Bike Team 2

Junior Design

03/04/2022

Project Checkoff

What is it: Our final system is an automatic bike light system which has automatic turn lights, fading brake lights and a dashboard with a small user interface. The system will be powered by a 12V rechargeable battery with an indication of the remaining battery life on the dashboard. The dash will also indicate speed that will be determined by the Hall-Effect sensor on the wheel. The dashboard also indicates if turn signals are on and holds both the kill and turn switches. The main enclosure contains the battery, PCB and microcontroller with hookups to the lights. The PCB is able to provide the proper power to both the microcontroller and lights, with the microcontroller doing most of the heavy lifting.

Datasheets:

- Switches: https://www.nkkswitches.com/pdf/MtogglesBushing.pdf
- Display: https://cdn-shop.adafruit.com/datasheets/865datasheet.pdf
- IMU: https://wiki.seeedstudio.com/Grove-6-Axis_AccelerometerAndGyroscope/
- Microcontroller: https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf
- LEDs: https://cree-led.com/media/documents/data-sheet-JSeries-6to36V-5050.pdf
- Hall-Effect Sensor: AH1815 (sparkfun.com)
- Battery: https://cdn.shopify.com/s/files/1/2364/9089/files/EP125.pdf?v=1610472863
- Charger: https://www.expertpower.us/products/epc122-2a

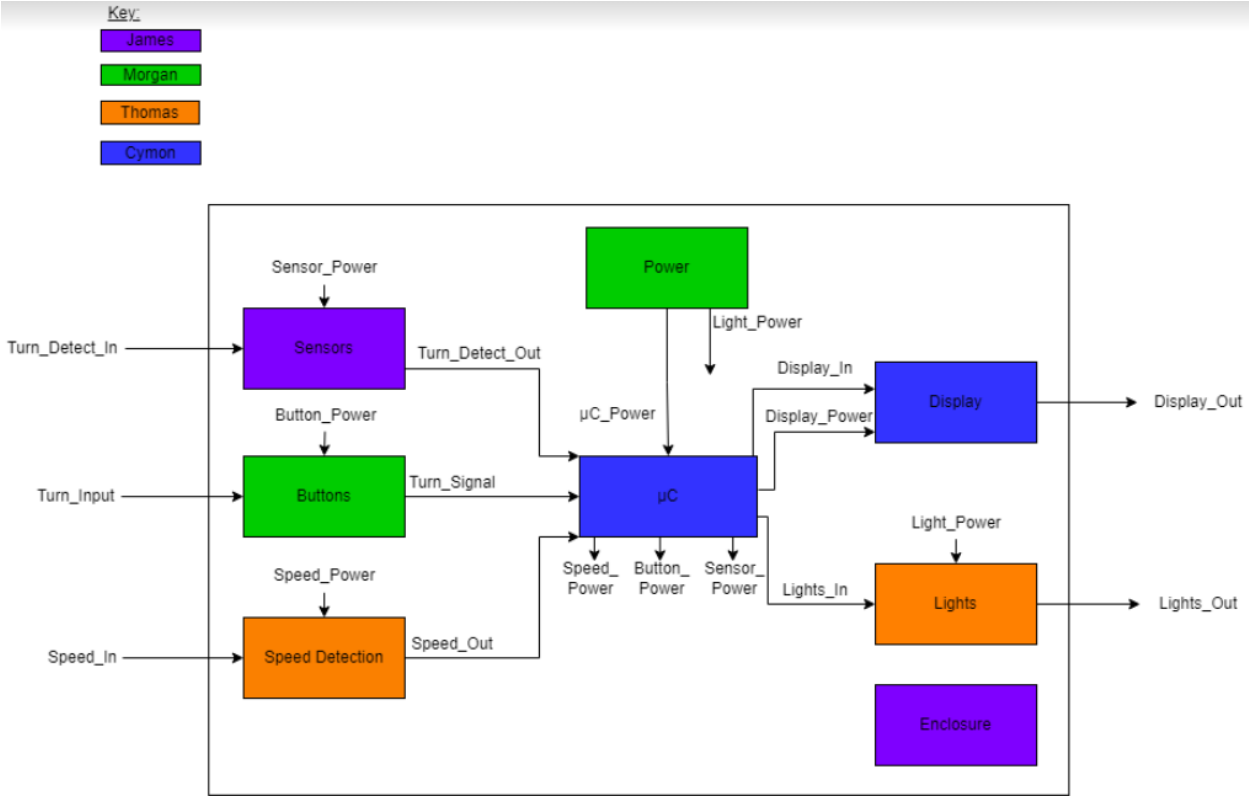
Table of Contents:

Bill of Materials:

Item	Reference Designator	Link	Quantity	Cost (Dollars)	Description	Vendor
SPDT Switch	U5A	NKK Switches	1	23.09	A switch that goes back to its neutral state, used for turning	NKK Switches
SPST Switch	U6A	NKK Switches	1	17.39	A regular switch used for power	NKK Switches
Adafruit 0.56" 4-Digit 7-Segment Display w/ I2C Backpack	U2	Display	1	11.95	A seven segment display used for the displaying the speed	Adafruit
Grove 6-axis IMU Module	U3	IMU	1	15.46	An IMU used for determining acceleration for turn detection	Seeed Studio
ESP32 Wroom Microcontroller	U1	ESP 32	1	10.99	A microcontroller used for handling all the various inputs and outputs	Expressif
JR5050 6-V P Class LEDs	D3, D4, D5	LED	3	1.80	The LEDs form the lights for turn and brake indication	CreeLED
Hall Effect Sensor	U4	Sensor	1	2.07	The sensor used for determining the speed	Texas Instruments
12V Battery	D6	Battery	1	35.00	The battery used for powering all the components	Expert Power

Battery Charger		Charger	1	22.00	Charges the battery	Expert Power
PCB			1	108.00	Distributes power properly through the system	Oshpark
PLA Filament		Filament	1	11.00	Used to create the main enclosure	Veeology
M3 Heat Set Inserts		Inserts	1	9.99	Used to attach the top and bottom of the enclosure	Albers LLC
Wire Glands		Glands	1	6.25	Used to protect the wires	NPT
Resistors	R1, R2, R3, R4, R5, R6, R7, R8	Resistors	3	10.99	Used to both voltage divide and provide consistent current	BOJACK
Basic LEDs	D1, D2	LEDs	2	1.08	Used to indicate turns to the user	Rohm Semiconductors
MOSFETs	M1, M2, M3	Mosfet	3	4.50	Used for sending power to lights	onsemi

Final Block Diagram:



Final Interface Table:

Interface	Specifications
Base_Power	Analog Signal V_Max = 12 V V_Min = 5 V I_Min = .1 mA I_Max = .41 A
Light_Power	Analog Signal V_Max = 6 V V_Min = 5 V I_Min = .1 mA I_Max = 400 mA
Sensor_Power	Power from the µC V_min = 3v V_max = 5v

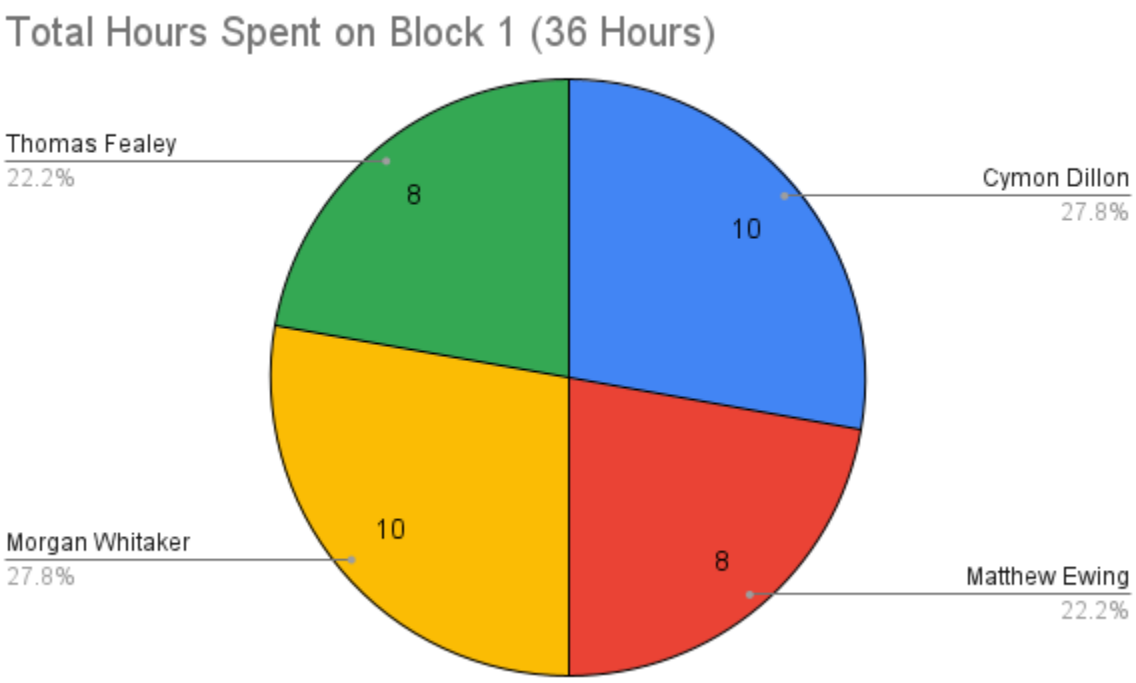
	Inom = .9mA
Turn_Detect_Out	Uses I2C word structure with clk and data signal* -word structure defined below Vmax: 3.3V Freq_Max: 5MHz Transfer rate: up to 100 kbit/s
Turn_Detect_In	Accelerometer and gyro information Detects decelerations Also detects turns being completed
Turn_Signal	Power returning to ESP Vmax = 3.3V Vmin = 3V Detected on rising edge
Turn_Input	Input to the turn signal Flipping it right is right Flipping it left is left Will return to middle on own
Lights_In	Analog Signal PWM Active-Low Vmax = 6V Vmin = 5V Imax = 400mA
Lights_out	Light from LEDs At least 100 Lumens Blinks left and right lights at 500 ms Brake light is solid when slowing down
Display_Power	Power from the µC Vmax: 3.3V Vmin: 3V Imax = 50 mA
Display_In	Digital signal I2C communication Uses I2C word structure with clk and data signal*

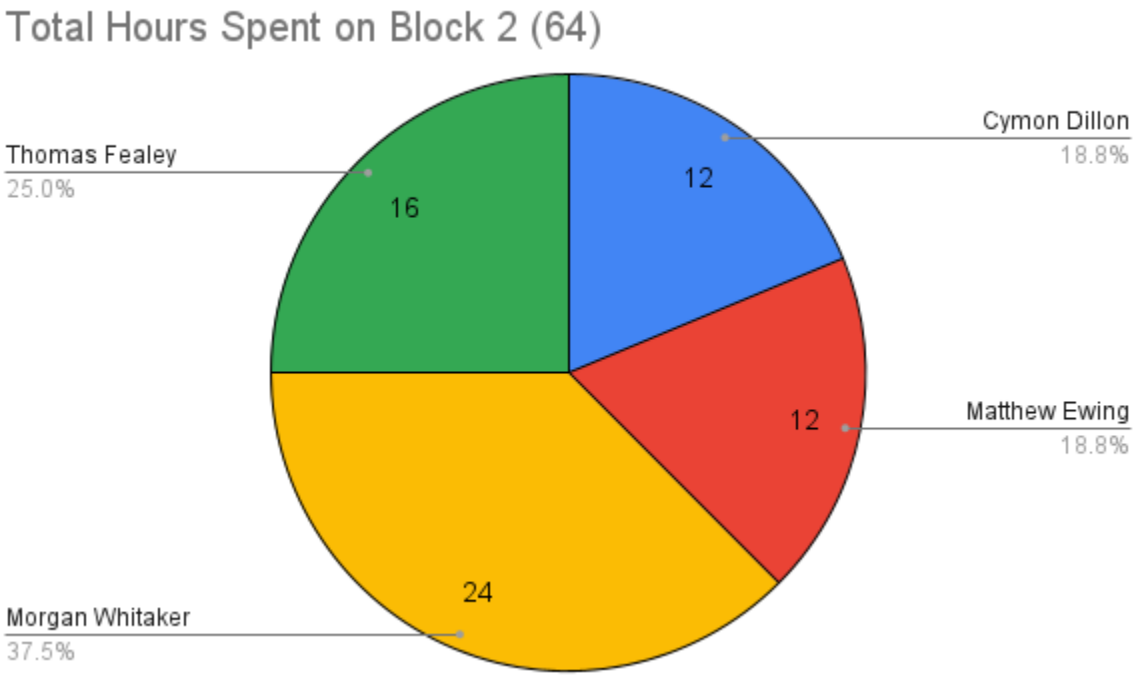
	-word structure defined below Vmax: 3.3V Freq_Max: 5MHz Transfer rate: up to 100 kbit/s
Display_Out	Light from LEDs Displays 2 digits with 2 decimal places Has a decimal point in the middle
Speed_Power	Power from the µC Vmax = 3.3V Vmin = 3V Imax = 50 mA
Speed_In	Analog Signal 0 ~ 20mph Signal when hall effect detects a magnetic field Makes the sensor output a 0 when field is detected Will be detected on falling edge
Speed_Out	Power returning to ESP Vmax = 3.3V Vmin = 3V Detected on falling edge
Button_Power	Power coming from ESP Used to signal turns by sending power back to µC Vmax = 3.3V Vmin = 3V Imax = 50 mA
µC Power	Vmin: 2.2V Vmax: 12V Vnom = 5V Inom: .5A

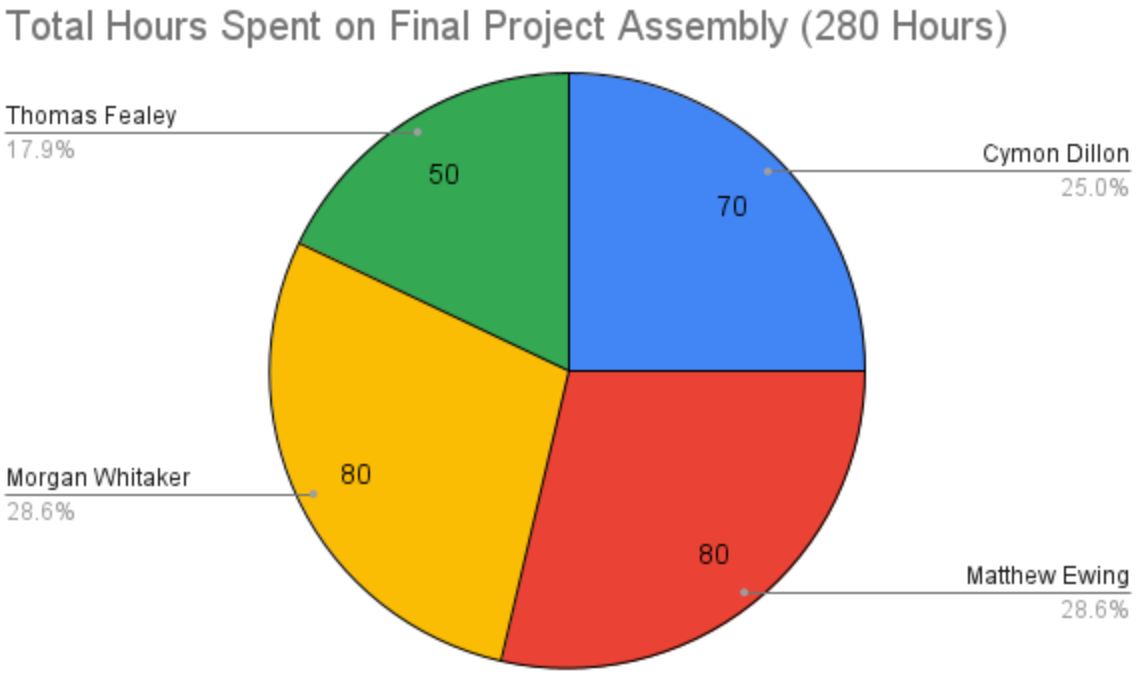
*I2C Word Structure:

Start bit
Device address(one byte)
data(at least one byte)
Stop Bit

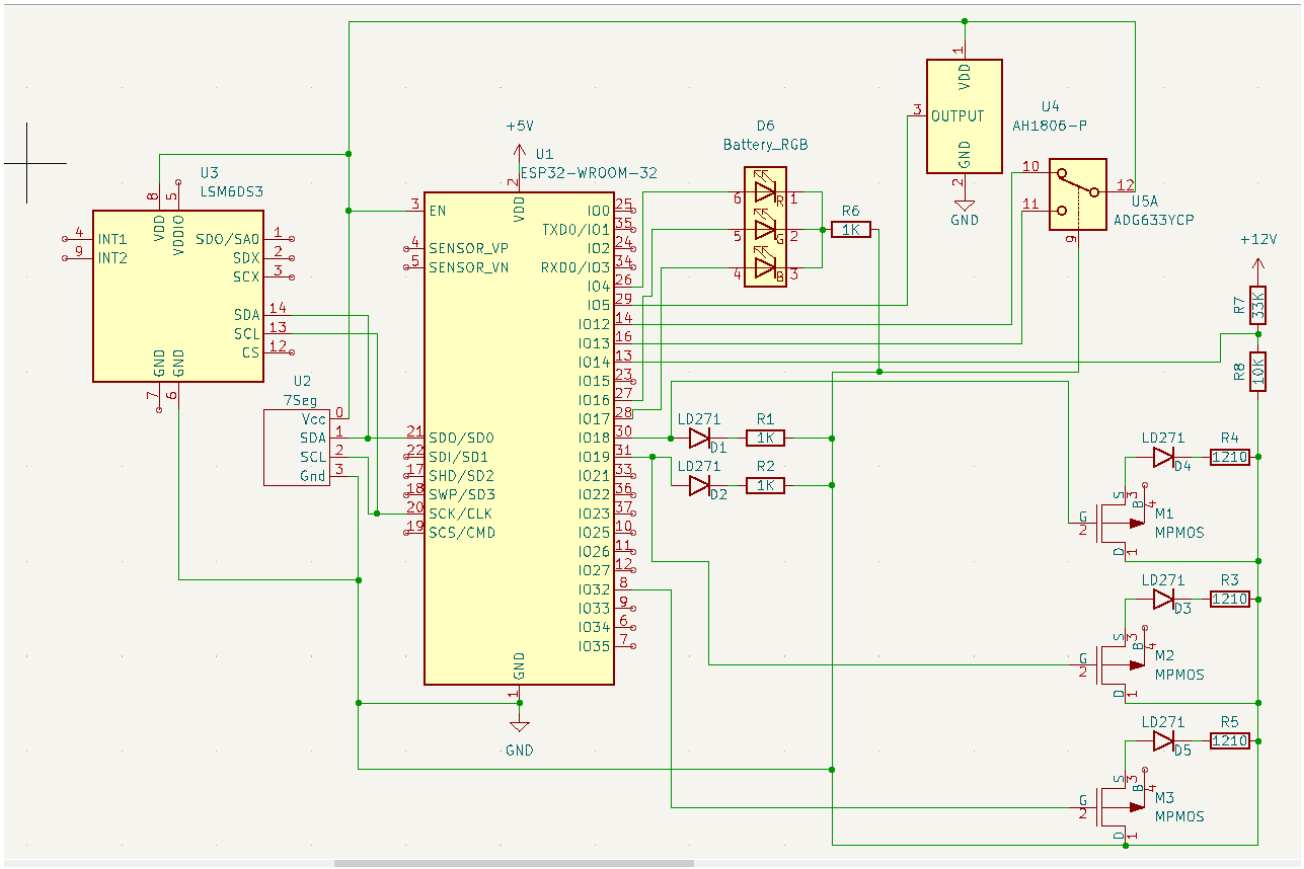
Time Report:


Total Hours Spent on Block 1 (36 Hours)
Thomas Fealey
22.2%
8
Cymon Dillon
27.8%
10
Morgan Whitaker
27.8%
10
Matthew Ewing
22.2%
8


Total Hours Spent on Block 2 (64)
Thomas Fealey
25.0%
16
Cymon Dillon
18.8%
12
Matthew Ewing
18.8%
12
Morgan Whitaker
37.5%
24

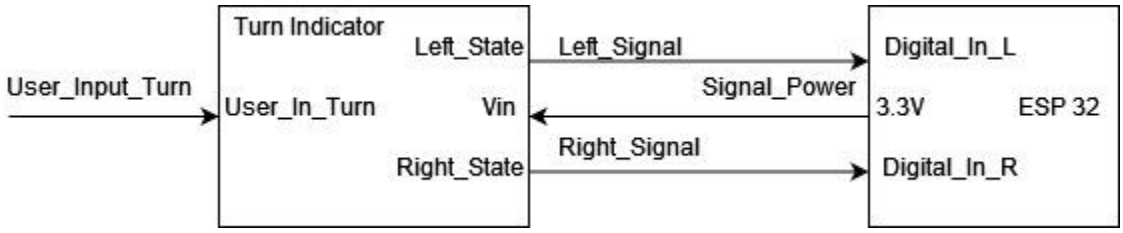

Total Hours Spent on Final Project Assembly (280 Hours)
Thomas Fealey
17.9%
50
Cymon Dillon
25.0%
70
Morgan Whitaker
28.6%
80
Matthew Ewing
28.6%
80

Final Block Schematic:

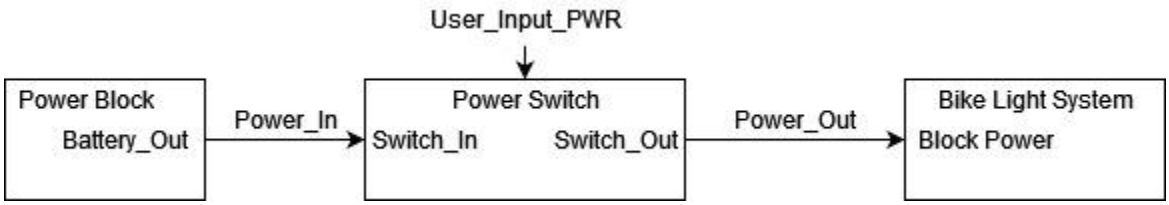


Individual Block Materials:

- **Switches:**
  - **Block Diagram:**
    - **Turn Indicator:**



    - **Power Switch:**



  - **Interface Table:**
    - **Turn Indicator:**

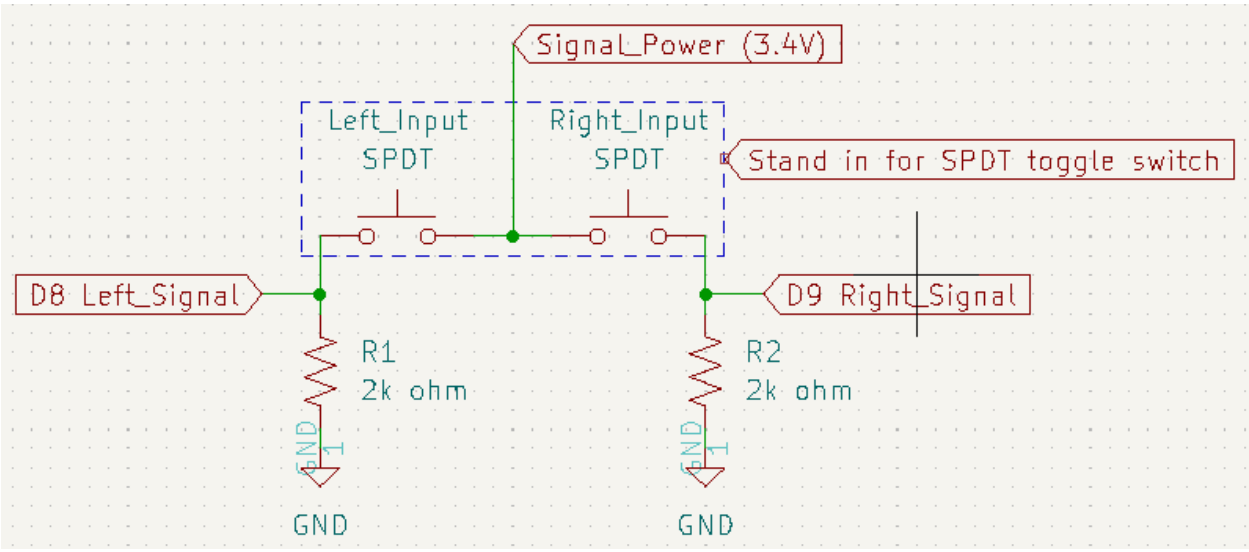
Interface	Specifications
User_Input_Turn	Physical Manipulation Pressure nom: 0.125N
Left_Signal	Digital Signal Vmax: 3.6V Vnom: 3.4V (Measured) Vmin: 3.0V
Signal_Power	Digital Signal Vmax: 3.6V Vnom: 3.4V (Measured) Vmin: 3.0
Right_Signal	Digital Signal

	Vmax: 3.6V Vnom: 3.4V (Measured) Vmin: 3.0V

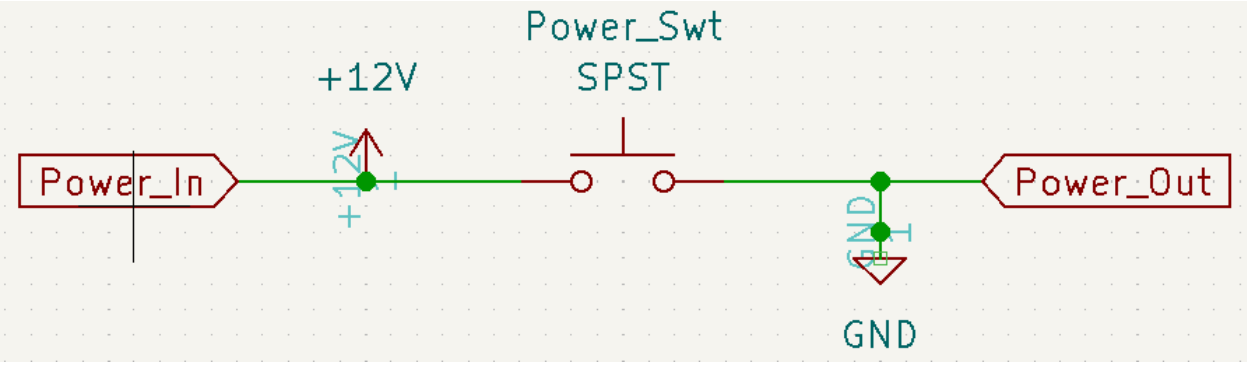
- **Power Switch:**

Interface	Specifications
User_Input_PWR	Physical Manipulation Pressure nom: 0.125N
Power_In	Digital Signal Vmax: 30V Vnom: 12V Vmin: 9V Imax: 3A Inom: 2A Imin: 0A
Power_Out	Digital Signal Vmax: 30V Vnom: 12V Vmin: 9V Imax: 3A Inom: 2A Imin: 0A

- **Block Schematic:**
  - **Turn Indicator:**



- **Power Switch:**

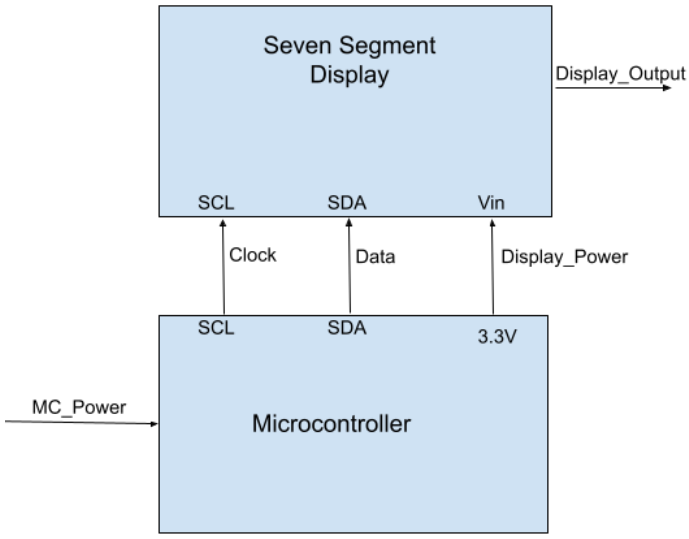


  - **Code:**

```
/*******************************
//  Morgan Whitaker
//  4, February 2022
//  ECE 342 Junior Design 2
//  Switch Block check off
*******************************/
//Setting up the digital pin names
const int left_in = 8;   //left turn indication input
const int left_out = 2;  //left turn output
const int right_in = 9;   //right turn indication input
const int right_out = 12;  //right turn output

int left_state = 0;         //initial state of the left turn indication
int right_state = 0;        //initial state of the right turn indication

void setup() {
  // put your setup code here, to run once:
  pinMode(left_in, INPUT);
  pinMode(right_in, INPUT);

  pinMode(left_out, OUTPUT);
  pinMode(right_out, OUTPUT);

  Serial.begin(9600);
}

void loop() {
  // put your main code here, to run repeatedly:
  digitalWrite(left_in, LOW);
  digitalWrite(right_in, LOW);
```

```
  //Conditional if left turn is entered
  if  (digitalRead(left_in) == HIGH){
      delay(500);
      Serial.println("Inside IF 1");
      digitalWrite(left_out, HIGH);          //Turn the light on for 1
second
      delay(1000);
      digitalWrite(left_out, LOW);           //Turn the light off
  }

  //Conditional if right turn is entered
  else if  (digitalRead(right_in) == HIGH){
      delay(500);
      Serial.println("Inside ELSE IF 2");
      digitalWrite(right_out, HIGH);         //Turn the light on for 1
second
      delay(1000);
      digitalWrite(right_out, LOW);          //Turn the light off
  }

}
```

- **Display:**
  - **Block Diagram:**

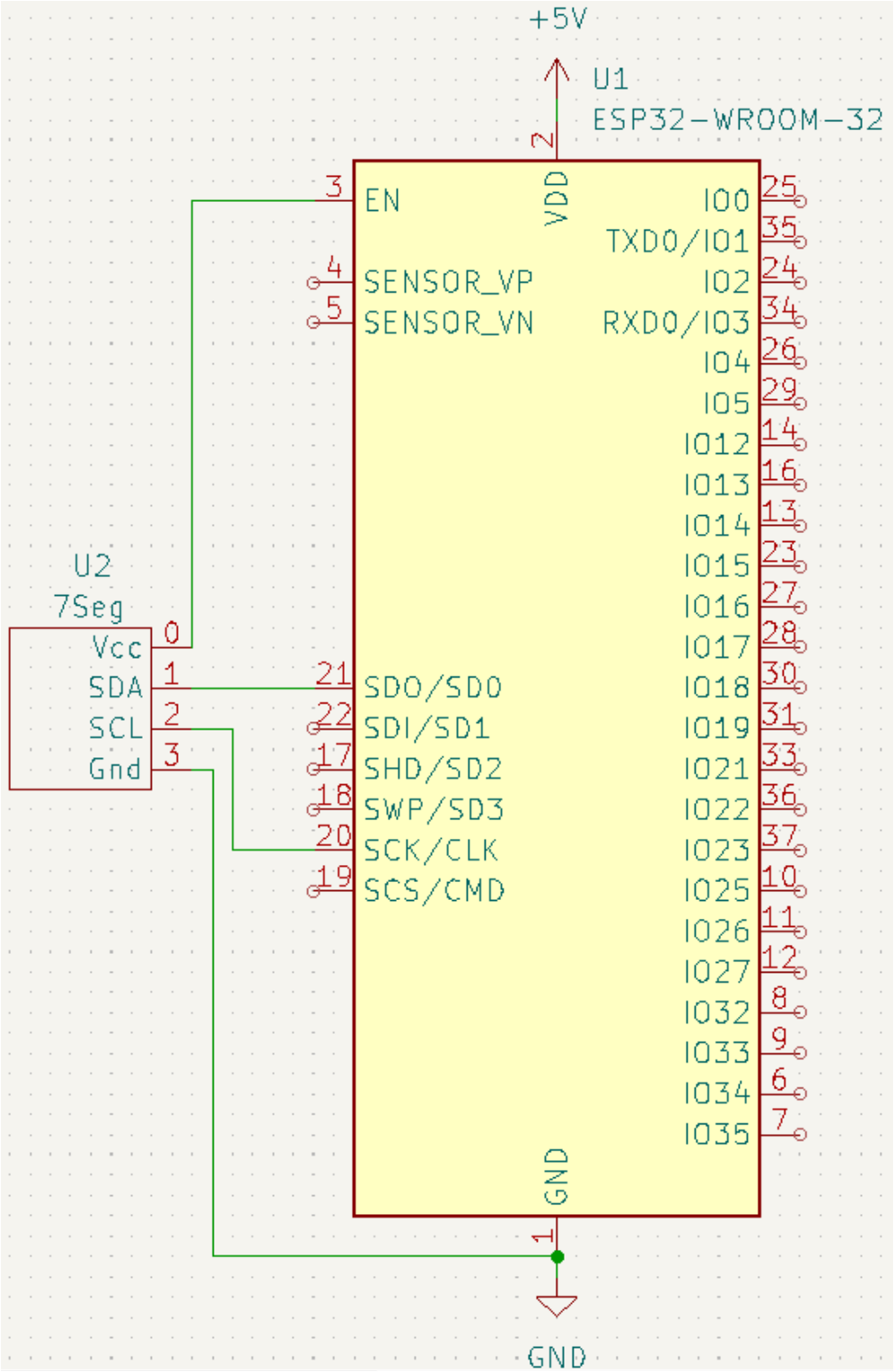


- **Interface Table:**

Interface	Specifications
Clock	Digital Signal I2C clk signal Vnom: 3.3V
Data	Digital Signal I2C data signal Vnom: 3.3V
Display_Power	Vmax: 5V Vmin: 3V
MC_Power	Vmin: 2.2V Vmax: 3.6V Inom: .5A
Display_Output	Displays 4-digits Goes to two decimal places Decimal point in middle

Note: There are also two signals going to a common ground, but these are not included here.

- **Block Schematic:**



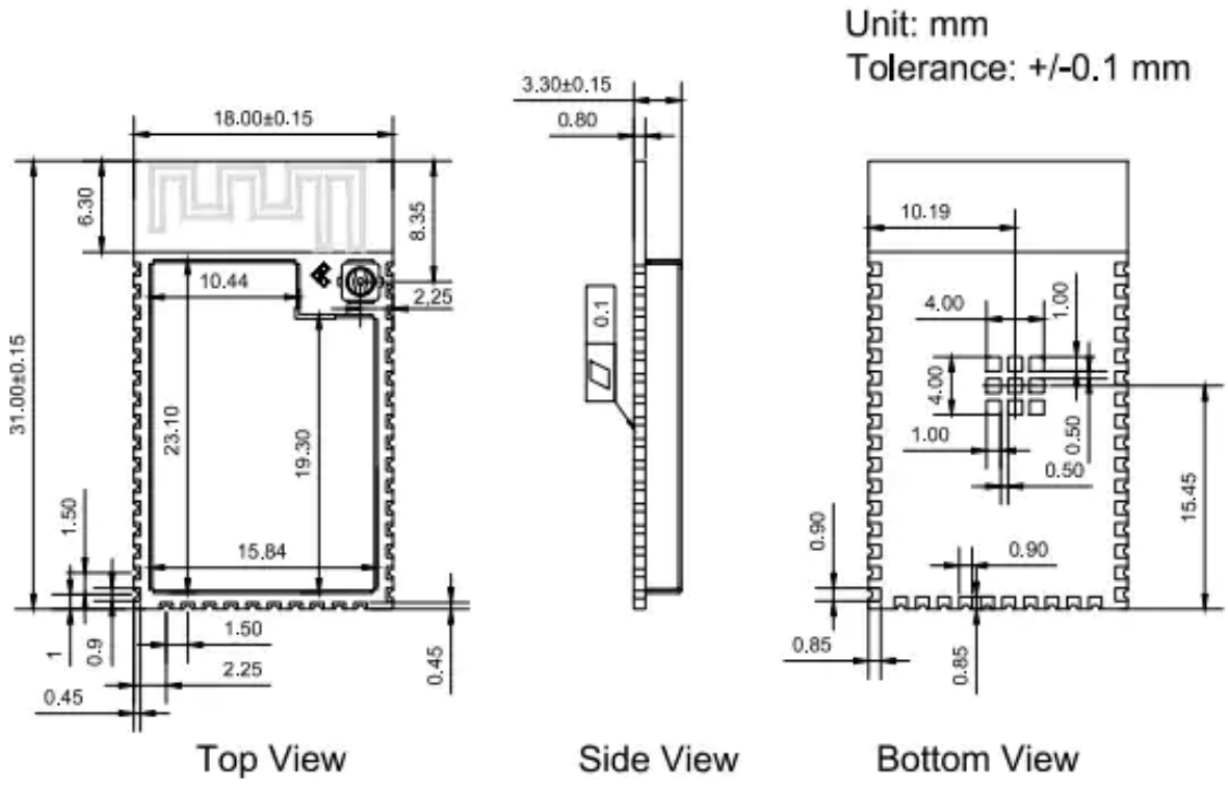
- **Code:**

```
/*********************************************************
*Author:Cymon Dillon
*Date:2/4/22
*Project:Bike Lights Team 2
*Libraries used: Adafruit_GFX, Adafruit_LEDBackpack
*********************************************************/

#include <Adafruit_GFX.h>
#include "Adafruit_LEDBackpack.h"

float count = 0; //Keeps count of time in ms
float Radius = 11;//radius of the wheel in inches
float Distance = (6.28*Radius); //circumference of the wheel
float BikeSpeed = 0;//The speed of the bike in mph
float SpeedSensor = 5; //is not 0 if the hall effect sensor senses a
full rotation

Adafruit_7segment matrix = Adafruit_7segment();


/*********************************************************
*Name:setup
*Purpose:Ran at the beginning of the program. Performs proper
*functions to communicate with the display. Sets a pin as an
*input, then displays all 0's to the display. Also sets up the
*communication to the serial monitor.
*********************************************************/
void setup() {

  matrix.begin(0x70); //set up for the display library

pinMode(SpeedSensor, INPUT);//input from the sensor, currently a button

// displays all 0's
matrix.writeDigitNum(0,0);
matrix.writeDigitNum(1,0,true);
matrix.writeDigitNum(3,0);
matrix.writeDigitNum(4,0);
matrix.writeDisplay();

Serial.begin(9600);
}
```

```
/*********************************************************
*Name:loop
*Purpose:If the button has been pressed, it runs the functions to
*calculate and display speed. It also resets the time count when
*necessary.
*********************************************************/
void loop() {

//if the button is detected to be pushed
if(digitalRead(SpeedSensor) == LOW){
  speedCalc(detect(count)); //find the speed based on the current time
  }

count = 0; //reset count

}

/*********************************************************
*Name:detect
*Purpose:Delays for debouncing purposes, then counts the time
*between button pushes.
*********************************************************/
float detect(float value){
  delay(400); //delay for button to finish being pushed
  value = value + 400;

  if(digitalRead(SpeedSensor) == LOW){
    return value; //if the button is pressed again, return
  }
  else{
   //if not pressed again, count ms and do again
    value++;
    delay(1);
    detect(value);
    }
  }

/*********************************************************
*Name:speedCalc
*Purpose:Takes in an elapsed time, and converts it into MPH for a
*set bike radius. Then displays this number.
*********************************************************/
void speedCalc(float elapsed){
   float BikeSpeed = (((Distance)/(elapsed))*56.8182);//find speed in
mph
```

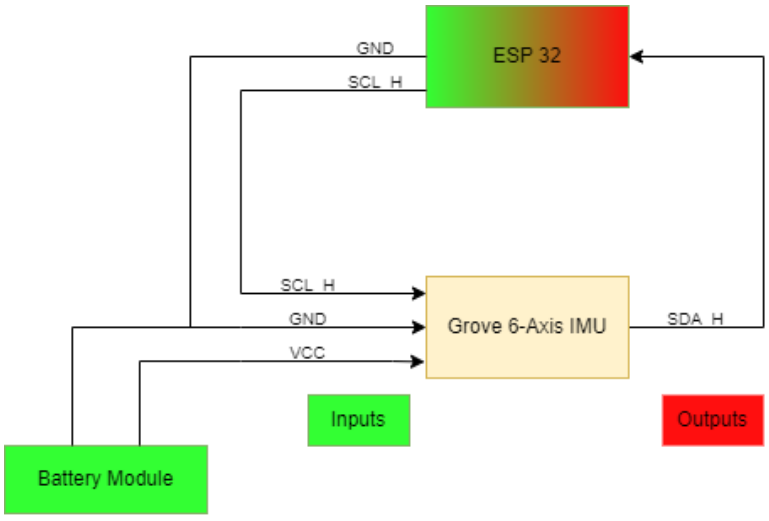
```
    //Print the values for testing purposes
    Serial.print("Time:");
    Serial.println(elapsed);
    Serial.print("Speed:");
    Serial.println(BikeSpeed);

    Display(BikeSpeed);//send the number to the display

     return;
    }

/*********************************************************
*Name:Display
*Purpose:Takes in a number, and displays it to the seven-segment.
*********************************************************/
void Display(float displayNum){
  //turn the value into an int
  int value = displayNum*100;

  //grab each digit
  int d4 = ((value%10));
  value = value/10;
  int d3 = ((value%10));
  value = value/10;
  int d1 = ((value%10));
  value = value/10;
  int d0 = ((value%10));
  value = value/10;

  //Diplay each bit
  matrix.writeDigitNum(0,d0);
  matrix.writeDigitNum(1,d1, true);//turns on the decimal
  matrix.writeDigitNum(3,d3);
  matrix.writeDigitNum(4,d4);
  matrix.writeDisplay();
  }
```

- **Mechanical Drawings:**



- **Turn Sensor:**

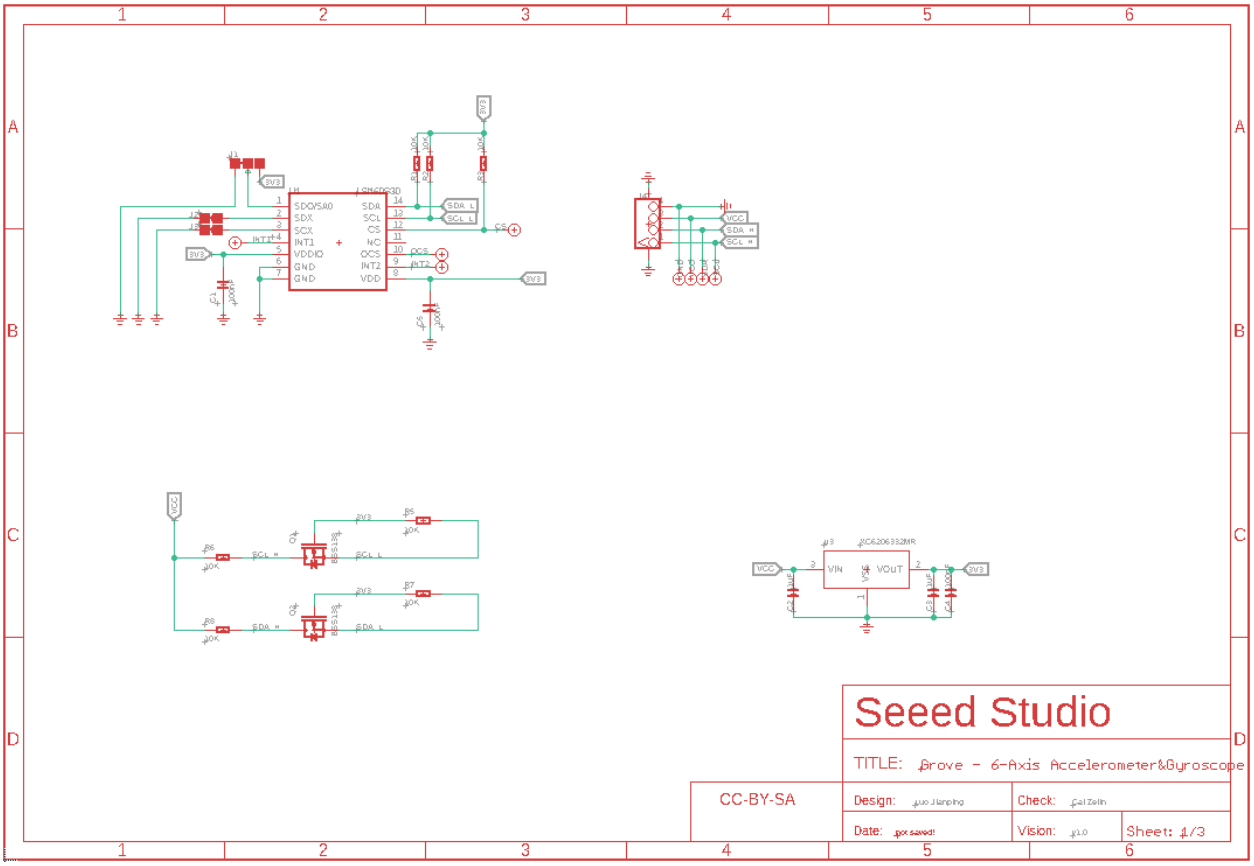
  - **Block Diagram:**



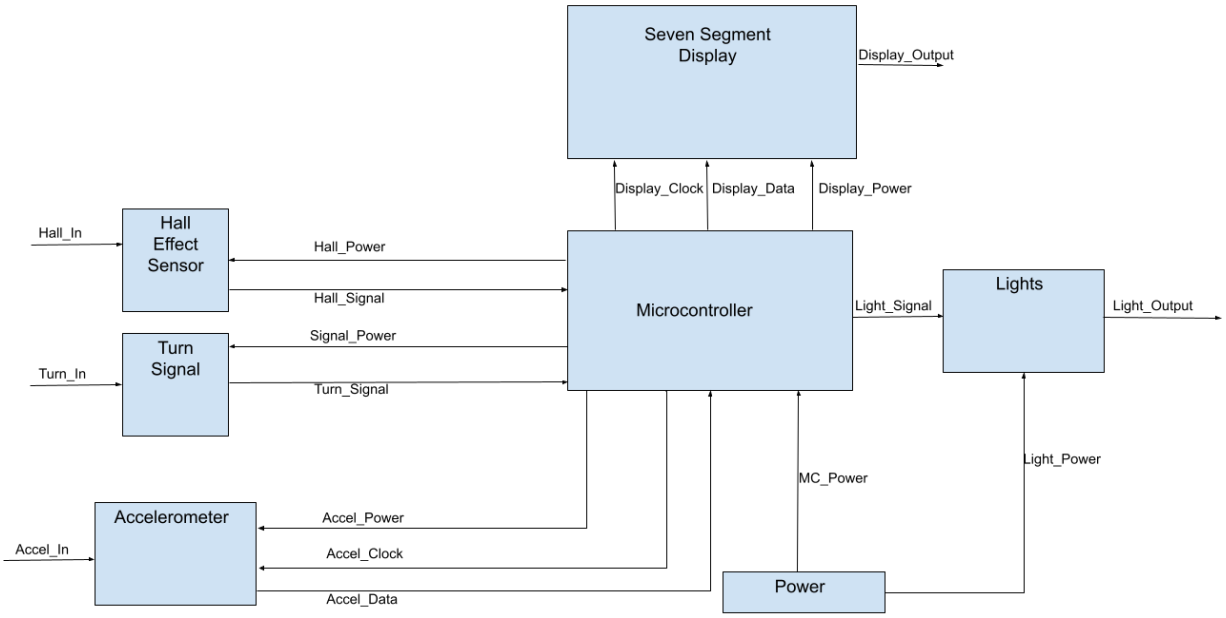
- **Interface Table:**

Net Name	Input/Output	Expected Value
VCC	Input	3.3v
GND	Input	0v
SDA_H	Output	Digital Signal w/ 3.3v max and 0v min, behavior based on I2C protocol
SCL_H	Input	Digital Signal w/ 3.3v max and 0v min, behavior based on I2C protocol

- **Block Schematic:**



- **Microcontroller:**
  - **Block Diagram:**



  - **Interface Table:**

Interface	Specifications
Display_Clock	Digital signal I2C communication Vmax: 3.3V Vmin: 3.0V Freq_Max: 4MHz ESP_Operating_Freq: 60 MHz -Can also run at 160MHz if set, but 8MHz by default
Display_Data	Digital Signal I2C data signal Vmax: 3.3V Vmin: 3.0V Uses I2C word structure with clk and data signal* -word structure defined below

	Transfer rate: up to 100 kbit/s
Display_Power	Power from the µC Vmax: 3.3V Vmin: 3V Imax = 50 mA
MC_Power	Vmin: 2.2V Vnom: 3.6V Vmax = 10V if powered through Vin pin Inom: .5A
Display_Output	Displays 4-digits Goes to two decimal places Decimal point in middle
Accel_Power	Power from the µC V_min = 3v V_Nom = 3.3V Inom = .9mA
Accel_Clock	Digital signal I2C communication Vmax: 3.3V Vmin: 3.0V Freq_Max: 4MHz ESP_Operating_Freq: 60 MHz -Can also run at 160MHz if set, but 8MHz by default
Accel_Data	Digital Signal I2C data signal Vmax: 3.3V Vmin: 3.0V Uses I2C word structure with clk and data signal* -word structure defined below Transfer rate: up to 100 kbit/s
Hall_Power	Power from the µC Vmax = 3.3V Vmin = 3V Imax = 50 mA

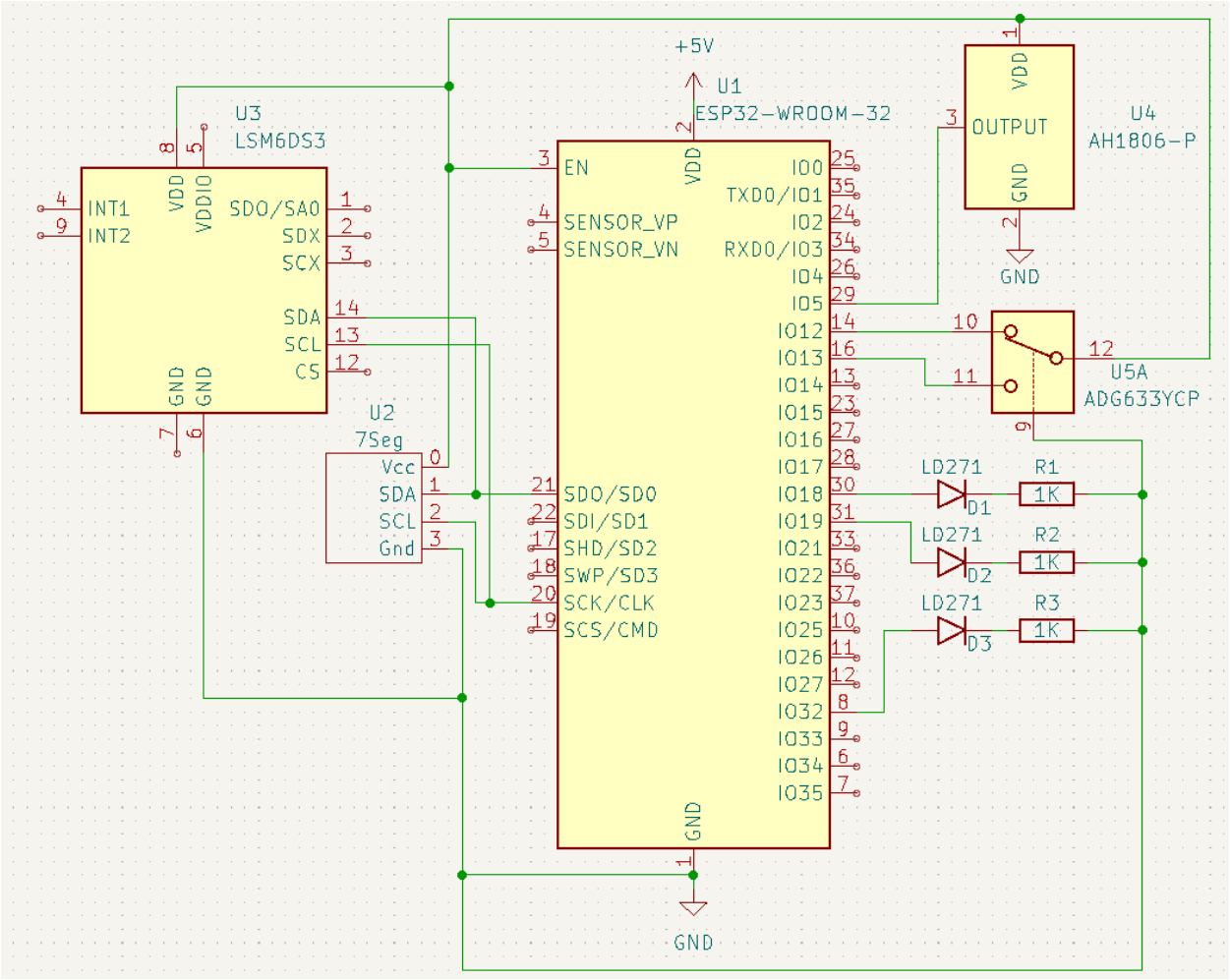
Hall_Signal	Power returning to ESP Digital signal Active Low Vhi = 3.3V Vlo = 0V Detected on falling edge
Turn_Signal	Power returning to ESP Digital signal Active High Vhi = 3.3V Vlo = 0V Detected on rising edge
Signal_Power	Power coming from ESP Vmax = 3.3V Vmin = 3V Imax = 50 mA
Light_Signal	Analog Signal PWM Vmax = 3.3V Vmin = 3V Max Duty Cycle: 256 where 256 = 100% Frequency:5000 Hz
Light_Output	At least 800 Lumens Blinks left and right lights at 500 ms Brake light is solid when slowing down
Hall_In	Signal when hall effect detects a magnetic field Makes the sensor output a 0 when field is detected Will be detected on falling edge
Turn_In	Input to the turn signal Flipping it up is right Flipping it down is left Will return to middle on own
Accel_In	Accelerometer and gyro information Detects decelerations Also detects turns being completed
Light_Power	Analog Signal

	V_Max = 12 V V_Min = 5 V I_Min = .1 mA I_Nom = 20mA

- **Microcontroller Pins:**

Pins	Purpose
D-12	**Left turn signal**
D-13	**Right turn signal**
D-32	**Brake Light**
D-5	**Hall-Effect sensor signal**
D-18	**Left turn signal light**
D-19	**Right turn signal light**
D-21	**I2c SDA pin**
D-22	**I2c SCL pin**
D-4	**Red pin on rgb diode**
Rx2	**Green pin on rgb diode**
Tx2	**Blue pin on rgb diode**
D-14	**Pin to read the battery life**

- **Block Schematic:**



- **Code:**

```
/*********************************************************
*Author:Cymon Dillon
*Date:2/4/22
*Project:Bike Lights Team 2
*Libraries used: Adafruit_GFX, Adafruit_LEDBackpack,LSM6DS3.h,
*                Wire.h
*********************************************************/
#include <Adafruit_GFX.h>
#include "Adafruit_LEDBackpack.h"
#include "LSM6DS3.h"
#include "Wire.h"
#define LED_BUILTIN 2


LSM6DS3 Accel(I2C_MODE, 0x6A);    //I2C device address 0x6A


float SpeedSensor = 5; //is not 0 if the hall effect sensor senses a full
```

```
                          rotation
int LeftSignal = 12; //used for when the left turn signal is turned on
int RightSignal = 13; //used for when the left turn signal is turned on
int LeftLight = 18; //left blinker light
int RightLight = 19; //right blinker light
int BrakeLight = 32; //Brake light
int Brightness = 0; //Controls duty cycle of brake light

// setting PWM properties
const int freq = 5000;
const int ledChannel = 0;
const int resolution = 8;

int leftCount = 0;//keeps count of how many time the left light blinks
int rightCount = 0;//keeps count of how many times the right light blinks

float Radius = 11;//radius of the wheel in inches
float Distance = (6.28*Radius); //circumference of the wheel
float BikeSpeed = 0;//The speed of the bike in mph

volatile float Time = 0; //Keeps Time of time in ms
unsigned long TotalTime = 0;//keeps track of the total program time
unsigned long LastTime = 0;//keeps track of time since last wheel rotation
unsigned long BlinkTime = 0; //tracks how long to blink
unsigned long Lefttime = 0; // tracks when to pull accel data for left turn
unsigned long Righttime = 0; // tracks when to pull accel data for right turn
unsigned long Braketime = 0; // tracks when to pull accel data for brake


volatile bool CalcSpeed = false; // if true, then you find the time since last
                                    rotation
volatile bool LeftBlink = false; //true if the left light should blink
volatile bool RightBlink = false; //true if right light should blink
volatile bool BrakeOn = false; //true the bike is slowing down

Adafruit_7segment matrix = Adafruit_7segment(); //sets up display

/*********************************************************
*Name:setup
*Purpose:Ran at the beginning of the program. Performs proper
*functions to communicate with the display. Sets a pin as an
*input, then displays all 0's to the display. Also sets up the
*communication to the serial monitor and pwm cahnnels.
*Finally, attaches several interrupts to pins and sets up the
*accelerometer.
*********************************************************/
void setup() {
matrix.begin(0x70);//sets up display

pinMode(SpeedSensor, INPUT); //Hall Effect sensor for speedometer
pinMode(LeftSignal, INPUT); //Left turn signal
pinMode(RightSignal, INPUT); //Right turn signal
pinMode(LeftLight, OUTPUT);//outputs to left led
pinMode(RightLight, OUTPUT);//outputs to right led
pinMode(BrakeLight, OUTPUT);//outputs to right led
```

```
//For the hall effect sensor
attachInterrupt(digitalPinToInterrupt(SpeedSensor), detect, FALLING);

//For left turn signal
attachInterrupt(digitalPinToInterrupt(LeftSignal), leftSignal, RISING);

//For Right turn signal
attachInterrupt(digitalPinToInterrupt(RightSignal), rightSignal, RISING);


// displays all 0's
matrix.writeDigitNum(0,0);
matrix.writeDigitNum(1,0,true);
matrix.writeDigitNum(3,0);
matrix.writeDigitNum(4,0);
     matrix.writeDisplay();


//Checks that accel is connected
    while (!Serial);
    //Call.begin() to configure the IMUs
    if (Accel.begin() != 0) {
        Serial.println("Device error");
    } else {
        Serial.println("Device OK!");
    }


//Used for pwm signal setup
ledcSetup(ledChannel, freq, resolution);
ledcAttachPin(BrakeLight, ledChannel);
ledcWrite(ledChannel, 0);//starts brake light off

Serial.begin(9600);
}

/*********************************************************
*Name:loop
*Purpose:Calculates the speed og the bike and displays it.
*Then, it checks for turns and braking, and turns the lights on
*and off when necessary.
*********************************************************/
void loop() {
 TotalTime = millis(); //keeps track of time since started

//If the speed interrupt happened
if(CalcSpeed == true){
  Time = TotalTime - LastTime;
  Time = Time + 250;
  LastTime = TotalTime;
  CalcSpeed = false;

}

//if the time between the interrupts is nonzero
if(Time != 0){
  Display(speedCalc(Time));
```

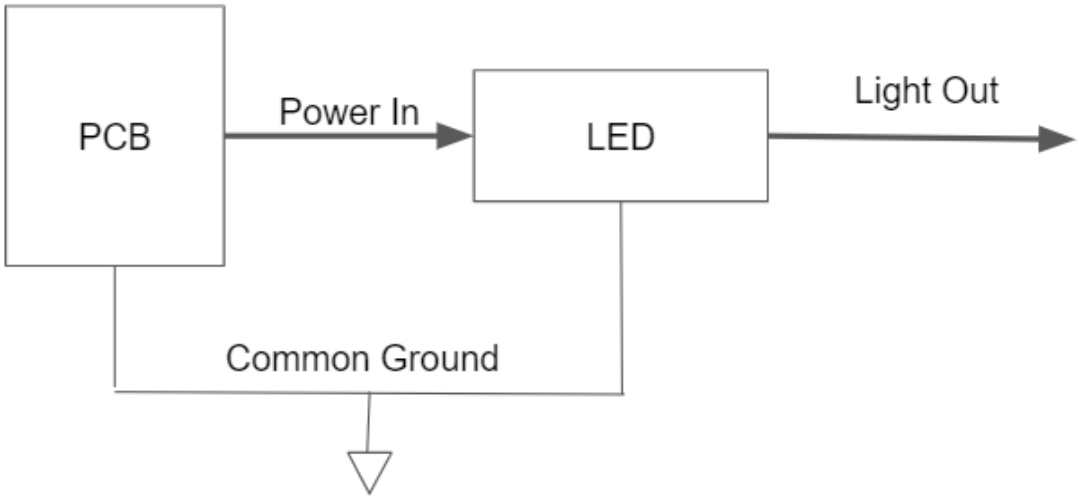
```
}

Time = 0;//reset Time

//if the left light should be blinking, do it
if(LeftBlink == true){
  //pull accel data
  leftBlink(); //blinks light
  LeftCheck(); //checks if left turn happens
  }

//if right light should be blinking, do it
if(RightBlink == true){
  rightBlink(); //blinks light
  RightCheck(); //checks if right turn happens
  }


  BrakeBlink(); //turns on brake light

}

/*********************************************************
*Name:lefttSignal
*Purpose:Used as the interrupt. If left signal pressed,
*changes the correct variable.
*********************************************************/
void leftSignal(){
  LeftBlink = true; //blinks left light
  RightBlink = false; //blinks right
  rightCount = 0; //resets the right count
  }

/*********************************************************
*Name:rightSignal
*Purpose:Used as the interrupt. If right signal pressed,
*changes the correct variable.
*********************************************************/
void rightSignal(){
  RightBlink = true; //blinks right
  LeftBlink = false; //blinks left
  leftCount = 0; //resets left count
  }

/*********************************************************
*Name:LeftBlink
*Purpose:Turns on the leftt blinker, and blinks it with .5 seconds
*on, .5 seconds off.
*********************************************************/
void leftBlink(){
  digitalWrite(RightLight, LOW); //turns off right light

  //Turns on lights for 500 ms, then off for 500ms
  if((TotalTime - BlinkTime) < 500){
    digitalWrite(LeftLight, HIGH);

  }
```

```
  else if(((TotalTime - BlinkTime) > 500) && (TotalTime - BlinkTime) < 1000){
    digitalWrite(LeftLight, LOW);

}
  else if((TotalTime - BlinkTime) > 1000){
    BlinkTime = TotalTime;
    leftCount++;

    //blink 5 times
    if(leftCount == 6){
    LeftBlink = false;
    leftCount = 0;
    }
  }

  }

/*********************************************************
*Name:RightBlink
*Purpose:Turns on the right blinker, and blinks it with .5 seconds
*on, .5 seconds off.
*********************************************************/
void rightBlink(){
  digitalWrite(LeftLight, LOW); //turns off left light

  //Turns on lights for 500 ms, then off for 500ms
if((TotalTime - BlinkTime) < 500){
  digitalWrite(RightLight, HIGH);

  }
else if(((TotalTime - BlinkTime) > 500) && (TotalTime - BlinkTime) < 1000){
digitalWrite(RightLight, LOW);

}
else if((TotalTime - BlinkTime) > 1000){
 BlinkTime = TotalTime;
 rightCount++;

 //blink 5 times
  if(rightCount == 6){
    RightBlink = false;
    rightCount = 0;
    }
  }

}

/*********************************************************
*Name:BrakeBlink
*Purpose:Turns on the brake lights with correct values.
*********************************************************/
void BrakeBlink(){

/* Here I will change the accel data into a brightness for the brake light.
I am not sure what values to expect yet but it will be something like this:

Brightness = Accel.readFloatAccelY();
```

```
*/

//Just reads the accel data every .5 S and also changes brake light
  if(TotalTime - Braketime > 500){
    Serial.print(" AccelY = ");
    Serial.println(Accel.readFloatAccelY(), 4);
    Braketime = TotalTime;
    ledcWrite(ledChannel, Brightness); //Send pwm signal to brakelight
    Brightness = Brightness + 16; //increase brightness

  //reset the brightness once it hits max
    if(Brightness > 256){
      Brightness = 0;
  }

    }
}

/*********************************************************
*Name:detect
*Purpose:Detects input from the hall effect sensor and changes
*necessary variables
*********************************************************/
void detect(){
  CalcSpeed = true; //tells speed to be calculated
}

/*********************************************************
*Name:speedCalc
*Purpose:Takes in an elapsed time, and turns it into an MPH
*speed
*********************************************************/
float speedCalc(float elapsed){
   float BikeSpeed = (((Distance)/(elapsed))*56.8182);//find speed in mph
   Serial.print(" Speed: ");
   Serial.println(BikeSpeed); //for testing purposes
   return BikeSpeed;
    }

/*********************************************************
*Name:Display
*Purpose:Takes in a value, parses out each digit, and displays
*them to the seven-segment display.
*********************************************************/
void Display(float displayNum){
  //turn the value into an int
  int value = displayNum*100;

  //grab each digit
  int d4 = ((value%10));
  value = value/10;
  int d3 = ((value%10));
  value = value/10;
  int d1 = ((value%10));
  value = value/10;
  int d0 = ((value%10));
```

```
  value = value/10;

  //Diplay each bit
  matrix.writeDigitNum(0,d0);
  matrix.writeDigitNum(1,d1, true);//turns on the decimal
  matrix.writeDigitNum(3,d3);
  matrix.writeDigitNum(4,d4);
  matrix.writeDisplay();
  }

/*********************************************************
*Name:LeftCheck
*Purpose:Checks if a left turn has happened, and changes
*necessary variables.
*********************************************************/
void LeftCheck(){
  //Just reads the gyro data every .5 S for now
if(TotalTime - Lefttime > 500){
    Serial.print(" GyroZ = ");
    Serial.println(Accel.readFloatGyroZ(), 4);
    Lefttime = TotalTime;
  }

/* Since I have not tested the accelerometer in the enclosure, I do not
 *  know what values to expect. Code will follow this form:
  if(left turn occurs)
  LeftBlink = false;
*/

}

/*********************************************************
*Name:RightCheck
*Purpose:Checks if a right turn has happened, and changes
*necessary variables.
*********************************************************/
void RightCheck(){

//Just reads the gyro data every .5 S for now
if(TotalTime - Righttime > 500){
   Serial.print(" GyroZ = ");
    Serial.println(Accel.readFloatGyroZ(), 4);
    Righttime = TotalTime;
  }

/* Since I have not tested the accelerometer in the enclosure, I do not
 *  know what values to expect. Code will follow this form:
  if(right turn occurs)
  RighttBlink = false;
*/
}
```

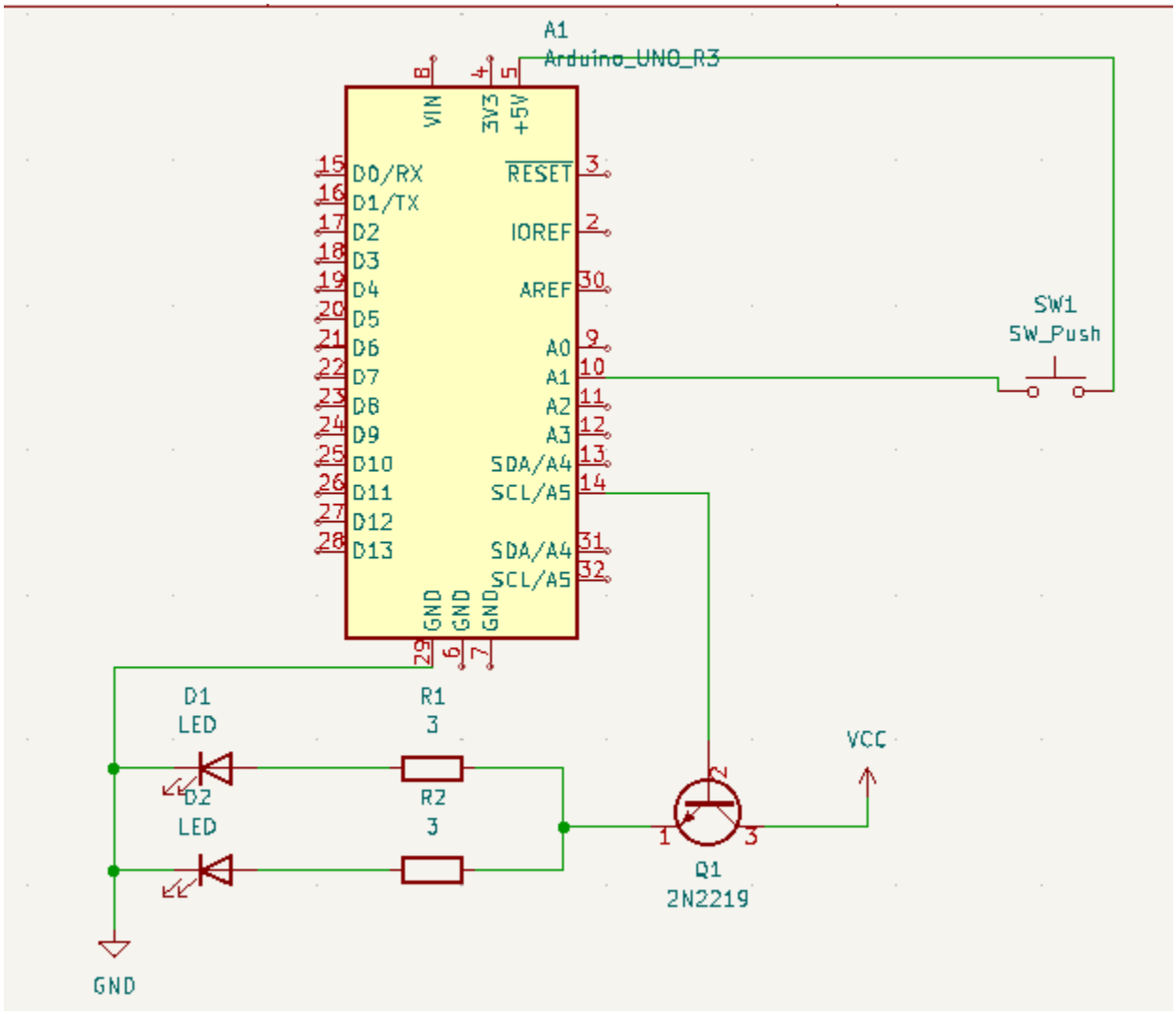
- **Lights:**
    - **Block Diagram:**



    - **Interface Table:**

Interface	Specifications
Power In	**Vin_min: 5.77V** **Vin_max: 6V** **I_in: 400mA**
Light Out	**400 Lumens**
Common Ground	**Ground**

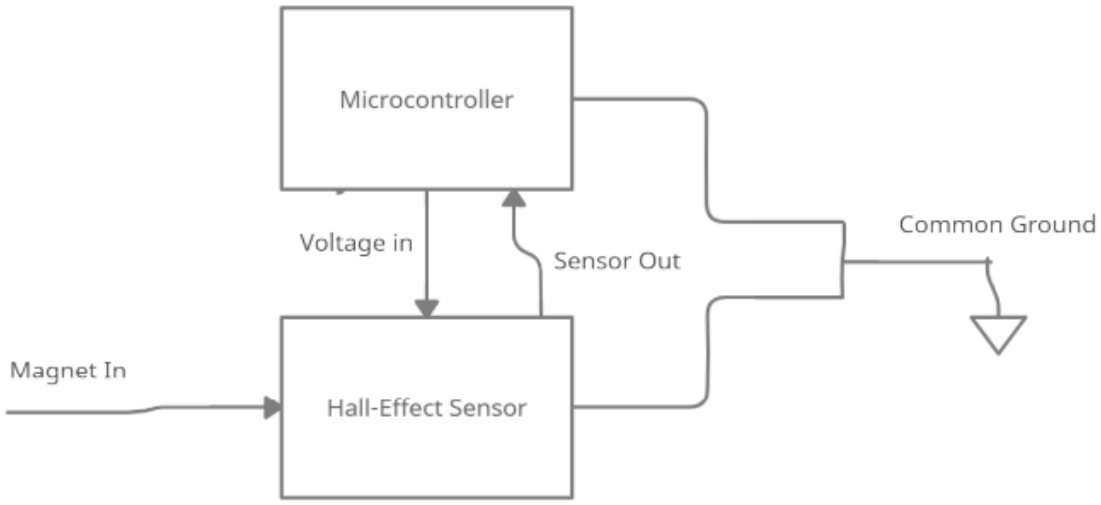
- **Block Schematic:**



- **Code:**

```
const int buttonPin = A1;
const int gatePin = A5;

int buttonState = 0;

void setup() {  //Establishes which pins are used as inputs or outputs
  pinMode(buttonPin, INPUT);
  pinMode(gatePin, OUTPUT);
  Serial.begin(9600);
}

void loop() { //Loops rapidly to check if the button is pressed and will make the output high if it has
  buttonState = digitalRead(buttonPin);
  if(buttonState == HIGH){
    digitalWrite(gatePin, HIGH);
    Serial.print("lol");
  }
  else{
     digitalWrite(gatePin, LOW);
  }
  delay(50);
}
```

- **Speed Detection:**
  - **Block Diagram:**

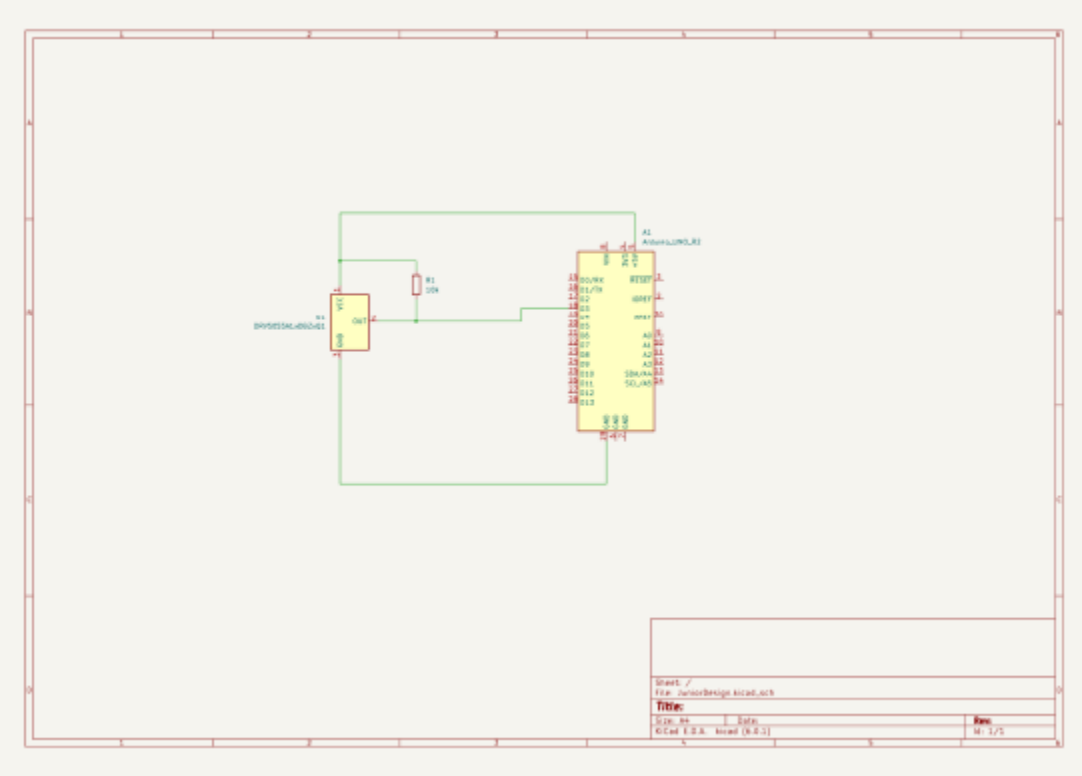


  - **Interface Table:**

Interface	Specifications

Magnet In	**Any Polarized Magnet**
Voltage In	**Vmax: 5.5V** **Vmin: 3V**
Sensor Out	**Current out: 2.5mA** **Vout_max: 5.5V** **Vout_min: 0V**
Common Ground	**Ground**

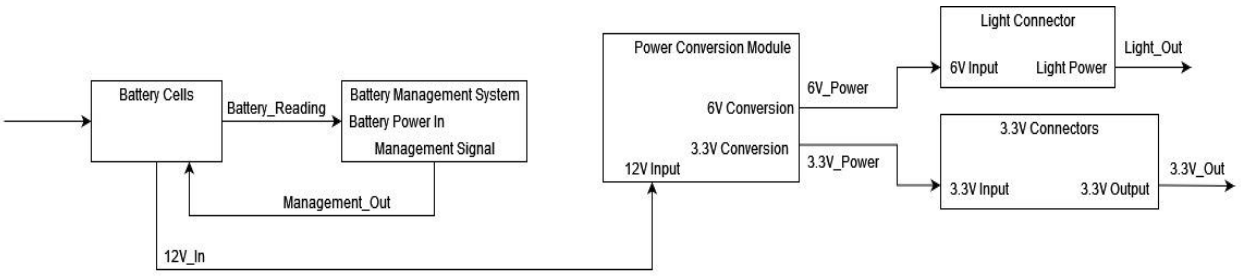
- **Block Schematic:**



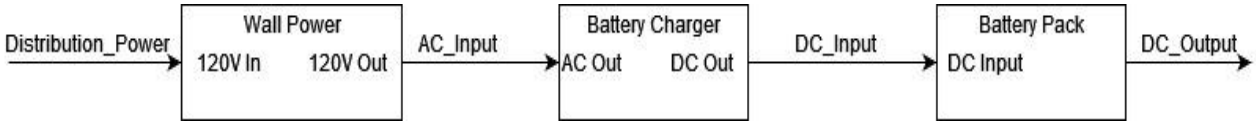
- **Code:**

```
/*
 * Arduino Uno Hall-Effect Sensor detection
 * Thomas Pealey
 * 2/4/2022
 */

volatile byte revolutions;
unsigned int rpm;
unsigned long timeold;
unsigned int magtime;
unsigned int Time;
const int magdetect = 2;

void setup() //Begins setup, mainly for interrupt and magnet detection
{
  Serial.begin(115200);
  pinMode(magdetect, INPUT);
  attachInterrupt(digitalPinToInterrupt(3), magnet_detect, FALLING);//Initialize the intterrupt pin (Arduino Uno digital pin 3) Calls magnet detect when magnet detected
  revolutions = 0;
  rpm = 0;
  timeold = 0;
}
void loop()//Counts the time passed in seconds
{
  /*if (half_revolutions >= 20) {
    rpm = 30*1000/(millis() - timeold)*half_revolutions;
    timeold = millis();
    half_revolutions = 0;
    Serial.println(rpm,DEC);
  }*/

  delay(1000);
  Time = Time + 1;
}
void magnet_detect()//This function is called whenever a magnet/interrupt is detected by the arduino
{
  revolutions = revolutions + 1; //Counts the number of revolutions
  Serial.println("Revolutions:");
  Serial.println(revolutions);
  Serial.println("Seconds passed");
  Serial.println(Time);
  Serial.println("detect");
  Time = Time - Time; //Resets the time when a magnet passes
}
```

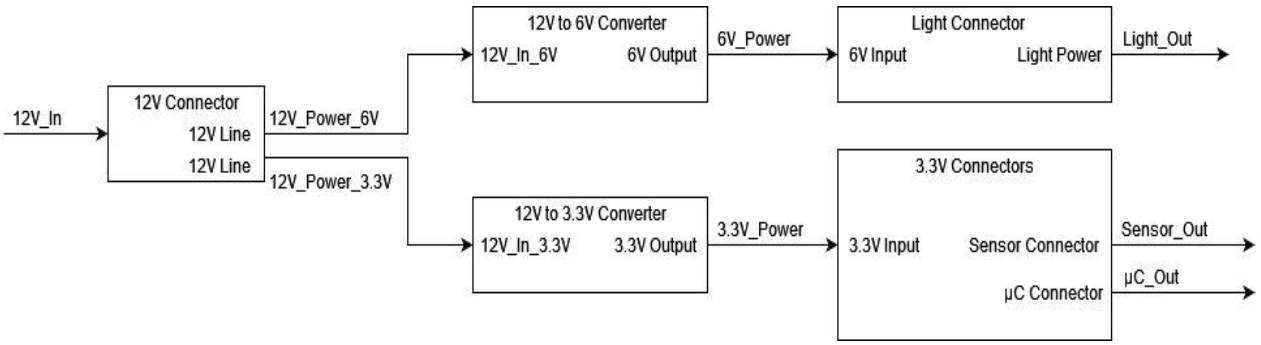
- **Power:**
  - **Block Diagram:**
    - **Battery Pack:**



- **Battery Charger:**



- **Power Conversion:**



- **Interface Table:**

  - **Battery Pack:**

Interface	Specifications
DC_Output	V_max: 14.6V V_nom: 13.8V V_min: 9.2V I_max: 10A I_nom: 7A I_min: 0A
Battery_Reading	(Not specified by supplier)
Management_Out	(Not specified by supplier)
12V_In	V_max: 13.8V V_nom: 12V V_min: 9.3V (I believe BMS will cut power before 9.2V is measured) I_max: 10A I_nom: 7A

	I_min: 2A
6V_Power	V_nom: 6V V_min: 5.77V I_max: 4A I_nom: 1.2A I_min: (Not specified by supplier)
3.3V_Power	V_max: 3.6V V_nom: 3.3V V_min: 3V I_max: 2A I_nom: 1.5A I_min: 0.8A
Light_Out	V_nom: 6V V_min: 5.77V I_max: 4A I_nom: 1.2A I_min: (Not specified by supplier)
3.3V_Out	V_max: 3.6V V_nom: 3.3V V_min: 3V I_max: 2A I_nom: 1.5A I_min: 0.8A

- **Battery Charger:**

Interface	Specifications
Distribution_Power	V_nom: 120V AC 50Hz-60Hz I_max: 15A I_nom: Device Dependant I_min: 0A
AC_Input	V_nom: 120V AC (No other specifications given)
DC_Input	V_nom: 14.4V DC I_nom: 2A I_min: 0.15A-0.25A
DC_Ouput	V_max: 14.6V V_nom: 13.8V

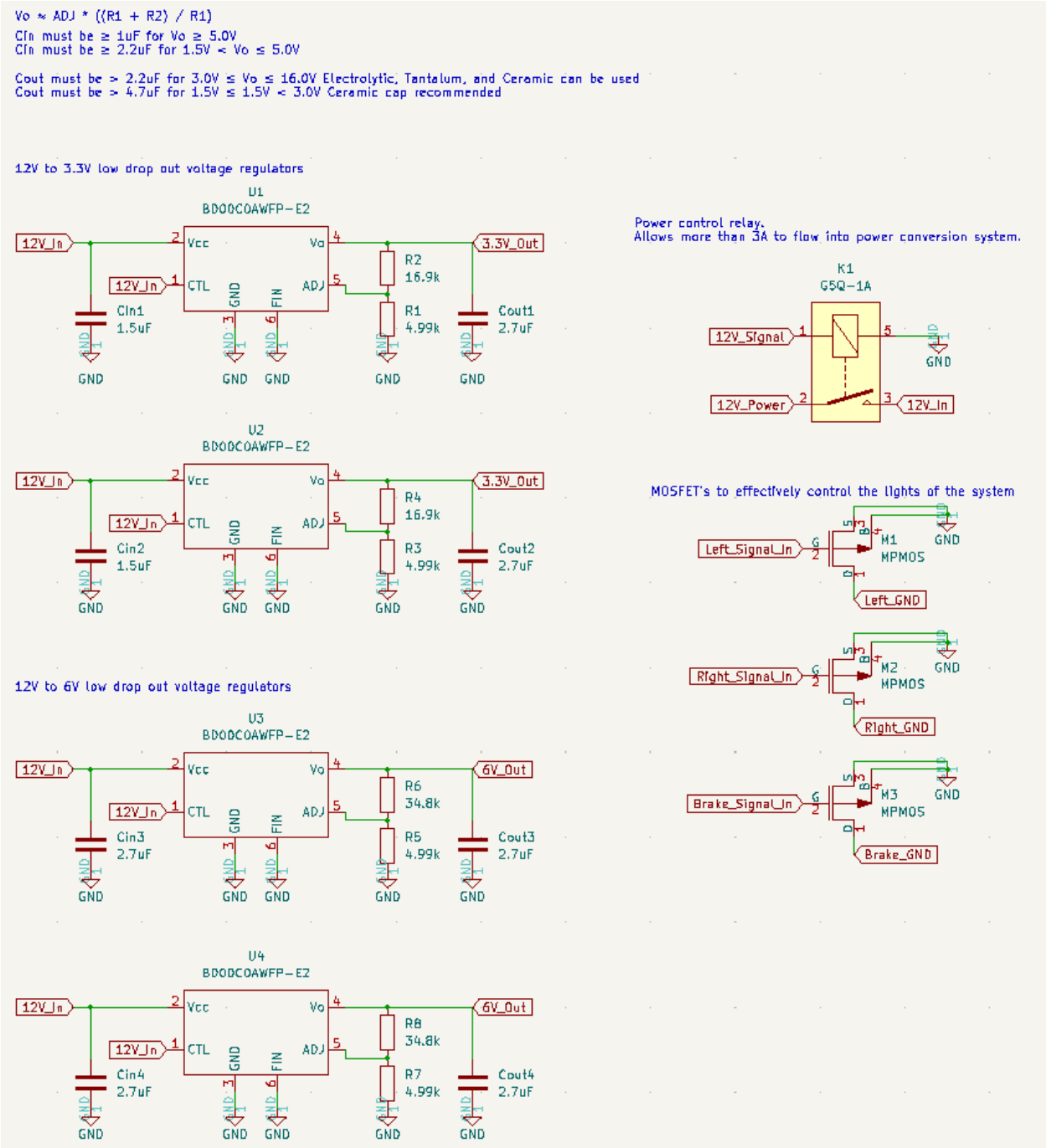
	V_min: 9.2V I_max: 10A I_nom: 7A I_min: 0A

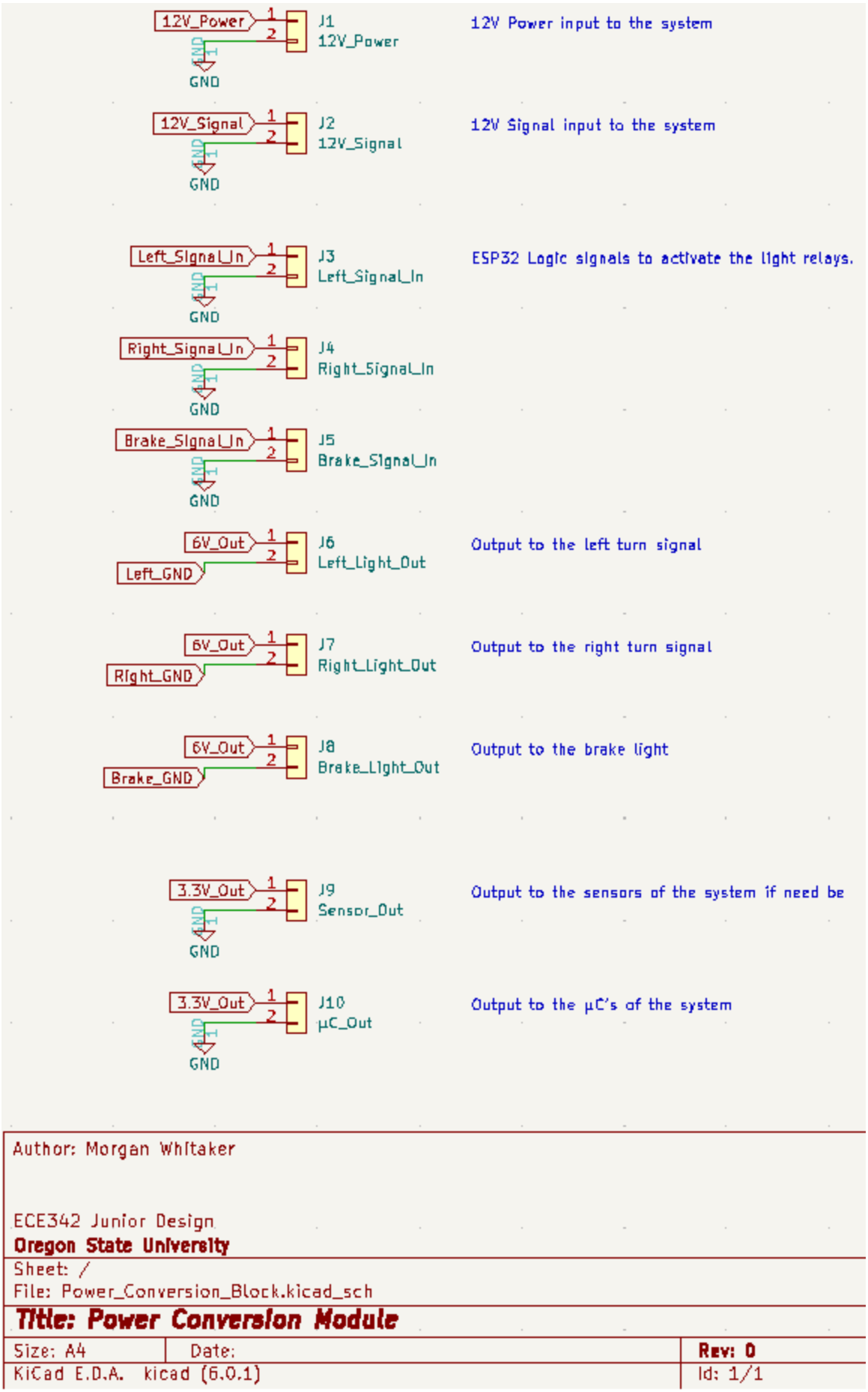
- **Power Conversion:**

Interface	Specifications
12V_In	V_max: 13.8V V_nom: 12V V_min: 9.3V (I believe BMS will cut power before 9.2V is measured) I_max: 10A I_nom: 7A I_min: 2A
12V_Power_6V	V_max: 13.8V V_nom: 12V V_min: 9.3V
12V_Power_3.3V	V_max: 13.8V V_nom: 12V V_min: 9.3V
6V_Power	V_nom: 6V V_min: 5.77V I_max: 4A I_nom: 1.2A I_min: (Not specified by supplier)
3.3V_Power	V_max: 3.6V V_nom: 3.3V V_min: 3V I_max: 2A I_nom: 1.5A I_min: 0.8A
Lights_Out	V_nom: 6V V_min: 5.77V I_max: 4A I_nom: 1.2A I_min: (Not specified by supplier)
Sensor_Out	V_max: 3.6V V_nom: 3.3V

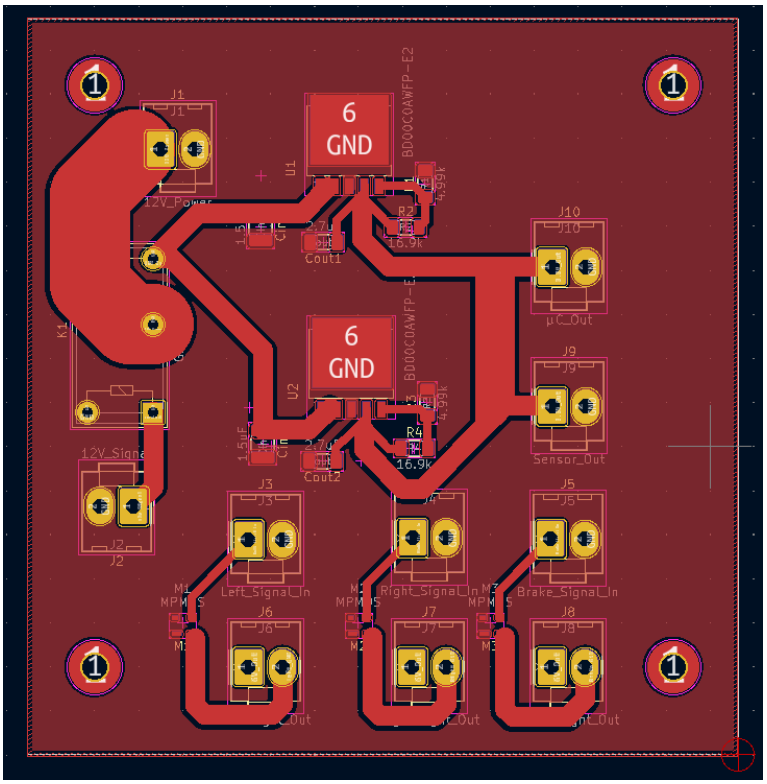
	V_min: 3V I_max: 0.8A I_nom: 0.6A I_min: 0.4A
µC_Out	V_max: 3.6V V_nom: 3.3V V_min: 3V I_max: 1.2A I_nom: 0.9A I_min: 0.4A

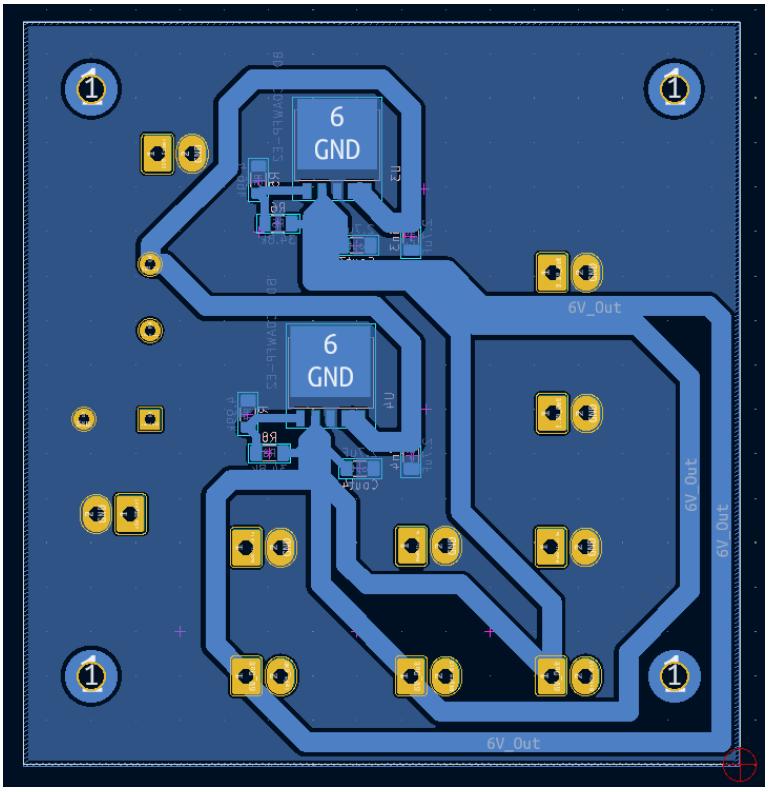
- **Block Schematic:**



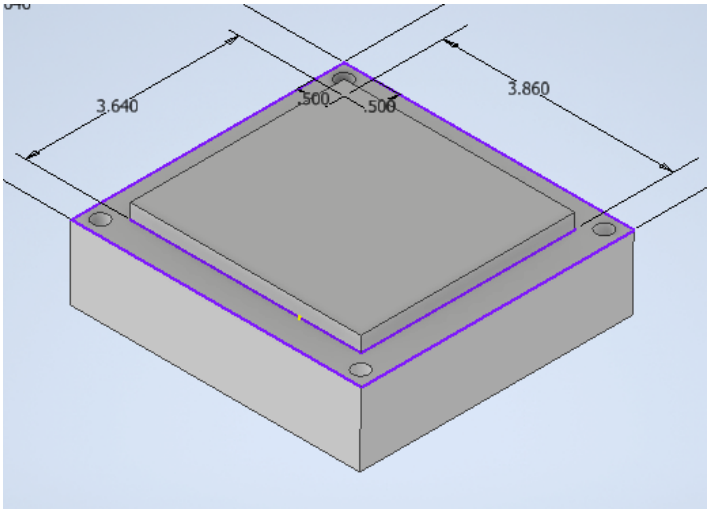

12V_Power
J1
12V_Power
12V Power input to the system
GND
12V_Signal
J2
12V_Signal
12V Signal input to the system
GND
Left_Signal_In
J3
Left_Signal_In
ESP32 Logic signals to activate the light relays.
GND
Right_Signal_In
J4
Right_Signal_In
GND
Brake_Signal_In
J5
Brake_Signal_In
GND
6V_Out
J6
Left_Light_Out
Output to the left turn signal
Left_GND
6V_Out
J7
Right_Light_Out
Output to the right turn signal
Right_GND
6V_Out
J8
Brake_Light_Out
Output to the brake light
Brake_GND
3.3V_Out
J9
Sensor_Out
Output to the sensors of the system if need be
GND
3.3V_Out
J10
µC_Out
Output to the µC's of the system
GND
Author: Morgan Whitaker
ECE342 Junior Design
Oregon State University
Sheet: /
File: Power_Conversion_Block.kicad_sch
Title: Power Conversion Module
Size: A4
Date:
Rev: 0
KiCad E.D.A. kicad (6.0.1)
Id: 1/1

- **PCB Layers:**

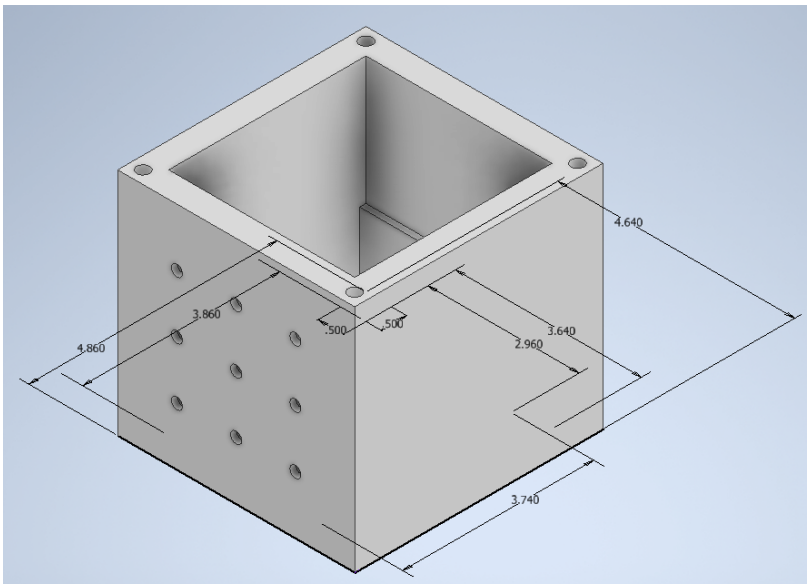

6
GND
J1
12V_Power
J10
µC_Out
J9
Sensor_Out
J3
Left_Signal_In
J5
Brake_Signal_In
J6
J7
J8
R4
16.9k
Cout1
Cout2


6
GND
6V_Out

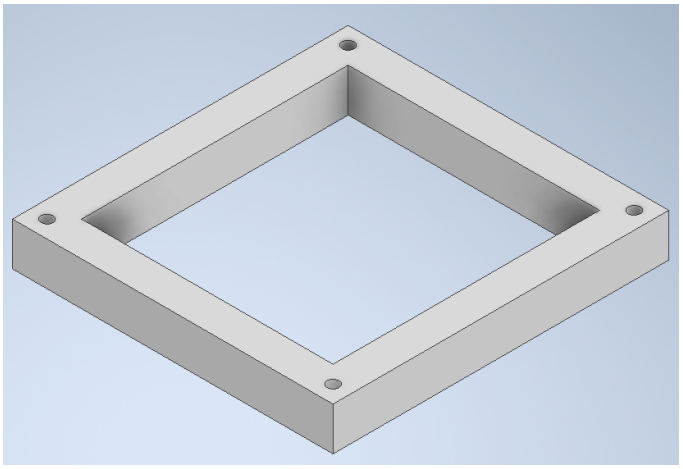
- **Enclosure:**
  - **Block Schematic:**
    - **Battery Enclosure Lid: LWH: 123.4mm x 118.9mm x 15.9mm**



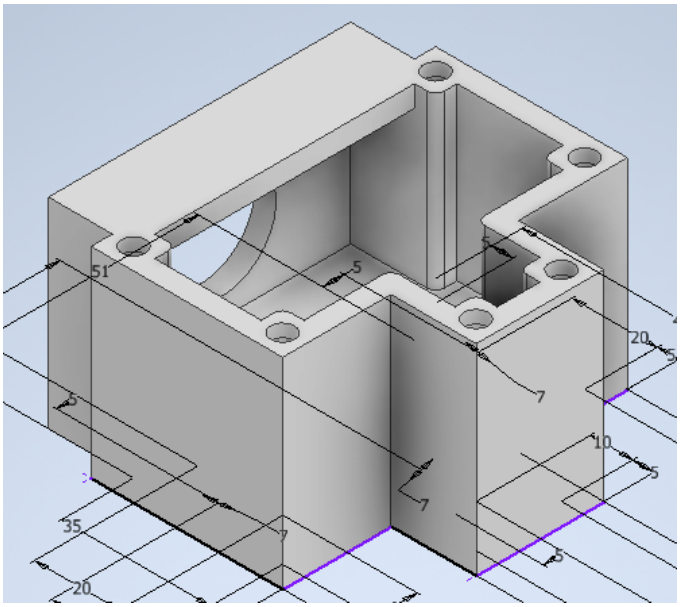
    - **Battery Enclosure: LWH: 123.4mm x 118.9mm x 114.3mm**



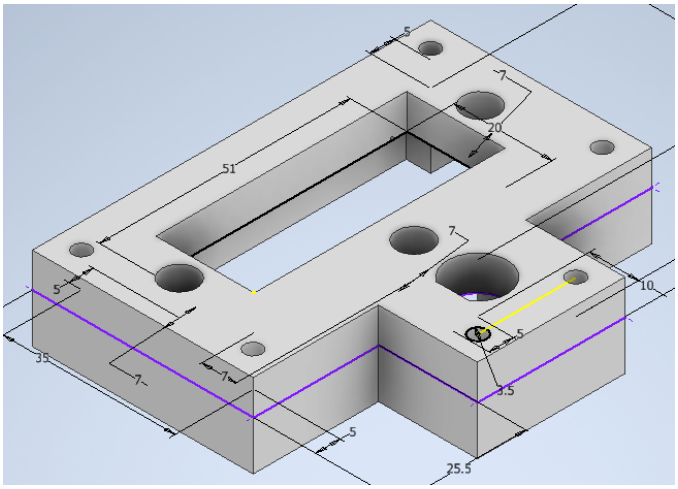
- **Battery Spacer: LWH: 123.4mm x 118.9mm x 15.9mm**



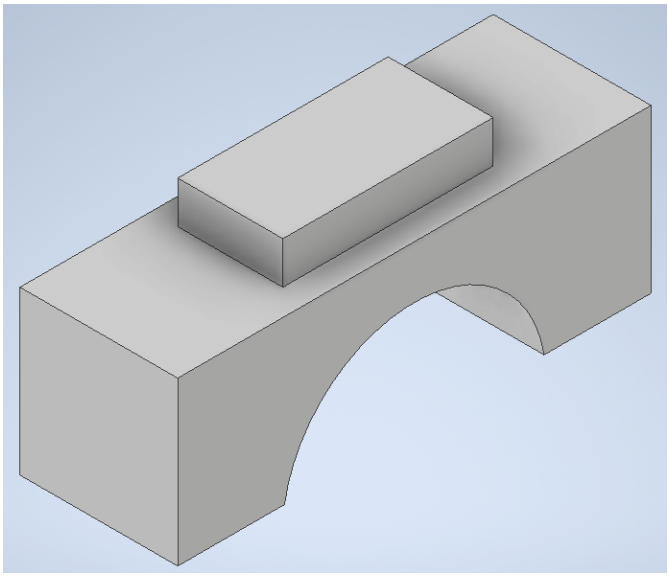
- **Dashboard Enclosure Main Body: LWH: 80mm x 81mm x 47mm**



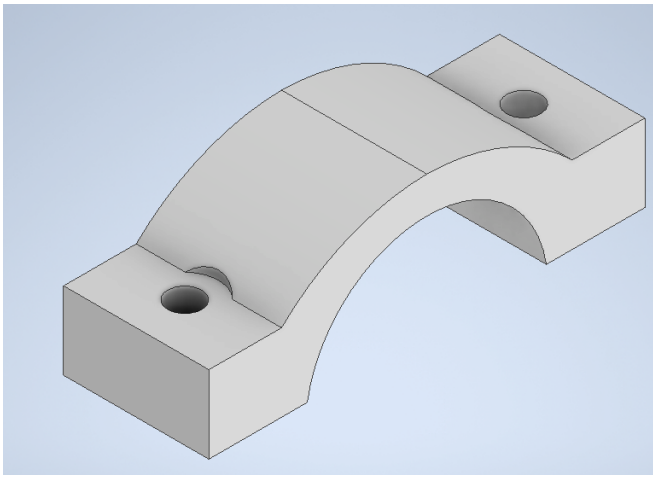
- **Dashboard Enclosure Lid: LWH: 65mm x 81mm x 17mm**



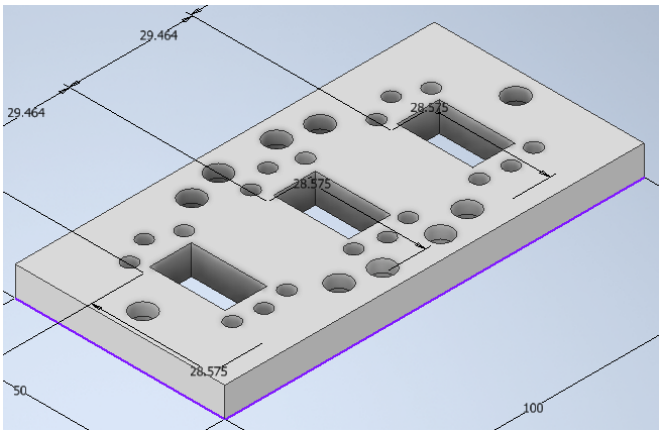
- **Dashboard Biscuit: LWH: 15mm x 45mm x 15.5mm**



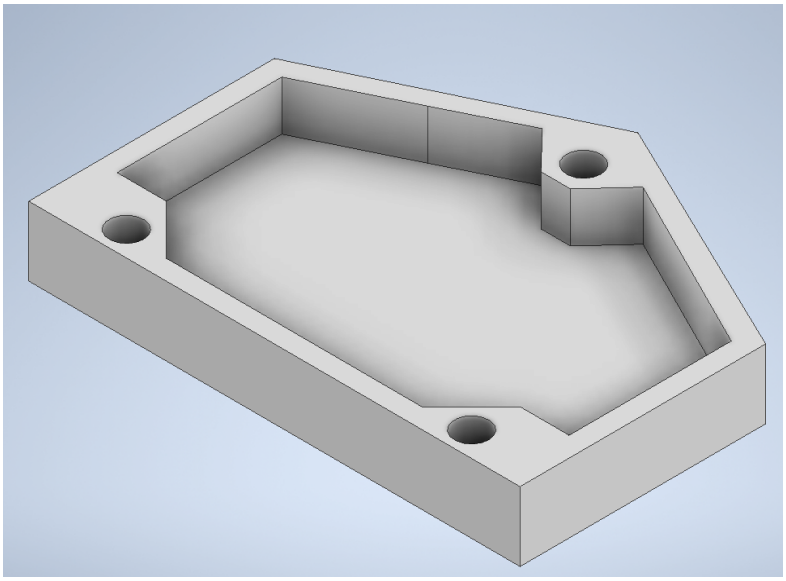
- **Dashboard Bottom Capture: LWH: 15mm x 45mm x 14.2mm**



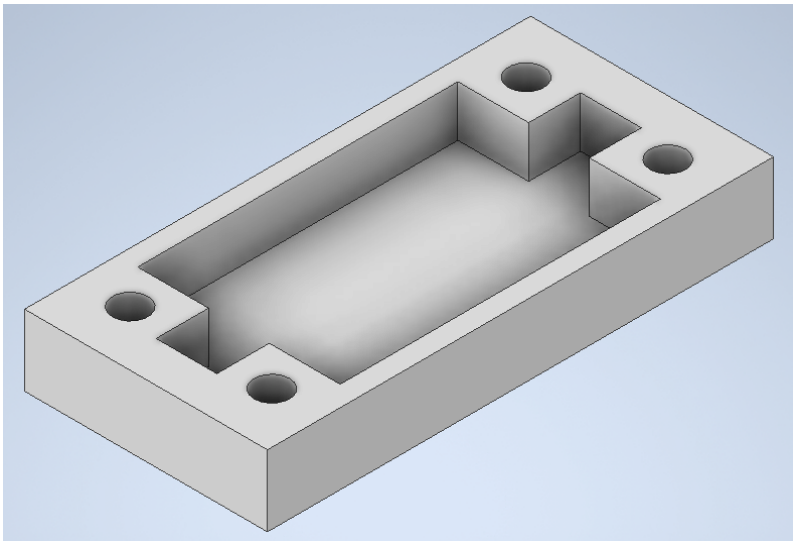
- **Tail Light Bracket: LWH: 50mm x 100mm x 7mm**



- **Left and Right Turn Indicator Cover: LWH: 50mm x 37mm x 7mm**



- **Brake Light Cover: LWH: 24mm x 50mm x 7mm**



Final Code:

```
/*********************************************************
*Author:Cymon Dillon
*Date:2/4/22
*Project:Bike Lights Team 2
*Libraries used: Adafruit_GFX, Adafruit_LEDBackpack,LSM6DS3.h,
*                Wire.h
*********************************************************/
#include <Adafruit_GFX.h>
#include "Adafruit_LEDBackpack.h"
#include "LSM6DS3.h"
#include "Wire.h"
#define LED_BUILTIN 2


LSM6DS3 Accel(I2C_MODE, 0x6A);    //I2C device address 0x6A


float SpeedSensor = 5; //is not 0 if the hall effect sensor senses
                         a full rotation
int LeftSignal = 12; //used for when the left turn signal is
                       turned on
int RightSignal = 13; //used for when the left turn signal is
                        turned on
int LeftLight = 18; //left blinker light
int RightLight = 19; //right blinker light
int BrakeLight = 32; //Brake light
int Brightness = 256; //Controls duty cycle of brake light
int PIN_RED = 4; // pin for red pin of battery led
int PIN_GREEN = 16; // pin for green pin of battery led
int PIN_BLUE = 17; //pin for blue pin of battery led
int BatteryPin = 14; // reads the battery power level



// setting PWM properties
const int freq = 5000;
const int ledChannel = 0;
const int resolution = 8;


int leftCount = 0;//keeps count of how many time the left light
                    blinks
int rightCount = 0;//keeps count of how many times the right
                     light blinks

float Radius = 13;//radius of the wheel in inches
float Distance = (6.28*Radius); //circumference of the wheel
float BikeSpeed = 0;//The speed of the bike in mph
```

```
float BatteryVal = 0; //keeps track of the battery power

volatile float Time = 0; //Keeps Time of time in ms
unsigned long TotalTime = 0;//keeps track of the total program
                            time
unsigned long LastTime = 0;//keeps track of time since last
                           wheel rotation
unsigned long BlinkTime = 0; //tracks how long to blink
unsigned long Lefttime = 0; // tracks when to pull accel data
                             for left turn
unsigned long Righttime = 0; // tracks when to pull accel data
                              for right turn
unsigned long Braketime = 0; // tracks when to pull accel data
                              for brake
unsigned long StopTime = 0; //tracks if the bike is stopped
int RBlinkCount = 180; // stops right blinker each blink is .5
                       seconds
int LBlinkCount = 180; // stops left blinker each blink is .5
                       seconds



volatile bool CalcSpeed = false; // if true, then you find the
                                    time since last rotation
volatile bool LeftBlink = false; //true if the left light should
                                    blink
volatile bool RightBlink = false; //true if right light should
                                     blink
volatile bool BrakeOn = false; //true the bike is slowing down

Adafruit_7segment matrix = Adafruit_7segment(); //sets up
                                                  display

/*********************************************************
*Name:setup
*Purpose:Ran at the beginning of the program. Performs proper
*functions to communicate with the display. Sets a pin as an
*input, then displays all 0's to the display. Also sets up the
*communication to the serial monitor and pwm cahnnels.
*Finally, attaches several interrupts to pins and sets up the
*accelerometer.
*********************************************************/
void setup() {
  matrix.begin(0x70);//sets up display

pinMode(SpeedSensor, INPUT); //Hall Effect sensor for
                               speedometer
pinMode(LeftSignal, INPUT); //Left turn signal
pinMode(RightSignal, INPUT); //Right turn signal
pinMode(LeftLight, OUTPUT);//outputs to left led
pinMode(RightLight, OUTPUT);//outputs to right led
pinMode(BrakeLight, OUTPUT);//outputs to right led
```

```
pinMode(PIN_RED,   OUTPUT);//outputs to red led pin
pinMode(PIN_GREEN, OUTPUT);//outputs to green led pin
pinMode(PIN_BLUE,  OUTPUT);//outputs to blue led pin
pinMode(BatteryPin, INPUT);//reads in battery power


//For the hall effect sensor
attachInterrupt(digitalPinToInterrupt(SpeedSensor), detect, FALLING);

//For left turn signal
attachInterrupt(digitalPinToInterrupt(LeftSignal), leftSignal, RISING);

//For Right turn signal
attachInterrupt(digitalPinToInterrupt(RightSignal), rightSignal, RISING);


matrix.writeDigitNum(0,0); // displays all 0's
matrix.writeDigitNum(1,0,true);
matrix.writeDigitNum(3,0);
matrix.writeDigitNum(4,0);
matrix.writeDisplay();


//Checks that accel is connected
    while (!Serial);
    //Call .begin() to configure the IMUs
    if (Accel.begin() != 0) {
        Serial.println("Device error");
    } else {
        Serial.println("Device OK!");
    }


//Used for pwm signal setup
ledcSetup(ledChannel, freq, resolution);
ledcAttachPin(BrakeLight, ledChannel);
ledcWrite(ledChannel, 0);//starts brake light off

digitalWrite(LeftLight, HIGH); //because board is active low
digitalWrite(RightLight, HIGH);//because board is active low


StopTime = millis(); //set it to the current milliseconds since start

Serial.begin(9600);
}

/*******************************************************
*Name:loop
*Purpose:Calculates the speed og the bike and displays it.
*Then, it checks for turns and braking, and turns the lights on
*and off when necessary.
```

```
**********************************************************/
void loop() {
 TotalTime = millis(); //keeps track of time since started

BatteryVal = analogRead(BatteryPin); //reads in battery power between 0
                                        and 3.3V
CheckBattery();


//If 3 seconds since last rotation
if((TotalTime - StopTime) >= 3000){
  Stopped();
 // Serial.println("stopped");

  }

//If the speed interrupt happened
if(CalcSpeed == true){
  Time = TotalTime - LastTime;
  Time = Time + 250;
  LastTime = TotalTime;
  CalcSpeed = false;

}

//if the time between the interrupts is nonzero
if(Time != 0){
  Display(speedCalc(Time));
}

Time = 0;//reset Time

//if the left light should be blinking, do it
if(LeftBlink == true){
  //pull accel data
  leftBlink(); //blinks light
  LeftCheck(); //checks if left turn happens
  }

//if right light should be blinking, do it
if(RightBlink == true){
  rightBlink(); //blinks light
  RightCheck(); //checks if right turn happens
  }


  BrakeBlink(); //turns on brake light

}

/**********************************************************
*Name:lefttSignal
```

```
*Purpose:Used as the interrupt. If left signal pressed,
*changes the correct variable.
*********************************************************/
void leftSignal(){
  LeftBlink = true; //blinks left light
    RightBlink = false; //blinks right
    rightCount = 0; //resets the right count
  }

/*********************************************************
*Name:rightSignal
*Purpose:Used as the interrupt. If right signal pressed,
*changes the correct variable.
*********************************************************/
void rightSignal(){
  RightBlink = true; //blinks right
   LeftBlink = false; //blinks left
   leftCount = 0; //resets left count
  }

/*********************************************************
*Name:LeftBlink
*Purpose:Turns on the leftt blinker, and blinks it with .5 seconds
*on, .5 seconds off.
*********************************************************/
void leftBlink(){
  digitalWrite(RightLight, HIGH); //turns off right light

  //Turns on lights for 500 ms, then off for 500ms
if((TotalTime - BlinkTime) < 500){
  digitalWrite(LeftLight, LOW);

  }
else if(((TotalTime - BlinkTime) > 500) && (TotalTime - BlinkTime) < 1000){
digitalWrite(LeftLight, HIGH);

}
else if((TotalTime - BlinkTime) > 1000){
 BlinkTime = TotalTime;
leftCount++;

//blink 5 times
  if(leftCount == LBlinkCount){
    LeftBlink = false;
    leftCount = 0;
    }
  }

  }

/*********************************************************
*Name:RightBlink
```

```
*Purpose:Turns on the right blinker, and blinks it with .5 seconds
*on, .5 seconds off.
**********************************************************/
void rightBlink(){
  digitalWrite(LeftLight, HIGH); //turns off left light

  //Turns on lights for 500 ms, then off for 500ms
if((TotalTime - BlinkTime) < 500){
  digitalWrite(RightLight, LOW);

  }
else if(((TotalTime - BlinkTime) > 500) && (TotalTime - BlinkTime) < 1000){
digitalWrite(RightLight, HIGH);

}
else if((TotalTime - BlinkTime) > 1000){
 BlinkTime = TotalTime;
 rightCount++;

 //blink 5 times
  if(rightCount == RBlinkCount){
    RightBlink = false;
    rightCount = 0;
    }
  }

}

/**********************************************************
*Name:BrakeBlink
*Purpose:Turns on the brake lights with correct values.
**********************************************************/
void BrakeBlink(){

//Just reads the accel data every .3 S and also changes brake light
  if(TotalTime - Braketime > 300){
    Serial.print(" AccelY = ");
    Serial.println(Accel.readFloatAccelY(), 4);
int brightCount = 0;
int dimCount = 0;

Serial.println(Brightness);


if(Accel.readFloatAccelY() > .35){ //if speeding up, dim
Brightness += 16;
}
else if(Accel.readFloatAccelY() < -.35){//if slowing down, get brighter
Brightness -= 16;
}
```

```
    Braketime = TotalTime;
//fix overflow
if(Brightness <= 0){
  Brightness = 0;
  }
  else if(Brightness >= 256){
  Brightness = 256;
  }

    ledcWrite(ledChannel, Brightness); //Send pwm signal to brakelight


    }
}

/*********************************************************
*Name:detect
*Purpose:Detects input from the hall effect sensor and changes
*necessary variables
*********************************************************/
void detect(){
  CalcSpeed = true; //tells speed to be calculated
  StopTime = TotalTime;
}

/*********************************************************
*Name:speedCalc
*Purpose:Takes in an elapsed time, and turns it into an MPH
*speed
*********************************************************/
float speedCalc(float elapsed){
   float BikeSpeed = (((Distance)/(elapsed))*56.8182);//find speed in
mph
   Serial.print(" Speed: ");
   Serial.println(BikeSpeed); //for testing purposes
     return BikeSpeed;
    }

/*********************************************************
*Name:Display
*Purpose:Takes in a value, parses out each digit, and displays
*them to the seven-segment display.
*********************************************************/
void Display(float displayNum){
  //turn the value into an int
  int value = displayNum*100;

  //grab each digit
  int d4 = ((value%10));
  value = value/10;
```

```
  int d3 = ((value%10));
  value = value/10;
  int d1 = ((value%10));
  value = value/10;
  int d0 = ((value%10));
  value = value/10;

  //Diplay each bit
  matrix.writeDigitNum(0,d0);
  matrix.writeDigitNum(1,d1, true);//turns on the decimal
  matrix.writeDigitNum(3,d3);
  matrix.writeDigitNum(4,d4);
  matrix.writeDisplay();
  }

/*********************************************************
*Name:LeftCheck
*Purpose:Checks if a left turn has happened, and changes
*necessary variables.
*********************************************************/
void LeftCheck(){
  //Just reads the gyro data every .3 S for now
if(TotalTime - Lefttime > 300){
   Serial.print(" GyroZ = ");
    Serial.println(Accel.readFloatGyroZ(), 4);

   if( Accel.readFloatGyroZ() > 100){
    Serial.println("stop turn");
LBlinkCount = leftCount + 10;//blink for 5 more seconds as turn occurs
   }
    Lefttime = TotalTime;
  }



}

/*********************************************************
*Name:RightCheck
*Purpose:Checks if a right turn has happened, and changes
*necessary variables.
*********************************************************/
void RightCheck(){

//Just reads the gyro data every .3 S for now
if(TotalTime - Righttime > 300){
   Serial.print(" GyroZ = ");
    Serial.println(Accel.readFloatGyroZ(), 4);
     if( Accel.readFloatGyroZ() < -100){
      Serial.println("stop turn");
RBlinkCount = rightCount + 10;//blink for 5 more seconds as turn occurs
   }
```

```
    Righttime = TotalTime;
  }
}

/*********************************************************
*Name:CheckBattery
*Purpose:Checks the battery power and changes led color
*********************************************************/
void CheckBattery(){

Serial.println (BatteryVal);
if(BatteryVal >= 2700){ //if battery between 66%-100%
  //display blue
  digitalWrite(PIN_RED, LOW);
  digitalWrite(PIN_GREEN, LOW);
  digitalWrite(PIN_BLUE, HIGH);

  }
else if(BatteryVal >= 1400){ //if battery between 66%-33%
  //display green
  digitalWrite(PIN_RED, LOW);
  digitalWrite(PIN_GREEN, HIGH);
  digitalWrite(PIN_BLUE, LOW);
  }
else{  //if battery between 33%-0%
  //display red
  digitalWrite(PIN_RED, HIGH);
  digitalWrite(PIN_GREEN, LOW);
  digitalWrite(PIN_BLUE, LOW);
  }
}



/*********************************************************
*Name:Stopped
*Purpose:When bike is stopped, display a speed of 0, and turn
*on the brake light
*********************************************************/
void Stopped(){
matrix.writeDigitNum(0,0); // displays all 0's
matrix.writeDigitNum(1,0,true);
matrix.writeDigitNum(3,0);
matrix.writeDigitNum(4,0);
matrix.writeDisplay();
StopTime = TotalTime;

Brightness = 0; // turn brake lights fully on
   ledcWrite(ledChannel, Brightness); //Send pwm signal to brakelight

  }
```