Bicycle Automated Turn Signal (BATSignal)

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

Final Report – ECE 445, Fall 2016
TA: Zipeng Wang
12/7/16
Group No. 3

Abstract

The Bicycle Automated Turn Signal (the BATSignal) is a wirelessly controlled bicycle turn and hazard warning signal that mimics the behavior of a car turn signal. The device takes the turn data from the user via buttons located on the handle bars and determines when the turn is completed through use of a gyroscope. Once the turn is completed, the turn signal automatically turns off. The current states of the turn signals are wirelessly sent to the LEDs located at the rear of the bicycle and displayed for any vehicle to see. This project was successfully implemented with the turn signals effectively switching off after a turn was completed.

Table of Contents

1. Introduction **4**
1.1 Statement of Purpose 4
1.2 Block Diagrams 4
2. Design **6**
2.1 Microcontroller 6
2.2 Software 6
2.3 Microcontroller (Decoder) 7
2.4 MPU-6050 (Gyroscope) 7
2.5 Power 8
2.6 RF Receiver/Transmitter 9
2.7 External Interface 9
2.8 Right/Left Turn Signal Switches 10
2.9 SD Card 10
3. Design Verification **11**
3.1 Gyroscope Data 11
3.2 Software Performance 12
3.3 Power Consumption 12
4. Cost **14**
4.1 Parts 14
4.2 Labor 14
5. Conclusion **15**
5.1 Accomplishments 15
5.2 Uncertainties 15
5.3 Ethical Considerations 15
5.4 Future Work 16
6. Citations **17**
7. Appendix **18**
7.1 Requirements and Verifications 18
7.2 Schematics and PCB Layouts 20
7.3 Pseudocode 22

1. Introduction

1.1 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 an economical wireless turn signal for bikers to use without needing to move their hands off the handlebars.

1.2 Block Diagrams

Our block diagrams are shown below. Our design consists of two separate PCBs, one located at the handlebars (Figure 1) and one located at the back of the bicycle, referred to as the seatpost (Figure 2).

The handlebar block diagram includes four submodules: the power system, control systems, user interface, and sensors. Our power system, consisting of a lithium ion battery and linear regulator, is used to power our microcontroller, transmitter, LEDs, switches, gyroscope sensor, and SD card reader. Our control systems consist of a microcontroller which takes data from the user interface switches and gyroscope and sends it to the transmitter for communication to the receiver on the back PCB. Our SD Card Reader was originally used for debugging and analysis of data from the gyroscope but was found to be unnecessary for our overall design. Within our user interface, the LED turn signal indicator takes data from the microcontroller for notifying the user if the turn signal was completed.

The seatpost block diagram includes the power system, control systems, and external interface. The power system on the back PCB is identical to that of the front PCB. It is used to power the microcontroller, receiver, and external interface LEDs. The microcontroller decodes the data sent by the transmitter and read by the receiver. This in turn is sent to the LEDs to create the turn signal and hazard signal.

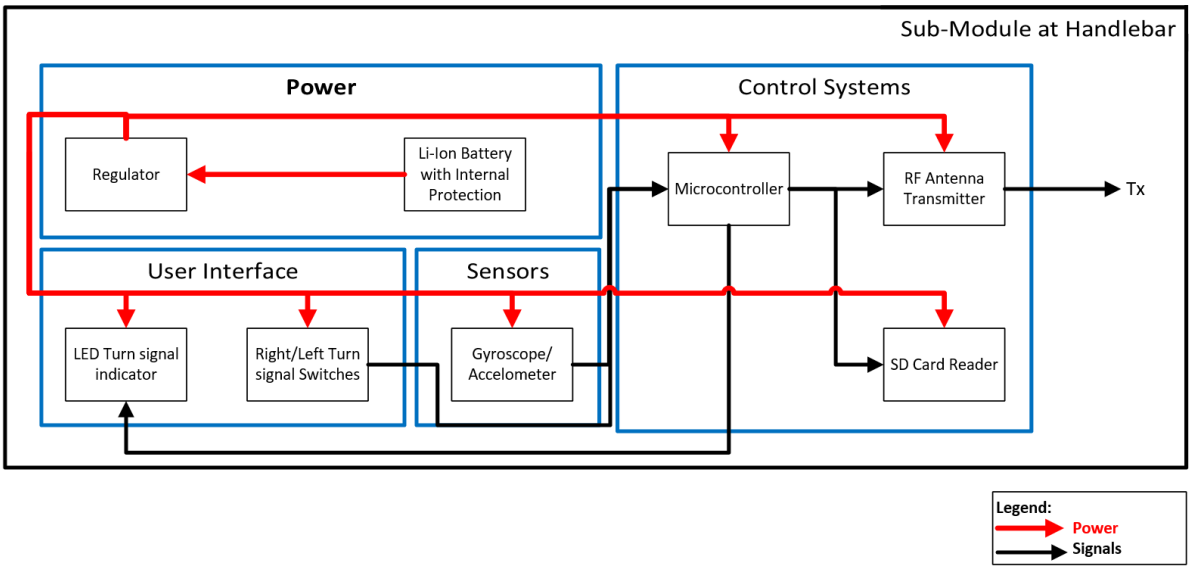


Figure 1 - Block Diagram of Handle-Bar Module

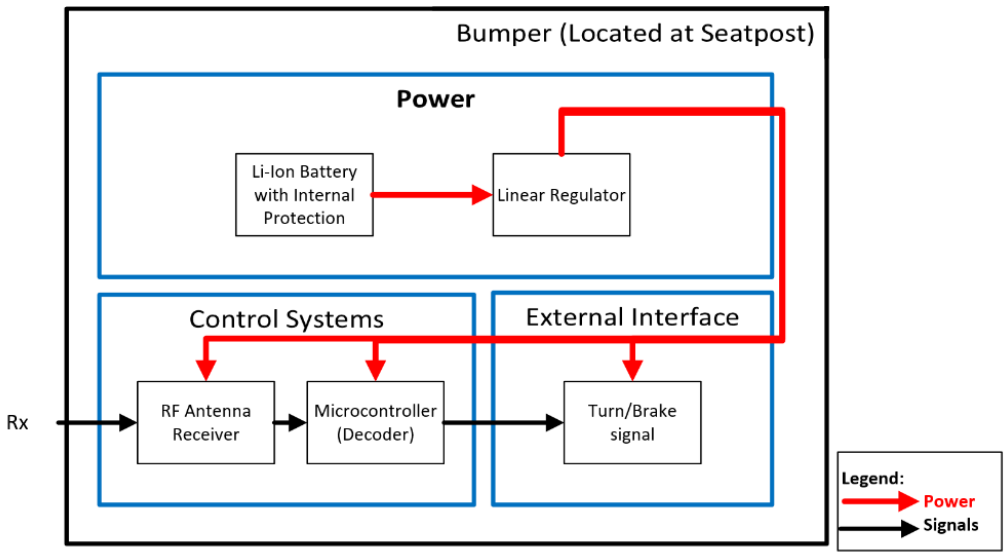


Figure 2 - Block Diagram of Seat-Post Module

2. Design

2.1 Microcontroller

The microcontroller (ATMEGA328P-U) collects all the inputs coming from the turn signal switches and MPU-6050 (gyroscope + accelerometer). Furthermore, it outputs signals to the transmitter for transmission of data to the Turn/Hazard signals located at the seatpost. The turn switches are connected to microcontroller pins PD7 and PB0 for receiving input data from the user. The user notification LEDs are connected to pins PD5 and PD6 of the microcontroller. The microcontroller then outputs the turn states for the user to see. The MPU-6050 chip uses the serial connection, I2C (SDA pin PC4, SCL pin PC5), for input of the gyroscope. The microcontroller uses pins PD2 and PD3 to communicate with the transmitter/receiver chip. Also, the microcontroller requires a 16MHz crystal as its frequency-determining component. The microcontroller is powered with a 3.3V source from the Power Module.

The microcontroller reads the turn switches through polling. The microcontroller starts a timer interrupt of 0.1 seconds for reading of periodic data from the gyroscope and calculation of the turn state. As the microcontroller detects that the turn is completed, it automatically turns the corresponding turning state off. The states of the turns are sent to the seatpost periodically every 0.1 seconds. The full schematic of the front PCB is shown in the Appendix Figure 11 for reference of microcontroller connections.

2.2 Software

The software located on the front PCB microcontroller uses right_turn state and left_turn state to control left/right turn state conditions and also uses the turn_state global variable to determine if the turn is completed or not. It reads the turn switch inputs controlled by the user by polling as seen from Figure 3. Once either the left or right switch is pressed, the state of the corresponding direction changes.

For example, if left_turn state is active and the left button is pressed, then left_turn state turns off and vice versa. Once the state changes, if the corresponding state becomes active, then the other state turns off. This way, at most only one of the direction states can be on at any time. Also, every time a switch is pressed, the yaw angle resets and turn_state becomes incomplete. To calculate whether a turn is completed, the software calls interrupts every 0.1 seconds. As seen from Figure 4, in every interrupt cycle, if either the left or right state is active, the program will call the check_turn() function to determine whether the turn is completed. If the turn is completed, then the corresponding direction state turns off and turn_state will become incomplete. At the end of the interrupt, the directions states will be send by the transmitter to the receiver.

The pseudocode for the check_turn() function can be viewed in the Appendix. The check_turn() function detects whether the turn is completed or not by incrementing numbers. If 18 successful turns have taken place within 0.1 second intervals in the direction in which the turn signal is activated, the turn is registered as completed. The bicycle is considered turning if the yaw angle of the bicycle increments by a minimum of 1 degree per iteration. This ensures that the turn can be detected and that the cumulative error that may arise over time is omitted. The reason we check 18 turns is that a typical turn takes an average of 2 seconds to complete. 18

turns correspond to a minimum of 1.8 seconds. There is also an error check of whether the user is continuing the turn at a given time. This is accomplished by verifying that the yaw angle does not increment for 1 second such that the turn is defined as stopped. When this happens, the software resets from the beginning.

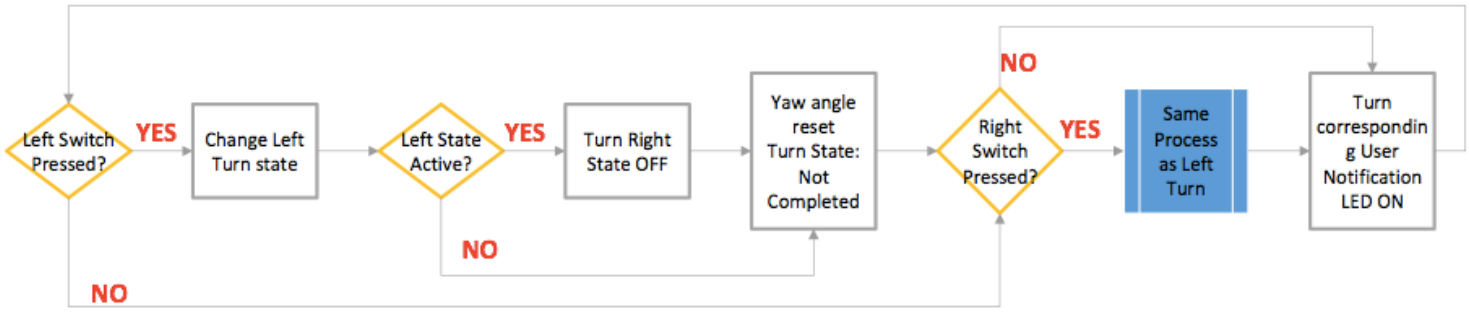


Figure 3 - Front PCB MCU Software Main Loop

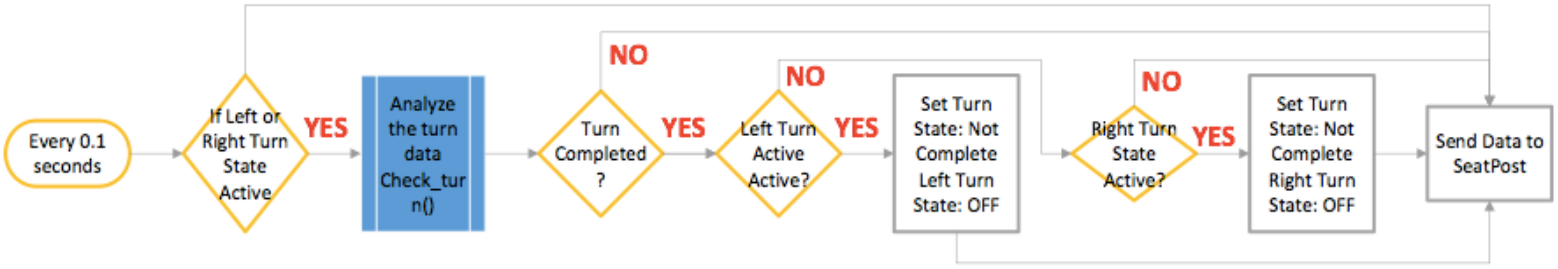


Figure 4 - Front PCB MCU Software Interrupt

2.3 Microcontroller (Decoder)

The microcontroller located at the seatpost receives serial data from the receiver. It decodes the data and feeds it to the Turn/Hazard signals located at the rear of the bicycle using pins PD6, PD7 and PB0. The schematic for this is shown in Figure 5.

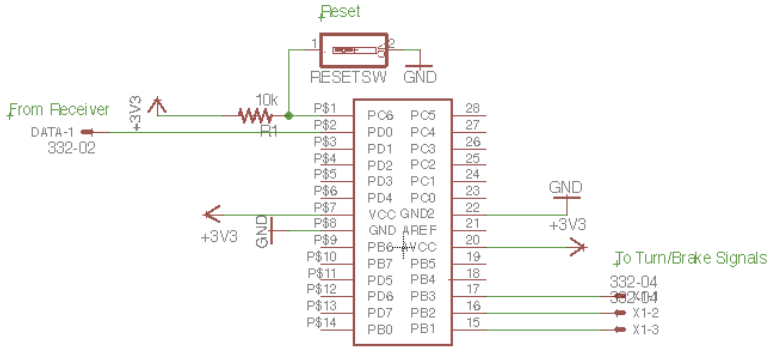


Figure 5 - Microcontroller Module (Decoder)

2.4 MPU-6050 (Gyroscope)

The InvenSense MPU-6050 chip is used for the sensor portion of the design. The chip includes a total of 6 axes, 3 axes for the gyroscope as well as 3 axes for the accelerometer. It

communicates with the microcontroller through an I2C serial connection (SDA, SCL). This module is used to analyze the bicycle motion to determine whether the cyclist is making a turn.

The design is powered with 3.3V from the Power Module. The schematic can be seen below in Figure 6.

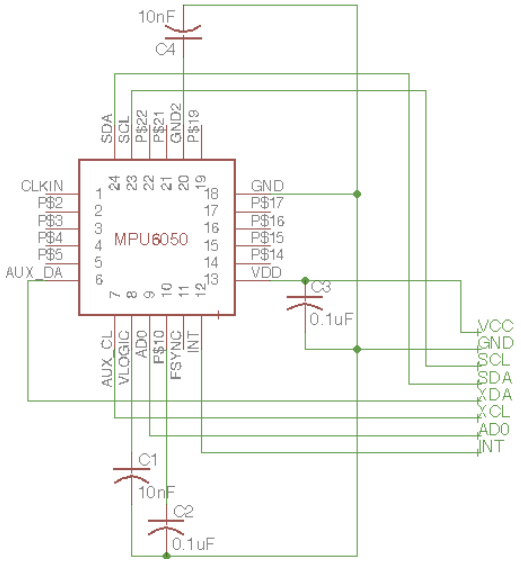


Figure 6 - Detailed MPU6050 Circuit

2.5 Power

The power supply provides power to each component block and consists of a single lithium ion battery (3.7 V, 2600 mAh nominal). We soldered this to a LP5912 linear regulator to regulate the voltage at a constant 3.3V. We required two of these power supplies, one for the circuitry at the handlebars and one for the circuitry at the seatpost. This circuit delivers around 9.6 Watts/hr at 500 mA output:

(2600 mAh * 3.7V)/1000 = 9.6 Wh

The schematic for the linear regulator is shown below in Figure 7 with pin outs as taken from the TI datasheet for LP512:

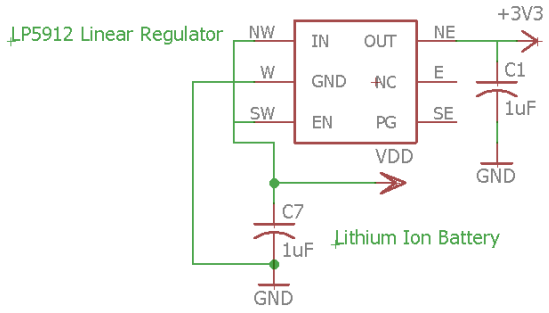


Figure 7 - Linear Regulator Schematic

2.6 RF Receiver/Transmitter

Our communication channel is comprised of two XBEE XB24-API-001 transceiver chips with the front-end chip serving as a transmitter and the back-end chip serving as a receiver. The chips' network parameters were set using the microcontroller. Using this chip to its full potential would allow us to communicate across a length of up to 300 feet outdoors and send up 250,000 bits per second. However, since we are only transmitting 80 bits per second across the length of a bike, our chips operate more reliably.

With this chip pair, one-way communication is achieved by the transmitter sending modulated carrier waves across the length of the bike and the receiver catching these waves. The carrier waves oscillate at a frequency of 2.4 GHz, allowing the use of small chip-embedded antenna, and are modulated by a byte of information construed by the front-end microcontroller. The transmitting chip only sends waves when the user activates one of the turn signals. The receiving chip, however, is continuously listening for incoming waves.

The single byte of information that is carried across the communication channel conveys the state of both turn signals. With 8 bits, each byte appears as the following sequence: 0 0 0 0 0 L 0 R. For example, if L = 1 and R = 0, then the communication link would be conveying that the left turn signal is on. Similarly, if L = 0 and R = 1, then the link would send that the right turn signal is on. Before each byte is sent, it is encoded by the front-end microcontroller. After each byte is received at the back-end device, it is decoded by the back-end microcontroller. The schematics for the transmitter and receiver are shown in the Appendix full PCB schematics (Figure 11 and Figure 12 respectively).

2.7 External Interface

The external interface on the back-end device mainly consists of three LED arrays which will display the two turn signals and the hazard signal. The design challenge is to get each LED array to display only when necessary. At first, our intention was to design a logic block to decode the received message in such a way that power is only sent to the correct array. Our initial design consisted of a shift register in conjunction with some additional logic elements. After experiencing some synchronization problems, however, we decided to proceed by using an additional microcontroller on the back-end device to decode the received message. In retrospect, this design alternative was far simpler and easier to implement. Albeit simple, this design was more expensive and would have a significant impact on cost in the case of mass production. The schematic for the external interface is shown below in Figure 8.

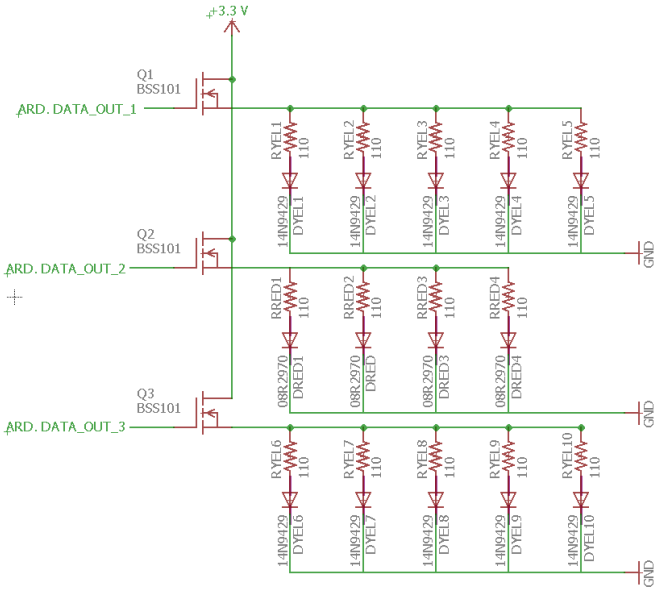


Figure 8 - Turn/Brake Signals with LED Design

2.8 Right/Left Turn Signal Switches

This is located on the handlebars of the bicycle. It is located at a reachable distance from the cyclist's hands. The switches send inputs to the microcontroller about the state of the switches.

2.9 SD Card

This module was used for debugging purposes. It was useful in recording the data we read from the MPU-6050 chip to analyze the gyroscope data. The SPI method for writing data was implemented. The circuit diagram of this module can be seen in Appendix Figure 11.

3. Design Verification

3.1 Gyroscope Data

Below is our gyroscope data for detection of a 100-degree left turn. Before the turn, the scale goes to +/- 10 degrees about the origin. This is due to both a sensitivity error and the user's slight angle variations. Once the turn starts, we can see a smooth increasing slope. As the turn comes to an end, the graph flattens out. As seen from Figure 9, as the check_turn() function detects 18 turns from the gyroscope data that are above the threshold for turn angle (1 degree), it verifies that the turn is completed and turns off the turn signal automatically.

Iterations	Yaw (deg)	Turn_count
80	9.42	0
81	11.46	0
82	14.06	0
83	16.82	1
84	20.58	2
85	24.59	3
86	27.89	4
87	31.49	5
88	35.15	6
89	37.57	7
90	40.21	8
91	41.7	9
92	45	10
93	50.63	11
94	53.66	12
95	58.81	13
96	66.31	14
97	72.96	15
98	80.97	16
99	86.35	17
100	89.54	18
101	93.84	19

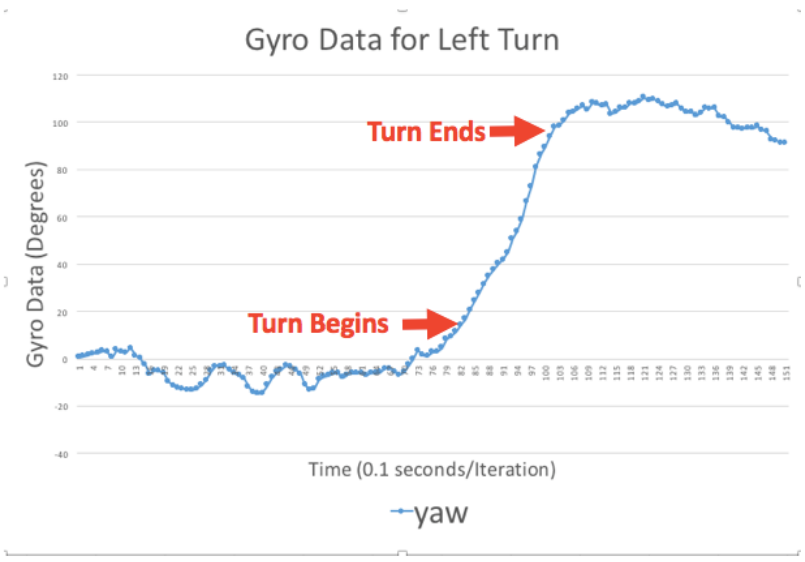


Figure 9 - Gyroscope Data and Turn State Detection

Below is further gyroscope data for detection of a 75-degree turn shown in Figure 10. The data is not as smooth as the previous Figure 9. This turn takes 2.4 seconds for completion. Although the data is not continuously incrementing, the software successfully detected that the turn had not terminated in the middle of the graph where a discrepancy is visible. It distinguished that the turn was still taking place until 75 degrees was achieved.

Iteration	Yaw (deg)	Turn_count
80	8.46	0
81	11.54	0
82	14.8	1
83	17.35	2
84	18.53	3
85	19.02	3
86	21.03	4
87	21.68	4
88	23.99	5
89	27.64	6
90	33.29	7
91	39.34	8
92	46.63	9
93	52.91	10
94	56.51	11
95	58.56	12
96	57.66	12
97	55.27	12
98	55.45	12
99	54.05	12
100	56.02	13
101	59.61	14
102	62.31	15
103	66.3	16
104	69.35	17
105	72.13	18
106	75.66	19

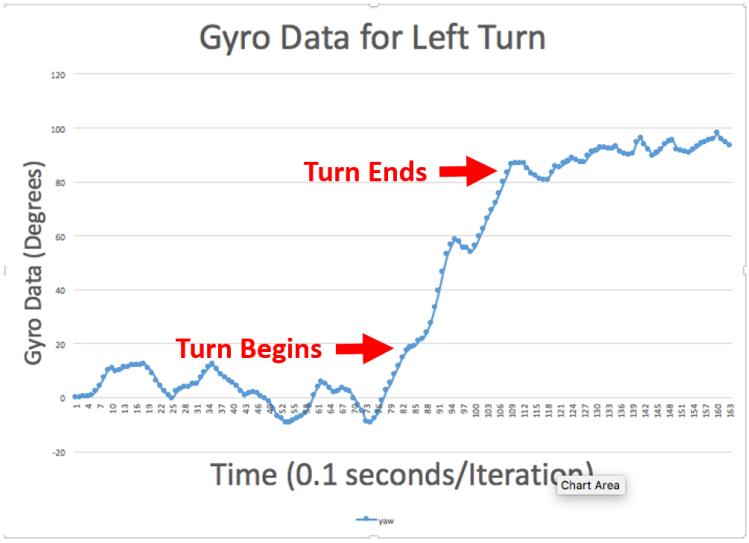


Figure 10 - Gyroscope Data for Rough Curve

3.2 Software Performance

Front PCB Software uses 68% (21,974 bytes) of the microcontroller program space (32,256 bytes)
Global Variables uses 75% (1,554 bytes) of dynamic memory (2,048 bytes) and 494 bytes for local variables.
Back PCB Software uses 12% (3,996 bytes) of the microcontroller program space (32,256 bytes)
Global Variables uses 17% (354 bytes) of dynamic memory (2,048 bytes)and 1,694 bytes for local variables

3.3 Power Consumption

MPU-6050 Power Consumption:

The MPU6050 is powered by a 3.3V source from the Power Module. It requires a range of 2.375 - 3.46 V to operate. In order to operate the gyroscope of the MPU-6050 chip, it requires a current of about 3.8A.

Power Required = 3.8 mA x 3.3V = 12.54mW

Microcontroller Power Consumption:

The Microcontroller is powered by 3.3V.

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 the microcontroller is active:

Maximum Power = 3.3V x 3.5mA = 11.55 mW

Minimum Power = 3.3V x 1.7mA = 5.61 mW

When the 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 Consumption:

The values used in the following calculations were gathered from the XBEE 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 Consumption:

The values used in the following calculations were gathered from the XBEE datasheet.

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

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

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

External Interface Power Consumption:

From the LED data sheets, the maximum power dissipation for a red LED is 130 mW and the maximum power dissipation for a yellow LED is 85 mW. Furthermore, from the BSS101 MOSFET datasheet, we gathered that the maximum power dissipation is 0.60 W. 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 drive the LEDs with 20 mA of current, we calculate their typical power consumption by multiplying the typical drive current with the 3.3 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. Cost

4.1 Parts

Part	Part Number	Unit Cost	Quantity	Total
LEDs	LED SuperBright	$.1475	20	$2.95
Microcontroller	Atmel ATMEGA328P-PU	$6.49	2	$12.98
Lithium Ion Battery (3.7V)	Ultrafire 18650 - 2600 mAh, 3.7V	$5.99	2	$11.98
Capacitors	Varies	$0.37	95	$3.57
16 MHz Crystal	HC49/US Package	$0.95	2	$1.90
Resistors	Varies	$.0091	150	$1.37
Transceivers (XBEE)	XB24-API-001	$19.00	2	$38.00
Inductors	Varies	$0.19	10	$1.90
Accelerometer/Gyroscope Chip	InvenSense MPU-6050	$2.95	1	$2.95
Switches	LED Tactile Button	$1.95	2	$3.90
Linear Regulator	595-LP5912-1.8DRVR	$1.18	2	$2.36
PCBs	PCBWay	$4.30	20	$86
BJTs	ST 2N3906	$0.14	3	$0.42
Female Pins	CONN RCPT .100’’ 10POS	$3.88	2	$7.76
Total				**$179.33**

4.2 Labor

Name	Hourly Rate	Total Hours Invested	Total = Hourly Rate x 2.5 x Total Hours Invested
Bora Cukurova	$32.50	150	$12,187.50
Natalya Bapst	$32.50	150	$12,187.50
Hanna Zayed	$32.50	150	$12,187.50
Total		450	$36,562.50

Total Cost → Parts + (ideal salary (hourly rate) × actual hours spent × 2.5) = **$36,741.83**

5. Conclusion

5.1 Accomplishments

We successfully implemented our design for the Bicycle Automated Turn Signal (the BATSignal). Our team created a turn signal design that could detect when a turn has been completed when testing in a laboratory environment. We tested our project on a bicycle with slow turning and achieved satisfactory and successful results. Furthermore, our design had an easy snap on and snap off attachment with Velcro to the bicycle. This is convenient not only for design testing but for cyclist usability. The LEDs on our design could be seen from a distance acceptable for vehicle visibility.

5.2 Uncertainties

We have not completed an official trial run on a road using our design on a bicycle. All experiments were performed in a laboratory environment. Therefore, there are uncertainties of practical usage in cases we could not foresee. Extensive testing in different environments and with multiple users would aid in determining the efficiency of our design. Furthermore, we are uncertain about how robust our design is in terms of lifetime and longevity of parts.

5.3 Ethical Considerations

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]*

5.4 Future Work

Future work on this project would include integrating the accelerometer functionality of the MPU-6050 chip to create a brake signal at the back of the bike. This would be useful for cyclists to signal to vehicles when they are slowing down or stopping abruptly. Furthermore, weatherproofing the bicycle turn signal and creating better housing for the PCBs would be the most practical furthering of this project.

6. 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.

7. Appendix

7.1 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 an 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 and Timer interrupt every 0.5 seconds or 0.5 seconds is a multiple of the interrupt period. **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

		signaling automatically with 90% efficiency.
Receiver and transmitter	**1.** The antenna can transmit and receive signals of at least three bits over the minimum length of the bike necessary (34 inches)	**1.** Test each bit combination 25 times (100 total) to determine if the receiver is receiving correctly and the transmitter is transmitting correctly **2.** Perform a link budget on the system by measuring transmitted power in dB, gains in dB and losses in dB **3.** Calculate by using the equation: Received Power (dB) = Transmitted Power (dB) + Gains (dB) − Losses (dB)
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 of 10 Degree/second error. **3.** Gyroscope can detect a frame angle change of 90 degrees	**1a**. Use a protractor/angle gauge on the ground along the y axis to measure the angle of the front wheel and verify that 45 degrees is the minimum detected **2a**. Use a protractor/angle gauge along the x axis to measure the angle of the bicycle tilt **3a**. Use a protractor/angle gauge along the frame axis to measure the angle of the bicycle's movement
User Interface	**1.** The left and right switches on the handlebars interface with the microcontroller **2.** The LED turns on and blinks when the turn signal is on	**1a**. Write a simple loop that checks that the switches are being registered when pressed **2a**. Test by writing a loop that checks if the turn signal is registered as on or off **2b**. Check if the LED has turned on when the turn signal is registered as on
External Interface	**1.** The turn signal lights up in the back and turns off when necessary **2.** The brake signal lights up in the back and turns off when necessary	**1a**. Test that the turn signal switches work to light the LEDs 100 times for both the right and left turn signals **1b**. Test that the LEDs turn off 100 times after the turn is completed

		2a. Test that the brake signal turns on when the bicycle is braking 100 times **2b**. Test that the brake signal turns off when the bicycle is moving again 100 times

7.2 Schematics and PCB Layouts

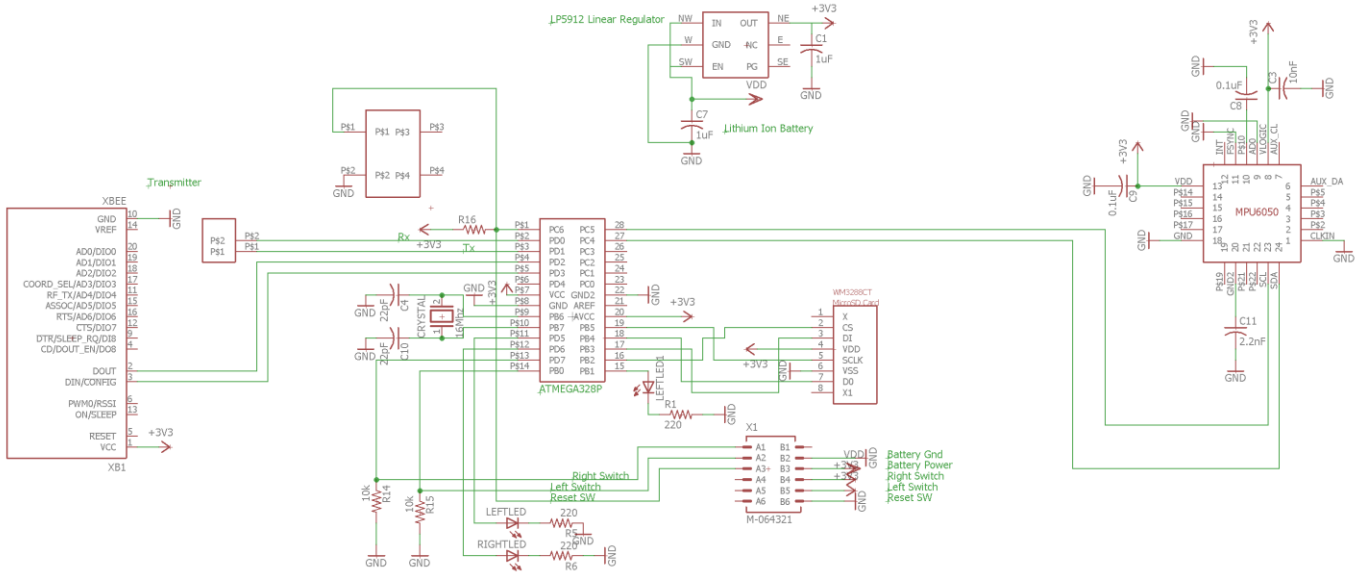


Figure 11 - Full Schematic for Front PCB

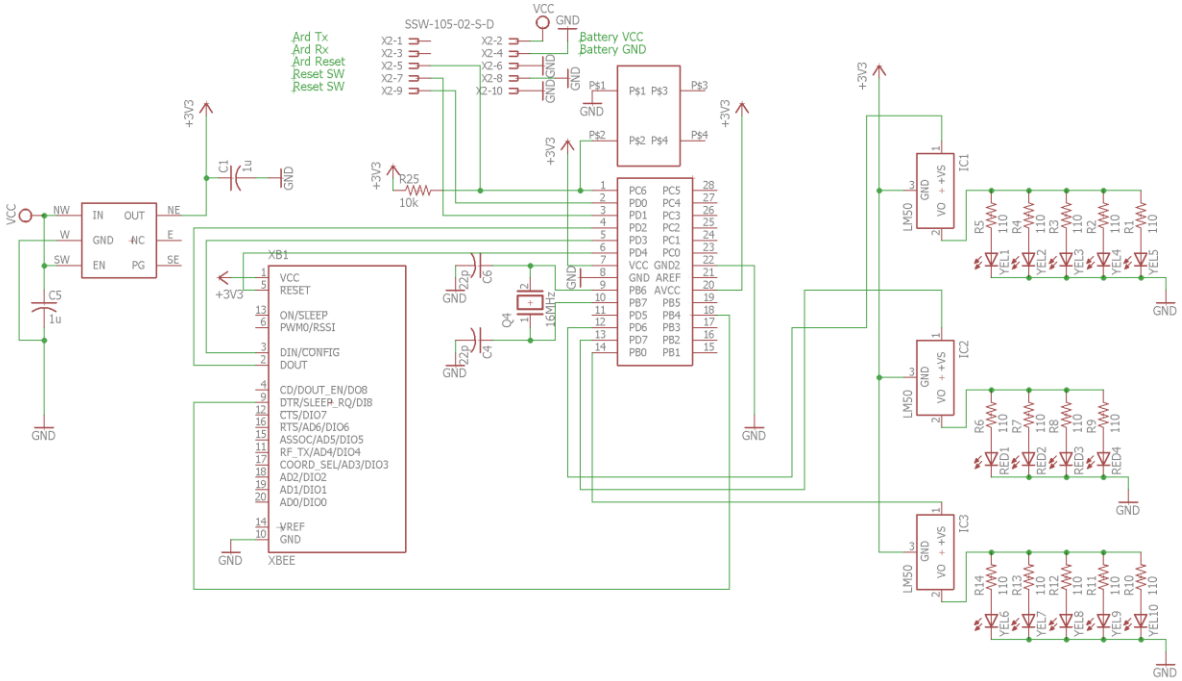


Figure 12 - Full Schematic for Back PCB

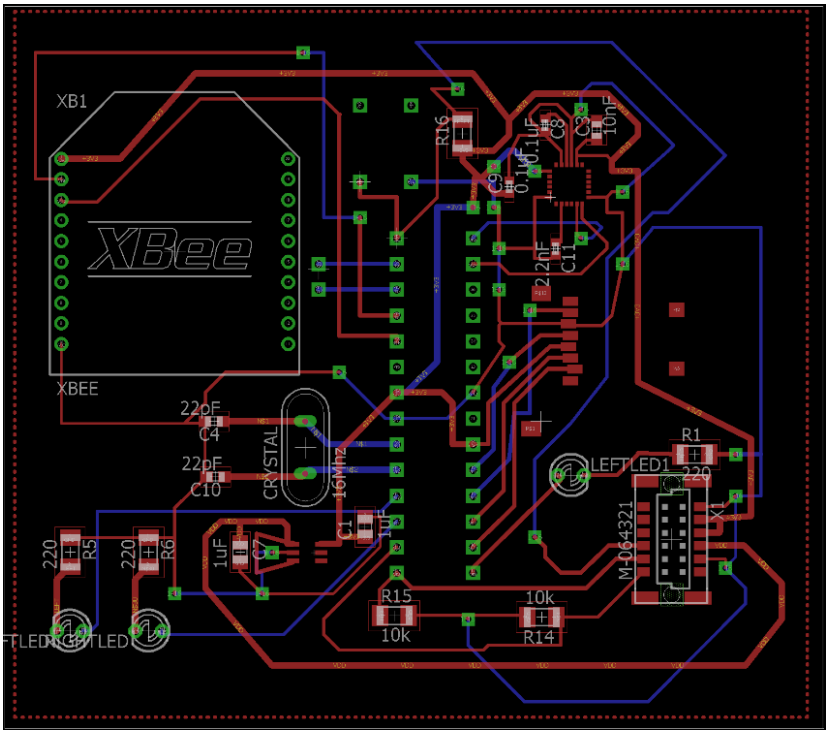


Figure 13 - Front PCB Layout

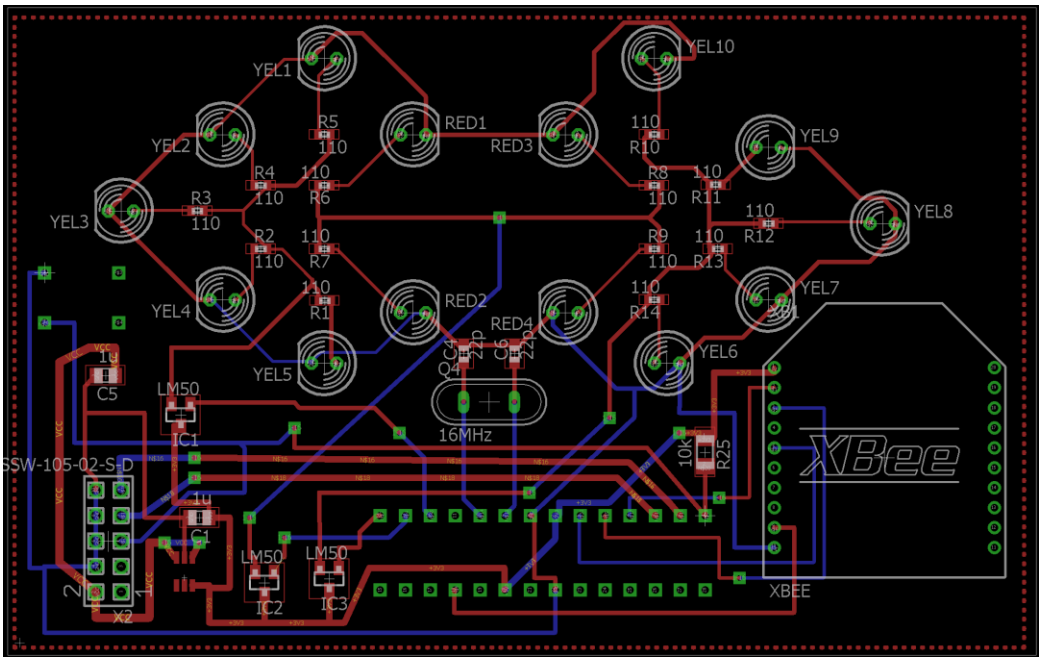


Figure 14 - Back PCB Layout

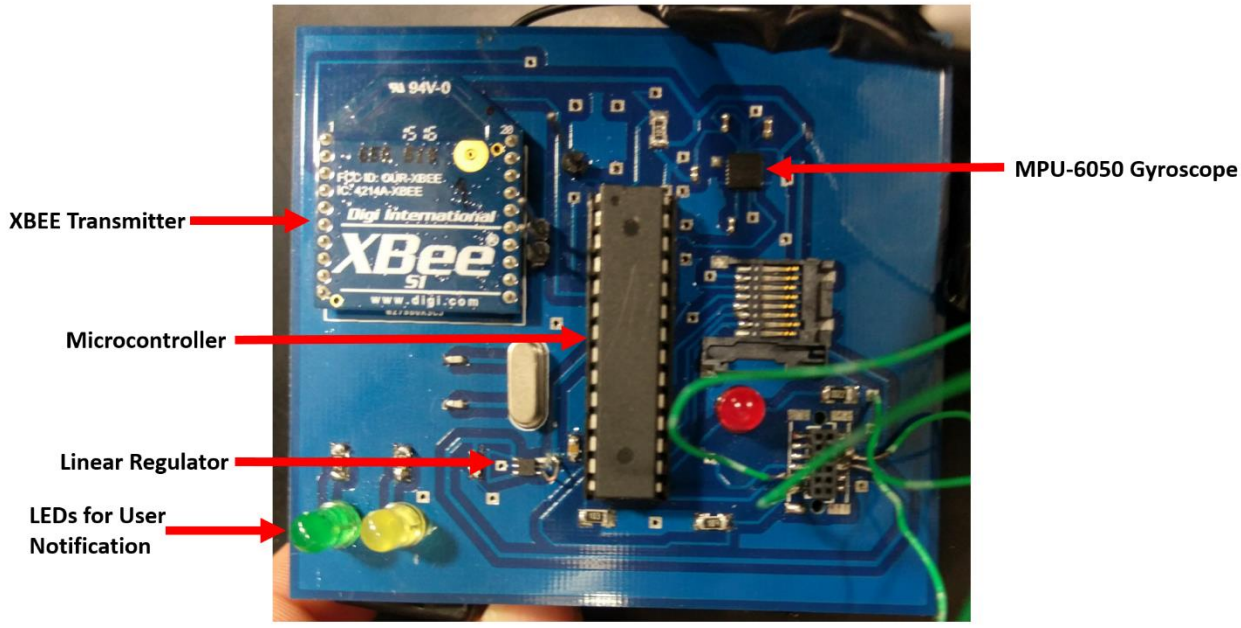


Figure 15 - Soldered Front PCB

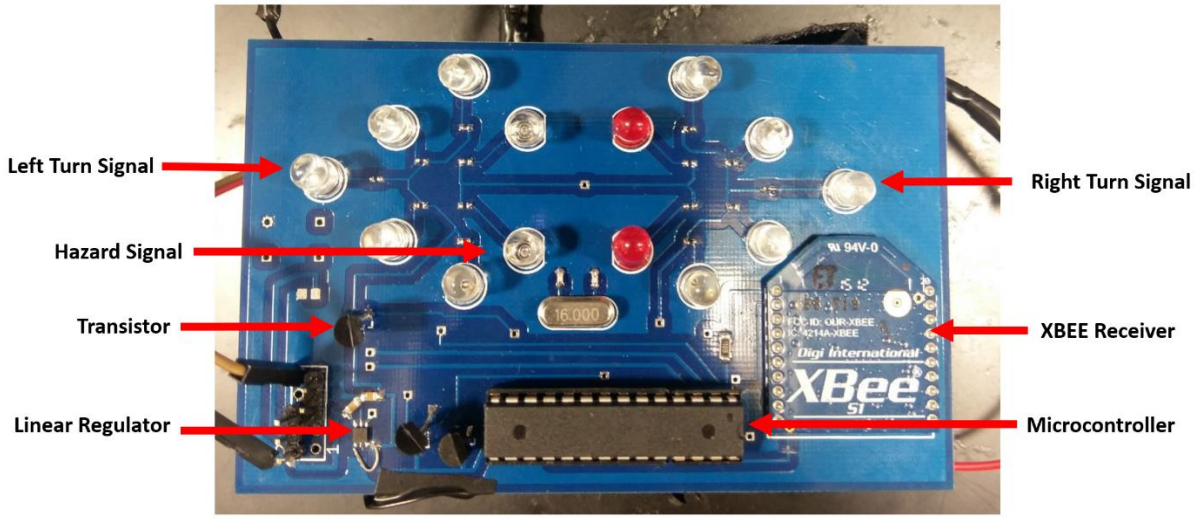


Figure 16 - Soldered Back PCB

7.3 Pseudocode

The pseudecode code for the check_turn() function is as follows:

```
uint8_t check_turn(){
If yaw>10 (bike is turning left) and left_turn state is active
If turn not completed
                If 18 successful turns detected
                        Turn completed
```

Reset continuity check // Reset error detection for no turn, signal on
Else if current yaw angle is 1 Degrees greater than the previous yaw angle
Count another successful turn
Reset continuity check //Reset bc. turn successful
Else if continuity check greater equal than 10 // No turn detected for 1 second
Reset number of successful turns // Bike is not turning
Else
Increment continuity check // Continuous counting no turns in turning
If yaw<-10 (bike is turning right) and right_turn state is active
//Apply same for right turn
Last_turn yaw angle is Current_turn yaw angle}