PORTABLE BCI STIMULATOR

By

Bonnie Chen

Randy Lefkowitz

Siyuan Wu

Final Report for ECE 445, Senior Design, Spring 2013

TA: Ryan May

01 May 2013

Project No. 17

Abstract

The goal of this project is to design a stimulator for a brain computer interface (BCI) that is portable in design. The design is to have 10 flashing LEDs, 5 for each eye, controlled via Bluetooth and mounted on a pair of goggles. The visual feedback will be recorded by signals from the user's brain in an Electroencephalography (EEG) system, and proper data acquisition will be handled by the BCI system.

Contents

1. Introduction

Brain Computer Interfaces (BCI) based on Electroencephalography (EEG) allow for the monitoring and analysis of ongoing brain activity in real time. The signals measured by this technology can be used to control user interfaces without the requirement of the human motor system. This technology can benefit those with paralysis and other severe disabilities.

1.1 Purpose

As of now, the majority of BCI systems are currently large and immobile, and therefore impractical for use in everyday life outside of a lab. There are several components to a BCI system such as data acquisition, a classification system, as well as stimulation, all of which must be made portable to create a portable BCI. To address this problem, we set out to create a portable stimulator that can communicate wirelessly with the parts that are monitoring brain activity. The stimulator will consist of LEDs flickering at predefined frequencies, with attention to luminescence (we don't want our LEDs to blind the user so it must be at the right intensity for each user) as well as controls to adjust the frequencies while maintaining accurate timing. Our design goals are to make the stimulation for the BCI and EEG portable and integrated wirelessly so that users are not confined to just a lab setting and that the system could be tested and used in different environments.

1.2 Functions

The circuit design is a glasses-mounted stimulator, which includes proper mounting for 5 LEDs around the right eye, creating convenience to the user and adding to the portability of the EEG and BCI system.

The stimulator takes inputs from a PC operator, and receives the LED blinking frequencies and intensities wirelessly through Bluetooth. Wireless communication with the PC enables the stimulator to be operable from a distance of around 30 ft.

Flashing lights can induce pain to the user's eyes. The stimulator prevents this by giving the user controls to vary the frequency and brightness of each LED. The frequencies of each LED controlled by the user can be variable from 1 to 99 Hz, which includes the useable operating limits of EEG detection of about 5-15 Hz. The intensity of each LED has 3 settings, all within a safe operating range, which gives the user sufficient control of light intensity coming from each LED.

1.3 Blocks

The stimulator can be broken up into multiple modules.

1.3.1 PC

The PC is where a user can change the LED frequencies and intensities using the serial monitor in the Arduino IDE. It also includes a built-in Bluetooth transmitter, which is used to send the data to the stimulator.

1.3.2 Wireless Receiver

The wireless receiver uses Bluetooth 2.0 protocol to receive the data from the PC and delivers it to the microcontroller.

1.3.3 Microcontroller

The microcontroller module is the brain of the project. We used an Arduino UNO to do all the computing and to control the flashing of the LEDs. The Arduino then outputs to the LED driver, which controls intensity.

1.3.4 LED Array

The LED array consists of 5 LEDs and a TI TLC-5940 LED driver. The LED driver is used to control the intensity of each LED using pulse width modification (PWM). The LEDs are flashed on and off by the microcontroller, and are mounted on a pair of glasses. The locations are adjustable, thanks to magnetic mounts.

1.3.5 Power

The power supply for the stimulator must be portable, so we used a 7.4V Lithium Ion rechargable battery. It powers the Arduino, which then supplies power to the Bluetooth receiver, LED driver, and LEDs.

2. Design

The implementation of the stimulator in the overall BCI system is shown in Figure 1. The brain signals from the user are being monitored by the EEG and is amplified and sent to the computer for Data Acquisition and Signal Processing. After this the signal is sent to the User Interface, which takes inputs for the controls to the stimulator. The control inputs from the user are transmitted to the stimulator from the computer and the resulting changes are fed back to the user being monitored with the EEG.

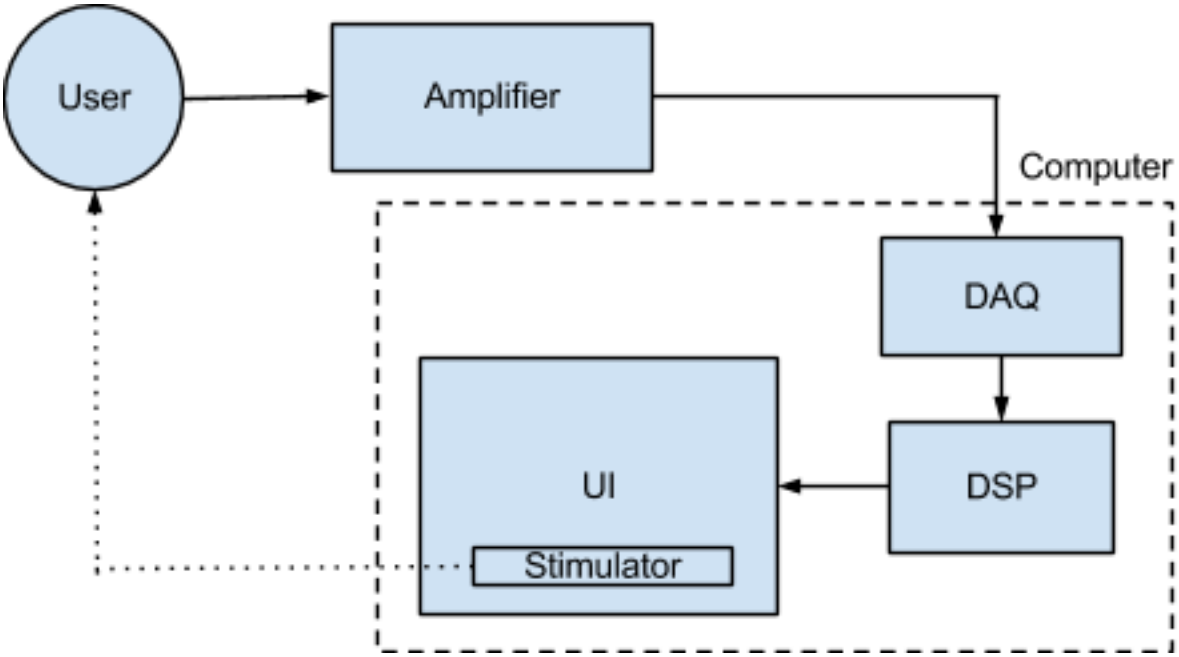


Figure 1: Full System Overview

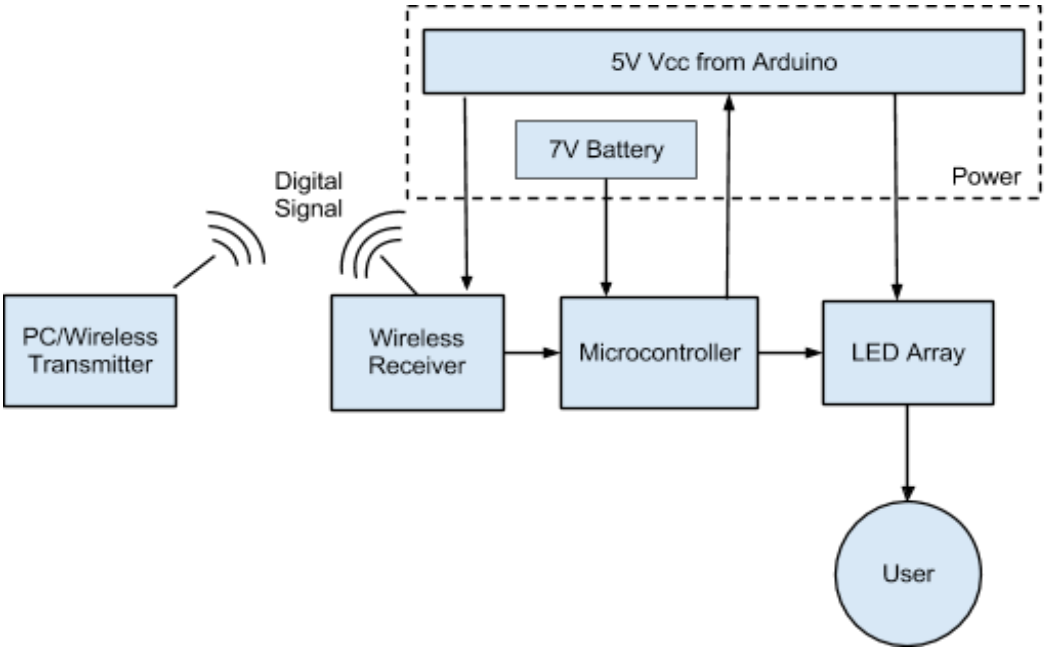


Figure 2: Top-Level Stimulator Schematic

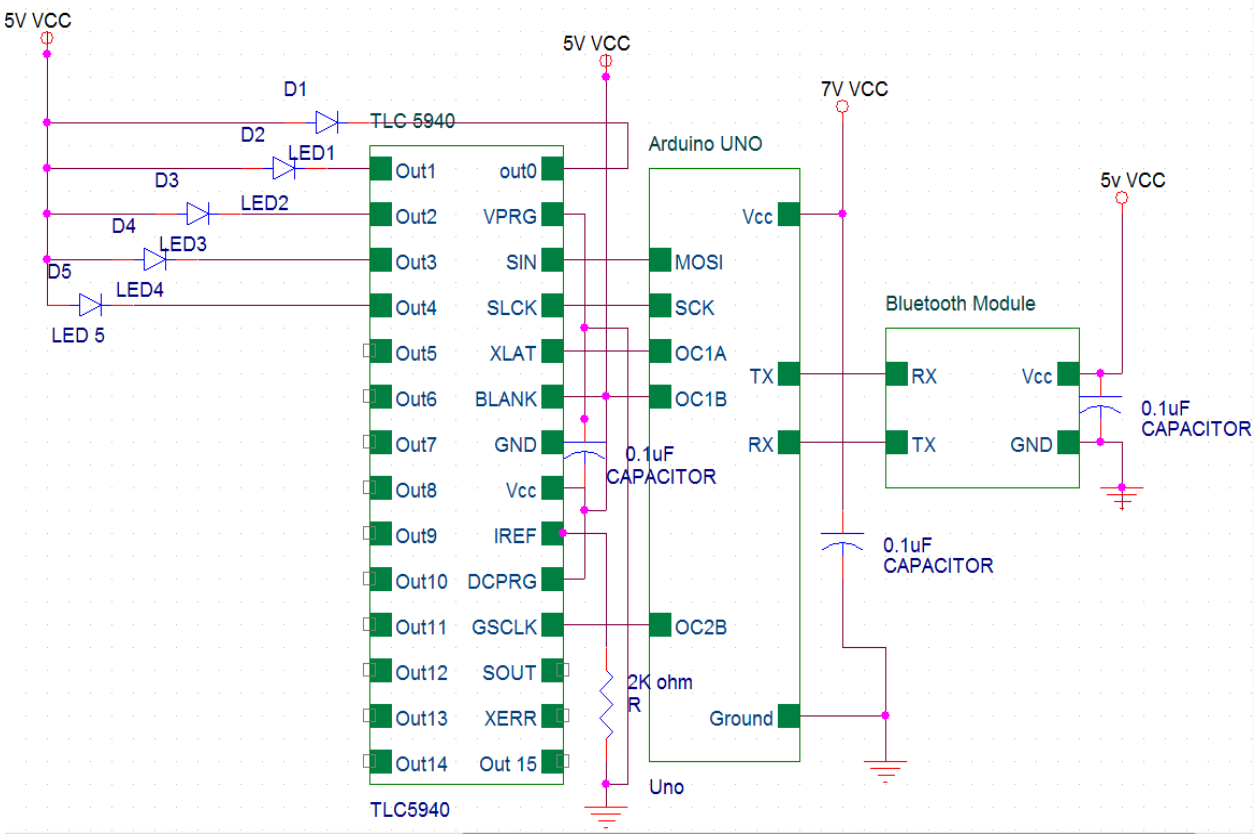


Figure 3: Stimulator Circuit Diagram

2.1 PC

The PC runs the Arduino IDE, and communicates with the Arduino via Bluetooth. When the serial monitor is opened, the user is shown the default LED frequencies, and is prompted with an option to change them. The user enters a simple command (such as LED0 5 to set LED 0 to 5Hz), and the command is verified, then sent to the microcontroller. Besides frequencies, this program also allows the user to change the intensity values of the LEDs between the 3 levels that we decided were safe and effective (INT 2). The PC also has a built-in wireless transmitter. Once paired with the receiver, the PC is able to communicate with the portable stimulator using Bluetooth 2.0 protocol.

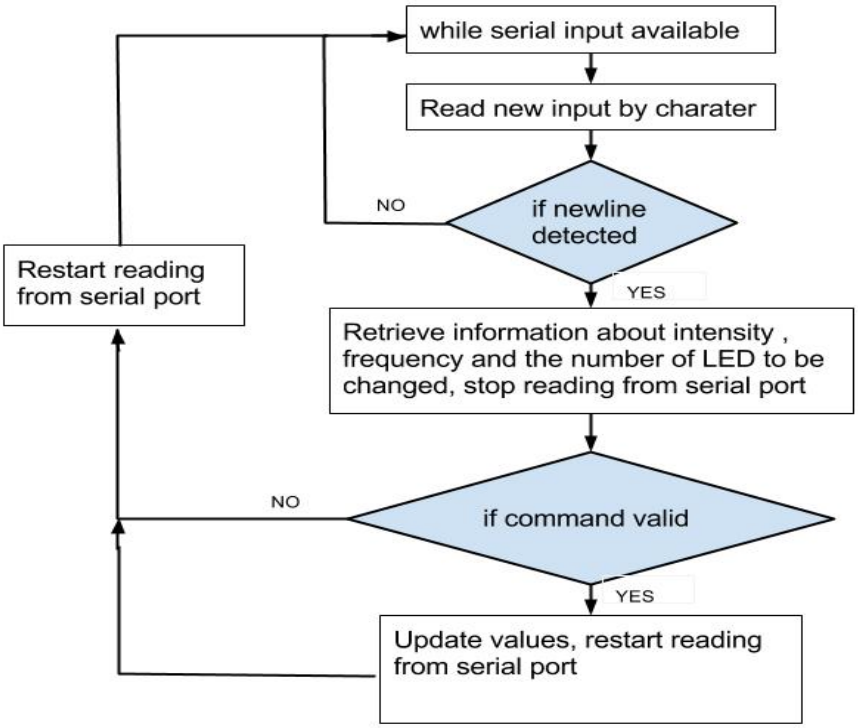


Figure 4: Input Algorithm Flow Chart

2.2 Wireless Receiver

The EGBT-046S wireless receiver is a UART serial TTL Bluetooth module. It has 4 channels: an input of 3.3V -5V, Ground, TX, RX. Both TX and RX have 3.3V outputs. Communication is set up by connecting the RX channel of the receiver to the TX channel of the Arduino and the TX of receiver to the RX of the Arduino. After the pairing is done through programming on the PC, data is sent through command lines on the pc to the Arduino. The wireless receiver will receive and interpret the multiple packets of serial data sent from the PC as if the two were connected by a wire, and translate them into a signal for the microcontroller to work with in outputting the correct frequency.

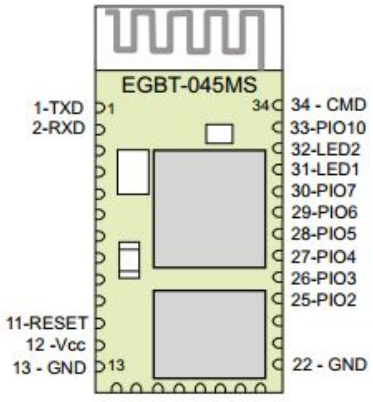


Figure 5: Bluetooth Receiver [3]

2.3 Microcontroller

The microcontroller used for the stimulator is the Arduino UNO R3. The controller input is serial data from the wireless receiver, and it outputs the frequencies and intensities chosen and set by the user to the LED driver through the PWM and Digital Output pins. It is powered by the 7.4V battery. After storing the frequencies and intensities as variables, the algorithm used first calculates the program's runtime, then it determines if it is time for each LED to toggle. It then sets the LED values using the TLC5940 library's tlc.set(channel,value) command, where channel is the LED number, and value is either off or on. After the new LED value is set the values are latched into the LED driver using tlc.update().

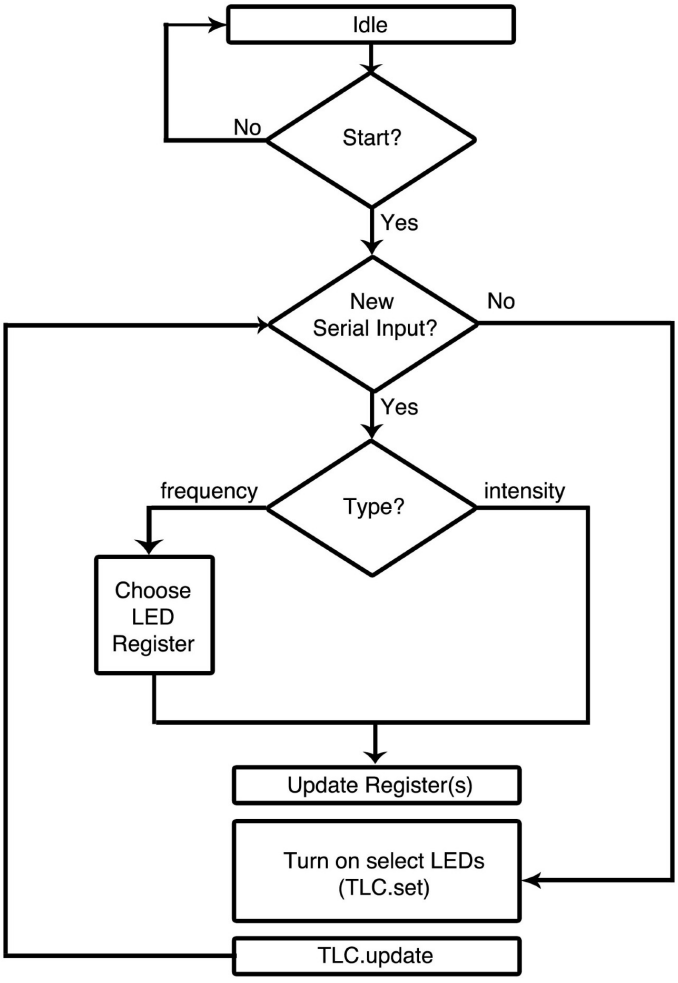


Figure 6: Timing Algorithm Flow Chart

2.4 LED Array

The LED array consists of an LED driver (TI TLC-5940), which sends the proper PWM signal outputs to 5 LEDs around an eye. The TLC-5940 is powered by 5V from the Arduino, and outputs to

each channel a voltage of 2-3.2V, which is the necessary voltage to light up the LEDs. The LED driver receives serial data from the microcontroller module and feeds the data to a connected array of 5 Lillypad LEDs, which are mounted onto a wearable glasses frame using magnets. The LEDs have a 2-3.2 voltage drop across each and a maximum forward current of 25mA. The LED driver has an IREF pin that is used to set a constant current on each channel's output. We placed a 2kohm resistor between the IREF pin and ground, limiting the output current of each channel to 20mA. The LEDs output the final blinking frequency to the user.

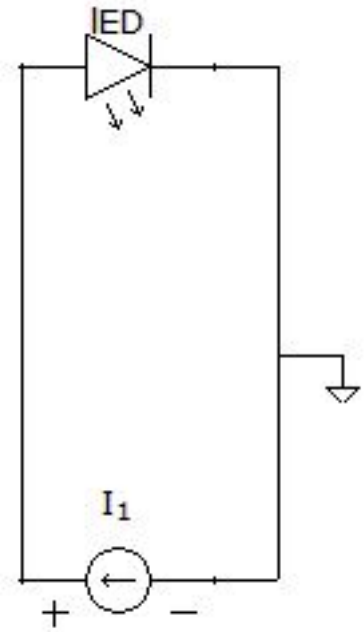


Figure 7: LED Equivalent Circuit

2.5 Power

All portable components in the stimulator require power from this module. The microcontroller is powered by a 7.4V 1250mAh lithium ion battery. Because of the voltage regulators in the Arduino Uno, the other components are connected through the 5V output pin on the Arduino. The TI TLC-5940 LED driver and the Bluetooth module have maximum input voltages of 5V, which is what we used to drive the 5 LEDs.

3. Design Verification

3.1 Bluetooth Receiver

We first added the Bluetooth device to the computer. The next step was to open a serial port that sends data to the Arduino. To test if the Bluetooth was working properly, we ran a test program on the Arduino that turns an LED on when we send a 1 from the computer wirelessly, and off when we send a 0.
No data was collected for the testing of this part.

3.2 Microcontroller

First we had to test if the command is properly received and the microcontroller can retrieve the information we need. We did that by typing in LED A ab or INT 1/2/3 and the Arduino printed the LED number A and frequency value ab back to the terminal to make sure the right value is retrieved before assigning them.

The next step is to calculate the runtime of each LED with the frequency given. With the frequency given, we calculated the time for each LED to be on and off (assuming 50% duty cycle), and when it should be on, we set the intensity the value we want. (the user can select 3 levels of intensity)

We verified the frequency by looking at its value in the IDE. We also looked at each LED's value over 1 second on the IDE, and make sure that they are on and off the correct number of times based on their frequencies. (Ex: LED0 has a value of 5 Hz and intensity=50%. LED0 should switch between 0 and 2048 5 times over 1 second).

3.3 LED Driver

The first thing to make sure is the LED driver must provide proper power dissipation to properly handle and control every LED. We monitored the current output by the TLC-5940 as well as the voltage across the LEDs to make sure that the current does not exceed 25mA which is the rating we are basing the R_{iref} value of $2k\Omega$ off of and that the voltage value is no greater than 3.2 V which is the maximum voltage to power the LEDs and no less than 2V which is the minimum voltage to power the LED.

The current draw from a single LED when we tested is 19.6 mA and when it is tested along with 4 four other LEDs the average of the current through is 19.45 mA. None of the LED has current more than 25 mA. And as expected, the frequency and intensity level, which is set by the PWM duty cycle, does not have a major effect on the value of the current drawn.

The next step is to test the actual frequency of the LED with an oscilloscope. First we chose to test low frequencies, 2 HZ, 4 HZ, 6 HZ, 8 HZ, 10 HZ :

Target Frequency (Hz)	Achieved Frequency (Hz)
2	1.996-2.004
4	3.98-4.02
6	5.5-6.5
8	7.9-9.1
10	10

Table 1: Low Frequency Bandwidth

Target Frequency	Achieved Frequency
20	20
40	38-42
60	59-63
80	77-83
99	100

Table 2: High Frequency Bandwidth

3.4 LED Array

We wanted to make sure each LED operates with an intensity that will not cause discomfort to the user. Verification for the LED intensities consisted of finding safe operating limits and environments in which the stimulator should be used. This was done first qualitatively by us looking at the LEDS, but to receive more quantitative data, there needed to be a proper recording method of finding the luminance values for the LEDs which requires a proper photodetector such as a photodiode or power meter to measure different intensity levels of the LEDs

3.5 Power Supply

Sufficient power must be provided for every component and must be efficient in delivering power that will allow maximum usage time of the stimulator for the user.

3.5.1 Bluetooth receiver

The power for the Bluetooth comes from the Arduino 3.3V output pin. We tested for a stable open circuit voltage output from the Arduino 3.3V pin using a multimeter connected from the output pin to ground and make sure the voltage does not vary over 5% to ensure proper power to the Bluetooth module, anymore than that could affect the behavior of the wireless transmission. The actual voltage on the receiver is 3.30V.

3.5.2 Arduino

The Arduino Uno was powered by a 7.4V LiPo battery, with which we measured using a multimeter connected from the positive battery terminal to ground to ensure that no less than 5% of that value is coming out. The actual voltage we read was 7.37V.

3.5.3 LED Driver

The LED driver has an input voltage of 5V from the Arduino, with which we measured with the multimeter in the same fashion as to how we measured the power for the Bluetooth module. The actual voltage on it was ~4.9v.

3.5.4 LEDs

We tested the currents drawn by the LEDs with a multimeter to make sure that no less than 0.5mA and no more than 25 mA of the required forward current was being measured. Any less would probably not be enough to power the LED and will be visible if the LED was not turning on, in which we would then step backwards in the power testing and analyze the driver to see if it was outputting the correct constant output current. The test data were included previously in the LED driver part.

3.6 PC

Make sure the computer can find the Bluetooth device and be able to add the device.

3.7 Overall testing:

3.7.1 EEG Classification Testing

The overall requirement of our project is the correct classification of LED frequency on an EEG system.

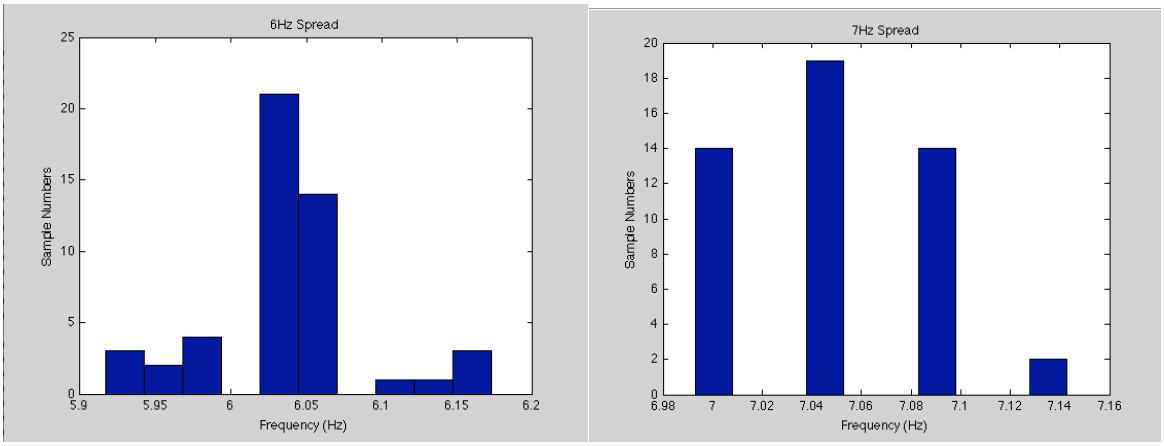
In the demo, all our frequencies were correctly classified. However different frequencies and intensities would have affected the results and response time and there are too much to consider and most of them are out of our control. So what we can do is analyze the frequency data with MATLAB for further testing.

3.7.2 Frequency analysis with MATLAB

To further analyze our results, we tested the frequency data with oscilloscope and record the frequency data of about 50 seconds. Then we use Matlab to do the analysis for the data. Here are the frequencies we tested, along with the mean and variance of the data:

1 Hz: $\mu = 1.0912$, $\sigma 2 = 0.41$
6 Hz: $\mu = 6.0694$, $\sigma 2 = 0.36$
7 Hz: $\mu = 7.1238$, $\sigma 2 = 1.37$
8 Hz: $\mu = 8.1937$, $\sigma 2 = 2.01$
9 Hz: $\mu = 9.2745$, $\sigma 2 = 5.22$
10 Hz: $\mu = 10.495$, $\sigma 2 = 10.77$

As we can see in Figure 8. the frequencies we have of our LEDs have a stable result when the frequency is low, but as the testing frequency gets higher, the results were more off.



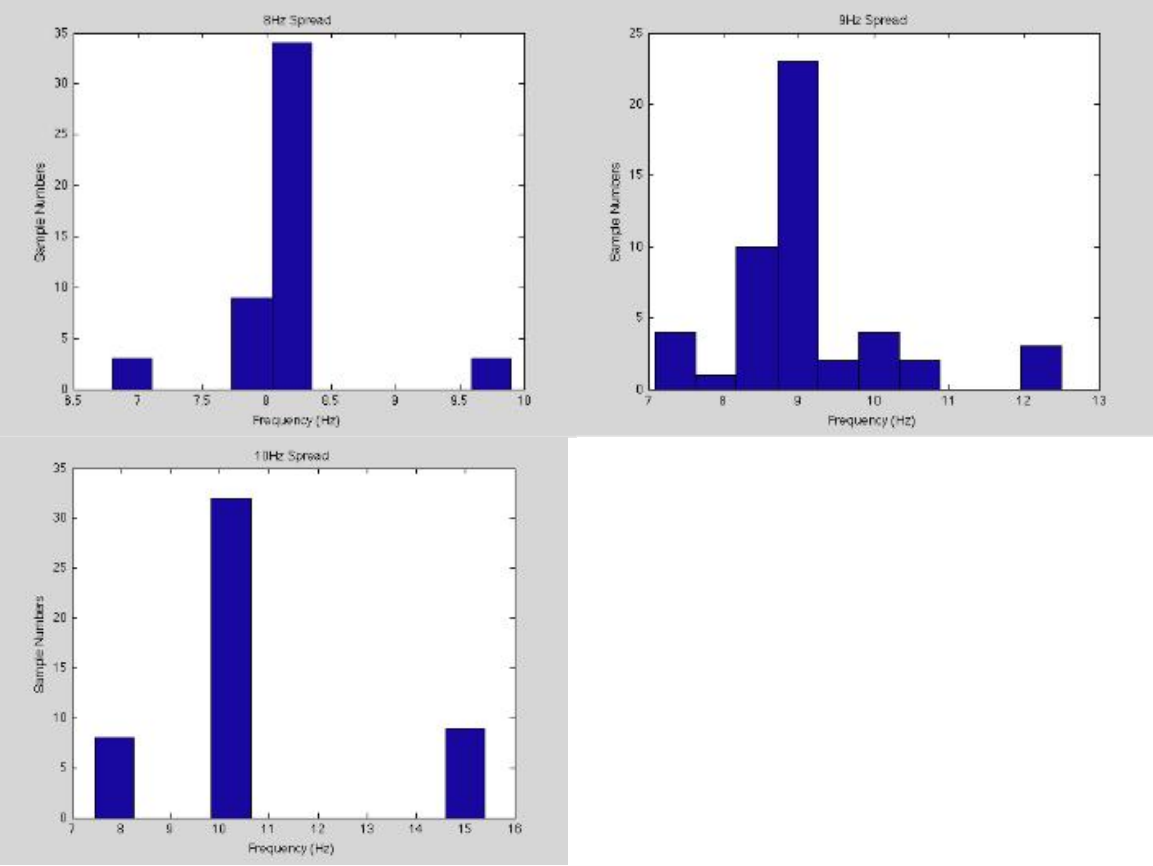


Figure 8: Frequency Bandwidth Samples

At lower frequencies, though the mean and variance looks better than higher frequencies, there are actually quite a lot of samples that are off from the frequencies we desired, but since they are not off by much, the result still provided good feedback from the EEG; as the frequency gets higher, we can see there are more samples at the frequencies we want but there are very few samples that are off by the desired frequency quite a lot. Though these samples made the mean and variance of the high frequency farther from the targeted frequency value, the actually result of EEG classification is still the same because the samples are only off for a very short period of time (~1ms) and for most of the time they were operating at the desired frequency.

3.7.3 Power Budget

Component	Imax (mA)	Voltage
Microcontroller	40 * 2 output pins	7-12V (ideal)
LEDs	20 * (10 LEDs)	3.2V (green, white)
Bluetooth Module	40	3.3V
LED Driver	120	5V
Total	440	

Estimated Usage time = 1250 [mAh] / 440 [mA] ≅ 3 hours of charge

3.7.4 Possible Improvements:

We had some noise when we tested the frequencies in Beckman. It's possible that the surrounding signals from other machines were affecting the results (there's a large magnet from the MRI machine in the basement of the building). A proper testing environment for our device would be outdoors since there would be less noise interfering with our device. Also the design for the stimulator is to be portable and used in various environments, so testing this aspect of the device would give us more qualitative and quantitative results for the frequency readings.

4. Costs

4.1 Cost Analysis

Part	Retail Cost ($)	Quantity	Actual Cost ($)
PCB	30.00	1	30.00
Lilypad Micro LEDs	3.95	2 (packs of 5)	7.90
Arduino Uno R3	39.00	1	39.00
TLC-5940	6.00	1	6.00
EGBT-0463 Bluetooth SMD Module	7.00	1	7.00
Venom 7.4V LiPo Battery	15.89	1	15.89
Venom LiPo Battery Charger	26.79	1	26.79
Barrel Jack Battery Connector	2.95	1	2.95
Glasses Mounting Frame	7.00	2	14.00
Total	**138.58**	**11**	**149.53**

Table 3: Parts Cost

Name	Hourly Rate	Total Hours	Total * 2.5
Siyuan Wu	$35.00	150	$13.125
Bonnie Chen	$35.00	150	$13.125
Randy Lefkowitz	$35.00	150	$13.125
Total	$105.00	450	$39,375

Table 4: Labor Costs

Component	*Cost*
Parts	$149.53
Labor	$39,375.00
Grand Total	**$39,524.53**

Table 5: Total Cost

5. Conclusion

5.1 Accomplishments

Our stimulator was able to achieve the proper frequency readings while interacting the the EEG system and the response time was within a range of 3-10 seconds for the user to match the frequencies running on the classifier program for the BCI. Even at the lowest light intensity settings of around 10/4096 of the PWM, the stimulator was able to provide a strong enough response from the user to be read. While the response time for lower intensities took around double the time to achieve the matching frequency signal as opposed to a stronger PWM value of 20/4096, the tradeoff for better user comfort preceded the response time.

5.2 Uncertainties

The testing results for higher frequencies (30-99 Hz) coming from the LED driver gave us greater variance and spread compared to the lower frequency range that is usually tested with the EEG (5-10 Hz).

We were also unable to attain the proper luminance measuring devices such as a power meter to get quantitative measurements for the intensity of each LED. While the demo could be done by testing on our own group members, we will need to find the proper luminance values that follow the ANSI Standards on Luminescence for safe operating limits on near-eye LEDs in order to operate the stimulator on other users.

5.3 Ethical Considerations

One safety concern is the use of the lithium-ion polymer battery. The worst-case scenario is the case in which the lithium battery explodes/expands because of an accidental short circuit in the PCB design which can cause the circuit to overheat. To prevent unwanted battery interactions with the face, we designed the mounting frame such that the battery will be placed in farthest away from the face by routing longer wires from each LED to the PCB which can be mounted or held by somewhere else on the user.

Also because the final output of our stimulator system is into the user's eyes, we must preserve the comfort and visual perception of the user, meaning that the LED array accounts for not only precise frequencies and duty cycle output, but also safe operating limits for various users such that no discomfort or harm will be done onto the user's vision through the use of the blinking lights. This required us to find proper testing and operating limits that are safe for long periods of exposure to blinking lights to the human eye. [6]

5.4 Future Work

While the project accomplished everything that was necessary to achieve the proper frequency readings from the EEG, there are still several areas for improvement that can help make the stimulator ready for use in industry and to be tested on a greater sample of users for research.

- More testing can be done with the EEG to determine the ideal threshold for response time, classification and stability of the system. This would be done through testing a greater range of frequencies and determining the thresholds of accuracy that is typically used in the Data Acquisition and processing of the EEG readings.

- Access to proper luminance testing devices would help us to secure the quantitative safety limits of near-eye LED operation. We would have liked to gather measurements with the proper candela values that are safe for people who are observing fast and bright blinking lights to avoid the possible occurrence of a seizure if the user is sensitive to certain pulses of light.

- The design of the mounting frame could be improved to create a more aesthetic and easily adjustable form to the stimulator, especially for the positioning of the LEDs on the frame. We would love to run the mounting frame through with someone knowledgeable in Industrial or Mechanical Engineering Design to improve this aspect of the stimulator.

6. References

[1] TLC-5940 Data Sheet. Texas Instruments
<http://www.ti.com/lit/ds/symlink/tlc5940.pdf>

[2] Arduino Uno R3 Schematic. SparkFun Electronics
<http://arduino.cc/en/uploads/Main/Arduino_Uno_Rev3-schematic.pdf>

[3] EGBT-045MS Bluetooth Module. Rasmicro
<http://www.rasmicro.com/Bluetooth/EGBT-045MS-046S%20Bluetooth%20Module%20Manual%20rev%201r0.pdf>

[4] Luminous Efficacy Tables. Georgia State University
<http://hyperphysics.phy-astr.gsu.edu/hbase/vision/efficacy.html>

[5] Photodiode Tutorial. Thorlabs
<http://www.thorlabs.com/newgrouppage9.cfm?objectgroup_id=285>

[6] Guidance for the reduction of photosensitive epileptic seizures cause by television. ITU
<http://www.itu.int/dms_pubrec/itu-r/rec/bt/R-REC-BT.1702-0-200502-I!!PDF-E.pdf>

[7] TLC5940 Arduino Library. Google Code
<http://code.google.com/p/tlc5940arduino/>

Appendix ARequirement and Verification Table

PC and Bluetooth Transmitter

Requirements	Verification
1.) Wireless communications must be configured through the PC to ensure proper transmission of the data	1.) For pairing the device, we find the Bluetooth device in control panel and add the Bluetooth device to PC. If it shows successfully added, it is connected to the PC.
2.) PC must be able to take in channels (LEDs or intensity) and values (frequencies or intensity values) from users, verify that they are valid, and store them.	2.) We will test that the input is being taken from the user, validated, and stored by inputting values we know are invalid and observing the expected error prompt, as well as giving valid values that we will then print back to the screen.
3.) The computer must translate the stored values into serial data.	3.) When a value is changed, the code will change the value of our 11 bit output variable. It will make 1 start bit, convert the channel into an address (next 3 bits), and the value of that channel into the last 7 bits. We will then print the new computer output to the screen and verify it is correct.

BlueTooth Receiver

Requirements	Verification
1.) Transmitter from the PC must be supported by the Bluetooth receiver	1.) Verify the computer has a Bluetooth transmitter that is either Bluetooth version 2.1+ or 2.0, 1.2, 1.1. Data will be sent to the Bluetooth receiver from the PC through PUTTY. Choose serial as connection type. Pick the right COM connection port and make the connection.
2.) Test if the Bluetooth receiver has received the data sent from the putty	2.) To test this, we will run program on the Arduino that turns an LED on when we send a 1 from the PC, and off when we send a 0.

3.) TTL Bluetooth Receiver must be able to receive data transmitted at a frequency that will not interfere with the operating frequencies for the LEDs	3.) After the receiver-end code on the Arduino is done, we will test and see if sending frequency signals that are higher than the upper bound operating frequency (100Hz +) to ensure that the transmission frequency of Bluetooth of ~2.4GHz will not affect the overall signal in functioning properly, monitoring the received data with an oscilloscope.

Microcontroller

Requirements	Verification
1.) After the testing for Bluetooth module is done, we will now test if the microcontroller has receive the correct data from Bluetooth module.	1.) To test this, since there is not much display to show what was received, we will test this part by viewing the data on the receiving pin using the Arduino IDE. For example, now we want to set LED0 to frequency 2HZ. The decoded data seen on the pin should be 1 000 0000010 start LED Value
2.) We will test that the serial data is properly decoded, and the correct values are stored in our variables.	2.) We will input a known 11 bit value to the decode function, and we will print out the values of our 5 LED and 1 intensity registers to verify that the correct value has been stored in the correct location.
3.) A counter will operate at Arduino clock frequency. The code will keep track of which LED should be on or off at each clock pulse.	3.) We will verify the counters speed by looking at its value in the IDE. We will also look at each LED's value over 1 second on the IDE, and make sure that they are on and off the correct number of times based on their frequencies. (Ex: LED0 has a value of 5 Hz and intensity=50%. LED0 should switch between 0 and 2048 5 times over 1 second).
4.) The data then needs to be output in a way that the TLC5940 can interpret (serial). We will be using the Arduino's tlc library (tlc.set and tlc.update) to send the data.	4.) We will verify that each time an LED value changes, tlc.set is called, and tlc.update is called at the correct intervals.

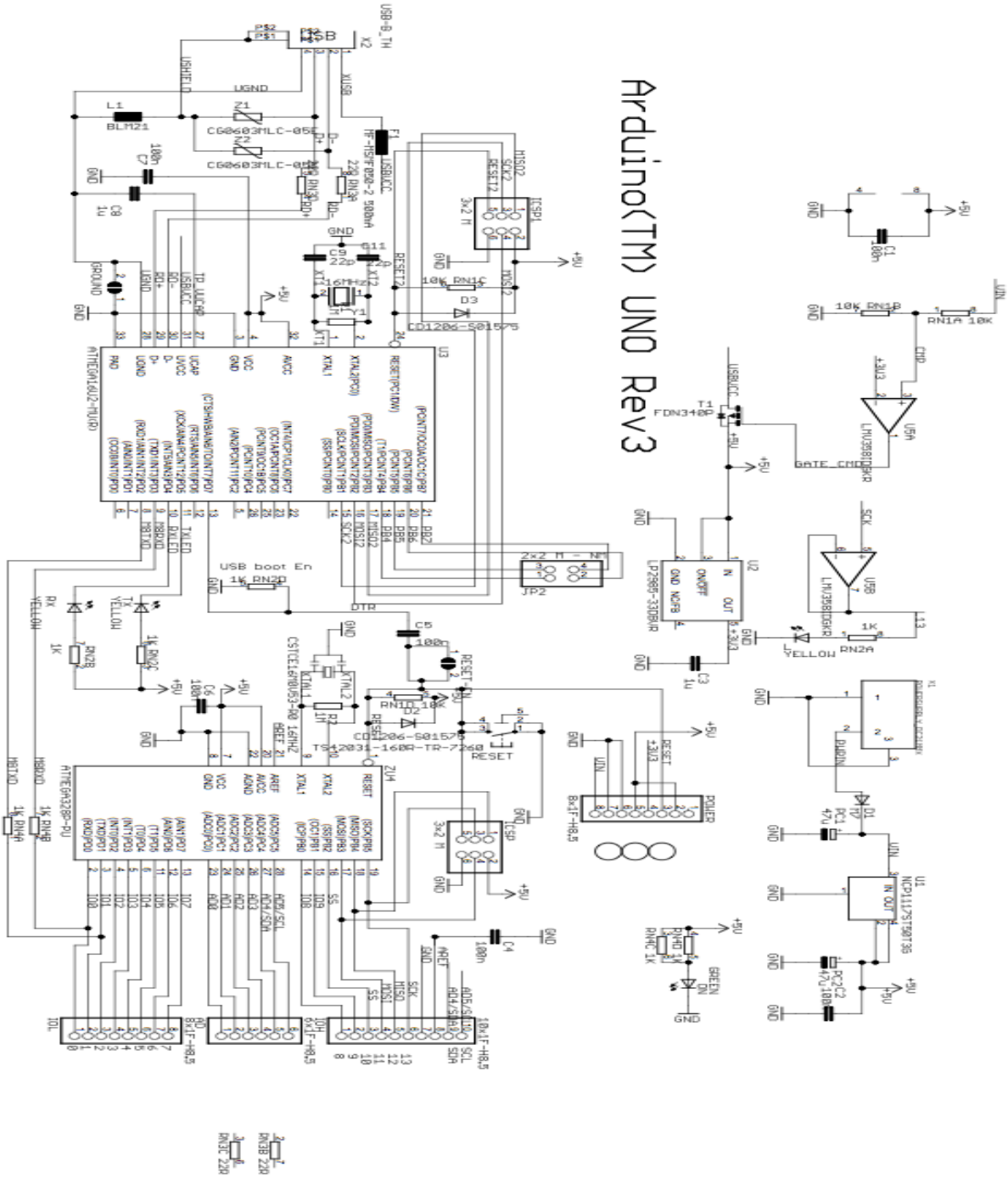
LED Array

Requirement	Verification
1.) The serial data sent to the LED driver will output the correct PWM signals to the desired output pins (10 total)	1.) We will set our GSCLK to a predetermined frequency, then change the value of GS register 1, and by monitoring the output of channel 1 on an oscilloscope, we will verify that the duty cycle being seen is the correct fraction of the 4096 count grayscale cycle (2048=50% of GSCLK, etc).
2.) LED driver must provide proper power dissipation to properly handle and control every LED	2.) Monitor the current output by the TLC-5940 as well as the voltage across the LEDs. Make sure that the current does not exceed 25mA which is the rating we are basing the R_{iref} value of $2k\Omega$ off of and that the voltage value is no greater than 3.2 V which is the maximum voltage to power the LEDs and no less than 2V which is the minimum voltage to power the LED
3.) Each LED must operate with an intensity that will not cause discomfort to the user	3.) Verification for the LED intensities will consist of finding safe operating limits and environments in which the stimulator can be used. This will be done first qualitatively by us looking at the LEDS, but to insure that this is achieved, there must be a proper recording method of finding the luminance values for the LEDs which will require a photodiode to measure different intensities. [4] a. The LED intensity will be measured with a photodetector and amplifier (optional). The photodiode (~4mm in diameter, the average size of the pupil) generates a current to give us the Power present on the photodiode at distance ~50mm from the photodetector. This is equivalent to the distance we are mounting the LEDs away from the user's eyes. We can then find the number of cds the LED is giving off at that distance. [5] b. If the current signal generated by the photodiode is too small, add an amplifier to measure the current. c. Test and measure several light intensities by changing the PWM value for each LED channel, record and measure which light colors and levels of intensity are the most comfortable to the user.

Power Supply

Requirement	Verification
Sufficient power must be provided for every portable component 1.) Bluetooth	1.) The power for the Bluetooth will be coming from the Arduino 5V output pin. We will need to test for a stable open circuit voltage output from the Arduino 5V pin using a multimeter connected from the output pin to ground and make sure the voltage does not vary over 5% to ensure proper power to the Bluetooth module, anymore than that can affect the behavior of the wireless transmission.
2.)Microcontroller	2.) The Arduino Uno will be powered by 7.4V battery, with which we will measure using a multimeter connected from the positive battery terminal to ground to ensure that no less than 5% of that value is coming out.
3.) LED Driver	3.) The LED driver will have an input voltage of 5V from the Arduino, with which we will measure with the multimeter in the same fashion as to how we measured the power for the Bluetooth module.
4.) LEDs	4.) We will test the currents drawn by the LEDs with a multimeter and make sure that no less than 0.5mA of the required forward current is being measured. Any less will probably not be enough to power the LED and will be noticed if it is not turning on, in which we will need to step backwards in the power testing and look at the driver and see if it is outputting the correct constant output current which is 120ma.
5.) Power supply must be efficient in delivering power that will allow maximum usage time of the stimulator for the user.	5.) Find the current draw at multiple loads using a multimeter, which should fall within 5% of the specified amps given by the data sheets, compare power usage with the given values for the current and voltages to the rated mAhs on the batteries to determine usage time for the stimulator (see section 3.5.3)

Appendix B Arduino UNO Datasheet [2]



Appendix C Code

```
#include <Tlc5940.h>
#include <tlc_animations.h>
#include <tlc_config.h>
#include <tlc_fades.h>
#include <tlc_progmem_utils.h>
#include <tlc_servos.h>
#include <tlc_shifts.h>

const int ledcount = 5;  //number of LEDs to control

//  SET INITIAL VARIABLES HERE
int intensity = 10;  //intensity value (probably 0-4096)
int freq[ledcount] = {20,6,7,8,9};  //set the LED frequencies here. Lower numbers give the best accuracy

int ledState[ledcount];  //each LED can be 0 or intensity
//int freq = 5;  //used for initial tests
//unsigned long interval = 1000000/freq;  //LED period in microseconds (used for testing)
unsigned long currentmicros = 0;  //start the counter at 0
unsigned long previousmicros[ledcount];  //array holding last time each LED changed
unsigned long interval[ledcount];  //how long to stay on/off to achieve the desired frequency
String name = "";
String st2 = "LED";
String st3 = "INT";

boolean nameEntered = false;
```

```
int lednumber;
int frequency;
char buff;
char buffa;
int j;


void setup()
{
Tlc.init();
Serial.begin(9600);
  Serial.println("Please enter command.");
for(int i = 0; i < ledcount; i++)
{
  interval[i] = 500000/freq[i];  //set all of the intervals
  previousmicros[i] = 0;        //initialize counters
  //Serial.println(freq[i]);      //printing frequencies and intervals just to make sure
  //Serial.println(interval[i]);
}
}

void timer()
{

  for(int i = 0; i < ledcount; i++)  //for each LED
  {
    if(currentmicros - previousmicros[i] >= interval[i])   //if it has been enough time since the last flip
    {
    // save the last time you blinked the LED
```

```
      previousmicros[i] = currentmicros;

    // if the LED is off turn it on and vice-versa:
      if (ledState[i] == 0)
        ledState[i] = intensity;
      else
        ledState[i] = 0;
     // Serial.println(freq[i]);  //print the frequency value each time it changes (just for debugging)
      Tlc.set(i,ledState[i]);  //Loads TLC register
      Tlc.update();            //latches data into LED driver
    }
  }
}

void loop()
{

while (Serial.available()) {
  char readChar = (char)Serial.read();

  // If the next character is a linefeed (enter)
  // the name is complete. Exit the while() loop.
  if (readChar == '\n') {
    nameEntered = true;
    continue;
  }
  // If the character wasn't enter, add it to the name string.
  name += readChar;
}
```

```
// If a name has been entered (followed by \n)
// print "Hello name!"
if (nameEntered) {
if (name.startsWith(st2)&& (name[3]>='0')&& (name[3]<='9') && (name[5]>='0')&&
(name[5]<='9')&&(name[4]==' ') && (name[6]>= '0')&&(name[6]<= '9'))
// if name[0,2] eq INT && name[5,8] are integers
{
  Serial.println("legal ");
  buff = name[3];
  lednumber = buff-'0';//!!!LED number!!!//
  buff = name[5];
  buffa= name[6];
  frequency = 10*(buff-'0')+(buffa-'0');//!!!!!frequency!!!//
  freq[lednumber] = frequency;
  interval[lednumber] = 500000/freq[lednumber];  //set all of the intervals
  for(j=0; j<ledcount;j++)
  {
    Serial.print (j);
    Serial.println(freq[j]);
  }
}
  else if(name.startsWith(st3)&&(name[3]==' ')&& ((name[4]=='1') ||(name[4]=='2')|| (name[4]=='3'))  )
  {     Serial.println("legal ");
     if (name[4] == '1')
     {
       intensity = 10;
            Serial.print (intensity);
```

```
    }
    else if (name[4] == '2')
    {
      intensity = 20;
            Serial.print (intensity);

    }
    else
    {
      intensity = 30;
            Serial.print (intensity);

    }
  }
else {Serial.println("illegal command");}

  // Once the name has been printed, erase it and start over.
  name = "";
  nameEntered = false;
}
while (!Serial.available())  //run loop for 100 seconds to get enough data
{
  currentmicros = micros();  //call micros from main function to ensure it gets whole program time
  timer();
}
}
```