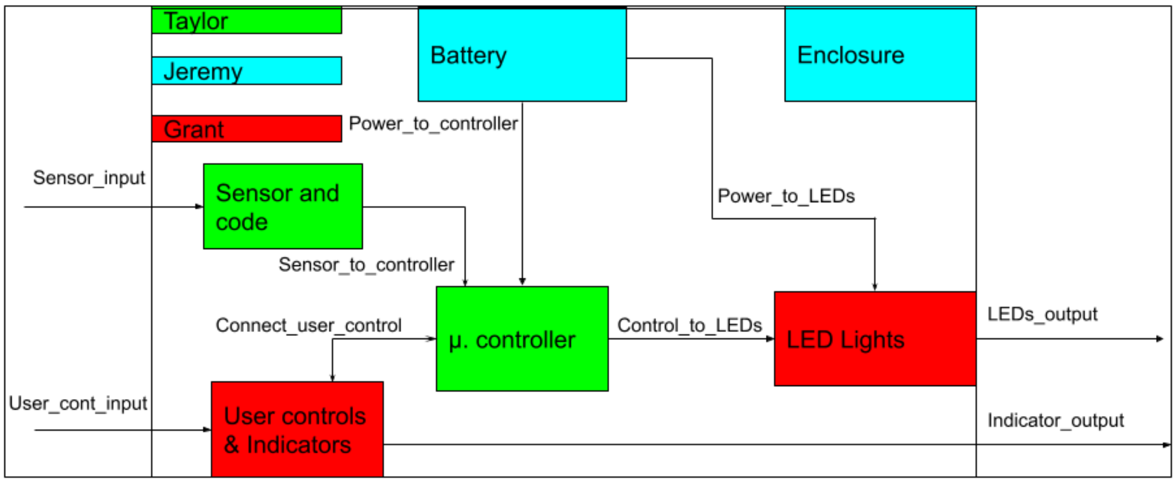
Bike Headlight and Taillight system



Interface Name	Specification
User_control_input	Two switches 2 [lbs/In^2]
Sensor_input	Gyroscope sensitivity: scale range: ±250, ±500, ±1000, and ±2000°/sec Accelerometer Sensitivity: scale range: ±2g, ±4g, ±8g and ±16g
Power_to_controller	V_{nom} = 5 [V] A_{nom} = 19 [mA]
Power_to_LEDs	V_{nom} = 12 [V] A_{max} = 700 [mA]
Sensor_to_controller	V_{nom} = 1.9-3.6 [V] A_{Nom} = 3.9 [mA] Multiple connections to Micro. Contr.
Connect_user_control	V_{nom} = 5 [V] A_{max} = 1.5 [mA]
Control_to_LEDs	V_{on} = 5 [V] V_{off} = 0 [V] A_{max} = 100 [nA]
Indicator_output	Visual: LEDs on / off

LEDs_output	Visual: LEDs on / off 380 Lumens Max

Grant Everson
Junior Design 2
February 4, 2022

This documentation is for the user controls of the bike project. These user controls will feature two LEDs and two membrane push buttons to allow the user to tell what indicator is on and to turn them on and off. The LEDs will flash in sync with the indicators on the back of the bike. Pressing a button will turn on the corresponding indicator and pressing it twice in quick succession will turn it off. This block will communicate with the microcontroller through the button presses and the microcontroller will control the flashing of the LEDs. Overall block view, black box, interface definitons, circuit design, pcb design, and total price are in the rest of the document below.

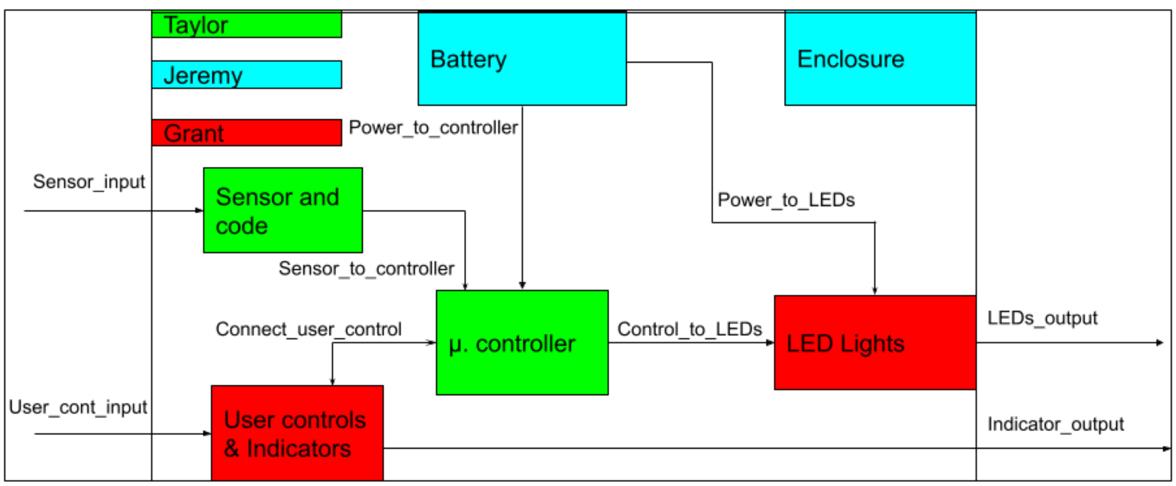


Figure 1: Top Level Diagram

User inputs = user_cont_input connect_ser_control = To / From microcontroller

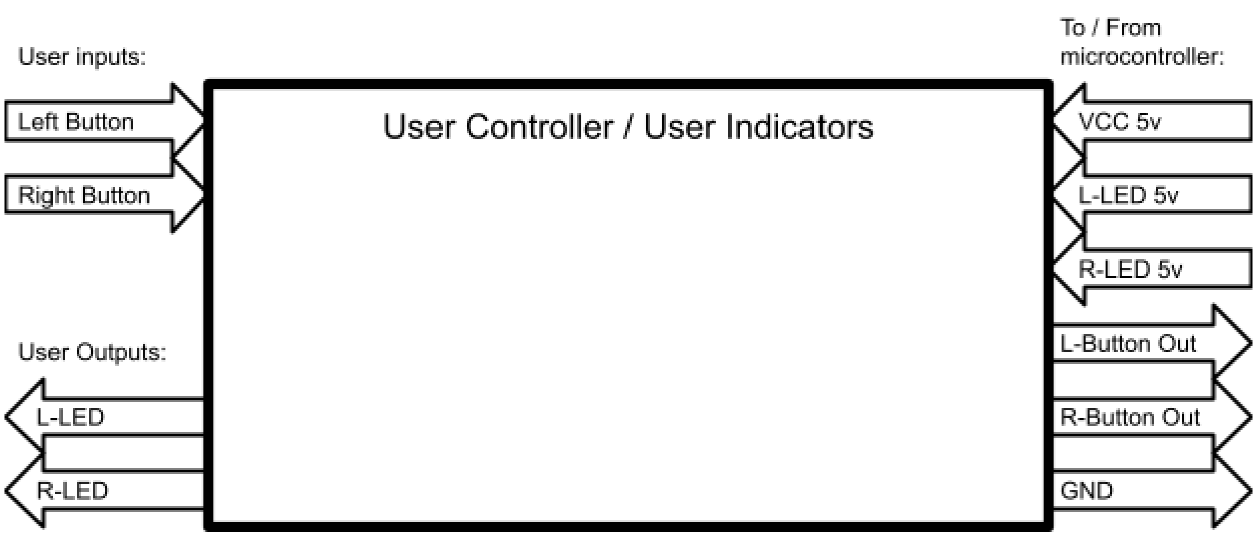


Figure 2: Black Box Diagram

Interface Name	Specifications:
Left Button (User input)	Human pressure input
Right Button (User input)	Human pressure input
L-LED (User output)	On / Off (Could blink with indicators)
R-LED (User output)	On / Off (Could blink with indicators)
VCC	V_{norm} = 5 [V] A_{norm} = 0.0 - 0.5 [mA]
L-LED (Controlled by signal from arduino)	V_{active} = 2 [V] A_{active} = 0.3 [mA] $V_{inactive}$ = 0 [V] $A_{inactive}$ = 0.0 [mA]
R-LED (Controlled by signal from arduino)	V_{active} = 2 [V] A_{active} = 0.3 [mA] $V_{inactive}$ = 0 [V] $A_{inactive}$ = 0.0 [mA]
L-Button Out (Output to arduino)	V_{active} = 5 [V] A_{active} = 0.5 [mA] $V_{inactive}$ = 0 [V] $A_{inactive}$ = 0.0 [mA]
R-Button Out (Output to arduino)	V_{active} = 5 [V] A_{active} = 0.5 [mA] $V_{inactive}$ = 0 [V] $A_{inactive}$ = 0.0 [mA]
GND	V_{norm} = 0 [V] A_{norm} = 1 [mA]

Table 1: Interface Specifications

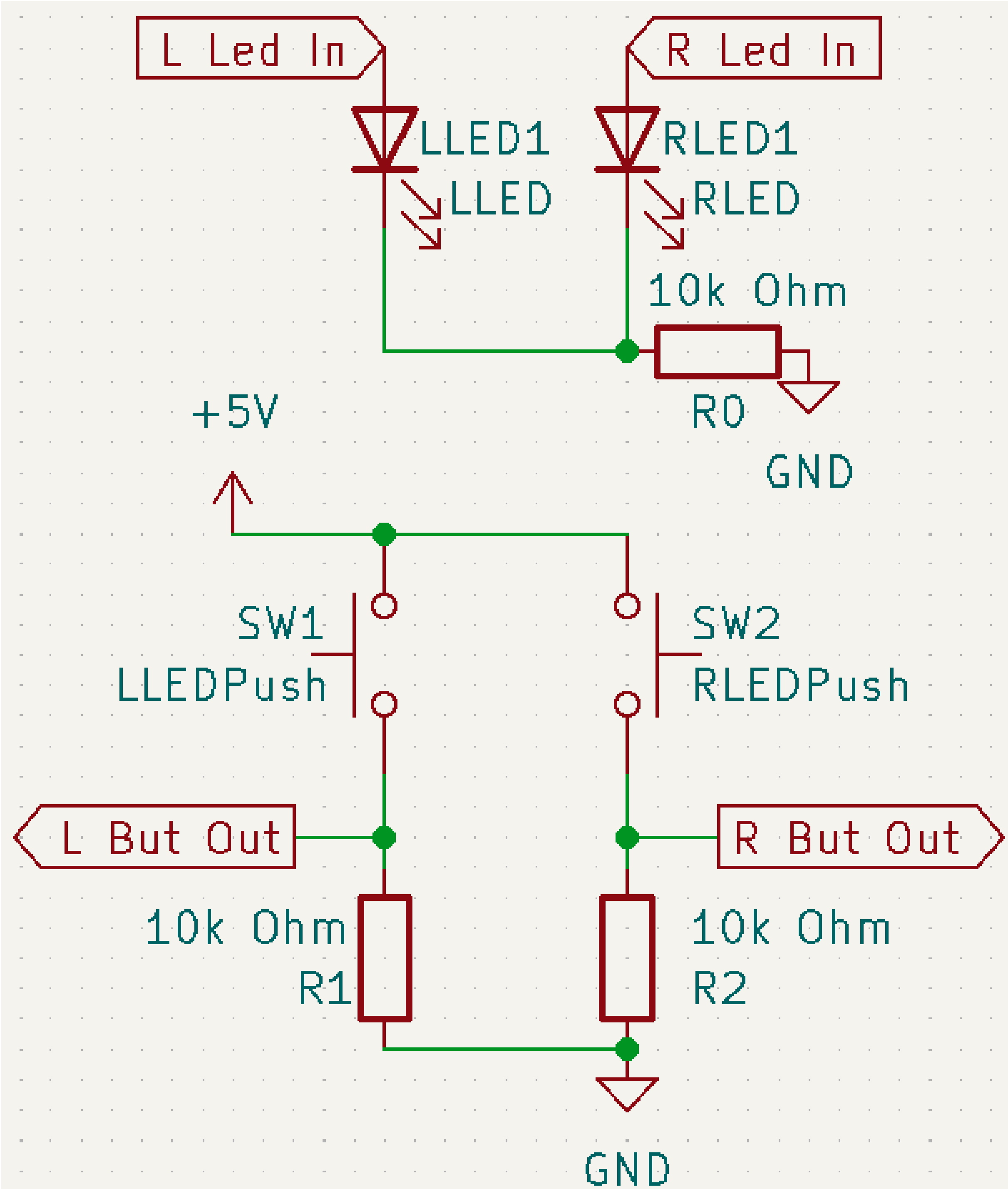


Figure 3: Schematic

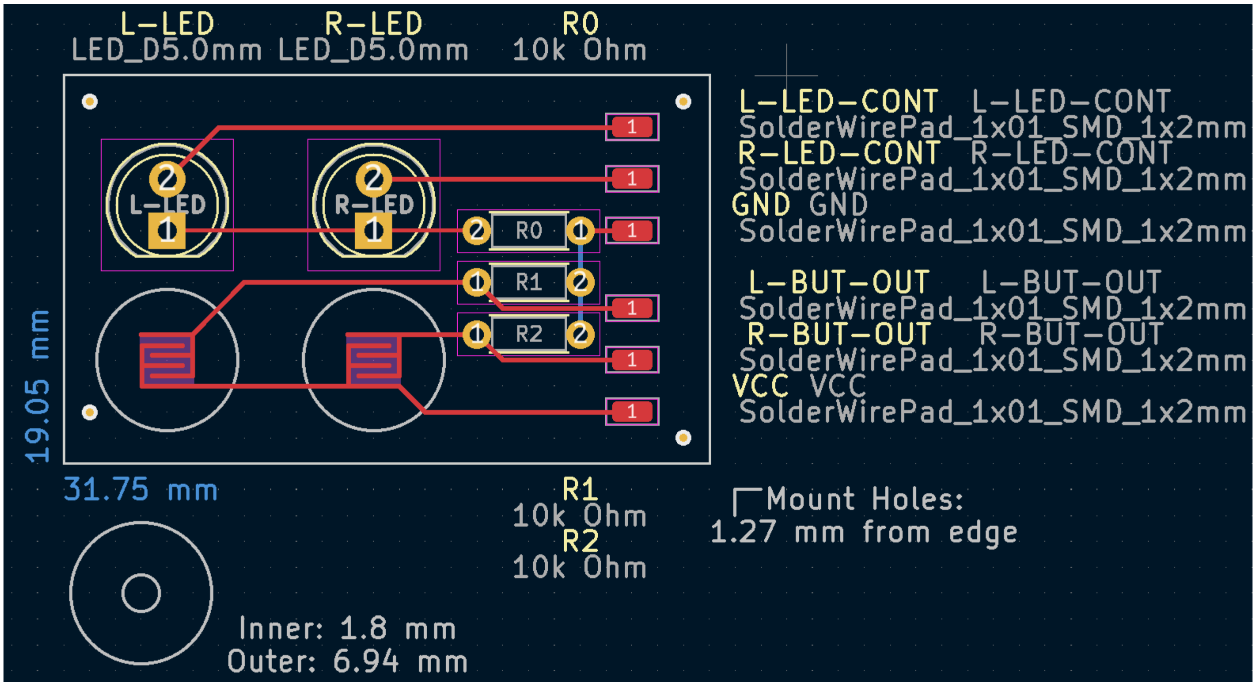


Figure 4: PCB Schematic with Dimensions

Quantity	Designator	Part:	Value	Price (individual)	Manufacturer
2	L-LED R-LED	LED	5mm 2V	$0.07	EDGELEC
2	Circles underneath LEDs	Silicon Conductive Pad Buttons	N/A	$0.18	Unbranded*
3	R0 R1 R2	Resistors	10k Ohm 0.5 Watt	$0.04	EDGELEC
1		PCB		$1.55	OSH Park
Total Cost:				$1.84	

Table 2: Parts Price List

*Buttons found at: https://www.ebay.com/itm/224076733119

Code: (Starts on next page)

```
// Grant Everson
// Bike Project User Controls test Script
// 1/31/2022
// Used to test the front user controls

// Corresponding pin definitions
const int but1 = 2;
const int but2 = 3;
const int led1 = 4;
const int led2 = 5;

// button states
int but1state = 0;
int but2state = 0;

void setup() {
 // Interface definitions for input and outputs
  pinMode(but1, INPUT);
  pinMode(but2, INPUT);

  pinMode(led1, OUTPUT);
  pinMode(led2, OUTPUT);

}

void loop() {
 // read the states of the button
  but1state = digitalRead(but1);
  but2state = digitalRead(but2);

 // Check if the button states are high
 // If so then set LED to high
 // Otherwise turn off the LED
  if (but1state == HIGH) {
        digitalWrite(led1, HIGH);
  } else {
        digitalWrite(led1, LOW);
  }
  if (but2state == HIGH) {
        digitalWrite(led2, HIGH);
  } else {
        digitalWrite(led2, LOW);
  }
}
```

Taylor Cole Brennan
Junior Design 2
Jan 28, 2022

Block 2: Sensor and Code

This is to document the Sensor and code block for the Junior design 2 bike project. The gyroscope/accelerometer has a connection to a voltage source with a max of 3.46 volts, ground and 3 analog connections to the microcontroller and receives analog information from the outside world. How it's coded, the brakes will activate after hitting a minimum deceleration value, based on the accelerometer input. Based on that value, the High and low value for the brake lights are divided causing a pulse width modulation. This division is taken into account when timing the turn signals as well. The turn signals are set to 1sec alterations. The turn signals need to be activated by user control, but after being activated, the turn signals won't turn off until the bike has turned so much (minimum turn amount) in order to detect early turn signallers.

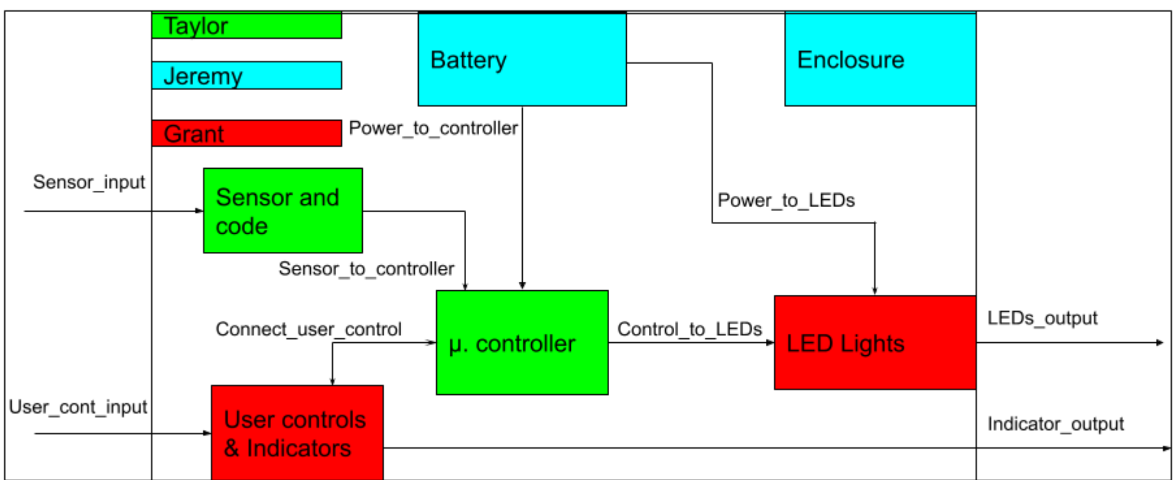


Figure 1: Top Level Diagram

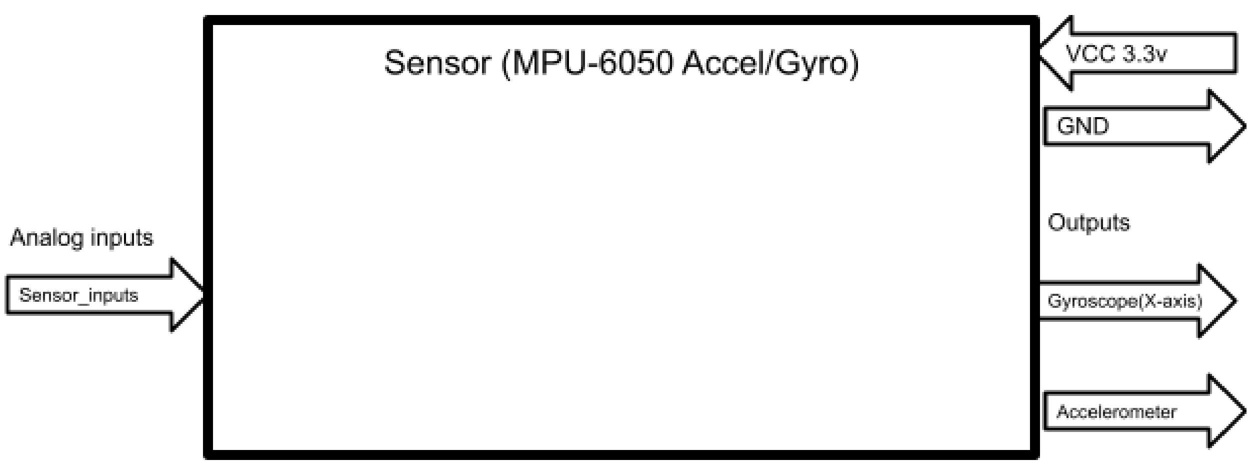


Figure 2: Black Box Diagram

Interface Name	Specifications:
Gyroscope_x_axis	scale range: ±250, ±500, ±1000, and ±2000°/sec Standby current: 5µA Gyroscope operating current: 3.6mA V_{norm} = 2.375 - 3.46 [V]
Accelerometer	scale range: ±2g, ±4g, ±8g and ±16g normal operating current: 500µA Low power current: 10µA (1.25Hz), 20µA (5Hz), 60µA (20Hz), 110µA (40Hz) V_{norm} = 2.375 - 3.46 [V]
Sensor_input	Gyroscope sensitivity: scale range: ±250, ±500, ±1000, and ±2000°/sec Accelerometer Sensitivity: scale range: ±2g, ±4g, ±8g and ±16g
Vcc	V_{norm} = 3.3 [V] A_{norm} = 19 [mA]
Gnd	V_{norm} = 0 [V] A_{norm} = 1 [mA]

Table 1: Interface Specification

Ref #	Qty.	Description	Item Model #	Manufacturer	Link	Price
2A	1	MPU-6050 MPU6050 3 Axis Accelerometer Gyroscope Module	3-01-0122-AB	HiLetgo	Amazon	$3.33 (3:$10)

Table 2: Parts Price List

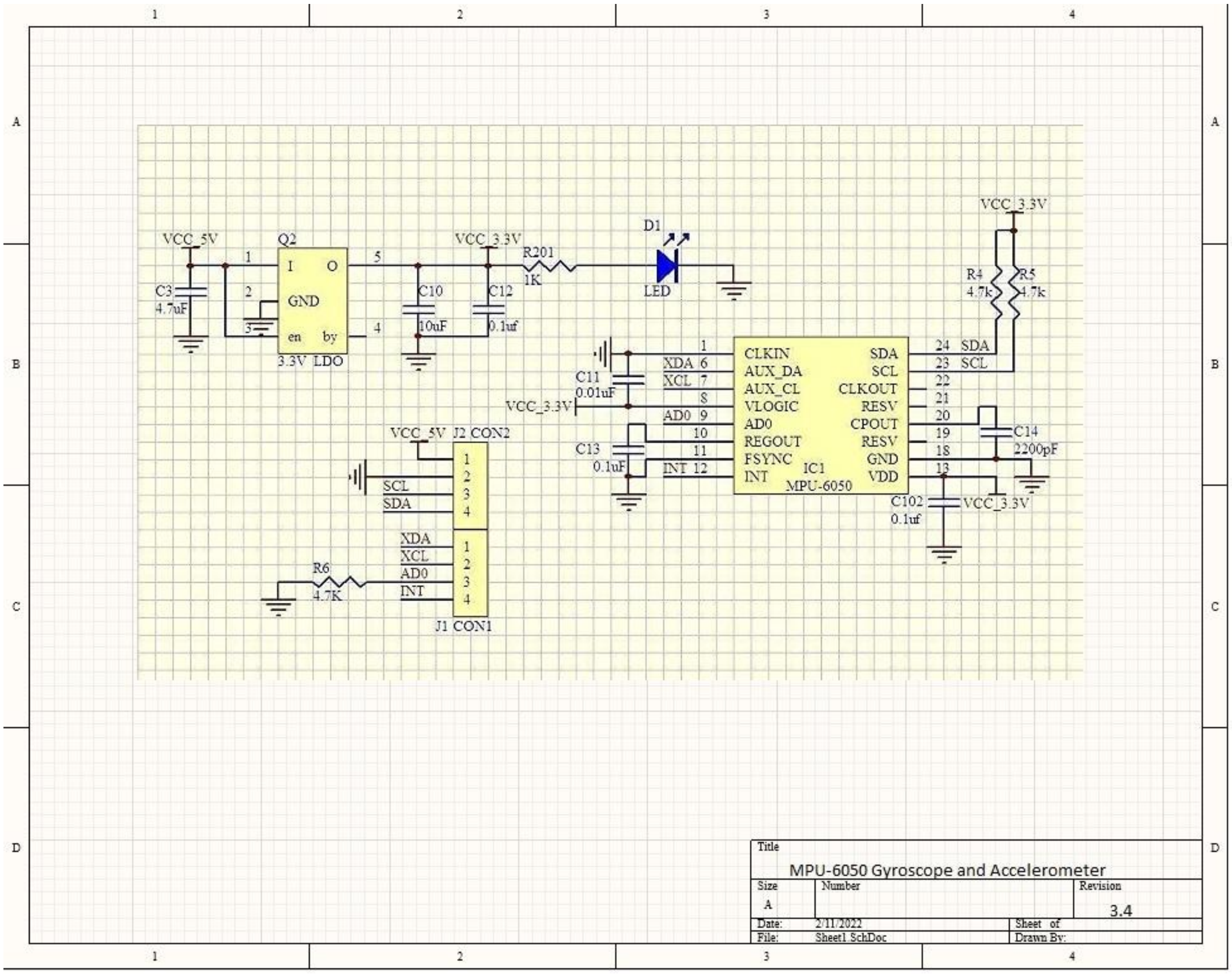


Figure 3: Design Schematic

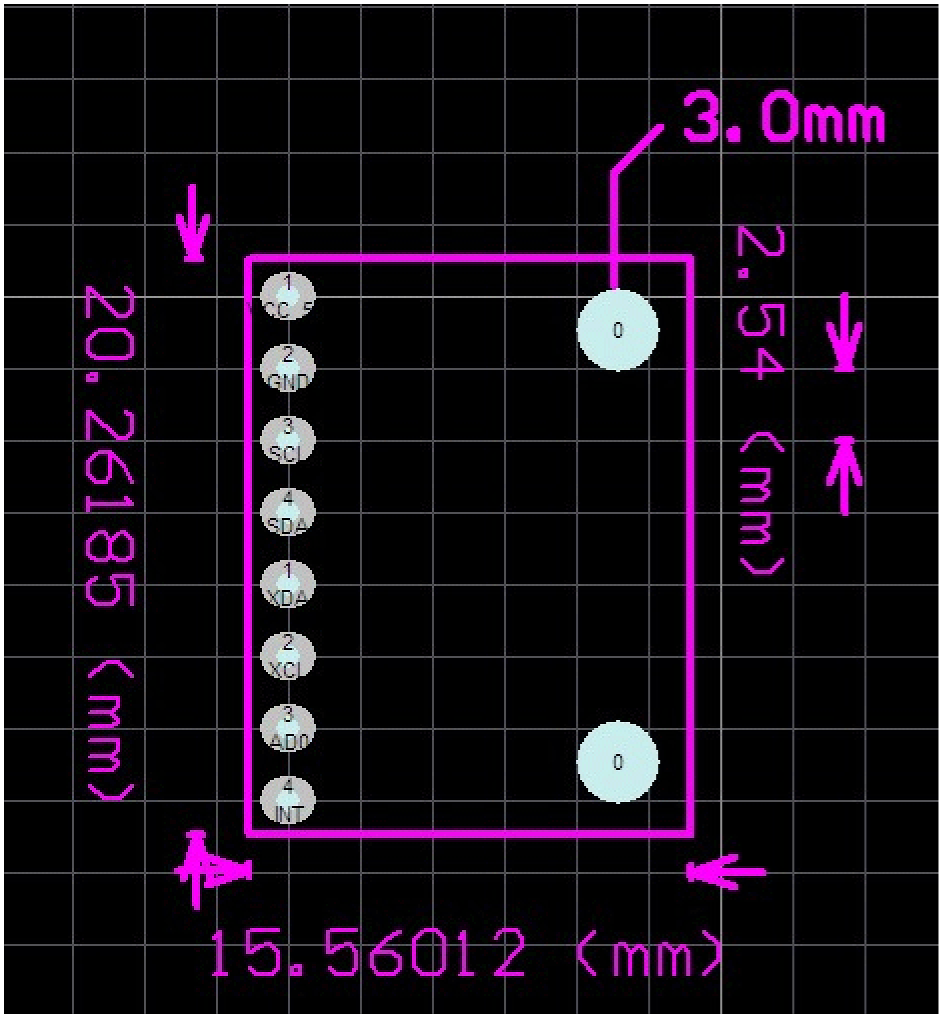


Figure 4: Mechanical Drawing

```
/*
Author:  Taylor Cole Brennan
Date:    February 11, 2022
Project: Auto Brake Lights, Bike project
Class:   Junior Design 2
*/
//inputs
float left_right = A0; //insert equation to get "prettier numbers, will need manual
calibration
float forward_back = A1; //insert equation to get "prettier numbers, will need manual
calibration
float accel = A3; //insert equation to get "prettier numbers, will need manual
calibration
int left_turn = 2;
int right_turn = 3;
//outputs
int left_turn_led = 4; //digital output, indicator light
int right_turn_led = 5; //digital output, indicator light
```

```
int left_tail_light = 6; //digital output, tail light
int right_tail_light = 7; //digital output, tail light
//global variables
float brake_start = 100; //dummy value need to be calc.
float left = -10.0;
float right = 10.0;
float forward = -10.0;
float backward = 10.0;
int left_hold, right_hold, left_dummy, right_dummy;

void setup() {
  // put your setup code here, to run once:
  pinMode(left_turn, INPUT); //digital
  pinMode(right_turn, INPUT); //digital
  pinMode(left_turn_led, OUTPUT); //digital
  pinMode(right_turn_led, OUTPUT); //digital
  pinMode(left_tail_light, OUTPUT); //digital
  pinMode(right_tail_light, OUTPUT); //digital

  Serial.begin(9600);
}

void loop() {
                                              //check values
  float brake_eq = brake/2;
  float turn = analogRead(left_right);
  float wheelee = analogRead(forward_back);
  float brake = analogRead(accel);
  int signal_left = digitalRead(left_turn);
  int signal_right = digitalRead(right_turn);
                                              //check turn signals
  if (signal_left) {left_hold=1; right_hold=0; left_dummy=0;
digitalWrite(left_turn_led, HIGH);}
  else if (signal_right) {left_hold=0; right_hold=1; right_dummy = 0;
digitalWrite(right_turn_led, HIGH);}
                                              //check if brake
  if (brake < brake_start){
      if (left_hold) brake_left(brake_eq);
      else if (right_hold) brake_right(brake_eq);
      else brake_lights(brake_eq);
  }                                           //no brake? Check if turn
  else if (left_hold) turn_left();
  else if (right_hold) turn_right();
                                              //check if turned fully
  if (left_hold || right_hold){
      if (turn < left && left_hold) left_dummy =1;
      else if (turn > right && right_hold) right_dummy =1;
```

```
  }                                            //check if straightened out
  if ( turn > left && left_dummy) {left_hold =0; digitalWrite(left_turn_led, LOW);}
  else if ( turn < right && right_dummy) {right_hold =0; digitalWrite(right_turn_led,
LOW);}
}

void brake_lights(float brake){                //brake only

       digitalWrite(left_tail_light, HIGH);
       digitalWrite(right_tail_light, HIGH);
       delay(500/brake_eq);
       digitalWrite(right_tail_light, LOW);
       digitalWrite(left_tail_light, LOW);
       delay(500/brake_eq);
}
void brake_left(float brake) {                 //brake and left turn
       digitalWrite(right_tail_light, HIGH);
                                      //1 sec of brake lights
       for (int i=0; i<=brake; i++){
       digitalWrite(left_tail_light, HIGH);
       delay(500/brake);
       digitalWrite(left_tail_light, LOW);
       delay(500/brake);   }
                                      //turn off turn signal
       digitalWrite(right_tail_light, LOW);
       //repeat loop for another sec
       for (int i=0; i<=brake; i++){
       digitalWrite(left_tail_light, HIGH);
       delay(500/brake);
       digitalWrite(left_tail_light, LOW);
       delay(500/brake);   }
}
void brake_right(float brake) {                //brake and right turn
       digitalWrite(left_tail_light, HIGH);
                                      //1 sec of brake lights
       for (int i=0; i<=brake_eq; i++){
       digitalWrite(right_tail_light, HIGH);
       delay(500/brake_eq);
       digitalWrite(right_tail_light, LOW);
       delay(500/brake_eq);       }
                                      //turn off turn signal
       digitalWrite(left_tail_light, LOW);
       //repeat loop for another sec
       for (int i=0; i<=brake_eq; i++){
       digitalWrite(right_tail_light, HIGH);
       delay(500/brake_eq);
       digitalWrite(right_tail_light, LOW);
```

```
        delay(500/brake_eq);       }
}
void turn_left() {                             //left turn only
        digitalWrite(left_tail_light, HIGH);
        delay(500);
        digitalWrite(left_tail_light, LOW);
        delay(500);

}
void turn_right() {                            //right turn only
        digitalWrite(right_tail_light, HIGH);
        delay(500);
        digitalWrite(right_tail_light, LOW);
        delay(500);

}
```

Figure 5: Code

Taylor Cole Brennan
Junior Design 2
Jan 28, 2022

Block 1: Microcontroller

This is to document the microcontroller block for the Junior design 2 bike project. The microcontroller will take in 5 inputs (3 analog, and 2 digital) and will have 4 outputs. How it's coded, the brakes will activate after hitting a minimum deceleration value, based on the accelerometer input. Based on that value, the High and low value for the brake lights are divided causing a pulse width modulation. This division is taken into account when timing the turn signals as well. The turn signals are set to 1sec alterations. The turn signals need to be activated by user control, but after being activated, the turn signals won't turn off until the bike has turned so much (minimum turn amount) in order to detect early turn signallers. The microcontroller delivers outputs to the turn signal indicators as well as the tail lights.

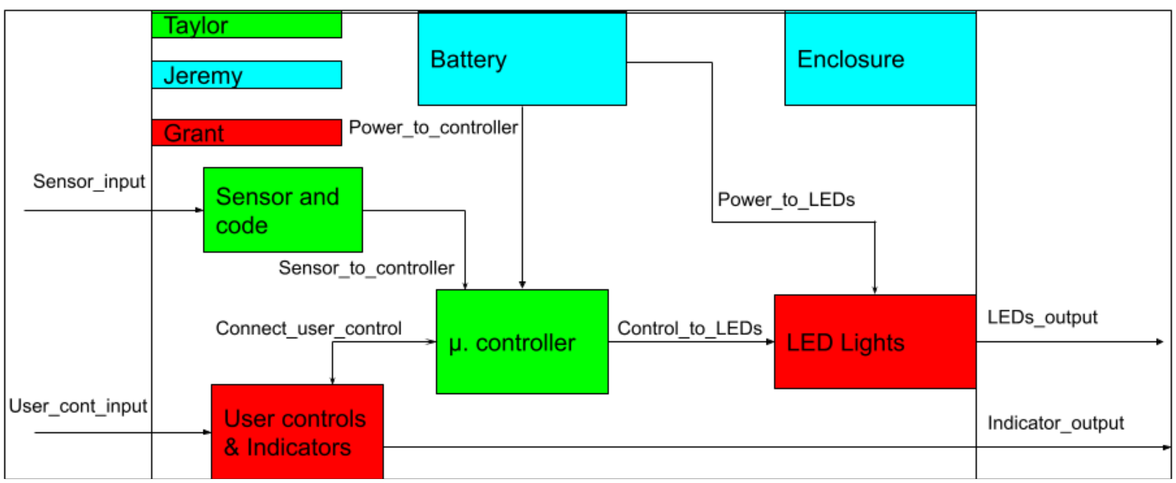


Figure 1: Top Level Diagram

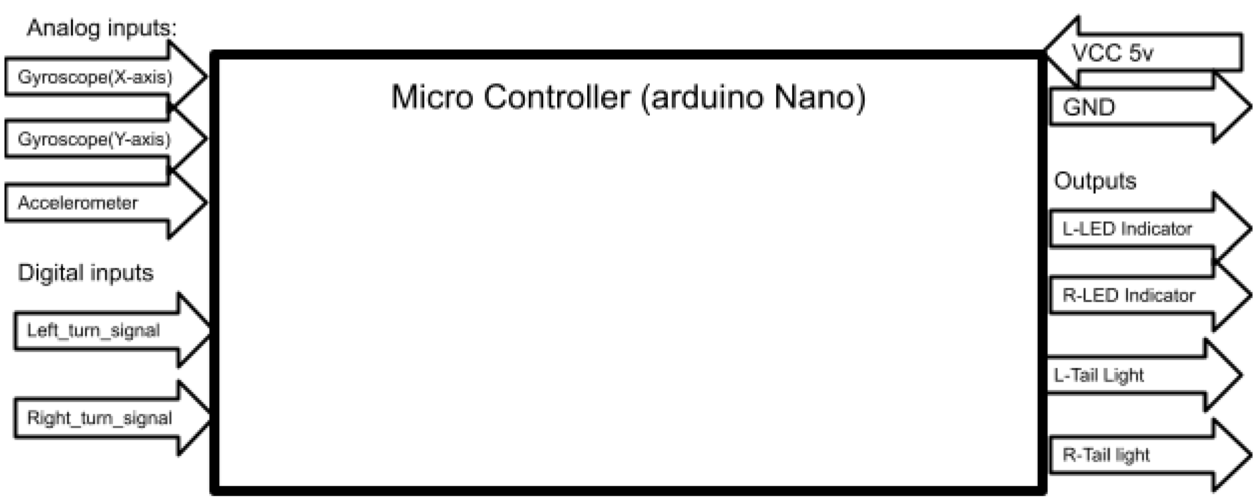


Figure 2: Black Box Diagram

Interface Name	Specifications:
Gyroscope_x_axis	scale range: ±250, ±500, ±1000, and ±2000°/sec Standby current: 5μA Gyroscope operating current: 3.6mA
Gyroscope_y_axis	scale range: ±250, ±500, ±1000, and ±2000°/sec Standby current: 5μA Gyroscope operating current: 3.6mA
Accelerometer	scale range: ±2g, ±4g, ±8g and ±16g normal operating current: 500μA Low power current: 10μA (1.25Hz), 20μA (5Hz), 60μA (20Hz), 110μA (40Hz)
left_turn_signal	V_{active}= 5 [V] A_{active}= 0.5 [mA] $V_{inactive}$= 0 [V] $A_{inactive}$= 0.0 [mA]
right_turn_signal	V_{active}= 5 [V] A_{active}= 0.5 [mA] $V_{inactive}$= 0 [V] $A_{inactive}$= 0.0 [mA]
Vcc	V_{norm}= 5 [V] A_{norm}= 19 [mA]
Gnd	V_{norm}= 0 [V] A_{norm}= 1 [mA]
L_LED_indicator	V_{active}= 5 [V] A_{active}= 0.3 [mA] $V_{inactive}$= 0 [V] $A_{inactive}$= 0.0 [mA]
R_LED_indicator	V_{active}= 5 [V] A_{active}= 0.3 [mA] $V_{inactive}$= 0 [V]

	$A_{inactive}$ = 0.0 [mA]
L_tail_light	V_{active} = 5 [V] A_{active} = 0.5 [mA] $V_{inactive}$ = 0 [V] $A_{inactive}$ = 0.0 [mA]
R_tail_light	V_{active} = 5 [V] A_{active} = 0.5 [mA] $V_{inactive}$ = 0 [V] $A_{inactive}$ = 0.0 [mA]

Table 1: Interface Specification

Ref #	Qty.	Description	Item Model #	Manufacturer	Link	Price
1A	1	Arduino Nano V3.0 ATmega328P	8541582500	LAFVIN	Amazon	$6.66 (3:$20)

Table 2: Parts Price List

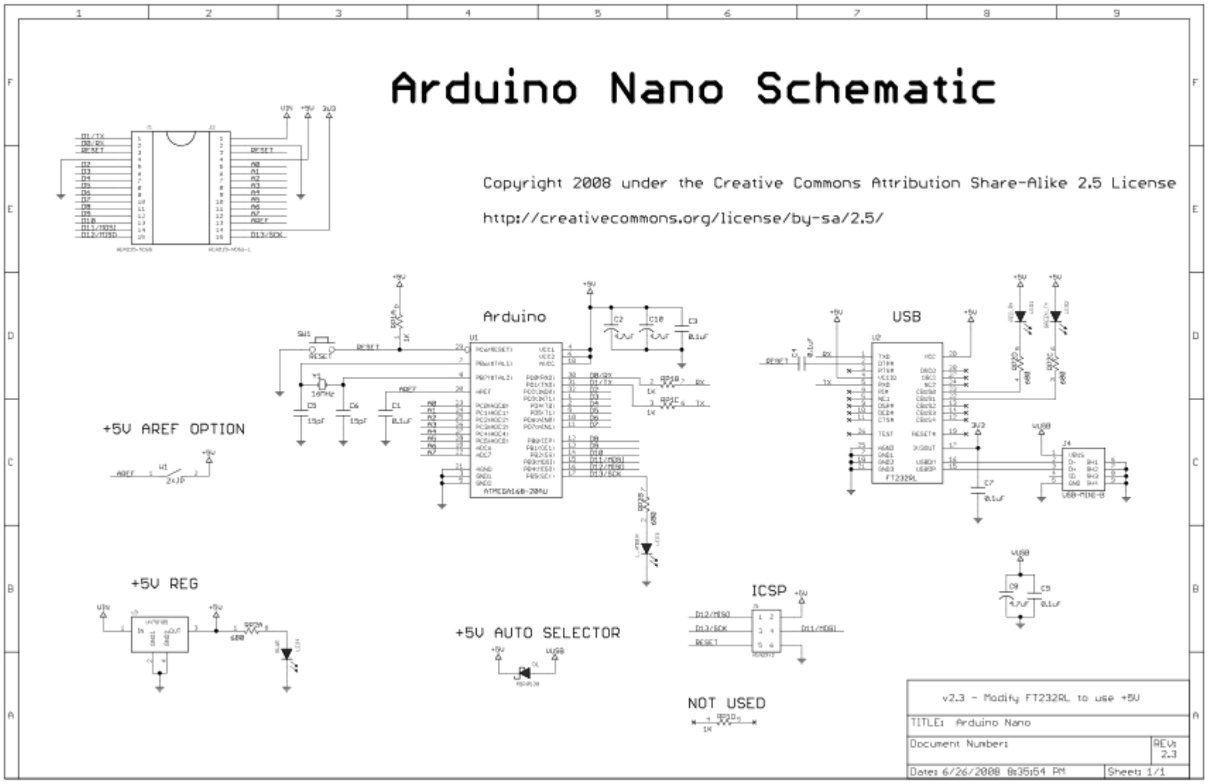


Figure 3: Design Schematic

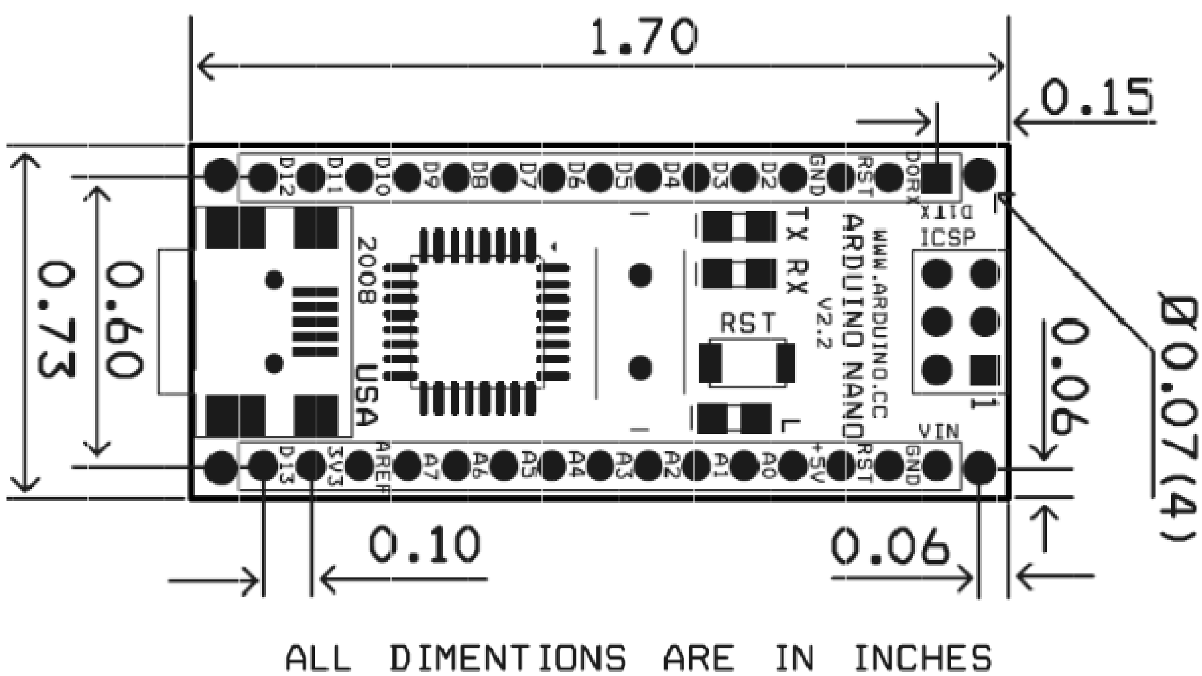


Figure 4: Mechanical Drawing

/*

```
Author:  Taylor Cole Brennan
Date:    January 28, 2022
Project: Auto Brake Lights, Bike project
Class:   Junior Design 2
*/

float left_right = A0; //insert equation to get "prettier numbers, will need manual
calibration

float forward_back = A1; //insert equation to get "prettier numbers, will need manual
calibration

//* float up_down = A2; commented out, most likely only need the two above*//

float accel = A3; //insert equation to get "prettier numbers, will need manual
calibration

int left_turn = digitalRead(1);

int right_turn = digitalRead(2);



int left_turn_led = 3; //digital output, indicator light

int right_turn_led = 4; //digital output, indicator light

int left_tail_light = 5; //digital output, tail light

int right_tail_light = 6; //digital output, tail light

float brake_start = 100; //dummy value need to be calc.


void setup() {

  // put your setup code here, to run once:

  pinMode(left_turn_led, OUTPUT); //digital

  pinMode(right_turn_led, OUTPUT); //digital

  pinMode(left_tail_light, OUTPUT); //digital

  pinMode(right_tail_light, OUTPUT); //digital

  Serial.begin(9600);

}
```

```
void loop() {

  float turn = analogRead(left_right);

  float wheelee = analogRead(forward_back);

  float brake = analogRead(accel);

  int signal_left = digitalRead(left_turn);

  int signal_right = digitalRead(right_turn);

  float brake_eq = brake / 2; //equation to determine amount of time for volts to LED

  //values to set parameters on turns, decelleration, tricks?

  float out = 0.0;

  float left = -10.0;

  float right = 10.0;

  float up = 10.0;

  float down = -10.0;

  int left_hold, right_hold;

  //check if the bike is slowing down

  if (brake < brake_start){

       do {

       //check if there is a turn signal

       if (signal_left) left_hold = 1;

       else if (signal_right) right_hold = 1;

       //right turn

       if (right_hold){

       //turn on led indicator, turn on turn light for 1 sec

        digitalWrite(right_turn_led, HIGH);
```

```
 digitalWrite(left_tail_light, HIGH);

//1 sec of brake lights

for (int i=0; i<=brake_eq; i++){

 digitalWrite(right_tail_light, HIGH);

 delay(1000/brake_eq);

 digitalWrite(right_tail_light, LOW);

 delay(1000/brake_eq);

}

//turn off turn signal

 digitalWrite(left_tail_light, LOW);

 //repeat loop for another sec

for (int i=0; i<=brake_eq; i++){

 digitalWrite(right_tail_light, HIGH);

 delay(1000/brake_eq);

 digitalWrite(right_tail_light, LOW);

 delay(1000/brake_eq);

}}


else if (left_hold){

//turn on led indicator, turn on turn light for 1 sec

 digitalWrite(left_turn_led, HIGH);

 digitalWrite(right_tail_light, HIGH);

//1 sec of brake lights

for (int i=0; i<=brake_eq; i++){

 digitalWrite(left_tail_light, HIGH);
```

```
        delay(1000/brake_eq);

        digitalWrite(left_tail_light, LOW);

        delay(1000/brake_eq);

       }      //turn off turn signal

        digitalWrite(right_tail_light, LOW);

        //repeat loop for another sec

       for (int i=0; i<=brake_eq; i++){

        digitalWrite(left_tail_light, HIGH);

        delay(1000/brake_eq);

        digitalWrite(left_tail_light, LOW);

        delay(1000/brake_eq);

       }}

       //no turn, only do the brake light calc

       else {

        digitalWrite(left_tail_light, HIGH);

        digitalWrite(right_tail_light, HIGH);

        delay(1000/brake_eq);

        digitalWrite(right_tail_light, LOW);

        digitalWrite(left_tail_light, LOW);

        delay(1000/brake_eq);

       }

       }while (brake < brake_start); //end loop if no longer slowing down, reset hold
values

       left_hold = 0;

    right_hold=0;

       }
```

```
 else if (signal_left){

      do {

      //set led indicator

      digitalWrite(left_turn_led, HIGH);

      left_hold = 1;

      int left_dummy =0;

      if (turn < left) left_dummy=1;

      //check if we have made a turn by passing a certain value, and 1 sec delay
between on/off values

      digitalWrite(left_tail_light, HIGH);

      delay(1000);

      digitalWrite(left_tail_light, LOW);

      delay(1000);

      //if we have turned so far, and are no long in the left turn range, set hold to
0 to end loop

      if (turn > left && left_dummy==true) left_hold=0;

      }while (left_hold);

      }

 else if (signal_right){

      do {

            //set led indicator

      digitalWrite(right_turn_led, HIGH);

      right_hold = 1;

      int right_dummy =0;

      if (turn > right) right_dummy=1;

          //check if we have made a turn by passing a certain value, and 1 sec delay
between on/off values
```

```
      digitalWrite(right_tail_light, HIGH);

      delay(1000);

      digitalWrite(right_tail_light, LOW);

      delay(1000);

       if (turn < right && right_dummy==true) right_hold=0;

            //if we have turned so far, and are no long in the left turn range, set
hold to 0 to end loop

       }while (left_hold);

 }
}
```

Figure 5: Code

```
/*
Author:  Taylor Cole Brennan
Date:    February 6, 2022
Project: Microcontroller test code
Class:   Junior Design 2

*/

float left_right = A0; //insert equation to get "prettier numbers, will need manual
calibration

float accel = A1; //insert equation to get "prettier numbers, will need manual
calibration

int left_turn_led = 2; //digital output, indicator light

int left_turn = 3;



void setup() {

  // put your setup code here, to run once:

  pinMode(left_turn_led, OUTPUT); //digital

  Serial.begin(9600);
```

```
}


void loop() {

  // put your main code here, to run repeatedly:

  float turn = analogRead(left_right);

  float brake = analogRead(accel);

  int signal_left =0;

  signal_left= digitalRead(left_turn);

if (brake < 300) Serial.print("Slowing Down");

else if (brake > 700) Serial.print("Speeding Up");

  Serial.print(brake);

if (turn < 300) Serial.print("Left");

else if (turn > 700) Serial.print("Right");

Serial.print(turn);

if (signal_left) {Serial.print("Turn Signal"); digitalWrite(left_turn_led, HIGH);

                    delay(3000); digitalWrite(left_turn_led, LOW);}

  Serial.print(brake);

}
```

Figure 6: Test Code for IO Validation

[YouTube Link]

Video 1: IO Validation

Since the microcontroller takes in both digital and analog inputs, the usage of multiple analog input and a single digital input is demonstrated in the video. The system only has digital outputs in the form of LED lights, so the use of a single LED light with switch control is demonstrated.

The inputs are also translated on the Serial Monitor to display the turning value and acceleration value.

Bike 8 Block Diagram Check - Jeremy Xu - Battery

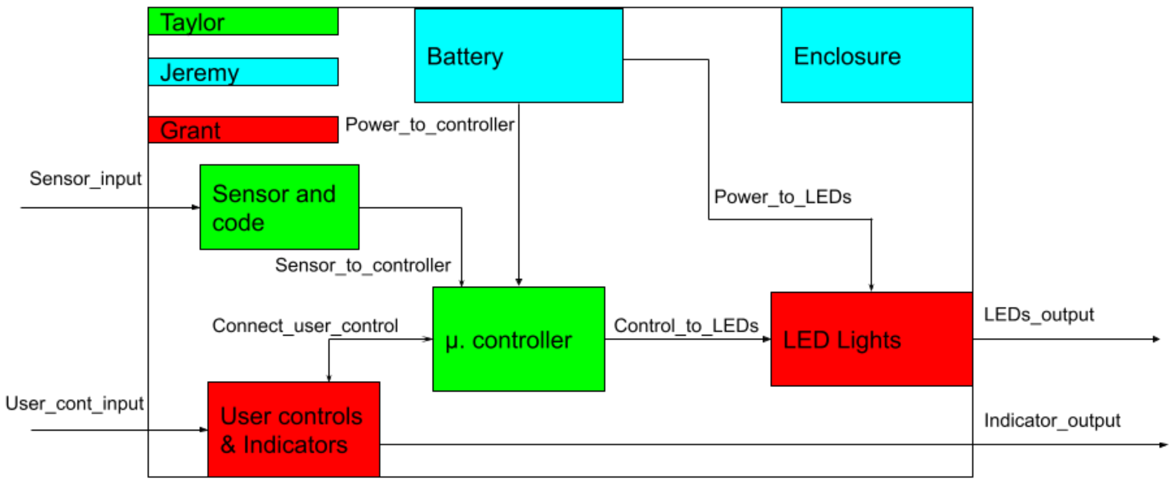


Figure 1: Top-level Diagram

This block diagram submission focuses on the battery for the Bike Turn Signal system. The battery that we are using is a 3000mAh lithium-ion battery that can supply 5V through its USB port and a maximum of 12V through its barrel jack. It also receives 12V through the same barrel jack to charge.

The battery is stored in a waterproof case and can be removed from the assembly to facilitate charging.

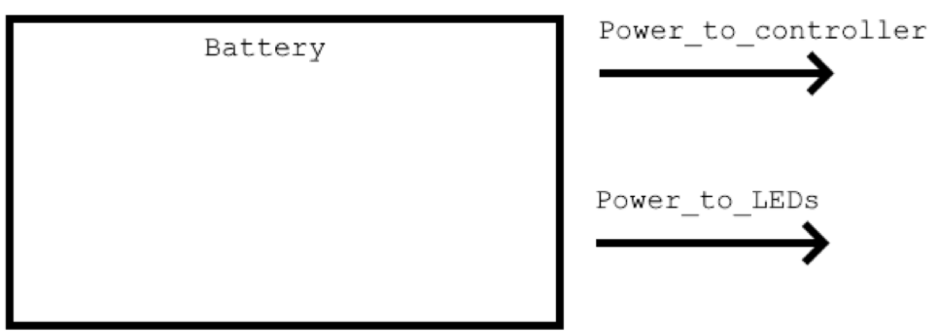


Figure 2: Black-box diagram

Battery specs	**Value**
Maximum V_{out}	$12V/5V$
Capacity	$3000\ mAh$
Dimensions	$105mm \times 64mm \times 24mm$

Table 1: Battery specifications

Interface Name	**Specifications**
`Power_to_controller` ($V_{cc,controller}$; `GND`)	$V_{controller,max} = 5V$ $A_{controller,max} \approx 19mA$ `GND` $= 0V$
`Power_to_LEDs` ($V_{cc,LED}$; `GND`)	$V_{LED,max} = 5V$ $A_{controller,max} \approx 42.54mA$ `GND` $= 0V$

Table 2: Interface specifications

Interface Name	Specifications:
Left Button	Open / Shorted circuit
Right Button	Open / Shorted circuit
L-LED (User output)	On / Off (Could blink with indicators)
R-LED (User output)	On / Off (Could blink with indicators)
VCC	V_{norm} = 5 [V] A_{norm} = 0.0 - 0.5 [mA]
L-LED	V_{active} = 2 [V] A_{active} = 0.3 [mA] $V_{inactive}$ = 0 [V] $A_{inactive}$ = 0.0 [mA]
R-LED	V_{active} = 2 [V] A_{active} = 0.3 [mA]

	$V_{inactive}$ = 0 [V] $A_{inactive}$ = 0.0 [mA]
L-Button Out	V_{active} = 5 [V] A_{active} = 0.5 [mA] $V_{inactive}$ = 0 [V] $A_{inactive}$ = 0.0 [mA]
R-Button Out	V_{active} = 5 [V] A_{active} = 0.5 [mA] $V_{inactive}$ = 0 [V] $A_{inactive}$ = 0.0 [mA]
GND	V_{norm} = 0 [V] A_{norm} = 1 [mA]

Table 3: Full interface specifications

Bill of Materials	**Cost**
TalentCell Rechargeable 12V 3000mAh Lithium ion Battery Pack	$28.79
ABS Plastic Dustproof Waterproof IP65 Junction Box Hinged Shell Universal Electrical Project Enclosure	$13.99
3/8" | 50ft Split Loom Tube	$10.99
100pcs Cable Zip Ties Heavy Duty 8 Inch	$5.49
Hot Glue Gun and Sticks, 30	$12.49
200 Pieces M4 Machine Screws and Nuts	$10.99
Total	**$82.74**

Table 4: Bill of Materials

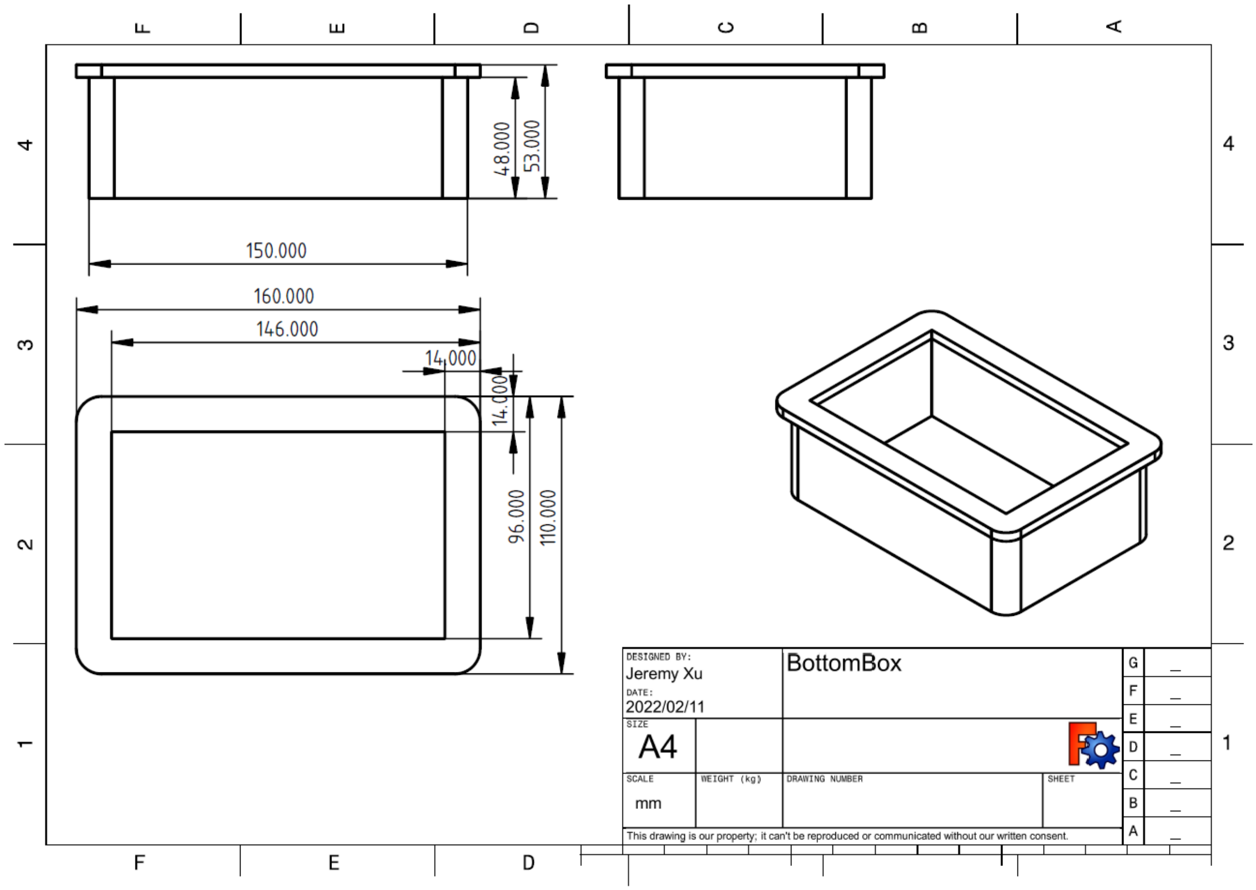


Figure 3: Battery enclosure, bottom

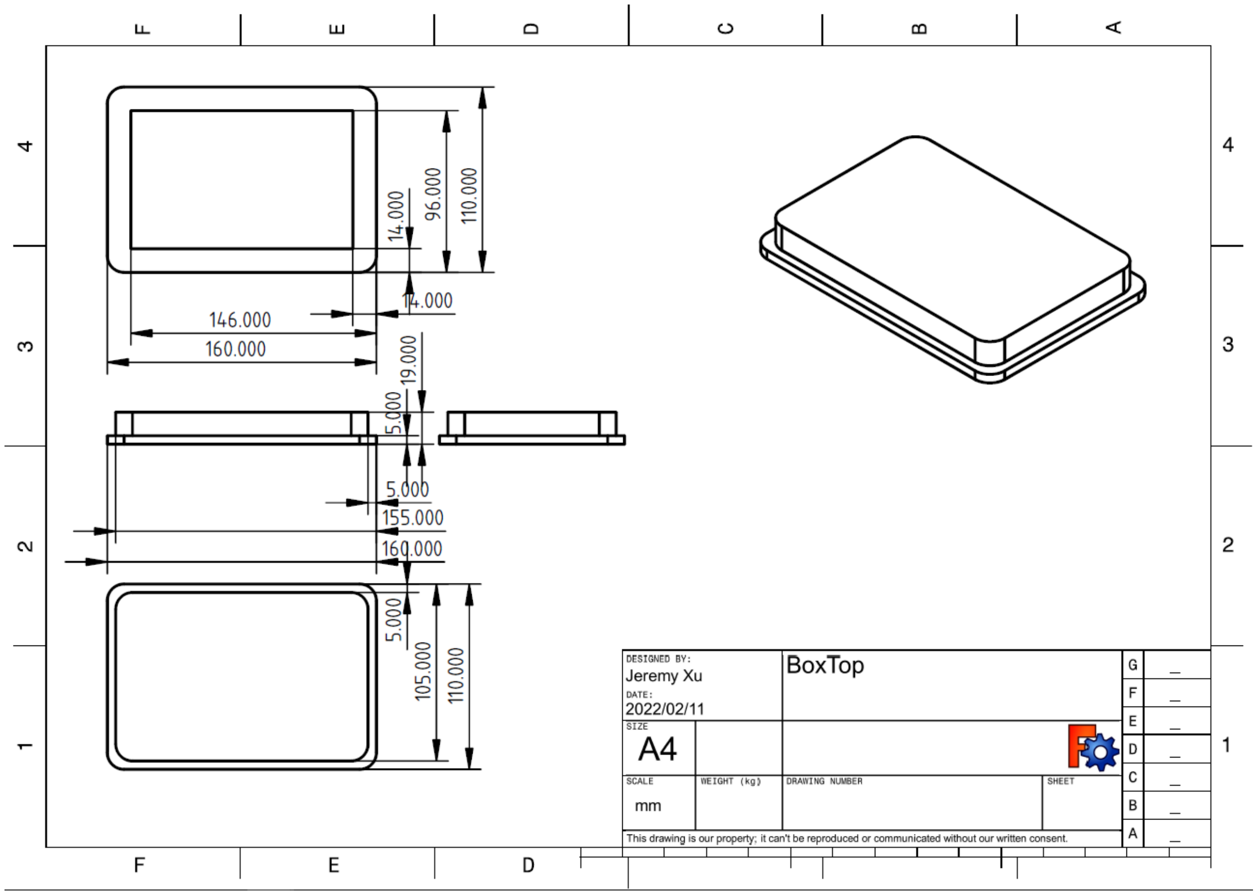
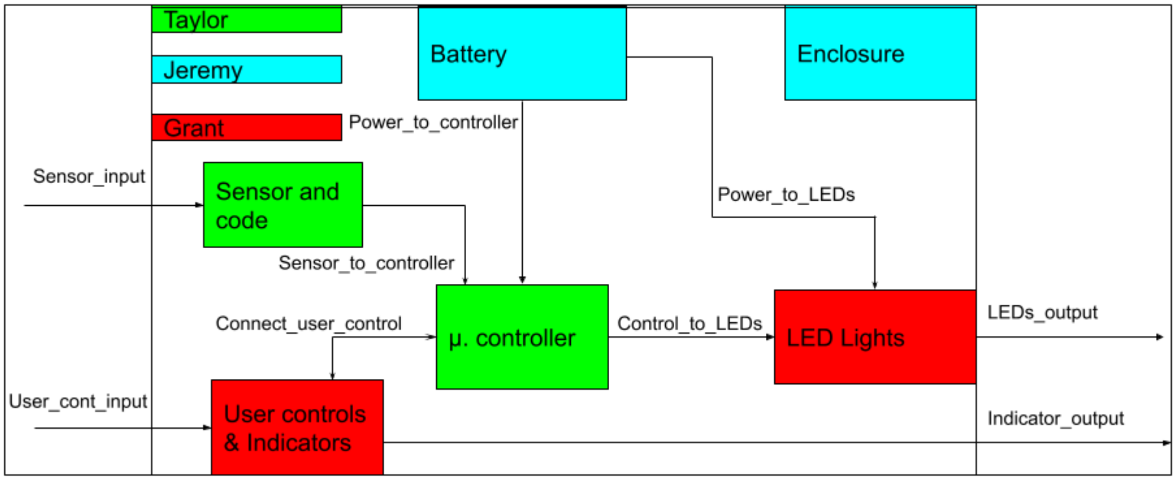


Figure 4: Battery enclosure, top

Bike 8 Block Diagram Check - Jeremy Xu - Enclosure



This block diagram submission focuses on the enclosure for the Bike Turn Signal system. Due to the lack of electrical components, there are no interfaces. Physical specifications have been provided instead. Technical drawings of the components have been provided on the following page and as a separate attachment to the submitted Canvas assignment.

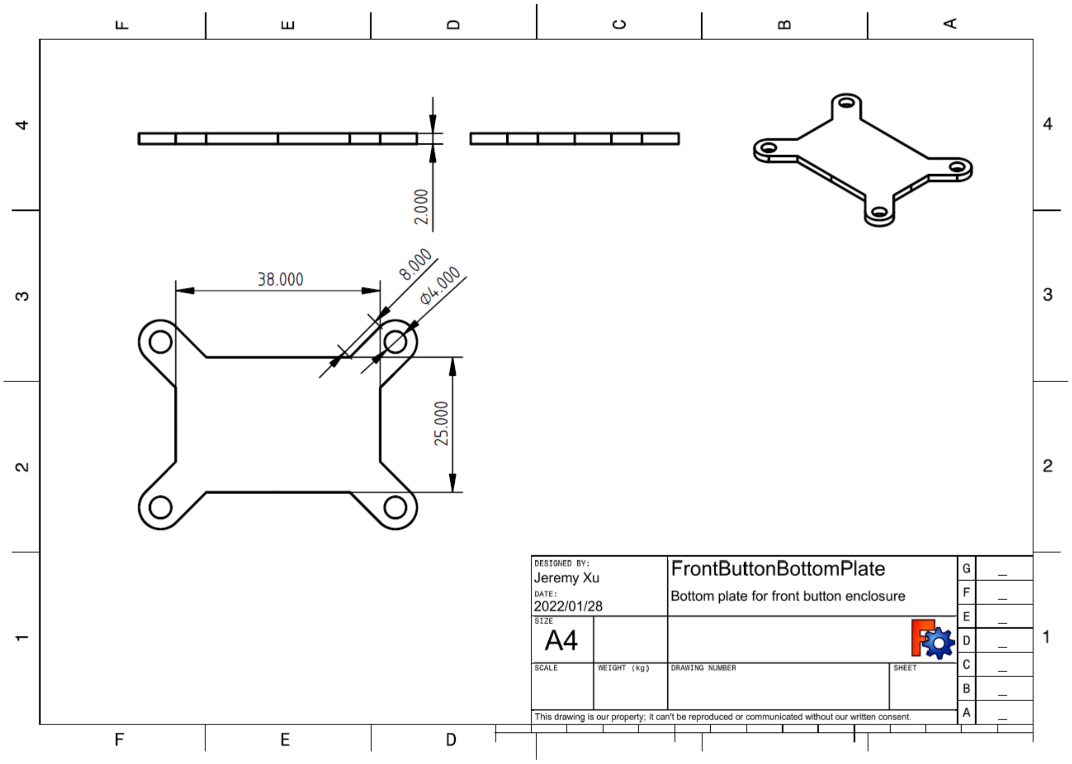
Front Button Enclosure Specs	**Dimensions**
Bottom Plate Primary Dimensions	38mm x 25mm x 2mm
Top Plate Primary Dimensions	38mm x 25mm x 6mm
Mounting Hardware	4x M4 x 12mm bolts, 4x M4 x 3.2mm nuts

Bill of Materials	Cost
4x 6-32 machine screws/nuts	$1.28
Total	**$1.28**

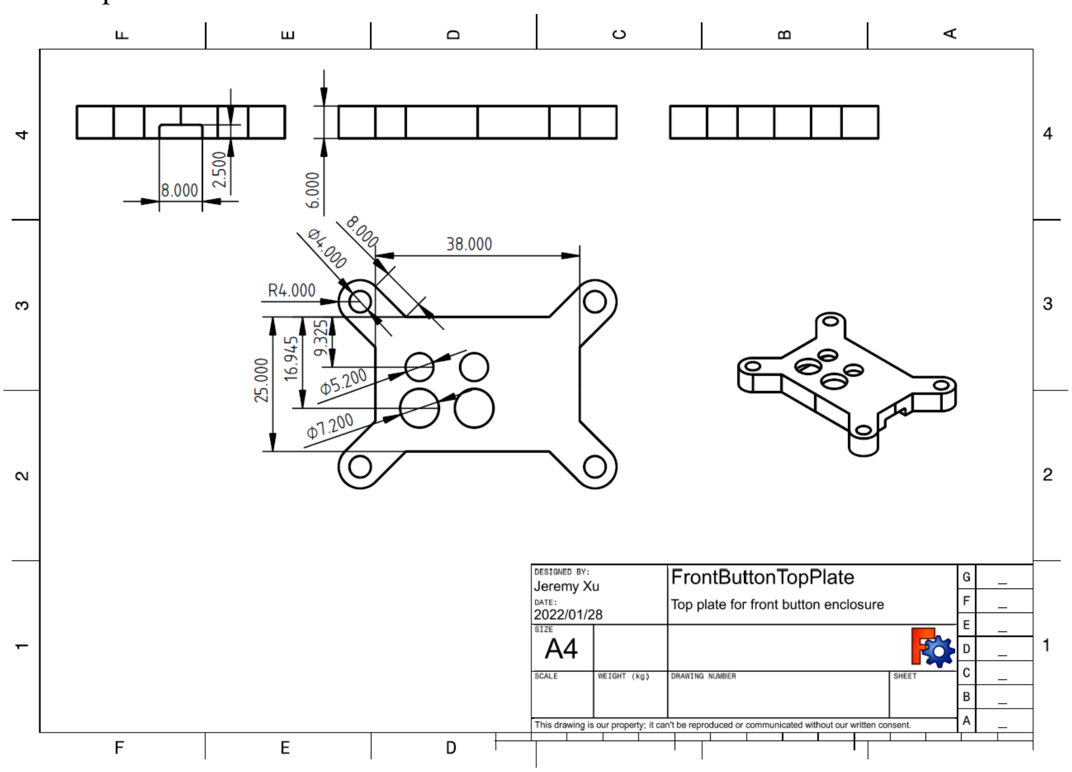
Interface Name	Specifications:
Left Button	Open / Shorted circuit
Right Button	Open / Shorted circuit
L-LED (User output)	On / Off (Could blink with indicators)
R-LED (User output)	On / Off (Could blink with indicators)
VCC	V_{norm} = 5 [V] A_{norm} = 0.0 - 0.5 [mA]
L-LED	V_{active} = 2 [V] A_{active} = 0.3 [mA] $V_{inactive}$ = 0 [V] $A_{inactive}$ = 0.0 [mA]
R-LED	V_{active} = 2 [V] A_{active} = 0.3 [mA] $V_{inactive}$ = 0 [V] $A_{inactive}$ = 0.0 [mA]
L-Button Out	V_{active} = 5 [V] A_{active} = 0.5 [mA] $V_{inactive}$ = 0 [V] $A_{inactive}$ = 0.0 [mA]
R-Button Out	V_{active} = 5 [V] A_{active} = 0.5 [mA] $V_{inactive}$ = 0 [V] $A_{inactive}$ = 0.0 [mA]
GND	V_{norm} = 0 [V] A_{norm} = 1 [mA]

Table 1: Interface Specifications

FrontButtonBottomPlate



FrontButtonTopPlate



Grant Everson
Junior Design 2
February 11, 2022

This documentation includes information on the circuit that will allow the high-power LED indicators to be controlled by an arduino. This circuit is designed to be controlled with pulse-width modulation in mind. This will allow the brightness of the LEDs to be variable with other factors, such as braking harder on the bike. The mosfets will allow the microcontroller to control a larger amount of current than is able to be output from the digital pins. The IRF520N was selected due to its ability to be turned off and on at a rapid pace along with its gate requiring 4v to turn on, which the 5v pins of the microcontroller can handle. The current design does not necessarily need this as the current through the LEDs is not at its maximum, but with lower resistor values this could be achieved.

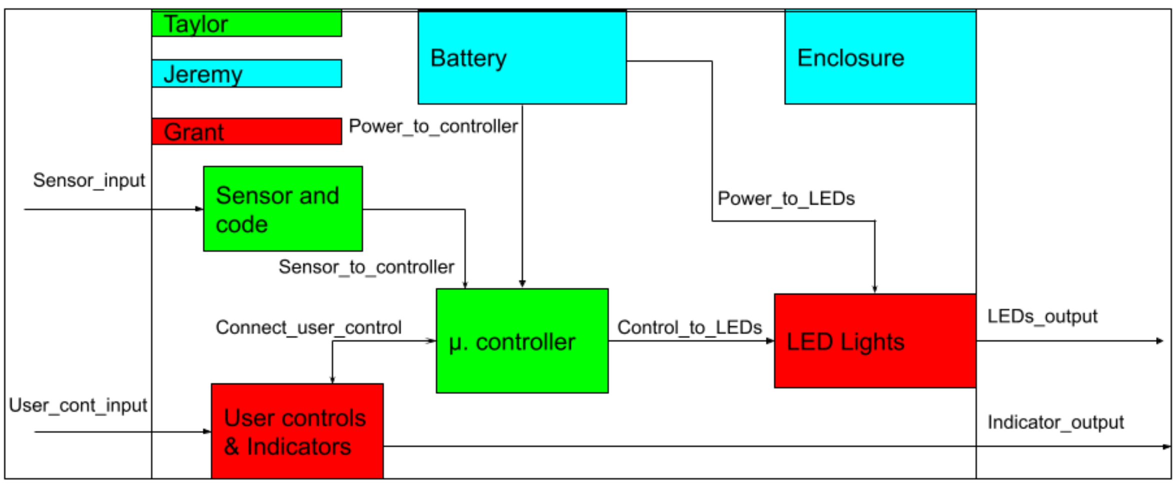


Figure 1: Top Level Diagram

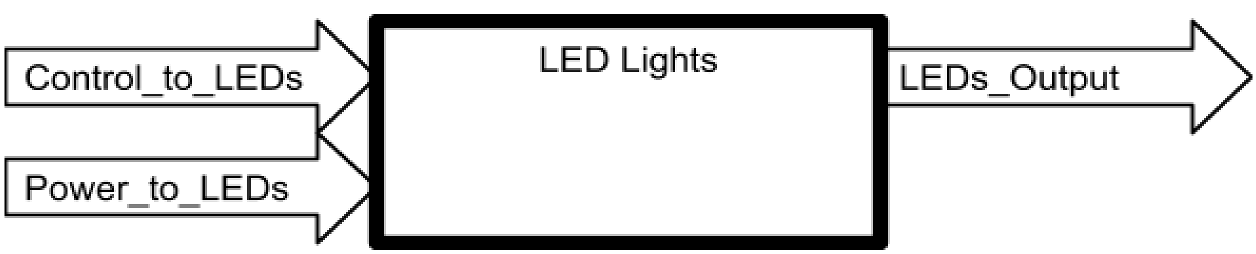


Figure 2: Black Box Diagram

Interface Name:	Sub Interface:	Specifications:
Control_to_LEDs	Arduino_In_Left	V_{Active} = 5 [V] I_{Active} = 100 [nA] $V_{Inactive}$ = 0 [V] $I_{Inactive}$ = 0.0 [A]

	Arduino_In_Right	V_{Active} = 5 [V] I_{Active} = 100 [nA] $V_{Inactive}$ = 0 [V] $I_{Inactive}$ = 0.0 [A]
Power_to_LEDs	VCC	V_{Nom} = 5 [V] I_{Nom} = 42.54 [mA] (This is 21.27 mA per LED) I_{max} = 350 [mA] (per LED)
	GND	V_{Nom} = 0 [V] I_{Nom} = 42.54 [mA] (This is 21.27 mA per LED) I_{max} = 350 [mA] (per LED)
LEDs_Output	DL LED	Visual: Active = On Inactive = Off
	DR LED	Visual: Active = On Inactive = Off

Table 1: Interface Specifications

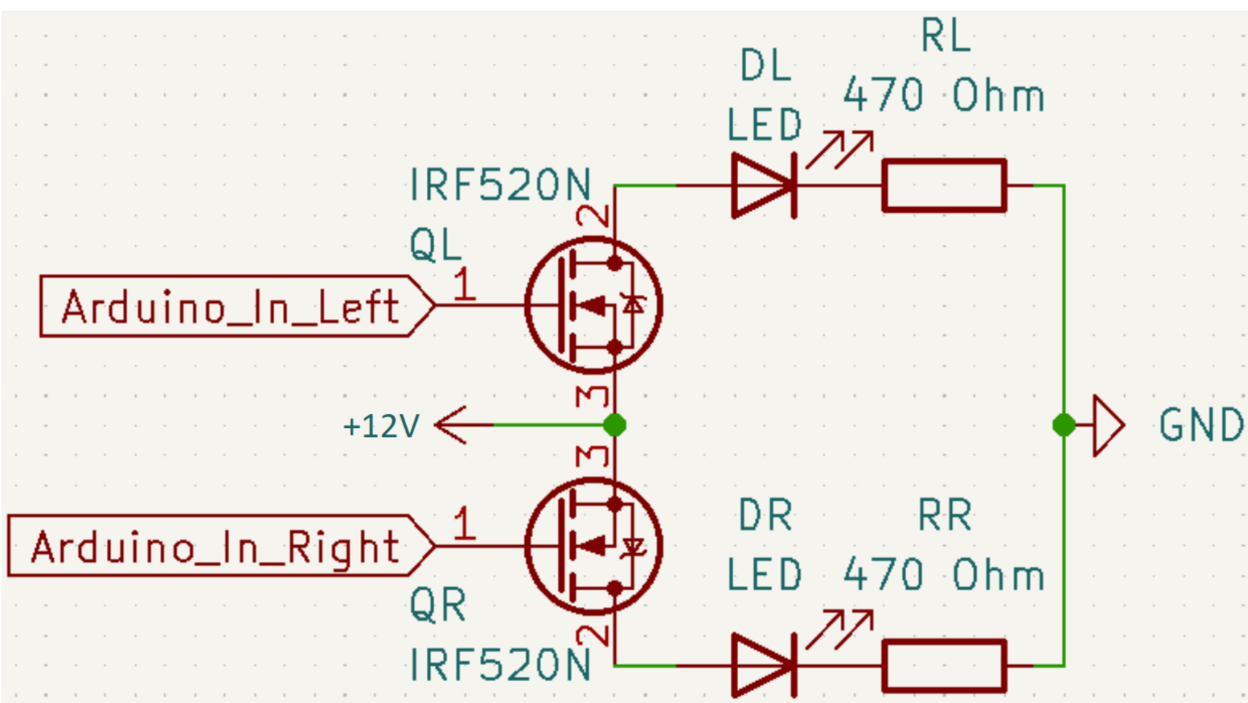


Figure 3: Arduino controlled LEDs Schematic

*Note 1: Both Arduino_In_Right and Arduino_In_Left are equivalent to Control_to_LEDs as this is actually a 2 wide bus. These arduino controls are +5v or 0v as they are digital.
*Note 2: Similar to above, the +12V and GND are equivalent to Power_to_LEDs as delivering power requires both hot and ground.

Quantity	Designator	Part	Value	Price (individual)	Manufacturer	Distributor
2	QL QR	Mosfet	IRF520N	$0.38	BOJACK	https://www.amazon.com/dp/B082J3F8HJ/
2	DL DR	LED	2.2 V (Typical) 350 mA (max) Color: White	$0.90	LEDGUHON	https://www.amazon.com/dp/B091C36CQN/
2	RL RR	Resistor	470 Ohm	$0.04	EDGELEC	https://www.amazon.com/dp/B07QG1V4BH/
Total Cost:				$1.32		

Table 2: Bill of Materials

Additional note:
Assisted with the following components:
- Battery selection & interfacing

Extra:
Waterproof enclosure for battery, sensors, and microcontroller. Jeremy was able to recreate the schematic for the waterproof box that we got from LMioEtool. This box is able to fit the battery, arduino, sensor, and the majority of wiring needed for the front controls and rear-indicators.This box will need to be modified to allow wires to go to and from the front controls and rear-indicators but this can be done simply by drilling a hole and sealing it up again with waterproof sealant.

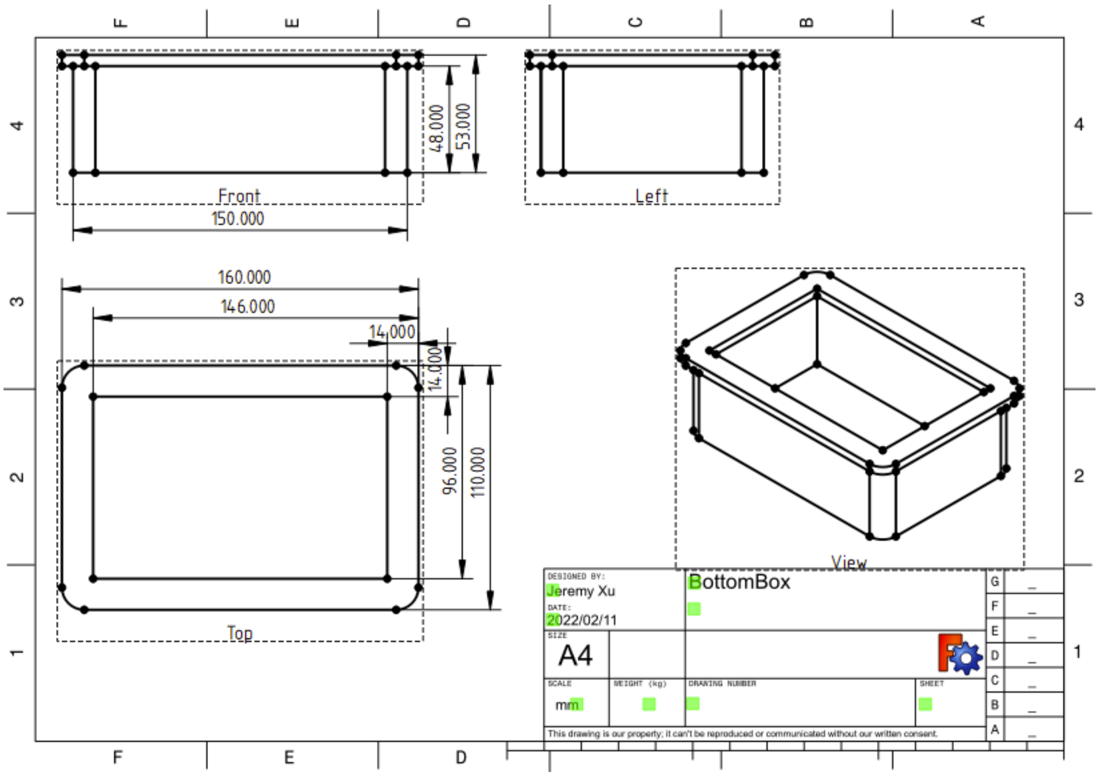


Figure 4: Waterproof box bottom

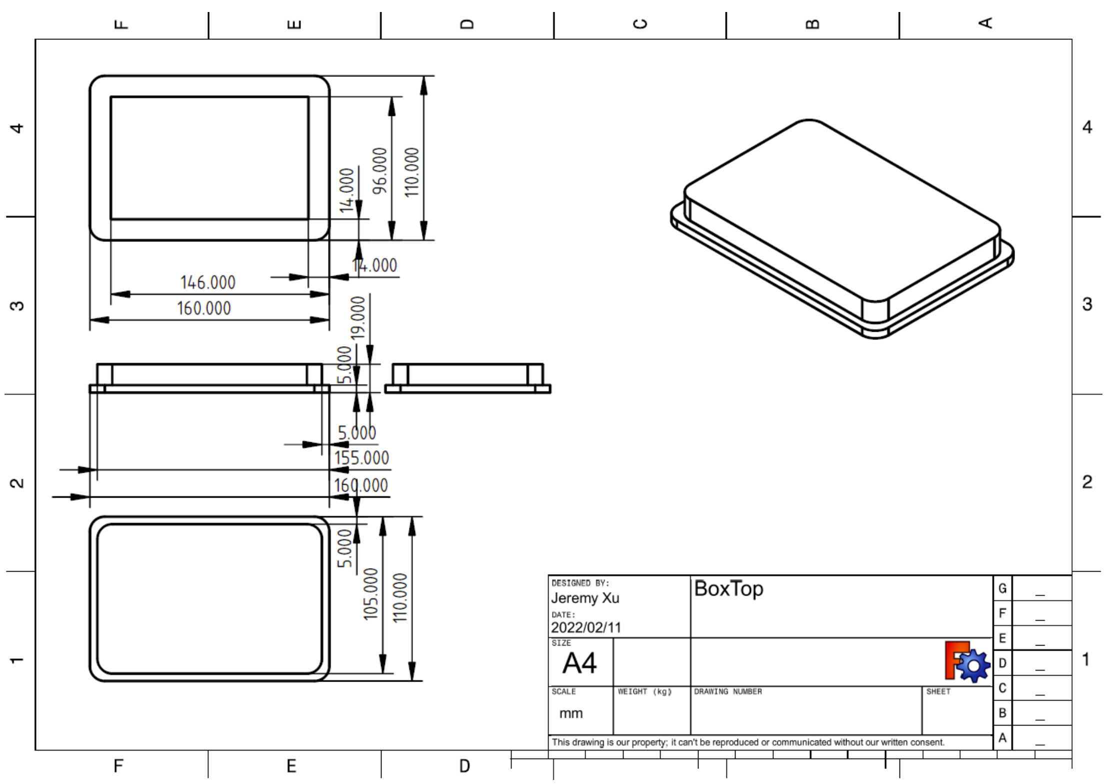


Figure 5: Waterproof box lid