**EE299: Final Project- Wireless Remote Mind Control Car**

**Daniel Jang**

**Michael Molina**

**Ehsan Sadeghmoghaddam**

University of Washington – Department of Electrical Engineering

6/4/2014

Table of Contents

[ABSTRACT 3](#_Toc389775893)

[INTRODUCTION 3](#_Toc389775894)

[DISCUSSION 3](#_Toc389775895)

[Design Specification 3](#_Toc389775896)

[Hardware Implementation 3](#_Toc389775897)

[Software Implementation 6](#_Toc389775898)

[TEST PLAN 8](#_Toc389775899)

[RESULTS 9](#_Toc389775900)

[SUMMARY 16](#_Toc389775901)

[CONCLUSION 17](#_Toc389775902)

[APPENDIX 17](#_Toc389775903)

[Sender 17](#_Toc389775904)

[Receiver 21](#_Toc389775905)

[EEG 26](#_Toc389775906)

# ABSTRACT

This lab report represent a design project that represent everything learn from the previous labs. In this design project, a remote and mind control car was successfully implemented. In the process of the implementation, Xbee shield, EEG sensors, Bluetooth, and analog joy stick are used to invent a RC car which would be able to move with the concentration level of a user wirelessly in addition to the joy stick.

# INTRODUCTION

The purpose of this lab was to utilize knowledge earned about the formal design process from the previous labs by incorporation of a new design project. In this design project, teams were required to use Xbee shields, sensors, and a not used sensor. Our team decided to design a remote control car which also would be able to be controlled by levels of concentration of the user via EEG sensor. In the end, our team implemented a design which the user would be able to remotely move the car to all directions via the joy stick and Xbee shields, and also he/she would be able to move the car forward by concentration on something wirelessly via Bluetooth.

# DISCUSSION

## Design Specification

In this project the requirement was to design and implement a system of choice. The system had to include two Arduinos, wireless communication between the boards, serial monitor usage, LED display, tilt sensor or temperature sensor, and two features the Arduino is capable of but not used in previous projects. For the project the design of an RC car being controlled using an EEG sensor was decided. The temperature sensor is used for shutting the motors down when it reached a threshold temperature, and the wireless communication is used to control the car via remote Arduino module. Two feature that are designed in the project is the custom character function for the LCD and the pulse width modulator output to control motors and a horn.

The input of the system is a PS2 joystick, two buttons for light and horn control, potentiometer for speed control and EEG sensor for another source of speed control, and temperature sensor.

The output of the system is to move a car and shine LEDs and make honking noises.

**Phase 1:**

* ***An EEG receiver (sensor):***
  + Picks up the brain waves via at least one sensor.

**Phase 2:**

* ***A controllable device:***
  + A car with forward and backward movement
  + Moves left right
  + Controllable via joy stick via Xbee shields

**Phase 3:**

* ***Integration of Phase 1 & 2:***
  + Robot movement via translation of the EEG signals

## Hardware Implementation

**Phase 1:**

In this phase, a Uno board was integrated with MindWave EEG sensor. These two were connect via the MindWave’s Bluetooth USB dongle. Refer to the picture below for the block diagram of the phase 1.

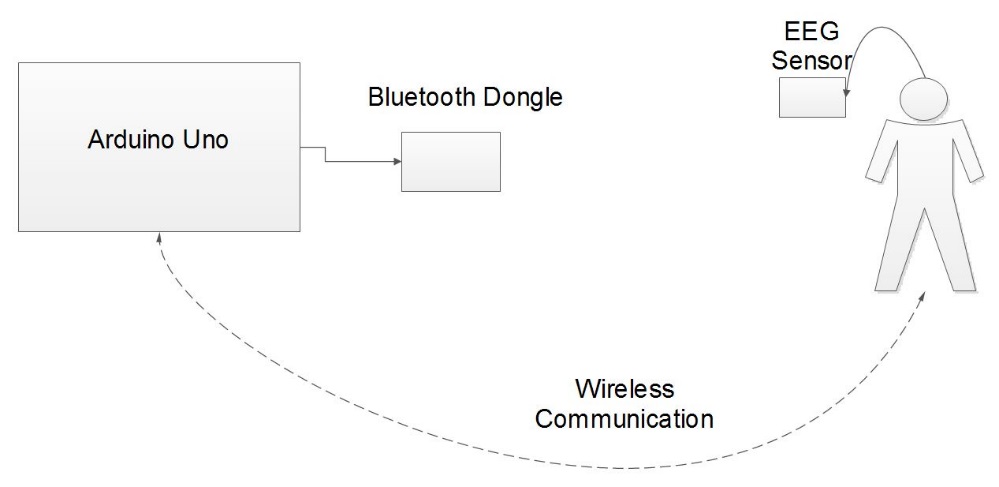
****

Figure 1

**Phase 2:**

In this phase, two Uno boards with Xbee shields were used to transmit data wirelessly. The basic idea is that the user is able to control the car wirelessly with a joy stick. Also for the car hardware, our team used a broken RC toy car to avoid extra costs. The picture below shows block diagram of the hardware implementation of phase 2.

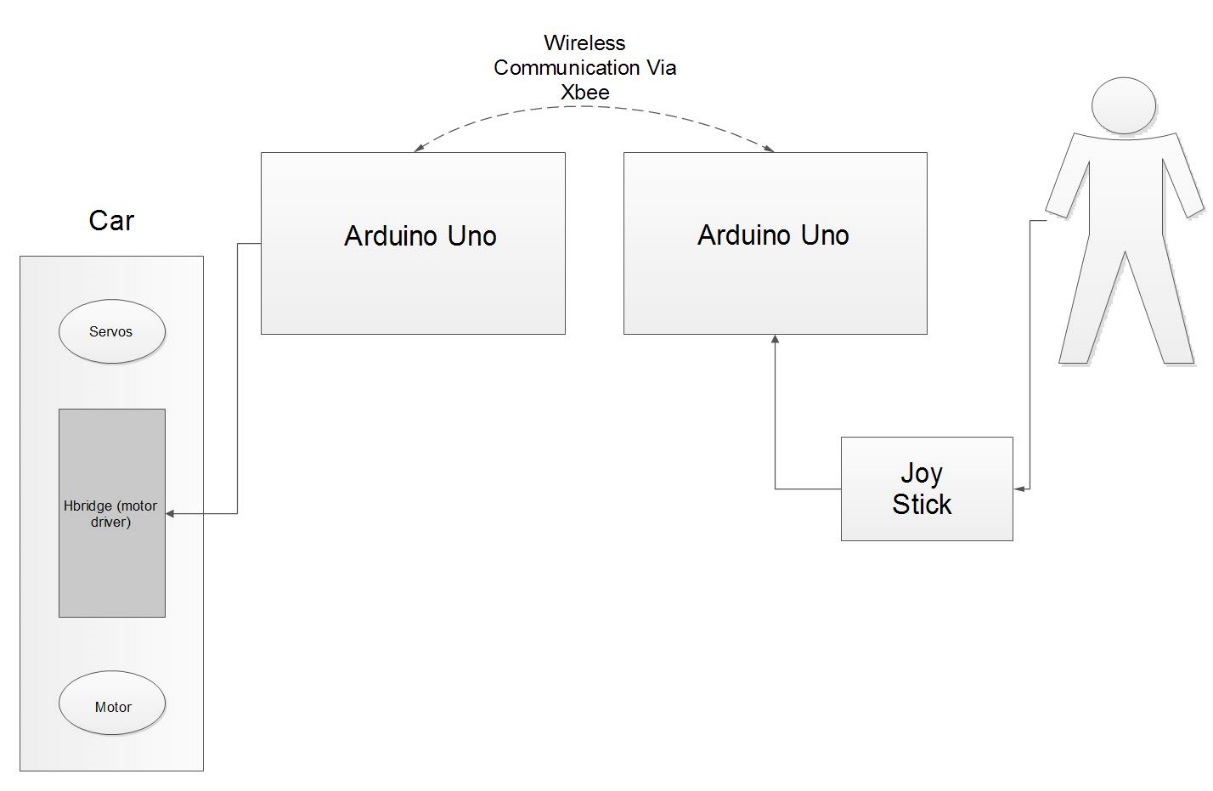
****

Figure 2

**Phase 3:**

In this phase, the phase 1 and phase 2 after successful prototyping were integrated into one project. Please refer to the picture below for the block diagram.

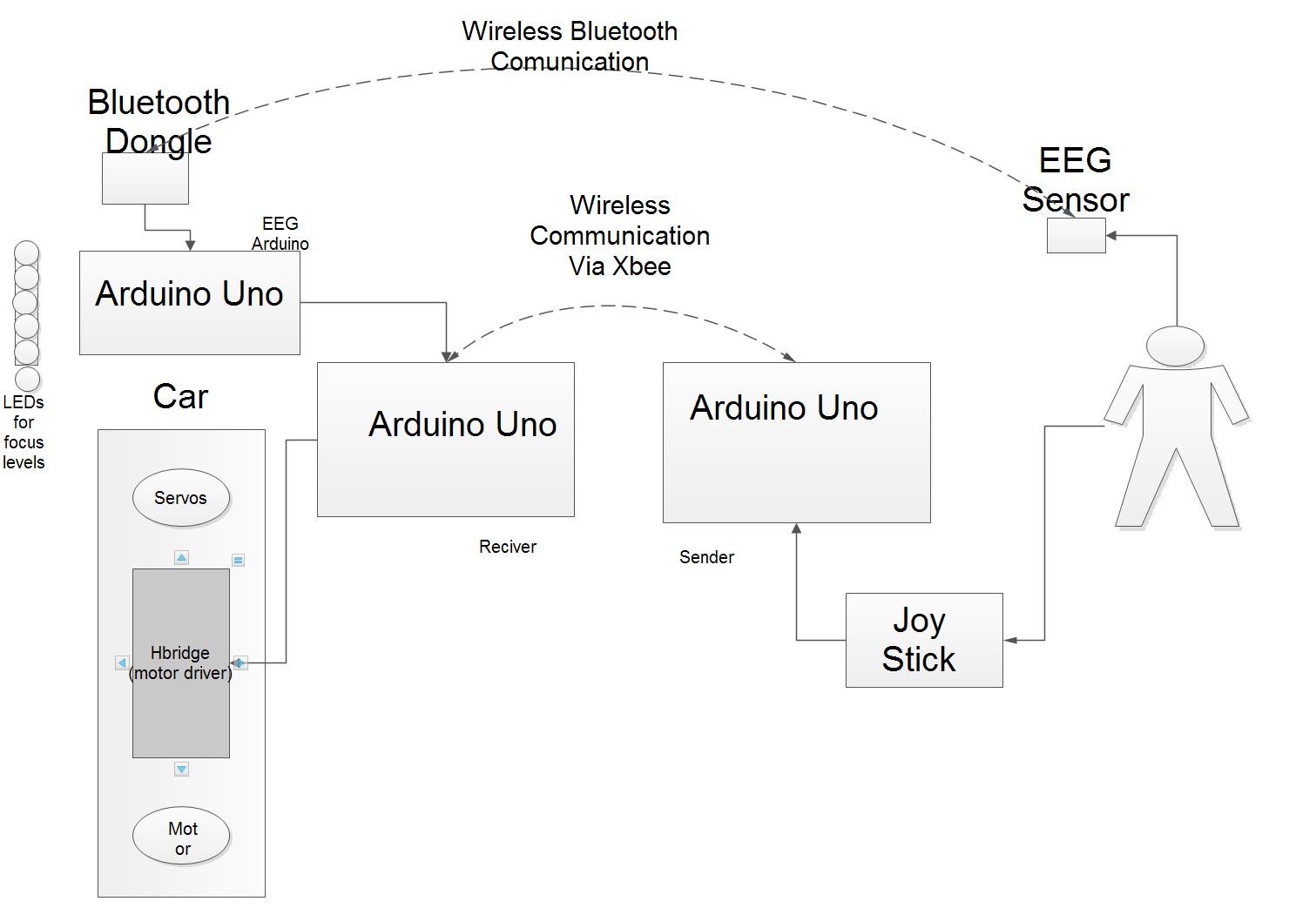
****

Figure 3

**Actual implantation Diagram:**

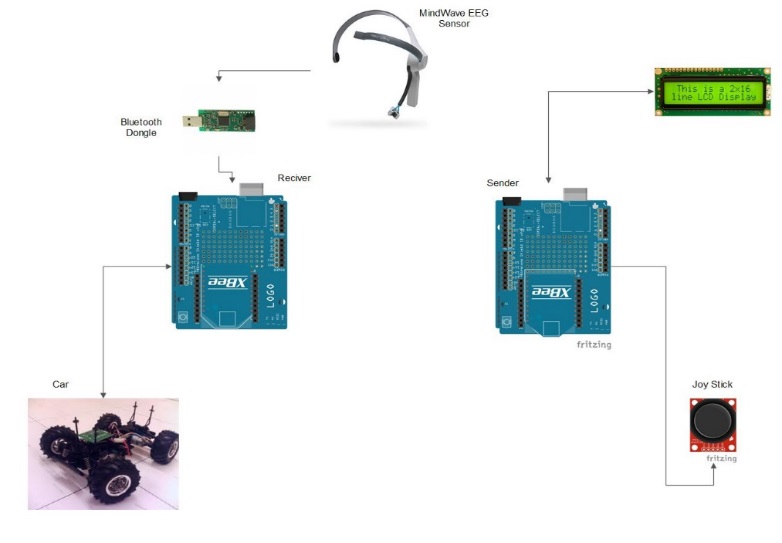
****

Figure 4

## Software Implementation

Our EEG RC car project consists of three codes: 1) Sender, 2) Receiver, 3) EEG.

**Sender**

The sender writes values from the peripherals (PS2 controller, headlights button, horn button) to the receiver. This module also handles the LCD to display the speed of the car and the surrounding temperature.

The PS2 Controller is a joystick that carries information through two important pins. The x and y pin. This is the x and y coordinates of the joystick. The values for x and y is from 0-1022. Where (511,1022) would mean the joystick is all the way up and in the middle. We broke this down into three cases. If X is greater than 597 that would indicate that the joystick is to the right, and less than 425 mean to the left. The values in between 425-597 would mean that the joystick is in the middle which signifies that the cars wheels are straight. The same applies for Y. The values greater than 597 would mean go, less than 425 mean reverse, and in between signifies stop. Depending on the case our implementation sends a character byte to the receiver along with printing a direction to the LCD:

// Stop case

if (directionY < highThreshold && directionY > lowThreshold)

{

Serial.write("M"); // stop

}

// Go case

else if (directionY < lowThreshold)

{

lcd.setCursor(14,0);

lcd.print((char)0);

Serial.write("g"); // go

}

// Reverse case

else

{

lcd.setCursor(14,1);

lcd.print((char)1);

Serial.write("r"); // reverse

}

Similarly depending on the read values of the horn and headlights pins, a different character byte is sent to the receiving Arduino.

Printing an arrow symbol on the LCD was a bit tricky. To do this the function createChar was used. This allowed the programmer to program each pixel an a given individual square on the LCD screen. A set array of size 8 with type byte was used to control which pixels were on to create the custom character. The sample code to set up one of the arrows is shown below:

byte left[8] = {

B00000,

B00100,

B01000,

B11111,

B01000,

B00100,

B00000,

B00000 };

Then to set up the Arduino to save the character the createChar function was used from the lcd library and then to print them the lcd.print function was used with a integer was passed in with a character cast related to the integer that was inputted into the createChar function.

This module also handles the temperature and speed. This is done by using the analogRead(speedPin) and the analogRead(tempPin) command. The speed is determined by a potentiometer and the temperature is determined using a temperature sensor. The values returned from analogRead of the potentiometer is 10 bits long (ranges from 0-1023) we want to fit this onto a byte so we first decided to divide this value by four. This will cause the numbers to range from 0-255. This way we can easily print it out onto the lcd and read the value and write the value to the receiver. Later, we decided that we want this speed value to range from 170-255 because of the threshold speed needed to move the car was 170. We decided to print the percentage of the range to eliminate these “magic numbers”. Simple math 100.0 \* (value - min) / (max - min) would give us the percentage. The implementation for temperature reading was a bit more experimental. We looked through several online resources and found the equations for the temperature in Celsius from the sensor. A little bit of experimenting allowed us to get the most accurate measurements.

**Receiver**

In this module the main purpose is to receive values sent by the sender module, process them, and then perform the necessary operation such as running the motor, sounding the horn, or turning on the lights.

The motors which are connected to an H-bridge can be controlled by writing high and low to the dir\_a and dir\_b pins while setting the pwm\_a and pwm\_b values. dir\_a and pwm\_a control the direction of the front wheels and dir\_b and pwm\_b control the car to either go forward, reverse, or completely stopped.

**EEG:**

This module reads the values from the Bluetooth dongle of MindWave via the Tx and Rx pins. Therefore, the third Arduino was implemented since the Xbee shields use the serial to communicate. This module was implemented on the EEG Arduino (please refer to the hardware implementation).

This module consist of a function connectHeadSet which will connect the Bluetooth and the Arduino if the EEG sensor is turned on within 3 seconds. The buadrate is set to max (115200) to transfer the maximum amount of the data. In the main loop of this module, getAttention is called to update the value of concentration of the user (float attention hold the value) which is in format of percentage (could vary from 0% to 100% shown as float numbers ranging from 0.0 to 1.0).

After update of the attention value, seven if states put the attention value in seven stages to show on LEDs and if the desired state of centration is reached, move the car forward. These stages are:

Stage 1: if(attention > 0 && attention < .10) turn off all display LEDS

Stage 2: if (attention > .10 && attention < .30) turn on the first LED

Stage 3: if(attention > .30 && attention < .50) turn on the second LED

Stage 4: if(attention > .50 && attention < .60) turn on the third LED

Stage 5: if(attention > .60 && attention < .70) turn on the fourth LED

Stage 6: if(attention > .70 && attention < .80) turn on the fifth LED (green LED) and move the car forward until the attention goes below .70

Stage 6: if(attention > .80 && attention < .90) turn on the sixth LED and move the car forward until the attention goes below .70

Stage 6: if( attention > .90) turn on the seventh LED and move the car forward until the attention goes below .70

Note that cars movement is controlled by the receiver Arduino. Therefore to signal forward, this module outputs a high signal which is then received by the receiver Arduino.

# TEST PLAN

Due to complexity of the project our team planed on divide testing into three phases.

Phase 1:

After the implementation, the module was tested by showing the result on multiple LEDs. Basically, the concentration was shown as different levels described in the software implementation. Our team plan to test this phase on number of people (about 6) to assure the accuracy of the design.

Phase 2:

In this phase, Xbee shield were tested since other teams had problems with the hardware. Then, a simple module was implemented to test the accuracy of the data transfer. After this stage, the joy stick was added to test the user wireless control, and this module was tested for cases of all directions (forward, backward, left, right, and combos). A second module was also implemented to test the accuracy of the hbridge which led to usage of different hbridges to support are car specs such as supply of enough current to the motors. Then the two modules were integrated and tested extensively for cases such range of the Xbee shields.

Phase 3:

Finally all modules were integrated into a complete design. Our plan here was to use black box testing to detect any defect. Therefore, sample number of people used our final design to provide feedback about any issues they might have noticed.

# RESULTS

**Phase 1 - EEG**

The bluetooth module that communicated with the EEG sensor was hacked to get the data from the module into the Arduino so it can be analyzed and used to control the motor to move an RC car. In the picture below the module shows wires coming out of the it that go into the Arduino.

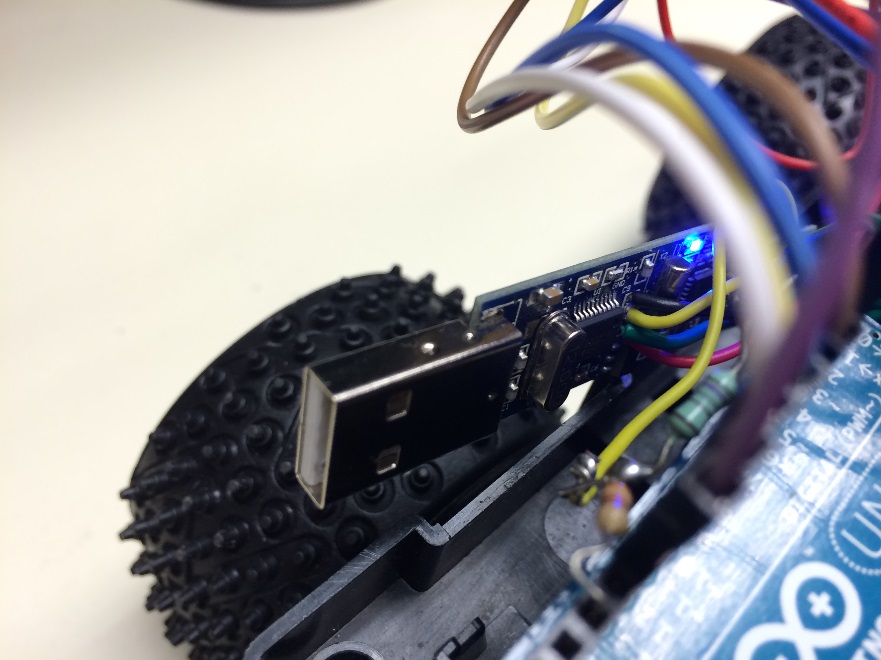


Figure 4. USB module for EEG sensor to connect via bluetooth

Originally the module connected via USB to show a graph a brain activity through software on the computer. In the picture below it shows that the LEDs used to show the concentration level it is receiving works. The specific picture shows the case when 3 out of the 5 LEDs are on showing that the concentration input is over 50%. This shows that the input of the

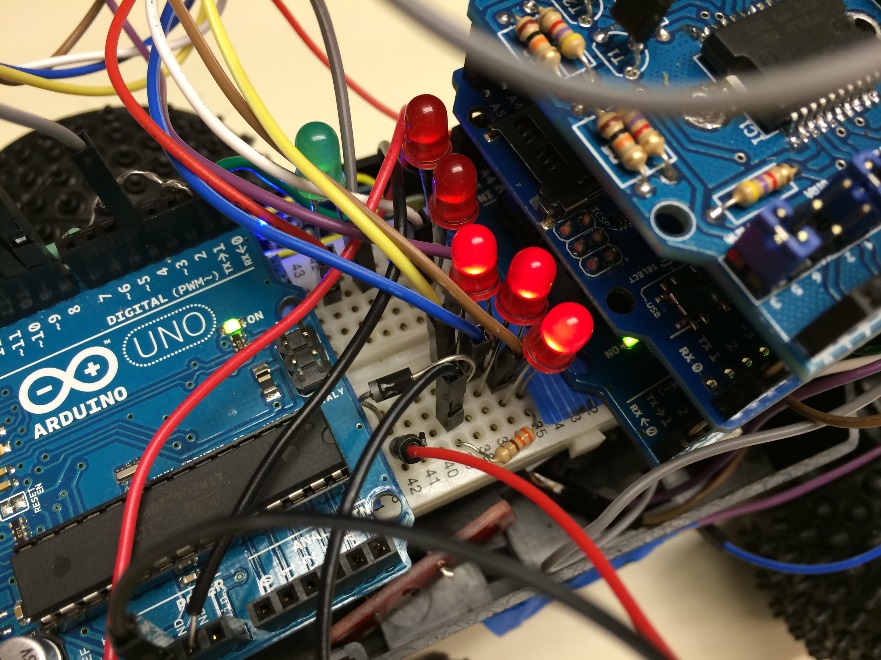


Figure 5. LEDs showing the level of concentration inputting through the EEG sensor

In the picture below and the picture of the bluetooth module, the LEDs are matching blue which means they are paired with each other so the sensor can send data that it reads.



Figure 6. EEG sensor pairing with the bluetooth module

**Phase II**

Using the XBEE on the Remote module of the car, a signal was sent to the XBEE on the Arduino on car module to turn on and off two LEDs for the headlights via button. This proves that the XBEE works and can communicate between the two Arduinos through wifi. The pictures shown below are the states of the headlights turning on and off. A buzzer was also implemented on the car module and controlled via wifi through the remote module and worked just as well as the lights.

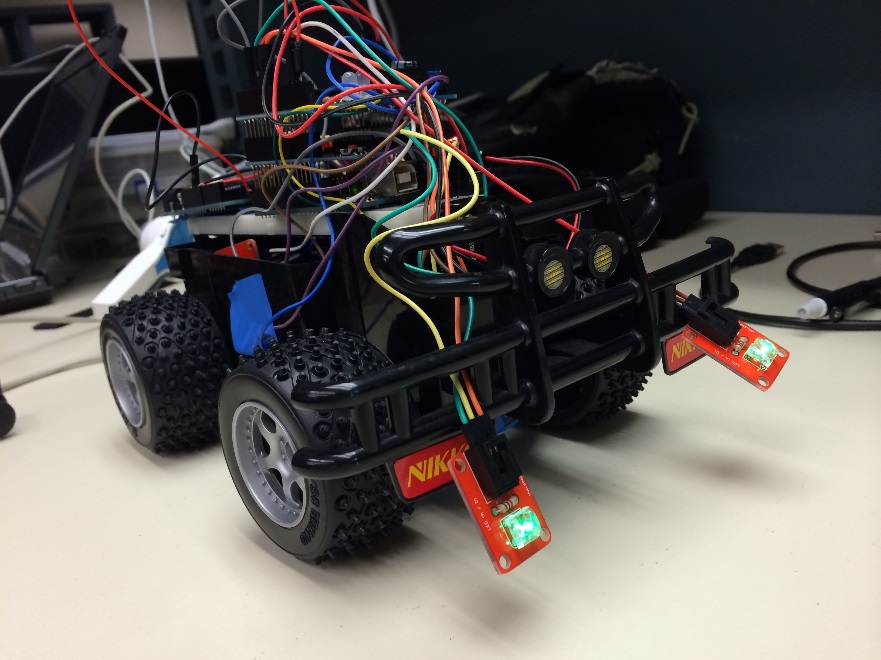
****

Figure 7. Headlights turned on

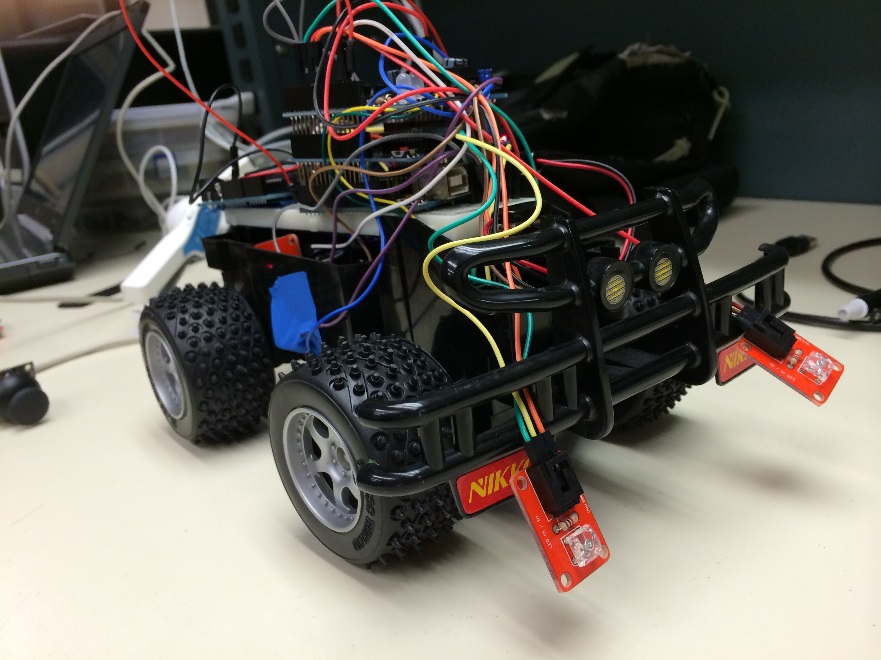
****

Figure 8. Headlights turned off

The pictures below show all of the input modules connected to the remote module Arduino that is being sent to the car module via wifi and the data being sent being printed to the LCD on the remote side. The potentiometer was used to control speed of motor imputing into the PWM and the temperature reaching over 80 degrees Fahrenheit made the car shut down and disable any movement controls from the joystick until it cooled down. Also the LCD prints the custom characters implemented for this project.

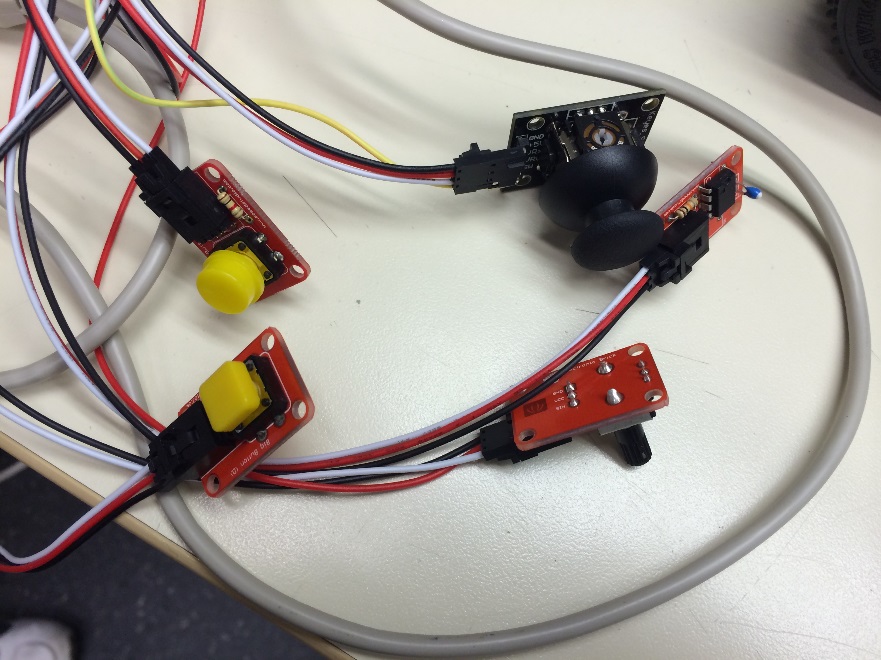


Figure 9. Round button for headlights, square button for horn, joystick for motor direction controls, temperature sensor, and potentiometer

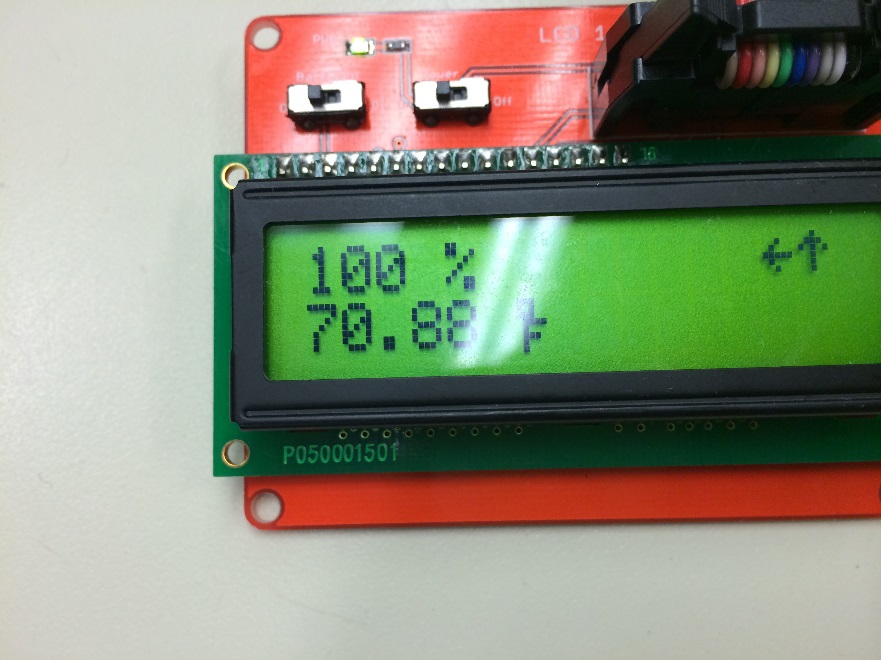


Figure 10. LCD screen printing out the motor speed, custom up, left, Fahrenheit characters, and temperature

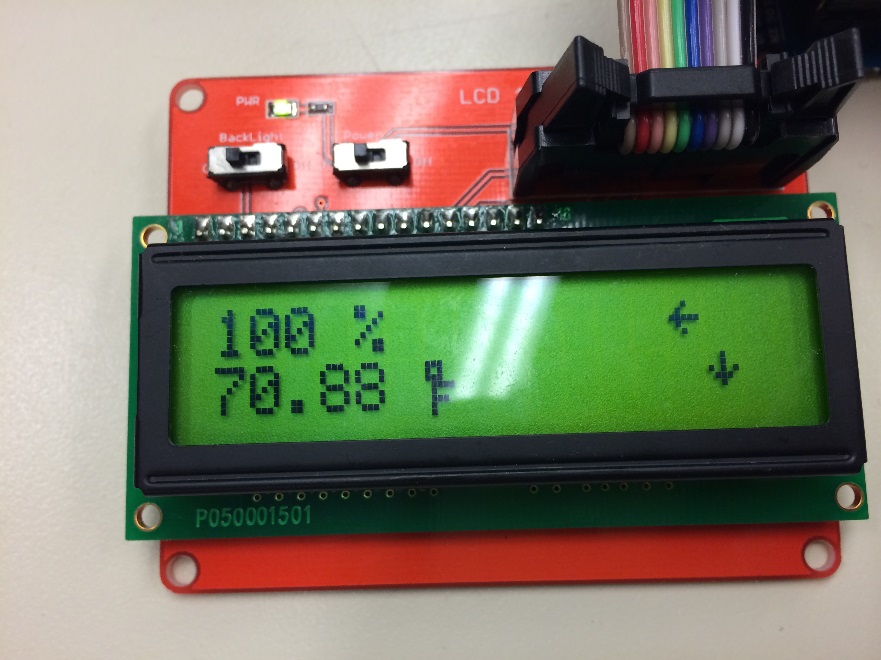


Figure 11. Custom characters left and down

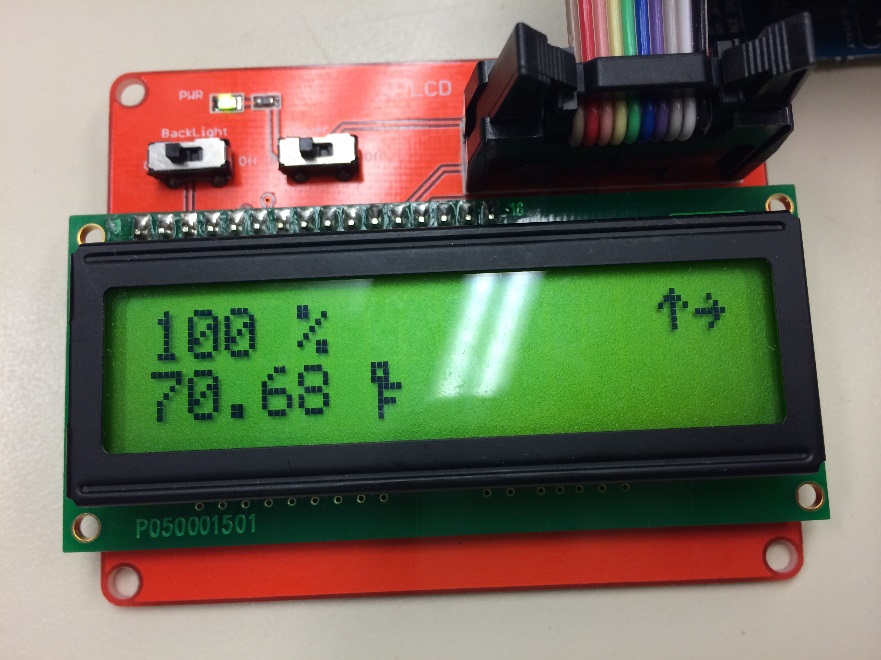


Figure 12. Custom characters up and right

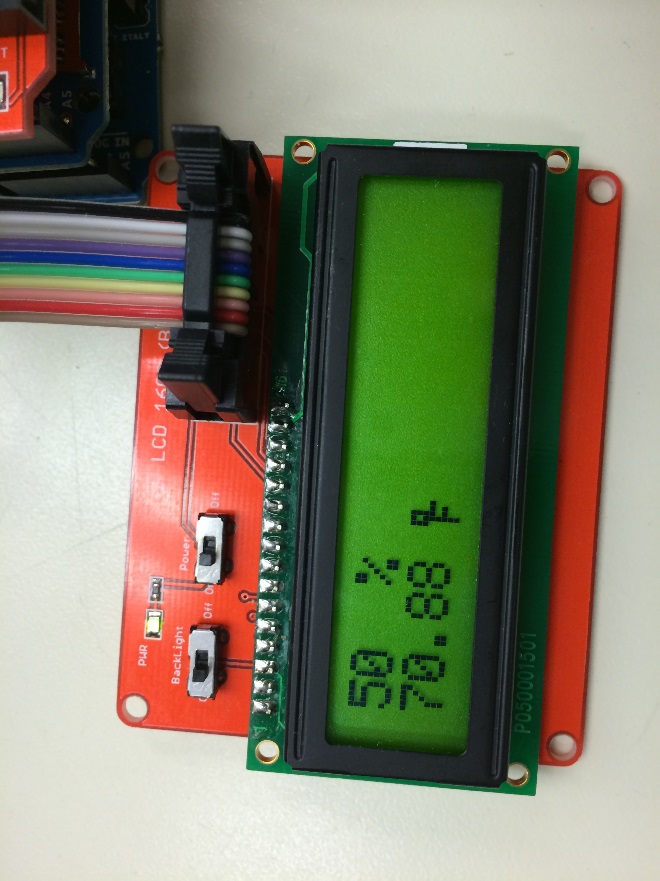


Figure 13. Case when motor speed is changed via potentiometer

To supply enough current for the car module to move and make it wireless, two different power supplies were used. Four double A batteries were used to power the two motors to turn and to move and a 9V batter was used to power the Arduino on the car module so it can become wireless. The pictures below show the battery supplies on the car.



Figure 14. Four double A batteries to power the motors via power shield

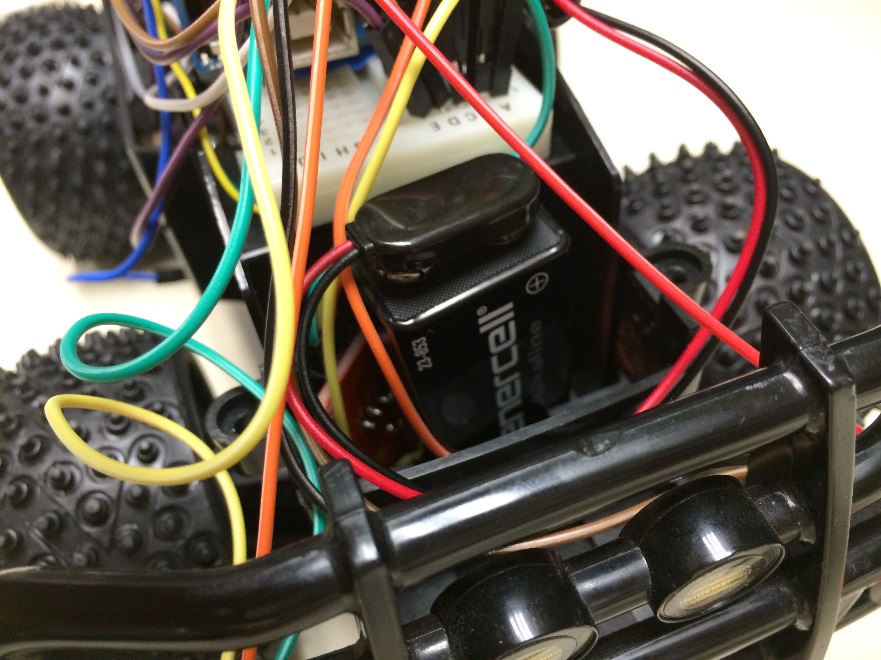


Figure 15. 9V battery used to power Arduino on board for controlling data received from remote

**Phase III**

To implement the EEG sensor with the remote control car system, a third Arduino was used to control an output pin connecting to the power shield on the other Arduino on the car to control the forward movement of the car. The third Arduino was powered via USB so the combined system was not wireless due to the lack of an extra 9V battery and adapter to the other Arduino. The picture below shows the complete car module after impleneting the EEG sensor that moved forward when concentration level reached over 70% (shown via LEDs).

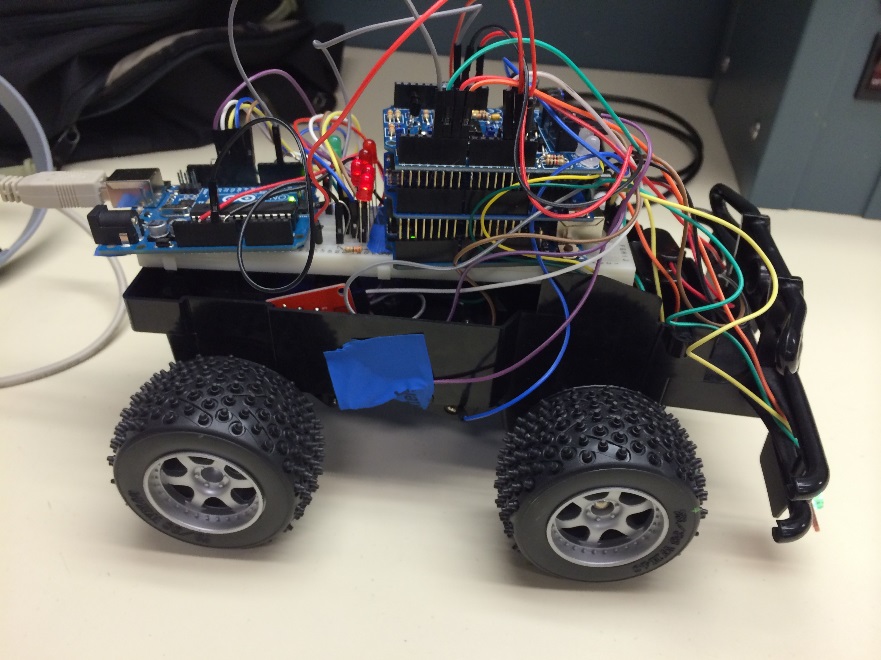


Figure 16. Final car module with both Phase I and Phase II implemented together

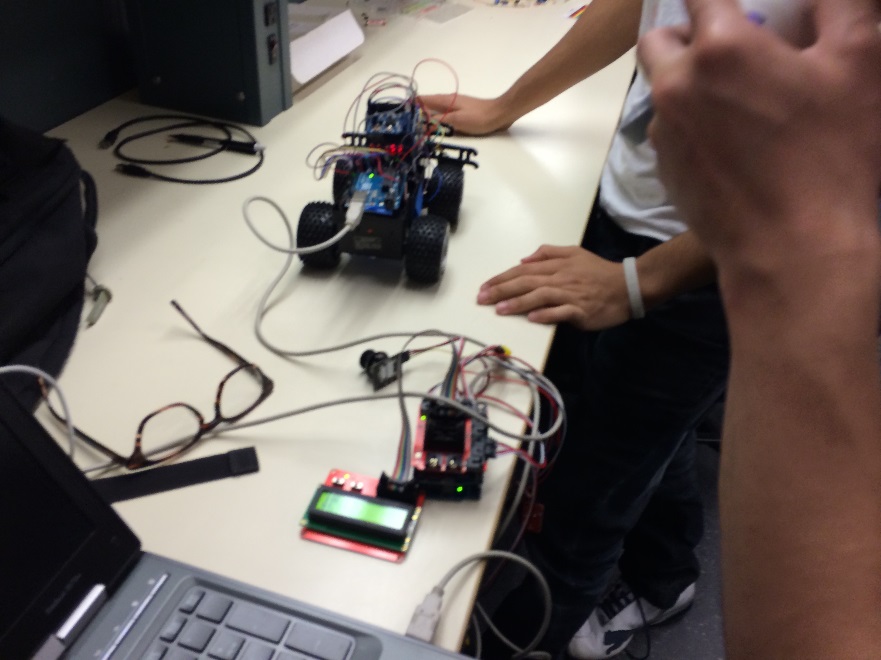


Figure 17. Final system with the remote module and car module

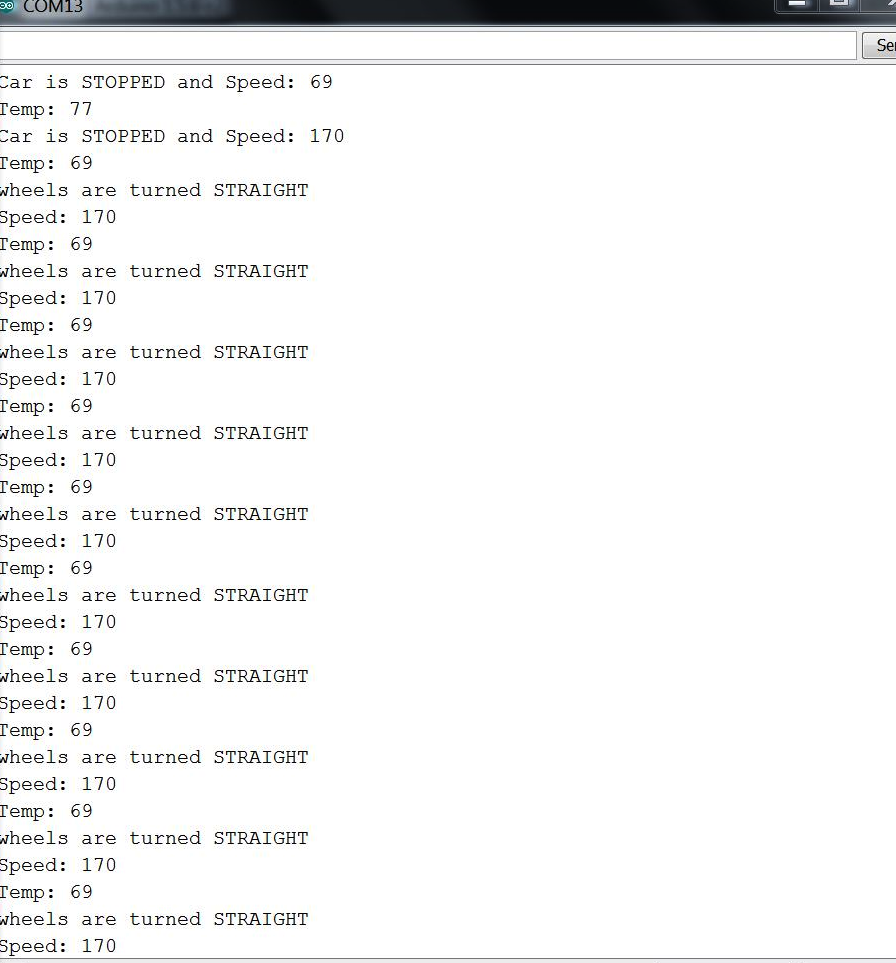


Figure 18. Serial Com debugging output

# SUMMARY

In summary our mind controllable RC car was a success. This project met the design specification requirements. Our design was programmed in the Arduino IDE using C programming language. We began the project by using an old RC car and stripping it until it consisted of just the base, wheels, and motors. We analyzed the behavior of the motors and figured out the conditions necessary to make the wheels go forward, reverse, stop, turn left, turn right, and straight. The next step was to provide the necessary power to the motors. We did so using a power shield for the Arduino. This power shield consists of a bridge that is able to invert the voltages and change intensity of the voltages using pwm. We then coded the sender and receiver modules. The receiver module consists of a wireless Arduino powered by a 9V battery. This module receives character bytes representing commands to be executed. Once this was implemented we moved on to the EEG module. The first part was to hack the USB dongle so that we were able to read from the headset via Bluetooth. The USB dongle connects to the third Arduino. This Arduino processes the incoming data and breaks the concentration level into percentages that would then be represented visually using LEDs. In this module there is a threshold percentage that would send a signal to the receiving Arduino to power the motor to go forward.

# CONCLUSION

In conclusion our project summarizes this course of microprocessors and the C programming language. Using three Arduino microprocessor and several peripherals we were able to wirelessly control an electronic car with our minds (and joystick)!. In this project we utilized the debugging skills we acquired throughout the course of this class. Some complications were getting enough power to the motors and the EEG mind-controlling module. The hardest part was the EEG module because when something was not working it was hard to tell whether it outcome of a tired (all-nighter) brain or poor coding. We were pleased with our final results but noted some possible improvements:

Improvements:

1. Drag speed racing: who can get to the finish line first using their concentration levels!
2. Variable speed changes with different concentration levels
3. Using two Arduinos instead of three

# APPENDIX

Sender

#include <LiquidCrystal.h>

#define lowThreshold 425

#define highThreshold 597

int headlights = 10; // pin for headlights

int horn = 9; // pin for horn

int xPin = 1; // pin for X direction

int yPin = 2; // pin for Y direction

int speedPin = 3; //analog input

int tempPin = 4;

int directionX; // stores the X direction from controller

int directionY; // stores the Y direction from controller

int speedy = 100;

int temp = 0;

// initialize the library with the numbers of the interface pins

LiquidCrystal lcd(2, 3, 4, 5, 6, 7, 8);

// arrow up

byte up[8] = {

B00100,

B01110,

B10101,

B00100,

B00100,

B00100,

B00000,

B00000

};

// arrow down

byte down[8] = {

B00100,

B00100,

B10101,

B01110,

B00100,

B00000,

B00000,

B00000

};

// arrow right

byte right[8] = {

B00000,

B00100,

B00010,

B11111,

B00010,

B00100,

B00000,

B00000

};

// arrow left

byte left[8] = {

B00000,

B00100,

B01000,

B11111,

B01000,

B00100,

B00000,

B00000

};

byte farhen[8] = {

B11100,

B10100,

B11100,

B01111,

B01000,

B01110,

B01000,

B01000

};

void setup() {

Serial.begin(9600);;

pinMode(headlights, INPUT);

pinMode(horn, INPUT);

// pinMode(xPin, INPUT);

// pinMode(yPin, INPUT);

// set up the LCD's number of columns and rows:

lcd.createChar(0, up);

lcd.createChar(1, down);

lcd.createChar(2, right);

lcd.createChar(3, left);

lcd.createChar(4, farhen);

lcd.begin(16, 2);

lcd.clear();

//analogReference(INTERNAL);

}

void loop() {

lcd.clear();

directionX = analogRead(xPin);

directionY = analogRead(yPin);

//--------------------------Y DIRECTION--------------------------------

// stop

speedy = analogRead(speedPin) / 12 + 170;

lcd.setCursor(0,0);

int speedyPercent = 100.0 \* (speedy - 170) / (255 - 170);

lcd.print(speedyPercent);

lcd.setCursor(4, 0);

lcd.print('%');

Serial.write(speedy);

lcd.setCursor(0,1);

int reading = analogRead(tempPin);

float voltage = reading / 1024 \* 500;

float tempC = reading / 9.31 - 32;

float tempF = (tempC \* 9.0 / 5.0) + 32.0;

lcd.print(tempF);

lcd.setCursor(6, 1);

lcd.print((char)4);

Serial.write((int)tempF);

if (directionY < highThreshold && directionY > lowThreshold)

{

Serial.write("M"); // stop

}

// go

else if (directionY < lowThreshold)

{

lcd.setCursor(14,0);

lcd.print((char)0);

Serial.write("g"); // go

}

// reverse

else

{

lcd.setCursor(14,1);

lcd.print((char)1);

Serial.write("r"); // reverse

}

//--------------------------X DIRECTION--------------------------------

// straight

if (directionX < highThreshold && directionX > lowThreshold)

{

Serial.write("s"); // straight

}

// left

else if (directionX < lowThreshold)

{

lcd.setCursor(13,0);

lcd.print((char)3);

Serial.write("l"); // left

}

// right

else

{

lcd.setCursor(15,0);

lcd.print((char)2);

Serial.write("r"); // right

}

// //---------------------------- HORN---------------------------------

if (digitalRead(horn) == HIGH)

{

Serial.write("h"); // horn

}

else

{

Serial.write("n"); // silent

}

// //---------------------------HEADLIGHTS-----------------------------

if (digitalRead(headlights) == HIGH)

{

Serial.write("M"); // switch state

}

else

{

Serial.write("k"); // keep state

}

//Serial.flush();

delay(180);

}

Receiver

#define period 10

//const int leftMotorPin = 9;

//const int rightMotorPin = 8;

//

//const int forwardMotorPin = 10;

//const int backwardMotorPin = 11;

int pwm\_a = 3; //PWM control for motor outputs 1 and 2

int pwm\_b = 9; //PWM control for motor outputs 3 and 4

int dir\_a = 2; //direction control for motor outputs 1 and 2

int dir\_b = 8; //direction control for motor outputs 3 and 4

int temp = 0;

int speedy = 0;

const int headlightsL = 6;

const int headlightsR = 7;

const int horn = 5;

int tilt = 12;

const int EEGpin = 4;

const int led = 13;

char go;

char oneDirection;

char hornChar;

char headlightsChar;

boolean headlightsOn = false;

boolean tiltFlag = true;

const char s = 'M';

void setup() {

Serial.begin(9600);

pinMode(pwm\_a, OUTPUT); //Set control pins to be outputs

pinMode(pwm\_b, OUTPUT);

pinMode(dir\_a, OUTPUT);

pinMode(dir\_b, OUTPUT);

pinMode(tilt, INPUT);

pinMode(headlightsL, OUTPUT);

pinMode(headlightsR, OUTPUT);

pinMode(horn, OUTPUT);

pinMode(EEGpin, INPUT);

pinMode(led, OUTPUT);

analogWrite(pwm\_a, 0);

analogWrite(pwm\_b, 0);

// pinMode(leftMotorPin, OUTPUT);

// pinMode(rightMotorPin, OUTPUT);

// pinMode(forwardMotorPin, OUTPUT);

// pinMode(backwardMotorPin, OUTPUT);

}

// the loop routine runs over and over again forever:

void loop() {

//Serial.print("Tilt: ");

//Serial.print(digitalRead(tilt));

//Serial.print('\n');

//EEG More 80% attention, go forward

if(digitalRead(EEGpin)){

digitalWrite(led, HIGH);

while(digitalRead(EEGpin)){

digitalWrite(dir\_b, HIGH);

analogWrite(pwm\_b, 210);

//Serial.flush();

}

}

if(!(digitalRead(EEGpin))){

digitalWrite(led, LOW);

//Serial.flush();

Stop();

straight();

}

if (temp > 79)

{

//Serial.println("Tilt Flag true");

tiltFlag = false;

Stop();

}

else

{

tiltFlag = true; //false

// Serial.println("Tilt Flag false");

}

if (Serial.available() > 0)

{

speedy = Serial.read();

Serial.print("Speed: ");

Serial.print(speedy);

Serial.print('\n');

}

if (Serial.available() > 0)

{

temp = Serial.read();

Serial.print("Temp: ");

Serial.print(temp);

Serial.print('\n');

}

if (Serial.available() > 0)

{

go = Serial.read();

// stop

if (go == 's' && tiltFlag == true)

{

Serial.print("Car is STOPPED and ");

Stop();

}

// go

else if (go == 'g' && tiltFlag == true)

{

Serial.print("Car is GOING and ");

forward(speedy);

}

// reverse

else if (go == 'r' && tiltFlag == true)

{

Serial.print("Car is REVERSING and ");

backward(speedy);

}

}

if (Serial.available() > 0)

{

oneDirection = Serial.read();

// staight

if (oneDirection == 's' && tiltFlag)

{

Serial.println("wheels are turned STRAIGHT");

straight();

}

// left

else if (oneDirection == 'l' && tiltFlag)

{

Serial.println("wheels are turned LEFT");

left();

}

// right

else if (oneDirection == 'r' && tiltFlag)

{

Serial.println("wheels are turned RIGHT");

right();

}

}

if (Serial.available() > 0)

{

hornChar = Serial.read();

if (hornChar == 'h')

{

analogWrite(horn, 50);

}

else

{

analogWrite(horn, 0);

}

}

if (Serial.available() > 0)

{

headlightsChar = Serial.read();

if (headlightsChar == s)

{

headlightsOn = !headlightsOn;

}

}

if (headlightsOn)

{

digitalWrite(headlightsL, HIGH);

digitalWrite(headlightsR, HIGH);

}

else

{

digitalWrite(headlightsL, LOW);

digitalWrite(headlightsR, LOW);

}

// Serial.print("go = ");

// Serial.println(go);

// Serial.print("left = ");

// Serial.println(oneDirection);

delay(180);

}

void left(){

digitalWrite(dir\_a, LOW);

analogWrite(pwm\_a, 225);

}

void right(){

digitalWrite(dir\_a, HIGH);

analogWrite(pwm\_a, 200);

}

void forward(int speedy){

digitalWrite(dir\_b, HIGH);

analogWrite(pwm\_b, speedSet);

}

void backward(int speedy){

int speedSet = speedy \* 2;

digitalWrite(dir\_b, LOW);

analogWrite(pwm\_b, speedSet);

}

void straight(){

// digitalWrite(dir\_a, LOW);

analogWrite(pwm\_a, 0);

}

void Stop(){

analogWrite(pwm\_b, 0);

}

EEG

// Thank to Kimmo Karvinen & Tero Karvinen for getAttention

const int tinyLedPin = 13;

// LED for level of concentration

const int led1 = 8;

const int led2 = 9;

const int led3 = 10;

const int led4 = 11;

const int led5 = 12;

const int moveForward = 4;

void setup()

{

pinMode(tinyLedPin, OUTPUT);

pinMode(led1, OUTPUT);

pinMode(led2, OUTPUT);

pinMode(led3, OUTPUT);

pinMode(led4, OUTPUT);

pinMode(led5, OUTPUT);

pinMode(moveForward, OUTPUT);

Serial.begin(115200);

connectHeadset();

}

void loop()

{

float att = getAttention();

if(att > 0 && att < .10){

digitalWrite(led1, LOW);

digitalWrite(led2, LOW);

digitalWrite(led3, LOW);

digitalWrite(led4, LOW);

digitalWrite(led5, LOW);

digitalWrite(moveForward, LOW);

}

if(att > .10 && att < .30){

digitalWrite(led1, HIGH);

digitalWrite(led2, LOW);

digitalWrite(led3, LOW);

digitalWrite(led4, LOW);

digitalWrite(led5, LOW);

digitalWrite(moveForward, LOW);

}

if(att > 0.30 && att < .50){

digitalWrite(led1, HIGH);

digitalWrite(led2, HIGH);

digitalWrite(led3, LOW);

digitalWrite(led4, LOW);

digitalWrite(led5, LOW);

digitalWrite(moveForward, LOW);

}

if(att > 0.50 && att < .60){

digitalWrite(led1, HIGH);

digitalWrite(led2, HIGH);

digitalWrite(led3, HIGH);

digitalWrite(led4, LOW);

digitalWrite(led5, LOW);

digitalWrite(moveForward, LOW);

}

if(att > 0.70 && att < .80){

digitalWrite(led1, HIGH);

digitalWrite(led2, HIGH);

digitalWrite(led3, HIGH);

digitalWrite(led4, HIGH);

digitalWrite(led5, LOW);

digitalWrite(moveForward, HIGH);

}

if(att > 0.80 && att < .90){

digitalWrite(led1, HIGH);

digitalWrite(led2, HIGH);

digitalWrite(led3, HIGH);

digitalWrite(led4, HIGH);

digitalWrite(led5, LOW);

digitalWrite(moveForward, HIGH);

}

if(att > 0.90){

digitalWrite(led1, HIGH);

digitalWrite(led2, HIGH);

digitalWrite(led3, HIGH);

digitalWrite(led4, HIGH);

digitalWrite(led5, HIGH);

digitalWrite(moveForward, HIGH);

}

}

/\*\*\* Headset \*\*\*/

void connectHeadset()

{

//setGreen();

delay(3000);

Serial.write(0xc2);

//setWhite();

}

byte readOneByte()

{

while (!Serial.available()) { // <7>

delay(5); // ms

};

return Serial.read();

}

float getAttention()

{ // return attention percent (0.0 to 1.0); negative (-1, -2...) for error

byte generatedChecksum = 0; // <8>

byte checksum = 0;

int payloadLength = 0;

byte payloadData[64] = {

0

};

int poorQuality = 0;

float attention = 0;

Serial.flush(); // prevent serial buffer from filling up // <9>

/\* Sync \*/

if (170 != readOneByte()) return -1; // <10>

if (170 != readOneByte()) return -1;

/\* Length \*/

payloadLength = readOneByte();

if (payloadLength > 169) return -2; // <11>

/\* Checksum \*/

generatedChecksum = 0;

for (int i = 0; i < payloadLength; i++) { // <12>

payloadData[i] = readOneByte(); // Read payload into array

generatedChecksum += payloadData[i];

}

generatedChecksum = 255 - generatedChecksum;

checksum = readOneByte();

if (checksum != generatedChecksum) return -3; // <13>

/\* Payload \*/

for (int i = 0; i < payloadLength; i++) { // <14>

switch (payloadData[i]) {

case 0xD0:

//sayHeadsetConnected();

break;

case 4: // <15>

i++; // <16>

attention = payloadData[i]; // <17>

break;

case 2:

i++;

poorQuality = payloadData[i];

if (200 == poorQuality) {

//setYellow(); // <18>

return -4;

}

break;

case 0xD1: // Headset Not Found

case 0xD2: // Headset Disconnected

case 0xD3: // Request Denied

case -70:

return -5;

break;

case 0x80: // skip RAW // <19>

i = i + 3;

break;

case 0x83: // skip ASIC\_EEG\_POWER

i = i + 25;

break;

} // switch

} // for

return (float)attention / 100; // <20>

}