Bicycle Automated Turn Signal (BATSignal)

Natalya Bapst (bapst2)
Bora Cukurova (cukurov2)
Hanna Zayed (jhzayed2)

Design Review – ECE 445, Fall 2016
TA: Zipeng Wang
10/14/16
Group No. 3

Table of Contents

1. Introduction…….....3
1.1 Statement of Purpose……..……...…..3
1.2 Objectives……..……...…3

2. Design and Block Diagrams……...4
2.1 Microcontroller……...4
2.2 MPU-6050 (Gyroscope & Accelerometer).....5
2.3 Power……..7
2.4 RF Transmitter……8
2.5 RF Receiver……....9
2.6 External Interface……..……9
2.7 Turn Signal Switches……10
2.8 SD Chip……..10

3. Calculations…….....11

4. Tolerance Analysis…….....13

5. Requirements and Verifications…….……...15

6. Parts & Schedule…….17
5.1 Parts……..…17
5.2 Schedule……..18

7. Ethics & Safety……....19

8. Citations……20

9. Appendix……21

1. Introduction

1.1 Motivation/Statement of Purpose:

Cyclists are often required to use their hands to indicate when they are about to make a turn. This is often difficult and dangerous, especially when biking at nighttime. Several products exist on the market for bicycle turn signals, however all of these products force the cyclist to have to move their hands off of their handlebars to indicate a turn. The current products on the market require the cyclist to have to use a control box located in between the handlebars. This is dangerous for the cyclist to be looking at the controls as opposed to the road. The purpose of our project is to create a wireless turn signal for bikers to use without needing to move their hands off the handlebars.

1.2 Objectives:

Goals:

- Ability to make turns while keeping hands on the handlebars and eyes on the road
- Notify a driver behind when braking or slowing down
- Ability for turn signal to be visible to a driver behind during night times

Functions:

- Wirelessly controlled turn signal
- After the turn is completed, turn signal automatically turns off
- Brake signal using an accelerometer
- Cyclist is notified while the turn signal remains on

Benefits:

- Cyclist can keep his hands on the handlebar at all times
- Cyclist will avoid dangerous situations that might arise from making a turn on a road
- Ability to ride safely at night

Switch turns off automatically after turn is completed

Features:

- Easily mountable bicycle turn signal
- Easily readable LED screen that shows when the turn signal remains on for the cyclist's convenience
- Bluetooth compatibility with the turn signal and turn signal switches

2. Design

Overall system block diagram:

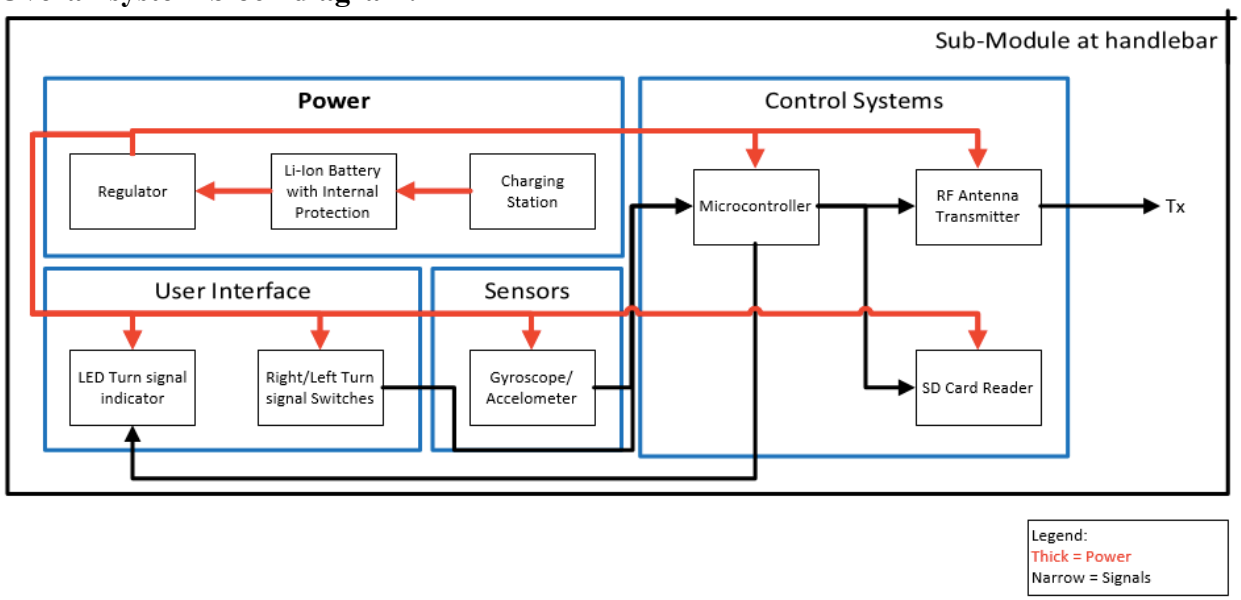


Figure 1 - Block Diagram of Handle-Bar Module

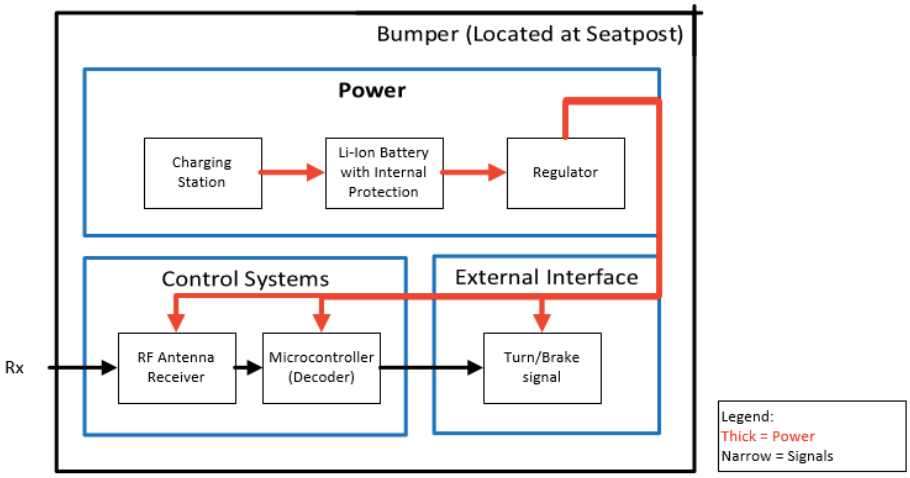


Figure 2 - Block Diagram of Seat-Post Module

2.1.a Microcontroller

The microcontroller (ATMEGA328P-U) will collect all the inputs coming from the turn signal switches and MPU6050 (gyroscope + accelerometer). Also, it outputs signals to the Transmitter, that transmit data to the Turn/Brake signals located at the seatpost. The turn switches will be connected to microcontrollers digital pins (pin 7 and 8) to get the input data. MPU6050 will use the serial connection, I2C (SDA pin 27, SCL pin 28) to input the accelerometer and gyroscope data and will use a digital pin 2 (pin 4) to trigger interrupts on the microcontroller. The microcontroller will use UART to communicate with the Transmitter. It will use the serial Tx (pin 3) connection of the microcontroller. The microcontroller will also control the sleep mode of

the Transmitter by controlling its PDN pin. It will put the transmitter into sleep mode when it is not transmitting data. Also, the microcontroller will use a 16MHz crystal as its frequency-determining component. The crystal will be grounded using 22pF capacitors as seen in the Figure below. The microcontroller uses to single female pin connected to Rx (pin2) with the sole purpose of loading program to the microcontroller. Additionally, it will have a reset switch (pin 1) to restart the program. The microcontroller will be powered from a 3.3V source that comes from the Power Module.

At any time, the microcontroller will read the accelerometer data of the bicycle. Once it determines that the bicycle decelerates above the threshold acceleration in -x-axis of the accelerometer, then it will transmit signal to the Transmitter to turn on the Brake signal. Another, function of the microcontroller is, once the biker presses the turn signal, it will output data to the transmitter to turn the corresponding turn signal. Then, it will start to analyze the MPU6050's gyroscope data. It will measure the yaw, pitch and roll angles of the bicycle to determine the turn. Once it detects that the turn is completed, it will send signal to the transmitter to turn off the turn signal automatically.

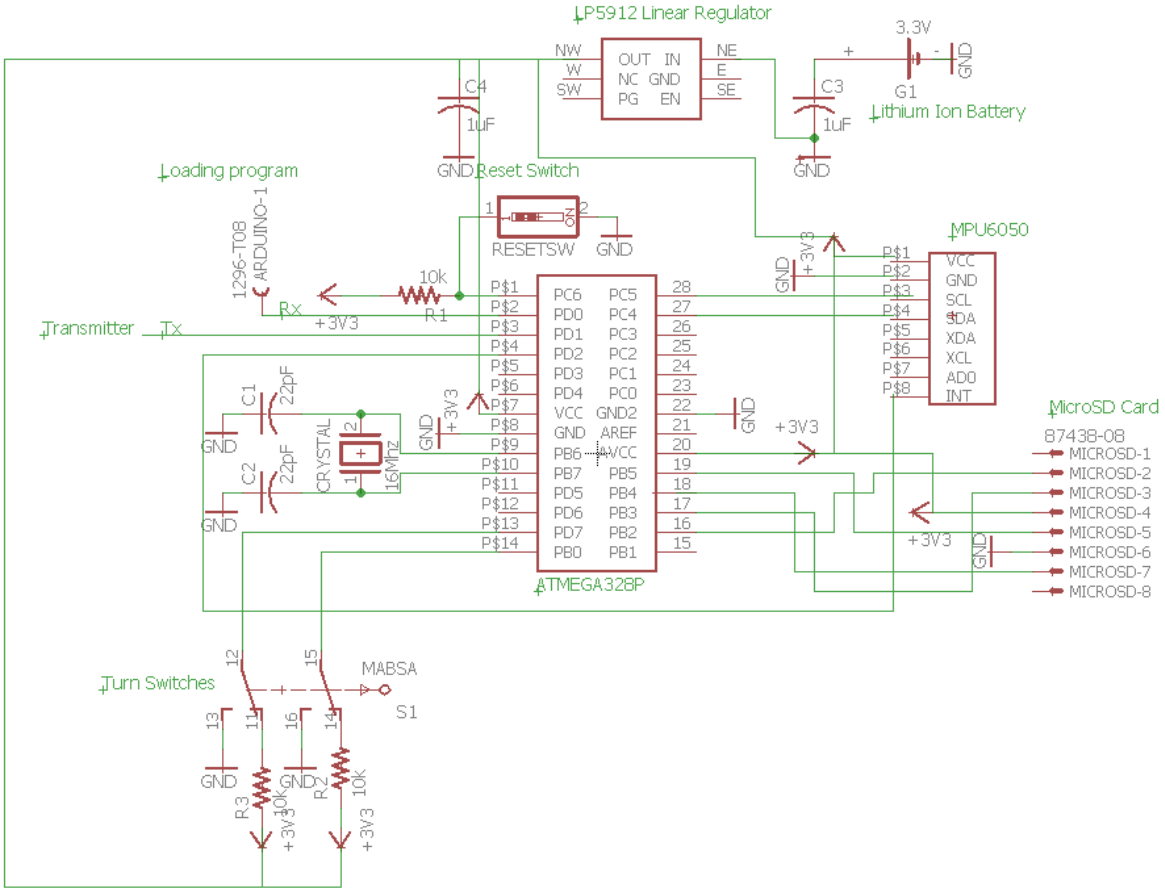


Figure 3 - Microcontroller Module Circuit

2.1.b Microcontroller (Decoder)

The microcontroller, that is located at the seatpost will receive a serial data from the Receiver. It will decode the data and feed it to the Turn/Brake signals located in the back end of the bicycle using the pins 15, 16 and 17.

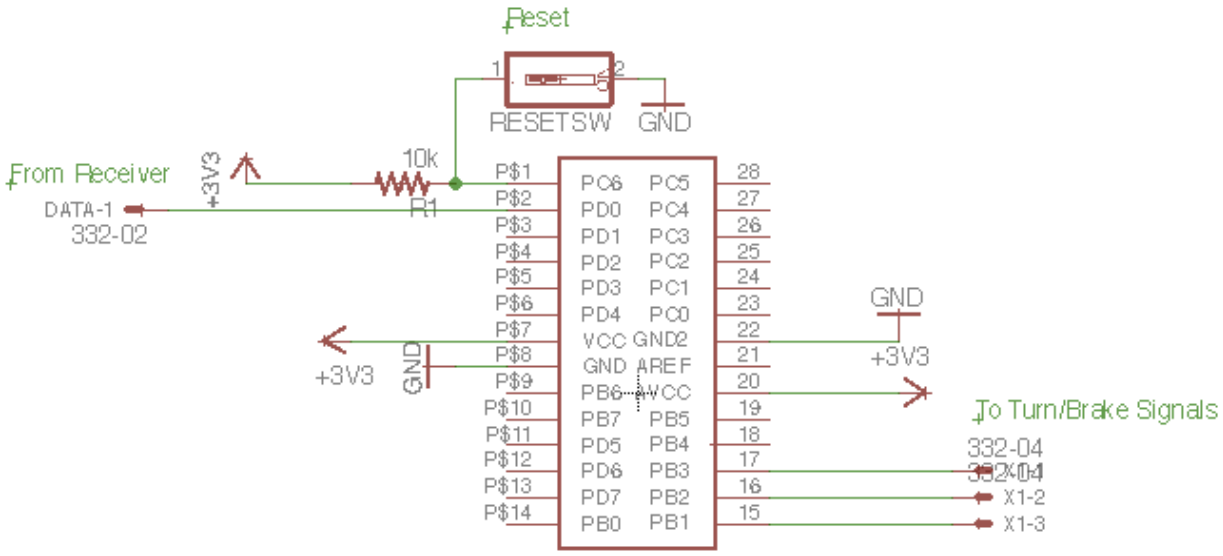


Figure 4 - Microcontroller Module (Decoder)

2.2 MPU-6050 (Gyroscope & Accelerometer):

The InvenSense MPU-6050 chip will be used for this portion of the design. The chip includes a total of 6 axis, 3 axis for gyroscope as well as 3 axis of accelerometer. It will communicate with the microcontroller by a I2C serial connection (SDA, SCL). Also, it will send an interrupt signal to microcontroller to indicate that there are data ready to be transferred from the FIFO buffer register to the microcontroller. This module will analyze the bicycle motion to determine whether the cyclist is applying brakes or making a turn.

The design will be powered by a 3.3V power from the Power Module. The electronic design can be seen from the figures and table below.

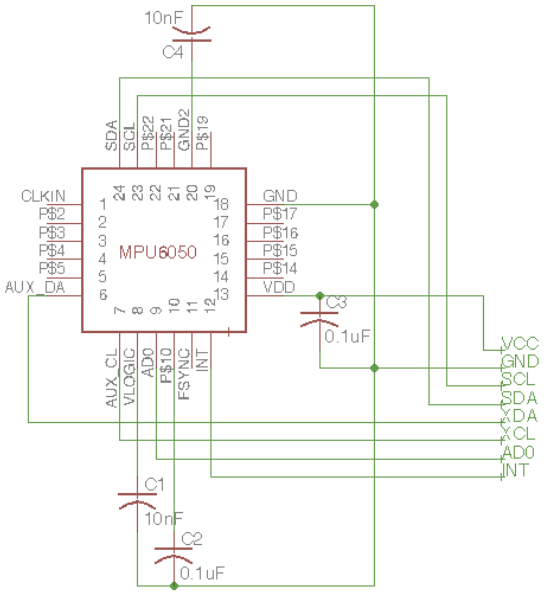


Figure 5 - Detailed MPU6050 Circuit

This module will analyze the bicycle motion. It will detect when a turn is taking place and when brakes are applied. The gyroscope feature of the MPU6050 chip will detect that cyclist making a turn by the measurement of the yaw-pitch-roll angle change. As seen in the figure below, the cyclist takes a turn between the 209 - 287 iterations of the turn. During the 90 Degree turn, we can observe that the yaw angle of the bicycle also shifted 90 degrees in the positive direction. Also, there was a temporary shift in the Pitch and Roll angles during the turn since cyclist also tilted to the side of the turn and rotate the handle bar. Therefore, during a turn, gyroscope reading shows the degree turn up to desired accuracy and it also provides shift in angle in pitch and roll angles to separate an actual turn from a slight change in direction over time.

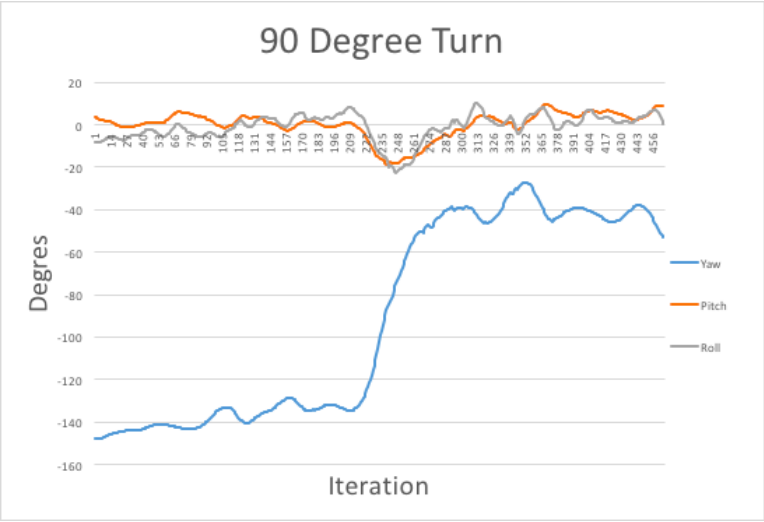


Figure 6 - Gyroscope reading during a 90 Degree turn to left

We will also be reading the accelerometer of MPU6050 chip to detect cyclist braking. As the cyclist decelerates higher than some threshold acceleration, which will be determined by testing braking acceleration data under various trains and conditions like asphalt, dirt, flat surface, downhill or uphill, the microcontroller will understand that the cyclist is braking.

2.3 Power:

The power supply supplies power to each component block and consists of a single lithium ion battery (3.7 V, 2000 mAh nominal). We will attach this to an LP5912 linear regulator in order to regulate the voltage at a constant 3.3V. We will require two of these power supplies, one for the circuitry at the handlebars and one for the circuitry at the bumper. The size of the overall design will be small since we are only using a single battery for each power supply. This circuit should deliver around 7.4 Watts/hr which should be enough to power each chip in our design:

$$(2000 \text{ mAh} * 3.7\text{V})/1000 = 7.4 \text{ Wh}$$

The schematic for the linear regulator is shown below with pin outs, taken from the TI datasheet for LP512:

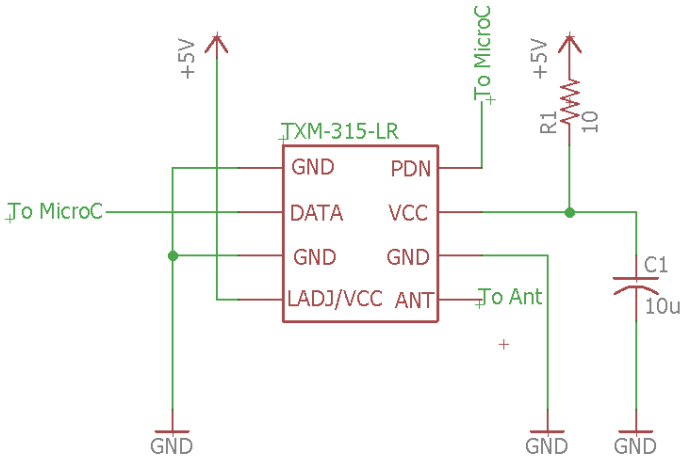


Figure 4 – Power System EAGLE Schematic

The simulation for Vin from 3.8 to 4.4V and output voltage ripple measurements is shown below, taken from the TI datasheet for LP512:

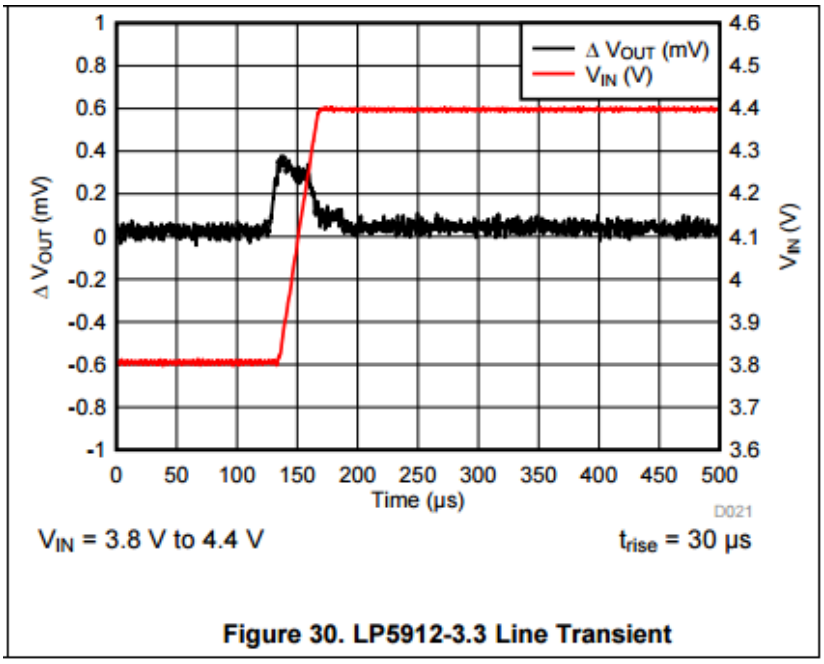


Figure 7 - Power Plot for Vin from 3.8V to 4.4V and Output Voltage Ripple

2.4 RF Receiver:

This block represents the receiving end of our front-end/back- end communication link. It consists of a single Linx RXM-315- LR chip, a simple power regulator for the chip, and a receiving antenna. The purpose of the receiver module is to pick up on the signals sent by the corresponding transmitter module mounted on the front-end device. It will then deliver the digital data to the External Interface block to that the proper turn signal and/or brake signal can be displayed. The communication link in this project consists of the receiver continuously running and searching for signals at 315 MHz. As such, the PDN pin of the receiver will be pulled up to Vcc.

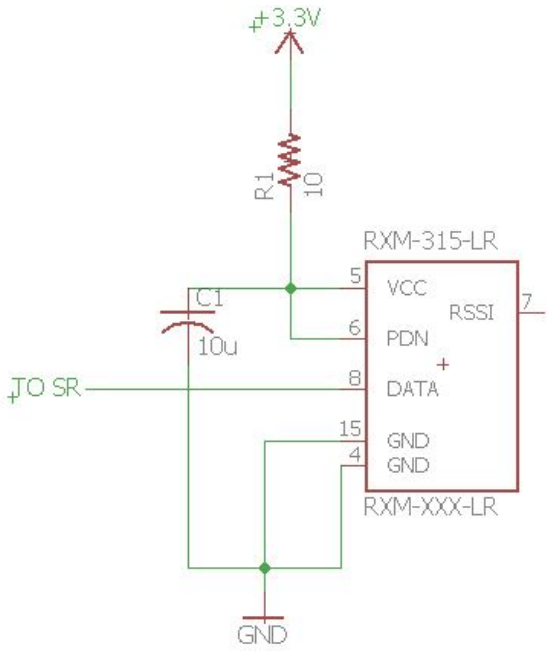
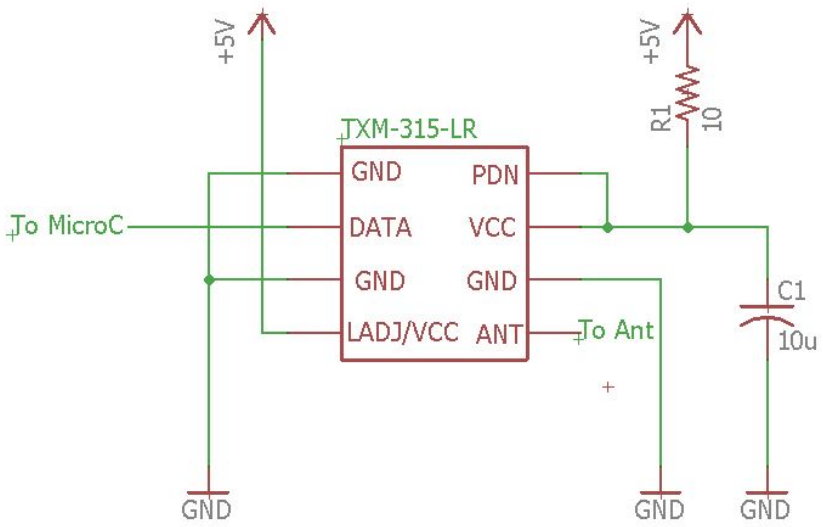


Figure 8 - RF Receiver RXM-315- LR

2.5 RF Transmitter

This block represents the transmitting end of our front-end/back-end communication link. It consists of a single Linx TXM-315-LR chip, a simple power regulator for the chip, and a transmitting antenna. The purpose of the transmitter module is to receive the signals sent by the microcontroller mounted on the front-end device and send them wirelessly to be picked up by the receiving module on the back-end device. In the communication link implemented in this project, the transmitter PDN pin will be connected to a digital output of the microcontroller. This will allow the transmitter to be turned on and to send signals only when it is needed (as dictated by the microcontroller), thus conserving power on the front-end device.



(Fix it)

Figure 9 – RF Transmitter TXM-315-LR

2.6 External Interface:

This block represents the device mounted on the rear of the bike. The board will be a PCB consisting of the LEDs and current-limiting resistors soldered onto it. The function of the board is to convey the turn signals to people behind the bicycle. The PCB will receive power from a battery mounted on the rear-device and will receive the turn signal data from the receiver, also mounted on the rear-device.

The turn signals will be composed of LEDs, where the left and right turn signals will each consist of 5 ultra-bright yellow LEDs arranged in an arrow structure, and the brake signal will consist of 4 ultra-bright red LEDs arranged in a square structure. This is shown below in Figure 8.

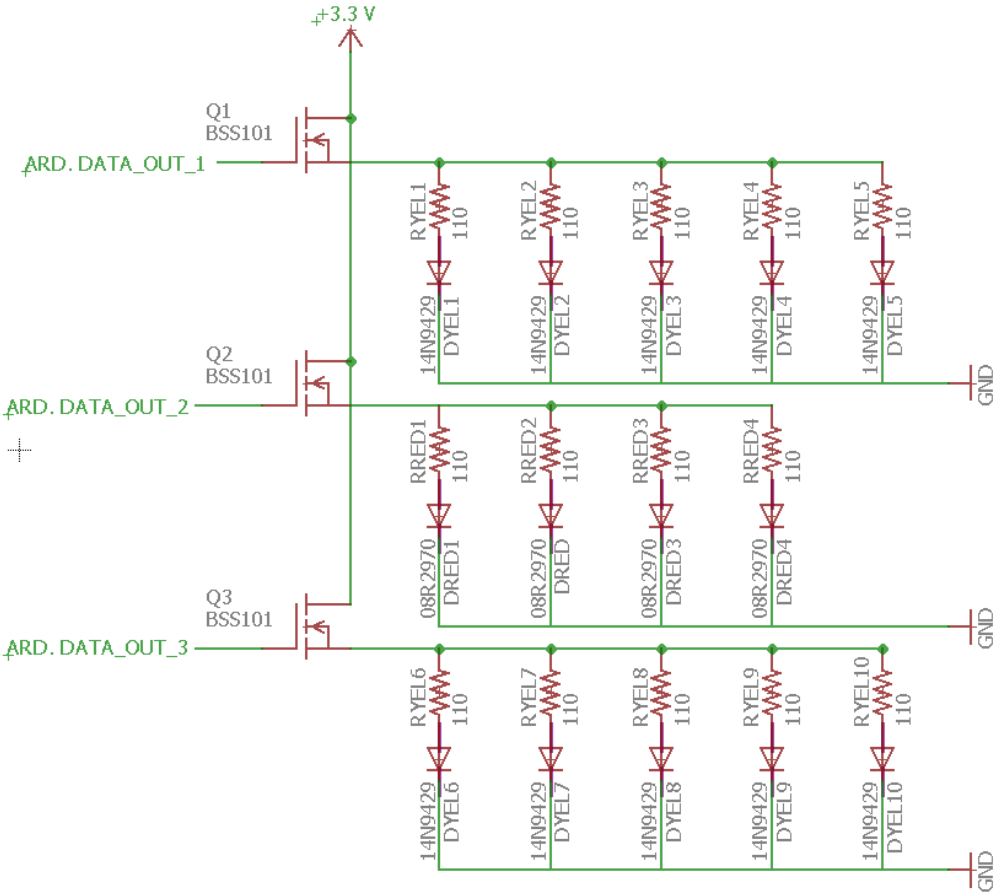


Figure 10 - Turn/Brake Signals with LED Design

The PCB will be mounted by attaching a universal-size clamp onto the bicycle seat post. Then an extender rod will be attached to the clamp so that the PCB can be situated behind the bicycle seat. See Figure 9 for a rough idea of this mechanical mounting scheme.

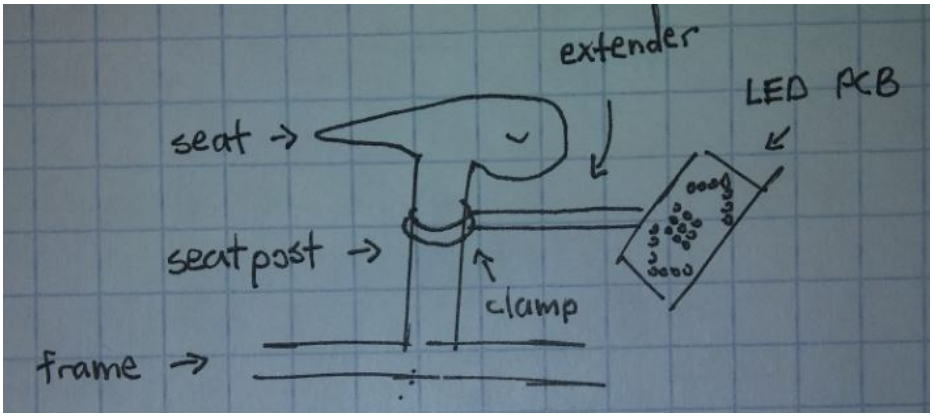


Figure 11 - Seat Post Module Design

2.7 Right/Left Turn Signal Switches:

This will be located on the handlebar of the bicycle. It will be located at a reachable distance from the cyclist's hands. The switches will send input to the microcontroller about the state of the switches.

2.8 SD Card:

This module will be required for debugging purposes. It will be useful in recording the data we read from the MPU6050 chip to analyze the data further. The SPI method for writing data will be implemented. The circuit diagram of the module can be seen from Figure 3.

3. Calculations

MPU-6050 Power Calculation:

The MPU6050 will be powered by 3.3V source from the Power Module. It requires a range of 2.375 - 3.46 V to operate. In order to operate Accelerometer + Gyroscope of the MPU6050 chip, it requires a current of about 3.8A.

Power Required = 3.8 mA x 3.3V = 12.54mW

Microcontroller Power Calculations:

The Microcontroller will be powered by 3.3V's
Current consumption at 3.3V and Active is 1.7-3.5mA
Current consumption at 3.3V and Idle is 0.3-1.5mA

Therefore, when microcontroller is active:

Maximum Power = 3.3V x 3.5mA = 11.55 mW
Minimum Power = 3.3V x 1.7mA = 5.61 mW

When microcontroller is idle:

Maximum Power = 3.3V x 1.5mA = 4.95 mW
Minimum Power = 3.3V x 0.3mA = 0.99 mW

Receiver Power Calculation:

The values used in the following calculations were gathered from the RXM-315-LR datasheet.

$$P_{Rx,min} = V_{min}I_{min} = (2.7\ V)(4.0\ mA) = 10.8\ mW$$

$$P_{Rx,typ} = V_{typ}I_{typ} = (3.0\ V)(5.2\ mA) = 15.6\ mW$$

$$P_{Rx,max} = V_{max}I_{max} = (3.6\ V)(7.0\ mA) = 25.2\ mW$$

Transmitter Power Calculation:

The values used in the following calculations were gathered from the TXM-315-LR datasheet.

$$P_{Tx,min} = V_{min}I_{min} = (2.1\ V)(3.4\ mA) = 7.1\ mW$$

$$P_{Tx,typ} = V_{typ}I_{typ} = (3.0\ V)(3.4\ mA) = 10.2\ mW$$

$$P_{Tx,max} = V_{max}I_{max} = (3.6\ V)(3.4\ mA) = 12.2\ mW$$

Calculations for Program Size:

Since the design of the program is not finalized, the minimum parameters are given below.
Program Size: 14,848 bytes/32,256 bytes
Global Variables: 578 bytes of dynamic memory
Local Variable: 1470 bytes

External Interface Calculation:

According to the LED data sheets, the maximum power dissipation for the red LED is 130 mW and the maximum power dissipation for the yellow LED is 85 mW. Furthermore, from the BSS101 MOSFET datasheet, we gather that the maximum power dissipation is 0.60 W. With these figures in hand, we can calculate the worst case power consumption for the external interface as follows:

$$P_{tot,worst} = P_{res}N_{res} + P_{max,FET}N_{FET} + P_{YLED}N_{YLED} + P_{RLED}N_{RLED}$$

$$= (0.25\ W)(3) + (0.60\ W)(3) + (10)(85\ mW) + (4)(130\ mW)$$

$$= 3.92\ W$$

Since we will be driving the LEDs with 20 mA of current, we can calculate their typical power consumption by multiplying the typical drive current with the 3.6 V operating voltage. Furthermore, we gather the typical power consumption for the MOSFETs from the datasheets as 0.45 W. Thus, we calculate the typical power consumption as follows:

$$P_{tot,avg} = (0.25\ W)(3) + (0.60\ W)(3) + (10)(45\ mW) + (4)(64\ mW) = 3.26\ W$$

4. Tolerance Analysis

Critical Component: Turn Signal Completion Detection
Acceptable Tolerance:

We have determined that the most critical portion of our turn signal is that it can detect that a turn has been completed and that the signal is turned off automatically. An acceptable tolerance would be detection of a turn angle at 90 degrees (turning onto a street). The angle of the bicycle frame (the Yaw) should experience a turn of 90 degrees with a 10% tolerance. Furthermore, the tilt angle should be 15 degrees with 10% tolerance. Finally, the handlebar turn angle should be 45 degrees with tolerance 10%. This tolerance level makes it unable to determine when the cyclist is changing lanes. It will only determine if the cyclist is making a full right or left turn.

Test Procedure:

We will test the bicycle on the road at a four-way intersection. Using an Arduino attached to an MPU-6050 chip sitting in the middle of the bicycle's handlebars, we tested the handlebar angle of turn, the tilt of the bike from side to side, and finally the turn angle of the entire bicycle's frame. From our data, we determined that the most effective way to determine if a bicycle has completed a turn is from the entire bicycle's frame angle of turn. This can be seen in the third graph titled "90 degree turn."

Presentation of Results:

We present our results in three graphs of data that show the angle of the bicycle's tilt, handlebars, and frame respectively over time. It can be seen that the turn angle for tilt is around 17 degrees (Figure 10) which falls within our tolerance. The yaw angle is 89 degrees which falls within our tolerance (Figure 11). The handlebar turn angle is 42 degrees which also falls within our range of acceptance (Figure 12).

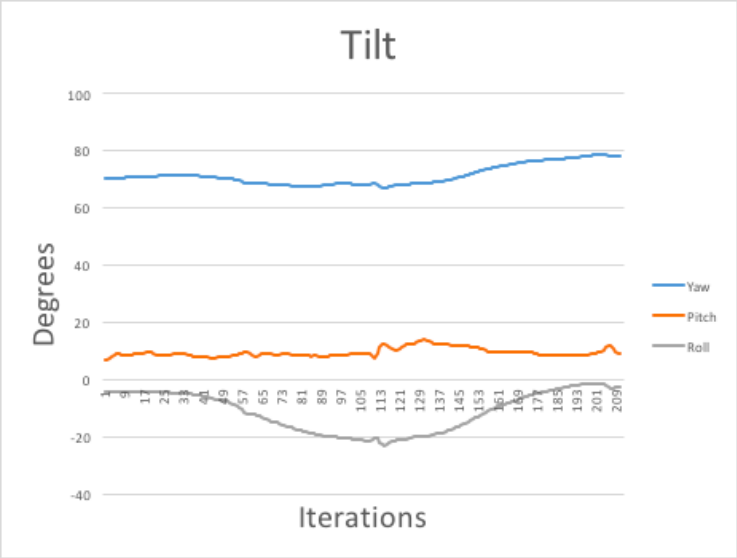


Figure 10 – Gyroscope reading for tilting bicycle

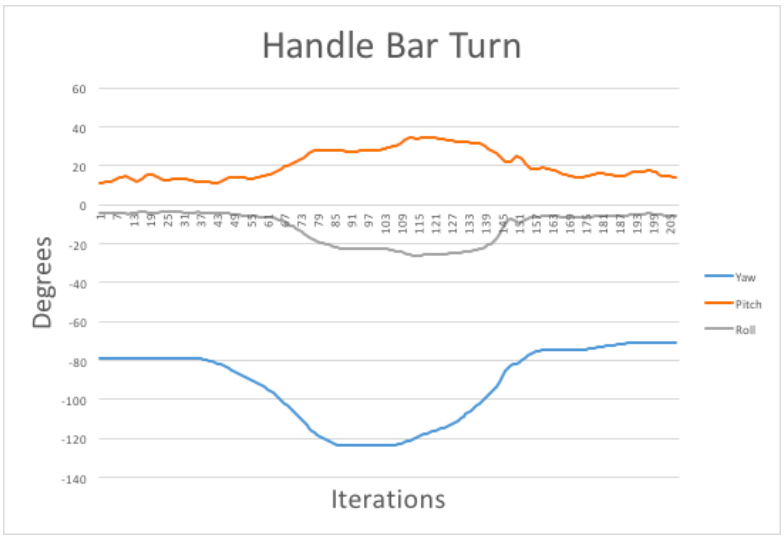


Figure 11 - Gyroscope reading for rotating handle bar of bicycle

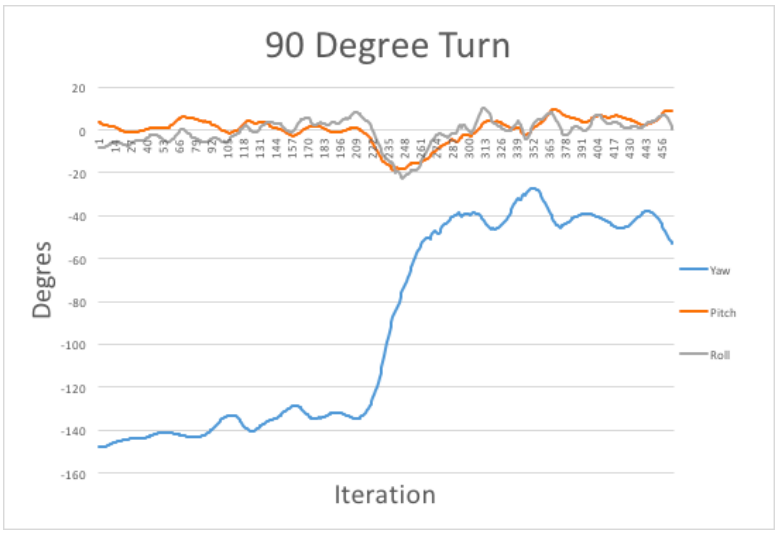


Figure 12 - Gyroscope reading for 90 Degree turn with a bicycle

5. Requirements and Verifications

Component	Requirements	Verifications
Power	**1.** The power supply voltage is 3.3V with a +/- 5% tolerance **2.** The power supply outputs 7.4 Wh with a +/- 5% tolerance	**1a**. Measure power supply output with oscilloscope to determine voltage ripple **1b**. Measure power supply output with multimeter to determine average voltage output **2a**. Measure power supply output with multimeter to determine average power
Microcontroller	**1.** The microcontroller has UART functionality for transmitting 16-bit serial bits using a 9600 baud rate to the transmitter when turn, left or right signal needs to be on. **2.** Timer interrupts should be provided every 0.5s for the turn signals. **3.** The software can read the data from the MPU6050 and switches and provide correct feedback to the back end microcontroller through Tx/Rx module.	**1a**. Send a 16-bit message from the front microcontroller to the transmitter **1b.** Use an oscilloscope to read the message that is being send to the DATA pin of the transmitter. **1c.** See if the serial data being send and the serial data that can be read matches. **2a.** Set a Timer interrupt every 0.5 seconds. **2b.** At every interrupt, toggle the value of the signal. **2c.** Continuously send this message through one of the microcontroller pin. **2d.** Use an oscilloscope to check to see a square wave with a period of 1s. **3a**. Confirm that once, either the left turn or right turn switches is pressed, corresponding turn lights toggles. **3b**. Once the cyclist applies a brake, the brake light flashes and stop flashing once the braking stops. **3c**. Once the cyclist completes a turn greater than 60 Degrees, the corresponding turn signal stops signalling automatically with 90% efficiency.

Receiver and transmitter	**1.** The communication link can transmit and receive signals of three bits over the minimum length of the bike necessary (34 inches) **2.** The PDN (power down) pin of the transmitter operates as intended	**1a.** Program the microcontroller to send each 3-bit word a total of 5 times (total of 40 sent signals) to the transmitter. **1b.** Verify that the receiver, which will be at least 34 inches away, correctly picks up on at least 35 of the transmitted signals by connecting an oscilloscope to the output data pin of the receiver. **2a.** Program the microcontroller to send (10) 3-bit signals to the transmitter WHILE the PDN pin is pulled high. Verify the transmitter sends out these signals by connecting a scope to the ANT pin of the transmitter. **2b**. Program the microcontroller to send (10) 3-bit signals to the transmitter WHILE the PDN pin is pulled low. Verify the transmitter does not send out any signals by connecting a scope to the ANT pin of the transmitter.
Accelerometer	**1.** Accelerometer can detect a full scale resolution of -2g to 2g acceleration **2.** It can detect when the cyclist is braking, decelerating at minimum 0.4g acceleration.	**1a.** We will measure the RPM of the bicycle wheel using an external accelerometer and calculate the acceleration. **1b**. Compare this result to the accelerometer used in our circuit **2a**. The sensor can detect the cyclist braking 80% of the time when the brake is engaged.
Gyroscope	**1.** Gyroscope can detect yaw, pitch and roll angles with a maximum bias error of 3% (~10.8 Degrees) **2.** Gyroscope should be able to detect a minimum of 180 Degrees of turn per second with a maximum	**1a.** Bring the gyroscope value of the angle at test to ~0 Degrees. **1b.** Use a gauge to specify turn angles every 10 Degrees for a 360 turn. **1c.** Slowly rotate the gyroscope to check whether desired rotation angles matches the desired angle markings.

	of 10 Degree/second error.	**2a**. Bring the gyroscope so that the angle at test reads a value of 0 Degrees. **2b.** Shift the gyroscope continuously on the ground every 1 second at a 180 Degree angle. **2c.** Read the value from the gyroscope and check whether it can accurately match the desired 180 degrees.
User Interface	**1.** The left and right switches on the handlebars interface with the microcontroller successfully with no debouncing	**1a**. Connect microcontroller to two testing LEDs. **1b.** In the microcontroller, match the input from the switches to the output that goes to the LEDs. **1c.** Test whether switches functions properly by comparing with the LEDs. **1d.** Also, count the number of times button is pressed and compare the results to make sure there is no debounce error.
External Interface	**1.** Each of the three LED blocks turns on/off when dictated by the back end microcontroller.	**1a**. Program the back microcontroller to send a square wave with period of 1 second and duration of 10 seconds. to each of the three MOSFETs. Verify that all LEDs turn on and off correspondingly. Repeat 3 times.

Point Distribution:

Module Name	**High Level Requirement**	**Points**
Microcontroller	-This module should successfully gather inputs from the switches and MPU6050 chip.	10 points

	-It should successfully analyze the data to determine the state of the bicycle (making a turn, finished the turn, braking) -It should be in communication with the Transmitter to control the turn signals.	
Microcontroller (Decoder)	This module should successfully read and decode the serial data from the receiver and control the blinking conditions of the Turn/Brake signals accordingly.	5 points
Transmitter	This module should successfully do a serial data transfer from microcontroller to receiver	7.5 points
Receiver	This module should successfully receive a serial data transfer to microcontroller to receiver	7.5 points
MPU6050	This module should successfully read the yaw, pitch, roll angles and the acceleration of the bicycle in the x, y and z axis.	5 points
Power to Front	This module should successfully supply steady state power to to the front-end circuit.	5 points
Power to Back	This module should successfully supply steady state power to the back-end circuit	5 points
External Interface (LEDs)	This module should successfully display the turn/brake signals that is easily visible to the surrounding	5 points
	Total	50 points

6. Parts & Schedule

6.1 Parts:

Part	Part Number	Unit Cost	Quantity	Total
LEDs - bought	LED SuperBright	$.1475	20	$2.95
Microcontroller - bought	Atmel ATMEGA328P-PU	$6.49	2	$12.98
Lithium Ion Battery (3.6V) - bought	Panasonic CGR19650CG	$5.99	2	$11.98
Capacitors - bought	Varies	$0.25	20	$5.00
16 MHz Crystal - **bought**	HC49/US Package	$0.95	1	$0.95
Resistors - **bought**	Varies	$.0475	30	$1.43
Antenna	Link Kit Module 433 MHz	$1.29	1	$1.29
Inductors	Varies	$0.19	10	$1.90
Accelerometer/Gyroscope Chip - **bought**	InvenSense MPU-6050	$2.95	1	$2.95
Switches	LED Tactile Button	$1.95	2	$3.90
Linear Regulator - **bought**	595-LP5912-1.8DRVR	$1.18	2	$2.36
Total				**$47.69**

6.2 Schedule:

Week	Task	Responsibility
9/11	Finalize and hand in proposal	Natalya Bapst
	Mock Design Review Sign up	Hanna Zayed
	Plan parts needed to be ordered	Bora Cukurova
9/18	Design receiver/transmitter	Hanna Zayed
	Order parts/Mock Design Review	Natalya Bapst
	Create circuit diagrams (EAGLE)	Bora Cukurova
9/25	Finalize circuit diagrams	Natalya Bapst
	Commence breadboarding	Bora Cukurova
	Design receiver/transmitter	Hanna Zayed
10/2	Breadboarding arduino	Bora Cukurova
	Breadboarding antenna	Hanna Zayed
	Design Review	Natalya Bapst
10/9	Determine antenna parameters	Hanna Zayed
	Determine MPU6050 parameters	Bora Cukurova
	Create Power System in EAGLE	Natalya Bapst
10/16	LED Assembly	Hanna Zayed
	Microcontroller Coding	Bora Cukurova
	Finalize PCB Design	Natalya Bapst
10/23	Antenna Assembly	Hanna Zayed
	Microcontroller Debugging	Bora Cukurova
	Microcontroller Assembly/Order PCB	Natalya Bapst
10/30	Antenna Assembly	Hanna Zayed
	Microcontroller Assembly	Bora Cukurova
	Antenna Testing	Natalya Bapst
11/6	Debugging Antenna	Hanna Zayed
	Debugging Code	Bora Cukurova
	3D Printing Casings	Natalya Bapst
11/13	Debugging Antenna	Hanna Zayed
	Debugging Hardware	Bora Cukurova
	Finalize Assembly	Natalya Bapst
11/20 (Thanksgiving)	Debugging Microcontroller	Hanna Zayed
	Debugging Wire Connections	Bora Cukurova
	Debugging Code	Natalya Bapst
11/27	Demonstration	Hanna Zayed
	Demonstration	Bora Cukurova
	Demonstration	Natalya Bapst

12/4	Final Proposal	Natalya Bapst
	Final Proposal	Hanna Zayed
	Final Proposal	Bora Cukurova

7. Ethics & Safety

Since our project is based on increasing the safety of the consumer, it is necessary that our project design will neither damage the components of the bicycle nor harm the user in anyway. This follows the first code of the IEEE Code of Ethics:

> *"To accept responsibility in making decisions consistent with the safety, health, and welfare of the public, and to disclose promptly factors that might endanger the public or the environment" [4]*

Throughout the design portion of our project, we will learn a great deal about wireless communications and power systems to ensure that our design operates safely and correctly. This will give our team competence in designing the circuitry and handling volatile components such as the Lithium Ion batteries used in our project. This follows the fifth and sixth codes of the IEEE Code of Ethics:

> *"To improve the understanding of technology; its appropriate application, and potential consequences; To maintain and improve our technical competence and to undertake technological tasks for others only if qualified by training or experience, or after full disclosure of pertinent limitations" [4]*

Finally, our competence in understanding the limitations of our power system and the circuitry that it must power is necessary, especially with the handling of Lithium Ion batteries. Knowledge of the tolerance levels of voltage and power are necessary for each component, from LEDs to microcontrollers and beyond. Through this, we follow the ninth code:

> *"To avoid injuring others, their property, reputation, or employment by false or malicious action" [4]*

Finally, we will credit any help received on this project and acknowledge and correct any errors throughout the course of our design such that the seventh code of the IEEE Code of Ethics is followed:

> "*To seek, accept, and offer honest criticism of technical work, to acknowledge and correct errors, and to credit properly the contributions of others" [4]*

8. Citations

[1] Arduino, "Arduino playground - MPU-6050," 2016. [Online]. Available: http://playground.arduino.cc/Main/MPU-6050. Accessed: Sep. 21, 2016.

[2] ATMEL, "ATMEGA328 Datasheet,". [Online]. Available: http://www.mouser.com/pdfdocs/Gravitech_ATMEGA328_datasheet.pdf. Accessed: Sep. 21, 2016.

[3] Contributors and J. 0, "Using EAGLE: Schematic,". [Online]. Available: https://learn.sparkfun.com/tutorials/using-eagle-schematic. Accessed: Sep. 21, 2016.

[4]"IEEE IEEE code of ethics," 2016. [Online]. Available: http://www.ieee.org/about/corporate/governance/p7-8.html. Accessed: Oct. 2, 2016.

[5]Newark, *CREE C503B-RAN-CY0B0AA2 LED, Red*, 2016. [Online]. Available: http://www.newark.com/cree/c503b-ran-cy0b0aa2/led-red-5mm-12cd-624nm/dp/08R2970. Accessed: Oct. 04, 2016.

[6]Newark, "MULTICOMP MCL053SYC LED, Yellow," 2016. [Online]. Available: http://www.newark.com/multicomp/mcl053syc/led-yellow-t-1-3-4-5mm-2-5cd-592nm/dp/14N9429. Accessed: Oct. 04, 2016.

[7] Received, "DC voltage regulator circuit,". [Online]. Available: http://www.electroschematics.com/922/dc-regulator/. Accessed: Sep. 21, 2016.

[8]Texas Instruments, "LP5912," 2016. [Online]. Available: http://www.ti.com/lit/ds/symlink/lp5912.pdf. Accessed: Oct. 02, 2016.

9. Appendix

Pseudo Code:

```
Global Variables:
left_turn_signal                    //left turn signal state
right_turn_signal                   //right turn signal state
turn_state_indicator          //HIGH when turn signal is activated and turn is about to happen
turn_state                                  //Indicates wheterh turn started, being made or
completed
MPU6050 control variables
MPU6050 rotation variables (x,y,z,yaw,pitch,roll)

Interrupt setups:
MPU6050 interrupt pin
Timer1

turn_detection()
{
        To be continued...
}
int MPU6050_data()
{
        Wait for MPU6050 interrupt is available
        Reset MPU6050 interrupt flag
        Check MPU6050 FIFO register overflow
        Else read all available data from FIFO register
        Calculate raw data for x,y,z,yaw,pitch,roll
        write raw data to SD Card
}
void setup()
{
        Initialize MPU6050
        Initialize Transmitter
        Initialize Timer1 ~ 1 second intervals
        Initialize SD_card
}

void loop()
{
        run MPU6050_data() function

        if acceleration at x-axis smaller than threshold acceleration
                set brake signal HIGH

        else
```

```
            set brake signal low

        if left_turn_switch is HIGH or right_turn_switch is HIGH
            set turn_state_indicator HIGH
            if left_turn_signal is HIGH
                set left_turn_signal HIGH
            else
                set right_turn_signal HIGH
            Reset Timer1 clock

        if turn_state_indicator is HIGH
            Run turn_detection()

        write all turn signal data to SD Card
        write all signal states to Transmitter
}

Interrupt: Timer1 (1 second intervals)
{
        if turn_state_indicator HIGH
            if right_turn_signal
                Toggle right_turn_signal
            else
                Toggle left_turn_signal
        else
            set right_turn_signal low
            set left_turn_signal low
        Set Timer1 flag low
}
```