4.1 Total Cost for Labor

4.2 Parts

Item	Quantity	Cost(\$)
Arduino Mega 2560	1	45.69
Screen shield for Arduino Mega 2560	1	19.68
SainSmart 3.2’’ TFT LCD with touch panel	1	17.99
Arduino Bluetooth Transceiver Module RS232	1	8.91
9V power adaptor	3	6.30
9V battery	3	9.63
LM348 Op Amp	4	5.00
LED (red)	5	3.60
100µF capacitor	4	2.40
300Ω resistor	4	1.64
10Ω resistor	4	1.60
47Ω resistor	1	0.47
56Ω resistor	1	0.38
68Ω resistor	1	0.49
82Ω resistor	1	0.76
Total		124.54

Table 4.2 Total Cost for Parts

4.3 Grand Total

Section	Total
Labor	$36,000
Parts	$124.54
Total	**$36124.54**

Table 4.3 Grand Total

Conclusion

5.1 Accomplishments

We have created a LED flashing circuit on PCB as the EEG signal stimulator, along with the Arduino board and the LCD in the center of it, which further connect to the Bluetooth chip on a separate PCB. The device works with the detection module and is able to demonstrate all the functionalities as intended. Fig. 5.1 below is a picture of our final product.

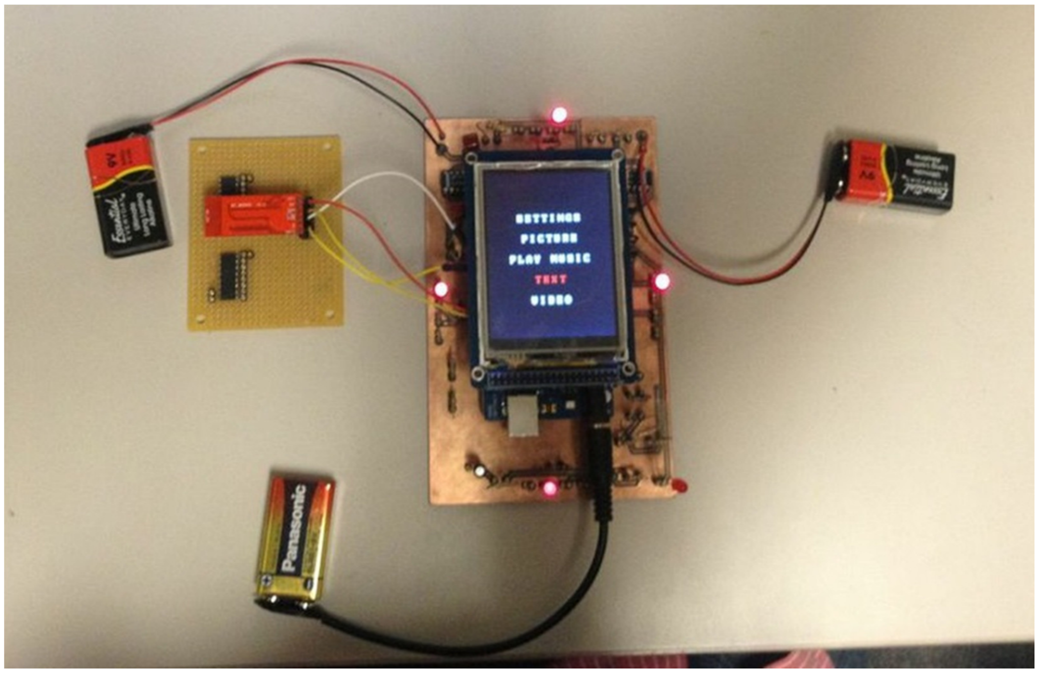


Fig. 5.1 Final Product

5.2 Uncertainties

Although the final design achieves the functionalities we expected, there are still a few uncertainties remained. First, although the battery life meets the requirements, after using the device for an extremely long time, the battery cannot supply the stable power. It is reflected in the waveforms of the input signals of LED lights. The waveform keeps drifting left and right, and the frequency keeps oscillating in a small range. The consequence is that LED lights driven by this kind of signal cannot effectively stimulate user's brain. Second, the user will usually stare at flashing LED lights for quite a long time, we are not sure if it is harmful to user's eyes. Last, we are using red LED lights in our circuit. Whether other colors of LED lights are better or worse than the red still needs to be checked.

5.3 Ethical Considerations

As electrical engineers, we always regard IEEE code of Ethics as our professional guideline. In all the stages of this project, from initial design to final implementation, we will strictly comply with the IEEE code of Ethics and inspect ourselves regularly to make sure we don't violate any of the requirements. Following are the ethical issues that we encounter in the project.

IEEE Code of Ethics	Relevance in Projects
1. To accept responsibility in making decisions consistent with the safety, health, and welfare of the public, and to disclose promptly factors that might endanger the public or the environment.	The purpose of our project is to give people an alternate way of interacting with machines, that is, by brain. It can make people's life more convenient, especially for the disables. The product uses flashing light source as stimulates, which may do harm to people's eyes. We are aware of that and will do various experiments on it until we get the proper flashing frequency and brightness. Also we will make it clearly in our instruction document.
3. To be honest and realistic in stating claims or estimates based on available data.	All the testing data will be recorded honestly and we will adjust our product based on it.
5. To improve the understanding of technology; Its appropriate application, and potential consequences.	Our project is based on the progress in EEG technology, which detects signals generated in human brains. We did a reasonable amount of research on it and understand how the technology works from a high level perspective. We have also seen its laboratory demonstration. In our project, we are focusing on applying the original lab

	application to our daily life, and exposing the potential value of the EEG technology to the world.
6. To maintain and improve our technical competence and to undertake technological tasks for others only if qualified by training or experience, or after full disclosure of pertinent limitations.	We are aware that our current experience and competence are only enough for building the prototype of our design. We will introduce the products to markets only after we gain more training and knowledge.
7. To seek, accept, and offer honest criticism of technical work, to acknowledge and correct errors, and to credit properly the contributions of others.	We are eager to ask for opinions about our product from both instructors and fellow students and we will credit the contributions of them. For advices which we think are valuable we will definitely take them. If other groups are seeking advice from our group, we will treat them with complete honesty.
9. To avoid injuring others, their property, reputation, or employment by false or malicious action.	We have taken the safety training online and have read through all the tutorials and we will strictly follow them all the time when we are in the lab.
10. To assist colleagues and co-workers in their professional development and to support them in following this code of ethics.	Everyone in our group values teamwork highly. Although each person has his own assignments for every week, we are willing to help any member who has difficulties in the project. Also we always remind each other of the importance of following the code of ethics.

Table 5.3 Relevant Ethical Considerations

5.4 Future Work

There is still a lot of space for this project to be improved in the future. First, logic circuitries or multiplexers could be added as a component to control the LED lights. For example, when a group of LED lights are not being used, it could be turned off by the logic circuit, in order to save some power. Besides, frosted plastic/glass covers on top of LED lights would also be helpful to defuse the light and make it more even for better stimulation. Another reason that this additional feature is desired is that, as mentioned in the Uncertainties, the defused light could be more eye-friendly and less harmful. In addition, during this semester, we did not have enough time to work more on the appearance of this product, as all the components and PCBs are not mechanically integrated with each other. Thus, it might also be a good idea to integrate Arduino and LCD better with PCBs to build a more integrated product.

5.5 Acknowledgments

We would like to thank James Norton for providing us with EEG technology and for his valuable help and suggestions. Without him we can never achieve the functionalities of the design. We would like to thank Professor Scott Carney and our TA Lydia Majure for their guidance throughout the process of the project, especially after the first design review. It was our hardest time and they helped us find the way. We would like to thank TA Ryan May and TA Dennis Yuan for their help with our PCB design. We would also like to thank ECE parts shop for teaching us the ironing skill. Without the help of people mentioned above and other TAs and classmates, we can never complete our project.

References

1. Documentation of SainSmart 3.2" TFT LCD + SainSmart TFT LCD Adjustable Shield For Arduino 2560 R3 1280 A082 Plug.
Retrieved from http://www.sainsmart.com/zen/documents/20-011-918/ITDB02_Graph16.rar.

2. Documentation of Arduino Mega 2560.
Retrieved from http://arduino.cc/en/Main/ArduinoBoardMega2560.

3. Library of UTFT.
Retrieved from http://www.henningkarlsen.com/electronics/library.php?id=51.

4. Pin connection between TFT 3.2" LCD and Arduino Mega 2560.
Retrieved from http://www.urel.feec.vutbr.cz/MIA/2011/Matyas/dl/240374PQ.pdf.

5. 555 Timer IC documentation.
Retrieved from http://en.wikipedia.org/wiki/555_timer_IC.

Appendix

Appendix A – Figures and Diagrams

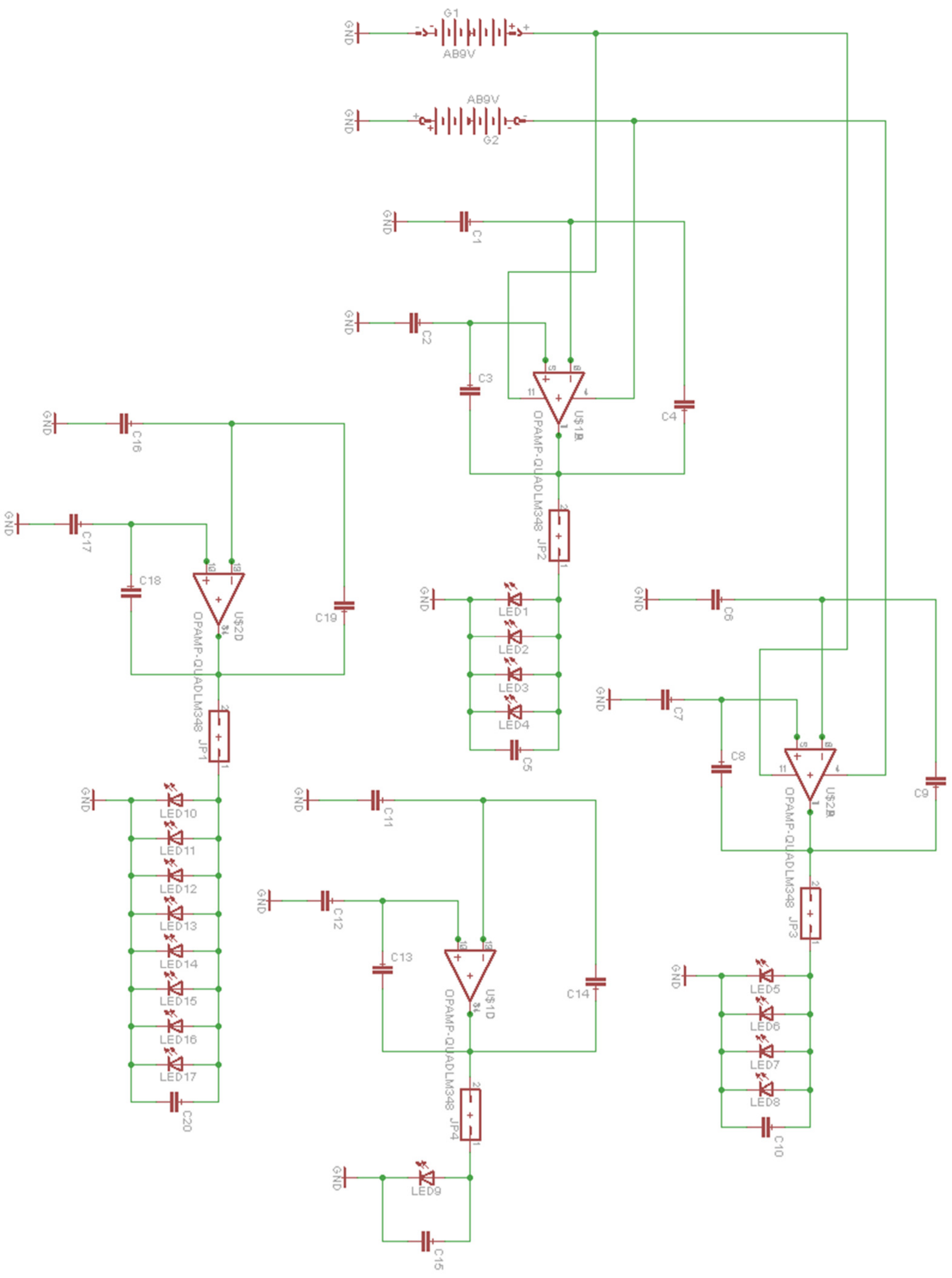


Fig.A.1 Schematics (the whole LED Flashing Circuit)

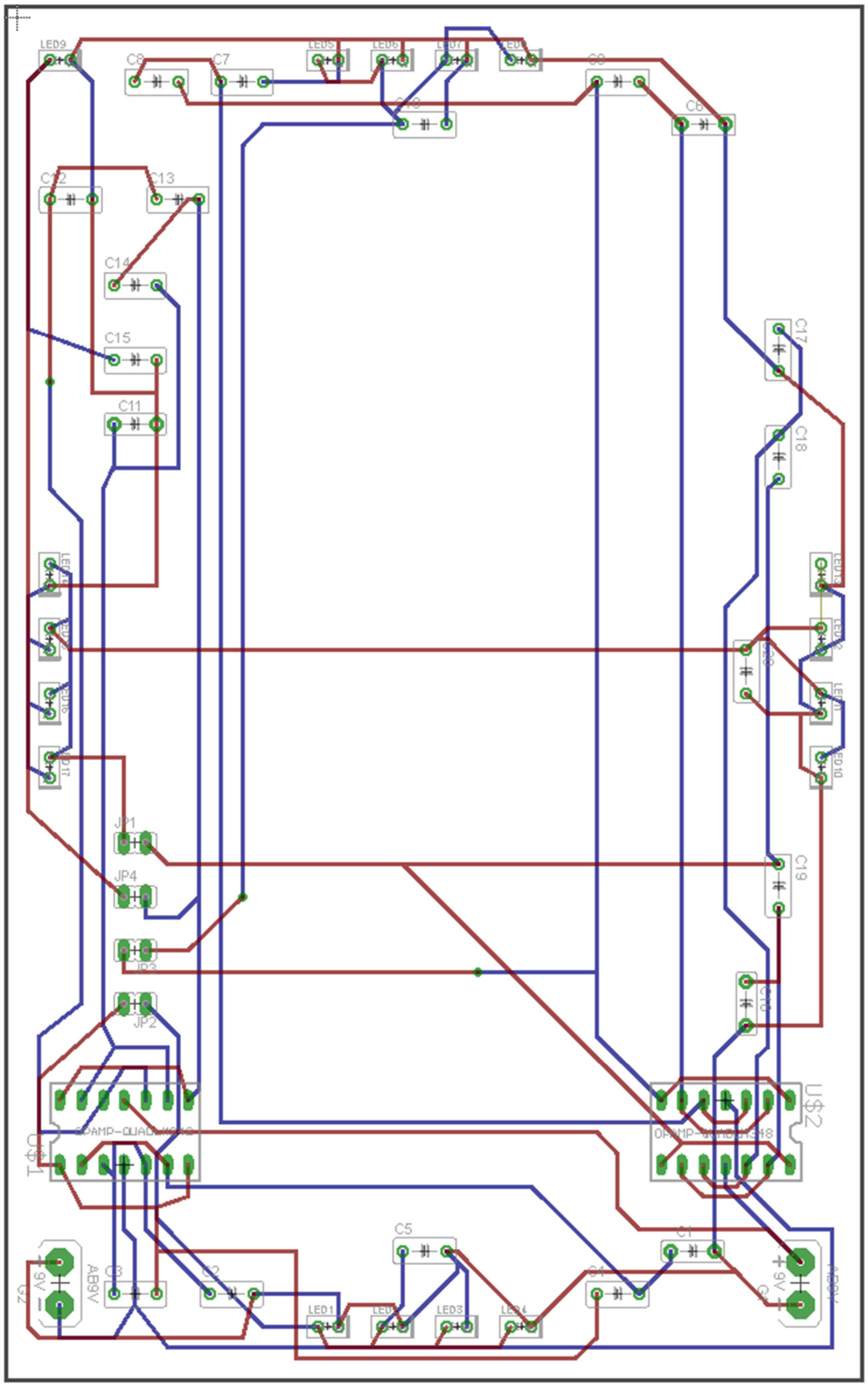
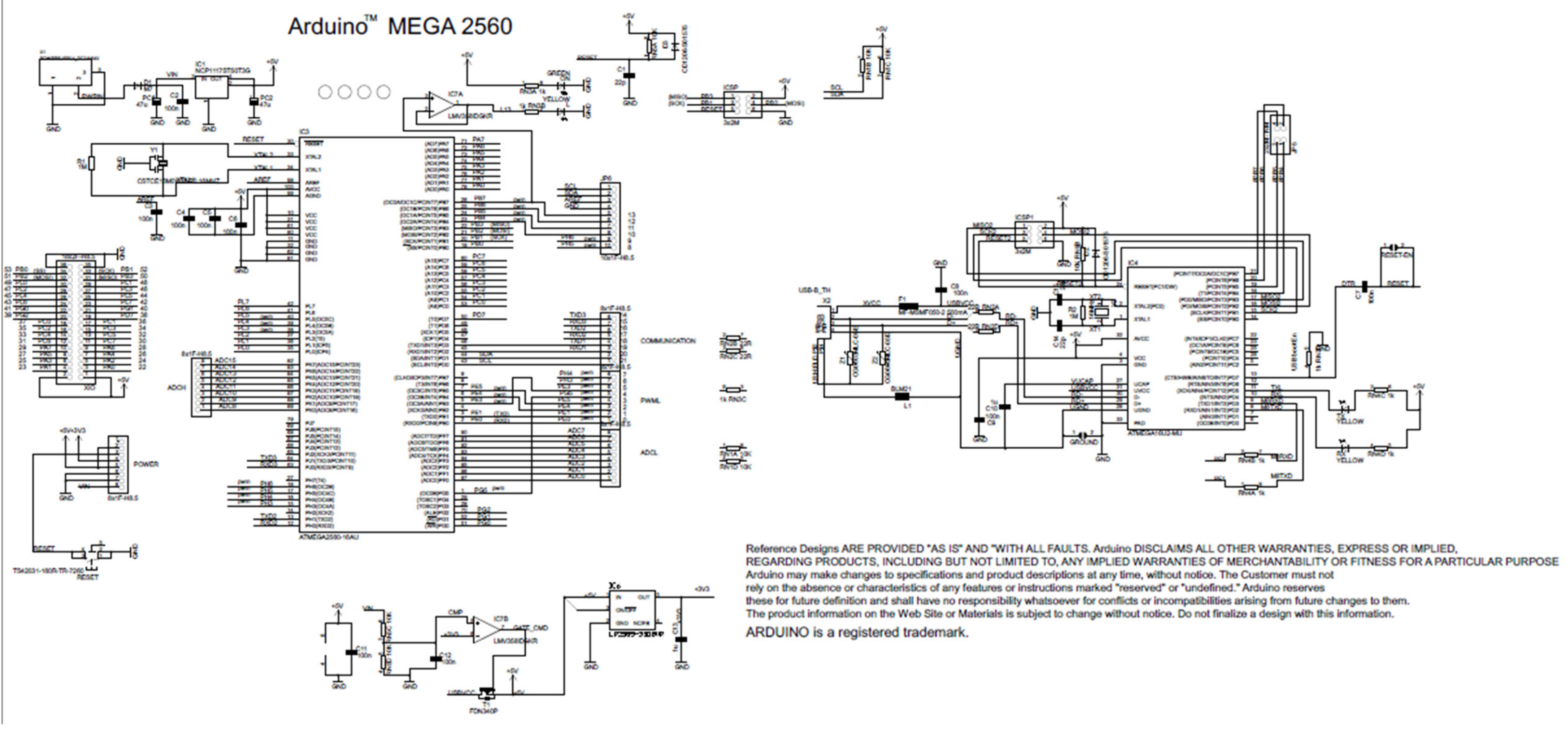


Fig.A.2 PCB Design (the whole LED Flashing Circuit)



Reference Designs ARE PROVIDED "AS IS" AND "WITH ALL FAULTS. Arduino DISCLAIMS ALL OTHER WARRANTIES, EXPRESS OR IMPLIED, REGARDING PRODUCTS, INCLUDING BUT NOT LIMITED TO, ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE
Arduino may make changes to specifications and product descriptions at any time, without notice. The Customer must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined." Arduino reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them.
The product information on the Web Site or Materials is subject to change without notice. Do not finalize a design with this information.
ARDUINO is a registered trademark.

Fig.A.3 Official Schematics (Arduino)

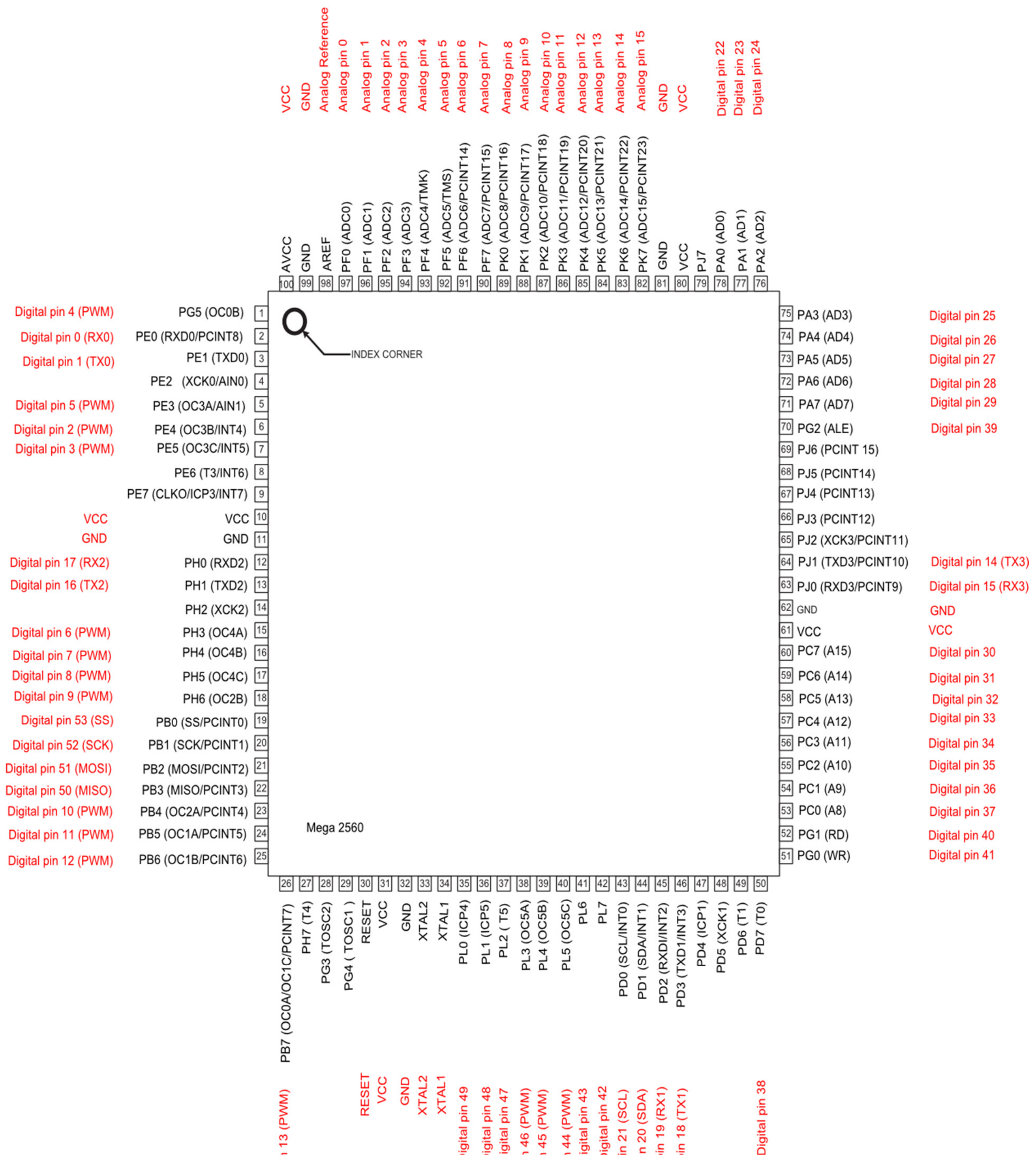


Fig.A.4 Pin map (Arduino)

Appendix B – Requirements and Verification Table

Module	Requirement	Verification
Display	1. LED lights should interact correctly with user (a) LED light blinks at 31, 33, 35, 37Hz stimulate different signals in human brain which can all be detected by detection module. (b) LED lights in the design layout (surrounding LCD) can stimulate signals in user brain when user stare at any of them. 2. LCD can display correct content (a) It should display specific content that microcontroller sends it (b) The transition between different contents should be smooth.	1. (a) - Set up detection module (one human user is required) - Connect LED circuit to make it blink at 31Hz - Put detector on the user - Ask user to stare at LED. Detector should receive a signal in user brain, record it. - Modify circuit to make LED flash at 33 Hz - Put detector on the user - Ask user to stare at LED Detector should receive a signal in user brain, record it. - Modify circuit to make LED flash at 35 Hz - Put detector on the user - Ask user to stare at LED Detector should receive a signal in user brain, record it. - Modify circuit to make LED flash at 37 Hz - Put detector on the user - Ask user to stare at LED Detector should receive a signal in user brain, record it. - Compare 4 signals, they should be distinguishable. - Repeat 10 times. The success rate should be 90%. (b) - Set up detection module (one human user is required) - Set up LED circuit and put LED lights in the final design style. - Turn on all LED lights to make them flash in their own

		unique frequencies (31,33,35,37Hz) - Put detector on user - Ask user to stare the top LED - Detection module should receive signal from user brain. - Check the signal. it should reflect top LED - Ask user to stare the left LED - Detection module should receive signal from user brain. - Check the signal. it should reflect left LED - Ask user to stare the right LED - Detection module should receive signal from user brain. - Check the signal. it should reflect right LED - Ask user to stare the bottom LED - Detection module should receive signal from user brain. - Check the signal. it should reflect bottom LED 2. (a) - Connect LCD with Arduino board - Write test program to display image on LCD - Program on microcontroller - Run the program - LCD should display the correct image - Write test program to display text on LCD - Program on microcontroller - Run the program

		- LCD should display correct text - Repeat for 20 times - The success rate should be more than 95% 2. (b) - Connect LCD with Arduino board - Write test program to display multiple image on the screen - Program on microcontroller - Run the program - LCD should display all images and have no glitch during transition. - Write test program to display multiple texts on LCD - Program on microcontroller - Run the program - LCD should display correct texts and have no glitch during transition. - Write test program to display image-text sequence on LCD - Program on microcontroller - Run the program - LCD should display correct images and texts and have no glitch during transition from image to text and from text to image - Repeat for 20 times - The success rate should be more than 95%
Microcontroller	1. Interface on LCD provides the correct level of interactions. (a) Software system programmed in microcontroller board has correct logistics and fluent control flow. (b) Content is displayed without significant delay.	1. (a) - Attach the microcontroller with the LCD only. - Use the touch panel as input to interact with LCD. - Start from the main menu. - Touch to select “play music” option.

	2. Accurate and lossless data transmission from the Bluetooth module to the microcontroller. Send data in parallel to both the microcontroller and a separate LED which are both connected to the Bluetooth module, compare the data received.	- If jumps to music screen, touch to select “back to main menu” option. - Repeat the above steps 20 times. Expected result: The screen reacts to all the control signals as designed in the software flow chart. The main menu screen transits to the music screen with a touch select of “play music” and the music screen transits back to the main menu screen with a touch select of “back to main menu”. At least 19 times of the test should have the correct transitions. Thus the tolerance is 5% of error rate. 1. (b) - Same setup as 1(a). - Start from the main menu. - Touch to transit to the music menu. - Measure the transition time and record. - Touch to transit back to the main menu. - Measure the transition time and record. - Repeat the above steps for 10 times. Expected results: The transition has a reasonable transition time, which is less than 0.5s. Among the 20 records, at least 19 of them should be less than 0.5s. If not, first check the software logic in the code. And then reboot the device and repeat the testing procedure.

		2. - Build a simple circuit including a data sender (a PC), Bluetooth module, the microcontroller and a single LED. - Also connect the LED with the Bluetooth receiver so that it reacts to what is received by the Bluetooth module. - Send data from the PC side. - Compare the reactions of both LED element and the microcontroller (which outputs to a LCD if necessary). - Wait for 20 seconds. - Repeat the above procedures for 10 times. **Expected result:** Every effective piece of data received from the detection module has the corresponding reaction by both the LED and the microcontroller, without the case that the reaction of LED is not reflected on the microcontroller. For each time a data is received from the sender on the LED, the microcontroller should also receive the same data. There is no tolerance on a data transmission failure to the microcontroller. If there is ever a failure, check the connection between the Bluetooth module and the microcontroller on both software and hardware level.
Communication	1. Bluetooth transmitter and receiver should be able to communicate in a quick and reliable manner.	1. (a) - Connect the Bluetooth receiver to the Arduino

	(a) Reliable transmission with sufficient distance apart (< 25 feet). (b) Accurate transmission with no or tolerable data loss. 2. Incorporate the response time in the detection phase into total communication cost. (a) LEDs with distinguishable flashing frequencies can be selected by brain wirelessly. (b) Nearly instantaneous (< 2.5 second) response on a single selection from brain to display.	board and upload the program to set up serial connection. - Send sample data from a computer using putty or other serial monitors - Increase the distance between the computer (Bluetooth transmitter) and Bluetooth receiver from 1 foot to 30 feet **Expected result:** Data transmission is reliable (no miss or error rates) within 25 feet. 1. (b) - Increase transmission rate (from 1 character per second to 16 character per second) and sample data size, - Transmission distance set at 20 feet **Expected result:** Data transmission is accurate up to 16 characters per second with no data loss. 2. (a) - Incorporate the whole detection module (signal collector, amplifier and analyzer) before the transmitter end and make selections by brain **Expected result:** Regions on the screen can be successfully selected as intended. 2. (b) - Record real response time from brain to display for several flashing frequencies

		Expected result: For distinguishable frequencies, response time can be as short as 1 second and at most 2.5 seconds.
Power	1. Supply steady input voltage to the system. (a) Capable to provide steady 9V input voltage to the Arduino board. (b) Capable to provide steady 3.3V input voltage to the LCD and the Bluetooth module. 2. Considerable battery time: capable to last 4 hours for worst case scenario. (i.e. continuously operating mode)	1. (a) - Build a simple test circuit including the power supply of 9V and resistors of various values. - Control the resistor values to obtain current values, which can be measured correspondingly. - Calculate the resulted voltage values based on the resistance and corresponding current values. **Expected result:** For all different pairs of resistor values and current values, the calculated voltage values should fall in the range: 8.0V to 10.0V. 1. (b) - Build a simple test circuit including the power supply of 3.3V (output of the embedded regulator on the Arduino board) and resistors of various values. - Control the resistor values to obtain current values, which can be measured correspondingly. - Calculate the resulted voltage values based on the resistance and corresponding current values. **Expected result:** For all different pairs of resistor values and current values, the calculated voltage

		values should fall in the range: 3.2V to 3.4V. 2 - Connect the power supply to the circuit. - The device keeps running in the operating mode. **Expected result:** The power does not run out until 4 or more hours later.

Appendix C – Arduino-based Display Code

```
#include <UTFT.h>

// Declare which fonts we will be using
extern uint8_t SmallFont[];
extern uint8_t BigFont[];
extern uint16_t image[];
extern uint16_t album[];
extern uint16_t horse1[];
extern uint16_t horse2[];
extern uint16_t horse3[];

// Uncomment the next line for Arduino Mega
UTFT myGLCD(ITDB32S,38,39,40,41);   // Remember to change the model parameter
to suit your display module!

int val;
int ledpin = 13;
int curr_state = 3;

int screen = 1;  //1-Home 2-Settings 3-Picture 4-Music 5-Text 6-Video

void setup()
{

  Serial.begin(9600);
  pinMode(ledpin,OUTPUT);

// Setup the LCD
  myGLCD.InitLCD(0);
  myGLCD.setFont(BigFont);

  myGLCD.clrScr();

  myGLCD.setColor(255, 255, 255);
  myGLCD.print(" SETTINGS ", CENTER, 80);
  myGLCD.print("  PICTURE  ", CENTER, 120);
  myGLCD.print("PLAY MUSIC", CENTER, 160);
  myGLCD.print("   TEXT   ", CENTER, 200);
  myGLCD.print("   VIDEO   ", CENTER, 240);

  myGLCD.setColor(255, 0, 0);
  myGLCD.print("PLAY MUSIC", CENTER, 160);

}

void loop()
{
        if(screen == 1)
        {
                in_menu_screen();
        }
        else if(screen == 2)
        {
                in_setting_screen();
        }
        else if(screen == 3)
        {
                in_picture_screen();
        }
        else if(screen == 4)
        {
                in_music_screen();
        }
        else if(screen == 5)
        {
                in_text_screen();
        }
        else if(screen == 6)
        {
                in_video_screen();
        }

}

void in_menu_screen()
{
        val = Serial.read();
  if(val=='w')
  {
    //myGLCD.InitLCD(0);
    myGLCD.setFont(BigFont);

    // myGLCD.clrScr();

    myGLCD.setColor(255, 255, 255);
    myGLCD.print(" SETTINGS ", CENTER, 80);
    myGLCD.print("  PICTURE  ", CENTER, 120);
    myGLCD.print("PLAY MUSIC", CENTER, 160);
    myGLCD.print("   TEXT   ", CENTER, 200);
    myGLCD.print("   VIDEO   ", CENTER, 240);

    if(curr_state == 1)
    {
      curr_state = 5;
      myGLCD.setColor(255, 0, 0);
      myGLCD.print("   VIDEO   ", CENTER, 240);
    }
    else if(curr_state == 2)
    {
      curr_state = 1;
      myGLCD.setColor(255, 0, 0);
      myGLCD.print(" SETTINGS ", CENTER, 80);
    }
    else if(curr_state == 3)
    {
      curr_state = 2;
      myGLCD.setColor(255, 0, 0);
      myGLCD.print("  PICTURE  ", CENTER, 120);
    }
    else if(curr_state == 4)
    {
      curr_state = 3;
      myGLCD.setColor(255, 0, 0);
      myGLCD.print("PLAY MUSIC", CENTER, 160);
    }
    else if(curr_state == 5)
    {
      curr_state = 4;
      myGLCD.setColor(255, 0, 0);
      myGLCD.print("   TEXT   ", CENTER, 200);
    }
  }
  else if(val=='s')
  {
    //myGLCD.InitLCD(0);
    myGLCD.setFont(BigFont);

    // myGLCD.clrScr();

    myGLCD.setColor(255, 255, 255);
    myGLCD.print(" SETTINGS ", CENTER, 80);
    myGLCD.print("  PICTURE  ", CENTER, 120);
    myGLCD.print("PLAY MUSIC", CENTER, 160);
    myGLCD.print("   TEXT   ", CENTER, 200);
    myGLCD.print("   VIDEO   ", CENTER, 240);

    if(curr_state == 4)
    {
      curr_state = 5;
      myGLCD.setColor(255, 0, 0);
      myGLCD.print("   VIDEO   ", CENTER, 240);
    }
    else if(curr_state == 5)
    {
      curr_state = 1;
      myGLCD.setColor(255, 0, 0);
      myGLCD.print(" SETTINGS ", CENTER, 80);
    }
    else if(curr_state == 1)
    {
      curr_state = 2;
      myGLCD.setColor(255, 0, 0);
      myGLCD.print("  PICTURE  ", CENTER, 120);
    }
    else if(curr_state == 2)
    {
      curr_state = 3;
      myGLCD.setColor(255, 0, 0);
      myGLCD.print("PLAY MUSIC", CENTER, 160);
    }
    else if(curr_state == 3)
    {
      curr_state = 4;
      myGLCD.setColor(255, 0, 0);
      myGLCD.print("   TEXT   ", CENTER, 200);
```

```
   }
}
      else if(val=='d')
      {
              myGLCD.clrScr();

              myGLCD.setFont(BigFont);

  if(curr_state == 1)
              {
                      myGLCD.setColor(255, 255, 255);
                      myGLCD.print("LANGUAGE", CENTER, 80);
                      myGLCD.print("BRIGHTNESS", CENTER, 120);
                      myGLCD.print("VOLUME", CENTER, 160);
                      myGLCD.print("ABOUT", CENTER, 200);
                      myGLCD.print("RESET", CENTER, 240);

                      screen = 2;
              }
              else if(curr_state == 2)
              {
                      myGLCD.setColor(255, 255, 255);
                      myGLCD.print("PICTURE", CENTER, 80);
                      myGLCD.drawBitmap(58, 120, 64, 64, image, 2);

                      screen = 3;
              }
              else if(curr_state == 3)
              {
                      myGLCD.setColor(255, 255, 255);
                      myGLCD.print("NOW PLAYING", CENTER, 80);
                      myGLCD.drawBitmap(58, 120, 64, 64, album, 2);
                      myGLCD.print("by Adele", CENTER, 240);
                      myGLCD.setColor(255, 0, 0);
                      myGLCD.print("Skyfall", CENTER, 200);

                      screen = 4;
              }
              else if(curr_state == 4)
              {
                      myGLCD.setColor(255, 255, 255);
                      myGLCD.print("SAMPLE TEXT", CENTER, 40);
                      myGLCD.print("SAMPLE TEXT", CENTER, 80);
                      myGLCD.print("SAMPLE TEXT", CENTER, 120);
                      myGLCD.print("SAMPLE TEXT", CENTER, 160);
                      myGLCD.print("SAMPLE TEXT", CENTER, 200);
                      myGLCD.print("SAMPLE TEXT", CENTER, 240);
                      myGLCD.print("SAMPLE TEXT", CENTER, 280);

                      screen = 5;
              }
              else if(curr_state == 5)
              {
                      myGLCD.setColor(255, 255, 255);
                      myGLCD.print("VIDEO", CENTER, 80);

                      int i;
```

```
                        for(i=0;i<10;i++)
                        {
                                myGLCD.drawBitmap(58, 120, 64, 64, horse1, 2);
                                delay(50);
                                myGLCD.drawBitmap(58, 120, 64, 64, horse2, 2);
                                delay(50);
                                myGLCD.drawBitmap(58, 120, 64, 64, horse3, 2);
                                delay(50);
                        }
                        screen = 6;
                }
        }
}

void in_setting_screen()
{
        val = Serial.read();
  if(val=='x')
  {
                myGLCD.InitLCD(0);
                myGLCD.setFont(BigFont);

                myGLCD.clrScr();

                myGLCD.setColor(255, 255, 255);
                myGLCD.print(" SETTINGS ", CENTER, 80);
                myGLCD.print("  PICTURE  ", CENTER, 120);
                myGLCD.print("PLAY MUSIC", CENTER, 160);
                myGLCD.print("   TEXT   ", CENTER, 200);
                myGLCD.print("   VIDEO   ", CENTER, 240);

                myGLCD.setColor(255, 0, 0);
                myGLCD.print(" SETTINGS ", CENTER, 80);

                screen = 1;
        }
}

void in_picture_screen()
{
        val = Serial.read();
  if(val=='x')
  {
                myGLCD.InitLCD(0);
                myGLCD.setFont(BigFont);

                myGLCD.clrScr();

                myGLCD.setColor(255, 255, 255);
                myGLCD.print(" SETTINGS ", CENTER, 80);
                myGLCD.print("  PICTURE  ", CENTER, 120);
                myGLCD.print("PLAY MUSIC", CENTER, 160);
                myGLCD.print("   TEXT   ", CENTER, 200);
                myGLCD.print("   VIDEO   ", CENTER, 240);

                myGLCD.setColor(255, 0, 0);
                myGLCD.print("  PICTURE  ", CENTER, 120);
```

```
               screen = 1;
       }
}

void in_music_screen()
{
       val = Serial.read();
  if(val=='x')
  {
               myGLCD.InitLCD(0);
               myGLCD.setFont(BigFont);

               myGLCD.clrScr();

               myGLCD.setColor(255, 255, 255);
               myGLCD.print(" SETTINGS ", CENTER, 80);
               myGLCD.print("  PICTURE  ", CENTER, 120);
               myGLCD.print("PLAY MUSIC", CENTER, 160);
               myGLCD.print("   TEXT   ", CENTER, 200);
               myGLCD.print("   VIDEO   ", CENTER, 240);

               myGLCD.setColor(255, 0, 0);
               myGLCD.print("PLAY MUSIC", CENTER, 160);

               screen = 1;
       }
}

void in_text_screen()
{
       val = Serial.read();
  if(val=='x')
  {
               myGLCD.InitLCD(0);
               myGLCD.setFont(BigFont);

               myGLCD.clrScr();

               myGLCD.setColor(255, 255, 255);
               myGLCD.print(" SETTINGS ", CENTER, 80);
               myGLCD.print("  PICTURE  ", CENTER, 120);
               myGLCD.print("PLAY MUSIC", CENTER, 160);
               myGLCD.print("   TEXT   ", CENTER, 200);
               myGLCD.print("   VIDEO   ", CENTER, 240);

               myGLCD.setColor(255, 0, 0);
               myGLCD.print("   TEXT   ", CENTER, 200);

               screen = 1;
       }
}

void in_video_screen()
{
       val = Serial.read();
  if(val=='x')
  {
              myGLCD.InitLCD(0);
              myGLCD.setFont(BigFont);

              myGLCD.clrScr();

              myGLCD.setColor(255, 255, 255);
              myGLCD.print(" SETTINGS ", CENTER, 80);
              myGLCD.print("  PICTURE  ", CENTER, 120);
              myGLCD.print("PLAY MUSIC", CENTER, 160);
              myGLCD.print("   TEXT   ", CENTER, 200);
              myGLCD.print("   VIDEO   ", CENTER, 240);

              myGLCD.setColor(255, 0, 0);
              myGLCD.print("   VIDEO   ", CENTER, 240);

              screen = 1;
        }
}
```

Appendix D – Bluetooth MATLAB Code

```
instrhwinfo('Bluetooth');

b = Bluetooth('S1', 1);

fopen(b);

up = 'w';

down = 's';

fwrite(b, up, 'uchar');

fwrite(b, down, 'uchar');

%fclose(b);

%clear(b);
```