ARDUINO CONTROLLED MOTORIZED LONGBOARD

By

Leon Ko

Kevin Lee

Daniel Moon

Final Report for ECE 445, Senior Design, Spring 2013

TA: Mustafa Mukadam

1 May 2013

Project No. 39

Abstract

Our group designed an Arduino controlled motorized longboard. The longboard uses a $LiFePO_4$ battery to provide power to a motor that is mounted to the back left wheel. The longboard is controlled by a remote control that has a joystick to accelerate and decelerate, as well as a brake button to drop the motor speed much quicker. A safety kill switch was installed onto the board to ensure that the board is not moving without a rider, and also stops the motor if the rider is to leave the board while moving. Assisted turning was also implemented into the board, allowing for smoother turns at consistent speed. The longboard is able to reach a minimum top speed of at least 13 miles per hour, and has a battery lifetime of at least 4 miles.

Contents

1. Introduction

Environmentally friendly, easy to use transportation is always a commodity that is sought after, especially if the mode of transportation can be used in cohesion with others such as the train or buses. The last leg of a commute is always usually done by means of human power, whether it is by bike, roller blades, or walking. Our Arduino controlled motorized longboard can solve this standing problem.

We designed and created an Arduino controlled motorized longboard. The longboard is controlled by a remote control with a joystick and brake button. The controller transmits these signals through Xbee chips to an Arduino that sends a pulse width modulation signal to a motor controller. The motor controller then accelerates or decelerates the motor accordingly.

In this report, we will discuss our Arduino controlled motorized longboard and the functions, features, and modules of the project, design and testing processes done on the longboard and costs of development.

1.1 Purpose

The goal of our project was to create an environmentally friendly mode of transportation that can be used as an independent electric vehicle for short distances or in cohesion with other modes of transportation to make commutes shorter by eliminating walking. We also desired to make longboarding a simpler and safer experience by implementing assisted turning, a safety kill switch, and having all movement controlled by a remote control. This product allows for quick transportation that is portable and easy to use, in an efficient and environmentally friendly way.

1.2 Functions and Features

1.2.1 Functions

This product will provide the rider the following benefits:

- Environmentally friendly mode of transportation
- Longboard that requires no energy provided by rider pushing board
- Easy control of acceleration and deceleration with simple remote controller
- Safer longboard experience due to kill switch
- Improved rider experience with assisted turning

1.2.2 Features

Our motorized longboard will include the following features:

- Speeds up to 13-20 mph
- Wireless remote control
- Joystick acceleration and deceleration
- Brake button
- Safety kill switch
- Assisted turning

1.3 Modules

Our project had many components to be designed, which are shown here:

1.3.1 Remote

The remote is the only user input for the motorized longboard. Three potential inputs exist on the remote: access to acceleration, deceleration, and breaking. Acceleration and deceleration will be achieved through manipulation of a joystick, with the magnitude of acceleration determined by how far the joystick is pushed. Breaking will be activated by pressing and holding a button. All input received by the remote will be transmitted wirelessly to the transceiver block.

1.3.2 Transceiver

The transceiver is the waypoint between the remote and Arduino micro-controller. While it does not perform any data manipulation or analysis, it is a vitally distinct section.

1.3.3 Arduino Micro-Controller

The micro-controller is where all the data analysis is performed. For acceleration and deceleration, it decodes the signal and passes the information via pulse width modulator to the motor. When the breaking button is held down, the micro-controller will decode the signal and activate the breaking. The last function the micro-controller manages is the kill switch. When the implanted flex sensor senses that the longboard is no longer bent (indicative of a person's weight on the board), the micro-controller will activate the breaking.

1.3.4 Motor

The motor receives the acceleration and deceleration signals from the Arduino micro-controller. Based on the pulse width modulator signal, the motor outputs the appropriate revolutions per minute. Furthermore, when regenerative breaking is enacted, the motor turns into a generator that will recharge the battery.

1.3.5 Communications (Transceiver)

The transceiver looks to provide instantaneous communication between remote and micro-controller.

1.3.6 Micro-Controller

The micro-controller performs all signal processing from the remote and data analysis from the flex sensor. The micro-controller is directly connected to the kill switch, which is a flex sensor. The micro-controller is also directly connected to turn assistance, which consists of a pair of infrared sensors. The micro-controller passes the information of whether the breaking button is pushed to the breaking block. Code is then run which sends information on how to transform the motor into a generator. Additionally, when the infrared sensors detect that the board is tilted, the micro-controller will use the pulse width modulator to decrease the rate of the motor. The micro-controller is powered by a $LiFePO_4$ battery.

1.3.7 Turn Assistance

Turn assistance consists of a pair of infrared sensors hard wired into the micro-controller. One sits on the left side and the other sits on the right side. Longboards tend to ride extremely close to the ground, and as a result it is possible to use an IR sensor to detect turns. When the user enters a sustained turn, one side of the longboard is closer to the ground, enough to increase the voltage output of an IR sensor. When the IR sensor tells the micro-controller that the longboard is turning, the micro-controller will decrease the PWM accordingly and send the new PWM to the motor. There will be a time delay and accepted voltage ranges instituted to allow for extenuating circumstances (such as turbulence, brief turns, etc).

1.3.8 Kill Switch

The kill switch consists of a flex sensor that is hard wired into the micro-controller. Because of the flexibility of longboards, the board tends to sink slightly at the center when there is a rider. The flex sensor takes advantage of this information, detecting when the board is no longer flexed (suggesting no rider). When there is no rider, the kill switch sends a signal to the micro-controller, enacting breaking. We will institute a reasonable time delay to allow for extenuating circumstances (i.e. the rider jumps).

1.3.9 Battery

The motorized longboard will utilize a $LiFePO_4$ battery, with a projected life of 4-8 miles.

2. Design

2.1 Top Level

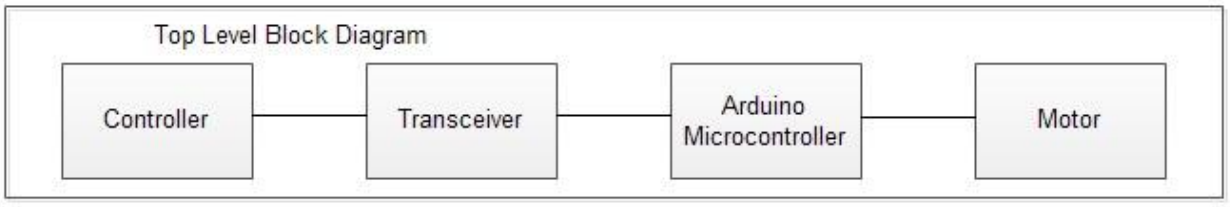

Figure 1. Top Level Block Diagram

2.1.1 Controller

The controller is the interface through which the user controls the motorized longboard. The longboard contains two primary functions: acceleration and braking. As a result, the controller featured two forms of input: the joystick and the button. The joystick was implemented with a dual axis potentiometer, although we only utilized the y-axis input. The button was simply a push button that provided a digital high when pressed.

2.1.2 Transceiver

The transceiver is the method through which input was relayed from the controller to the microcontroller. We decided to implement the transceiver with two XBEE 802.15.4 1mW radios due to their affordability and supposed ease of use. We soon realized that most projects used one microcontroller with each XBEE module, and simply used them to relay information. Our plan of using the XBEE to do some basic processing necessitated some coding of the XBEEs. Similarly, feeding inputs directly into the XBEE created some compatibility issues that needed resolution.

The two modules communicated with the default 9600 baud rate. Hypothetically we could have opted to increase the baud rate, as the sacrifice in range would have had a marginal impact on our project. However, an increased baud rate leads to more errors, regardless of the parity byte that the XBEE uses for error detection. A higher baud rate is also more susceptible to interference. The final deciding factor was maintaining the same baud rate between the XBEEs and serial output to the computer, where 9600 is the default entry point.

2.1.3 Arduino Microcontroller

The microcontroller converts the user inputs, interprets them, and outputs a PWM to the motor. We decided to use the Arduino Uno, as the project did not require heavy computing and only required a couple inputs and outputs. Furthermore, the Arduino is a very versatile device, easily adapting to the other modules we chose. It had the ability to communicate with XBEEs via an independently purchased XBEE shield, and could output the appropriate signal to the electronic speed control in the motor controller.

2.1.4 Motor

The motor is the end point of the user's input, propelling the longboard. We opted to use Great Planes Rimfire 1.60 Outrunner Brushless Motor for its power. A stated goal was efficient transportation, and the Outrunner delivered exceptional power. A brushless motor was chosen because of higher efficiency, higher torque, and reduced mechanical wear. As the motor is one of the most expensive parts, we wanted to select one that would not require regular replacement.

In order to match and order compatible parts, many values must be calculated. Due to our requirement of minimum top speed being 13 mph, we will use this minimum top speed to calculate the desired RPM of the wheel. The In Heat 80A wheels chosen have a diameter of 75 mm. So:

$$RPM_{wheel} = \frac{S}{C(60)} = \frac{20921.5\ m/hr}{(.075\pi)(60)} = 1479.89\ RPM \qquad (2.1.4)$$

where S is the desired minimum top in meters per hour and C is the circumference of the wheel in meters. From the above calculation, we see that about 1500 RPM is desired from our wheel. Next, we must calculate the required RPM of the motor to go along with our wheels. Assuming we use a gear ratio of 4:1 from motor to wheel (the gear ratio should not be too high since this could cause clutter under the longboard), the motor RPM would then be:

$$RPM_{motor} = G.R.x\ RPM_{wheel} = 4\ x\ 1500 = 6000\ RPM \qquad (2.1.4)$$

where G.R. is equal to gear ratio from motor to wheel. So now, a motor with a RPM of 6000 is desired. At this point, the calculation for motor and battery becomes a problem of availability of products. Motors have a kV rating which stand for RPM/volts, and a motor with a lower kV rating are generally more expensive, but give more torque. A rather inexpensive motor was found with a rather low kV rating, and so the RimFire 1.60 Outrunner was chosen for this project. The kV rating for this motor is 250 kV.

2.2 Specific Level

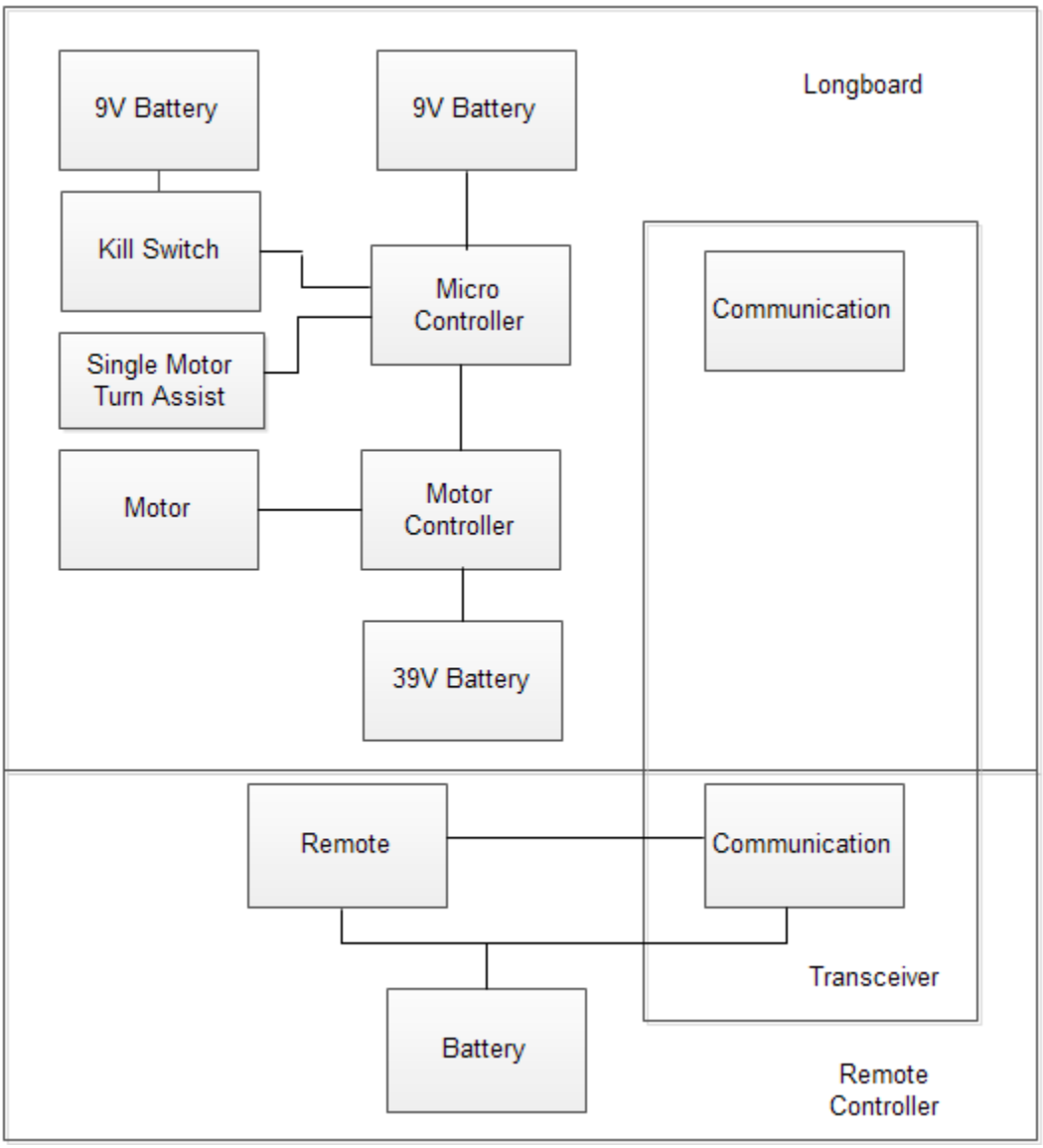

Figure 2. Specific Level Block Diagram

2.2.1 Battery

The controller was powered by 12 volts, with a regulator stepping the voltage down to 5 volts in the rails. Initially the controller used 6 volts from the batteries, but we quickly found the load was far too high, causing battery life to be unacceptably short. An intermediary step of 10 volts was also tested, but the final voltage of 12 volts was found to be most successful. A second regulator stepped the voltage from 5 volts to 3.3 volts for use with the transmitter.

2.2.2 Remote

The controller was comprised of a standard breadboard housed in a clear plastic casing. The clear casing was chosen intentionally to contribute to user experience. The XBEE adapter that connected the XBEE to the breadboard contained a blue LED that revealed signal strength.
The potentiometer initially connected to the 5 volt rails, but we found that the resulting output was incompatible with the transceiver. As a result, the potentiometer was attached to the 3.3 volt regulator already in place for the transmitter. The button simply connected to the 3.3 volt regulator and ground.

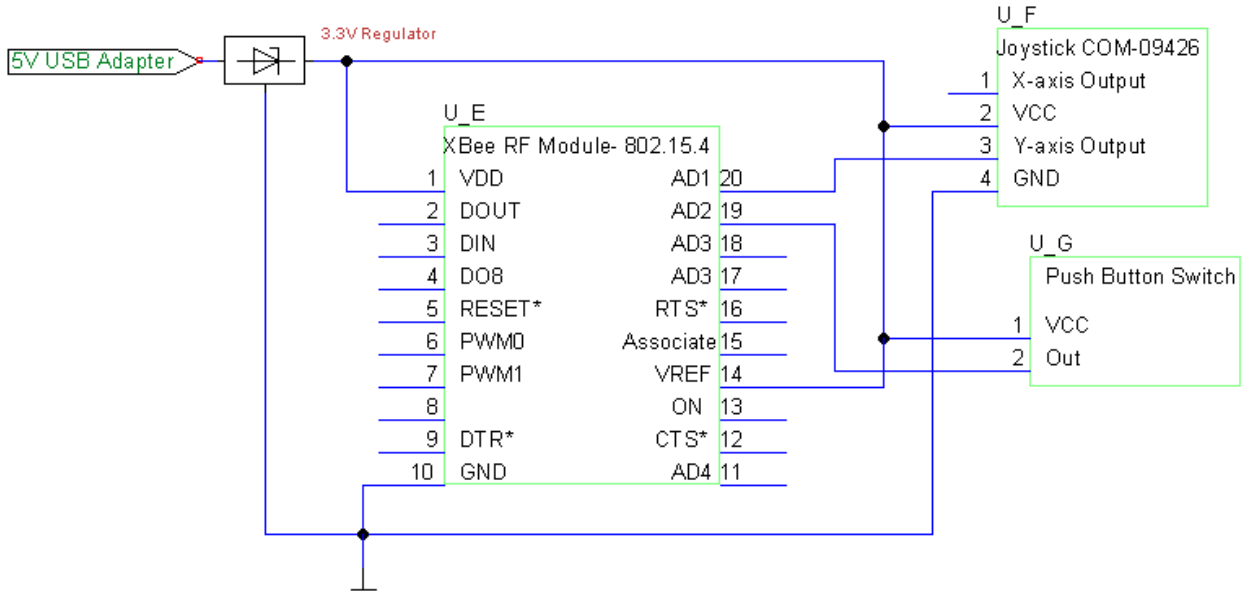

Figure 3. Remote Controller Circuit

2.2.3 Communication (Remote Controller)

Most of the XBEE compatibility issues came on the remote controller side of the transceiver. The primary issue came from the potentiometer, which was designed to output a voltage from zero to five volts. However, the XBEE analog input could only handle up to 3.3 volts. As a result, the output PWM was consistently too high. Thankfully, the potentiometer functioned similarly when placed between ground and 3.3 volts.

2.2.4 Communication (Longboard)

On the longboard, the XBEE did not offer problems so much as the XBEE shield that was used to interface between the XBEE and the Arduino Uno. The shield purchased was the Libelium XBEE shield, which unfortunately was a deprecated product. As a result, the Arduino was incapable of simultaneously corresponding with the XBEE and the computer. While this will not create an issue with the final functionality of the project, it vastly increased the difficulty when it came to debugging and implementing the project. Ultimately, no permanent solution was discovered: the only course of action was to unplug the shield while uploading code.

The only problem that came from the longboard XBEE was the inability to directly transmit the analog signal into the Arduino. Whereas a XBEE inputs a true analog signal, it outputs a PWM from which the analog can be derived. However, the Arduino Uno lacks the ability to simply read a PWM signal. Although the `PulseIn` function helped, it was extremely unreliable, especially for PWM inputs with 0% or 100% duty cycle. As a result, we were forced to include a low pass filter circuit to convert the output of the XBEE into a true analog signal, and then feed it back into the Arduino Uno.

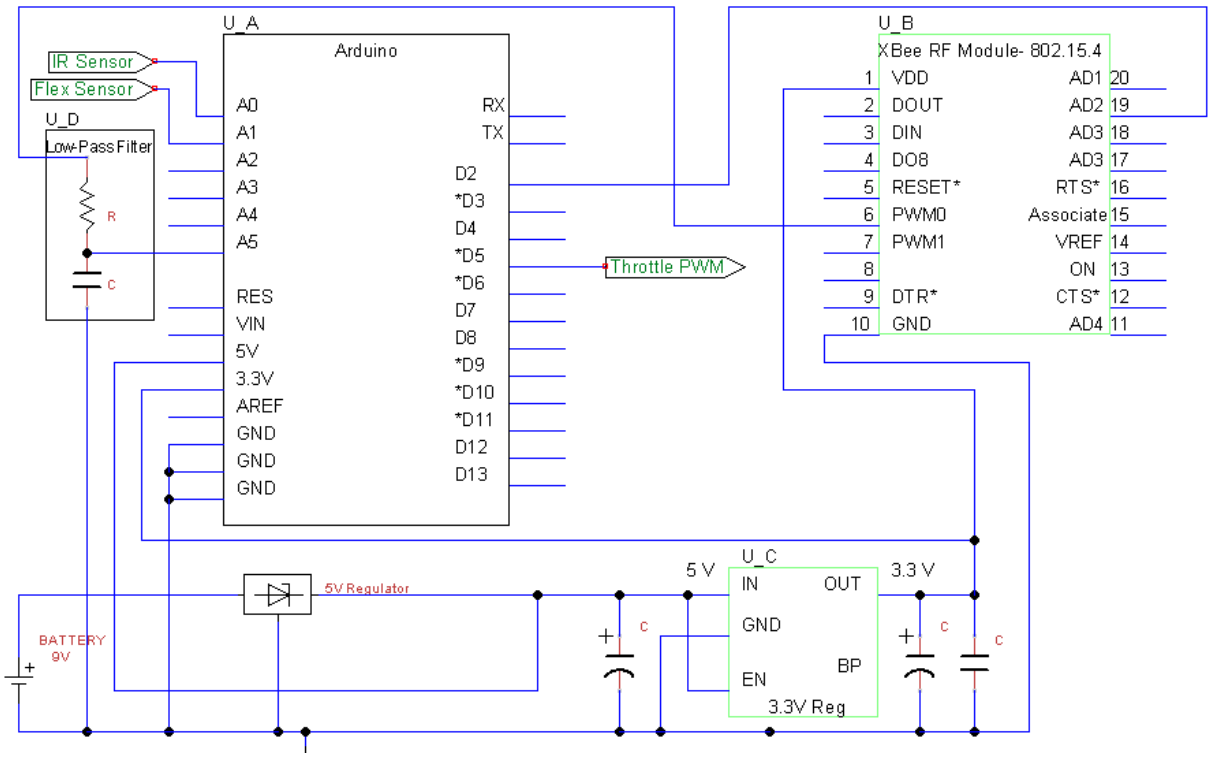

Figure 4. Micro-Controller Circuit

2.2.5 9V Battery (Kill Switch)

We decided to use separate batteries for the Arduino, kill switch, and IR sensors. Admittedly it will be a bit strange having them be consumed at different rates, but it proved to be unwise to place them all on one battery. All three modulescontinuously draw current, and would force users to continuously replace the battery. The decision to choose 9 volts discussed

2.2.6 Kill Switch

The kill switch is implemented by a flex sensor running along the underside of the longboard. If the longboard does not have a rider, the board and the flex sensor are straight. If the longboard has a rider, the flex sensor bends, changing its output. However, the flex sensor's output is its resistance, which the Arduino does not take as an analog input. As a result, we created a small voltage divider circuit that would convert the resistance into voltage.

Initially we chose to input 6 volts into the voltage divider, but the change in voltage between an occupied and unoccupied board was too low. Having a slightly lighter rider would cause the flex sensor to be unreliable. As a result, we increased the voltage to 9 volts, causing the change in voltage to be over 500 ohms so long as the rider was heavier than 140 pounds. The longboard itself is not considered safe for users less than 140 pounds, so this was an acceptable threshold.

2.2.7 Single Motor Turn Assist

Turn assist was implemented with a single infrared (IR) sensor on the left side of the board. Were the user to engage in a left turn, the IR sensor would be closer to the ground and output a higher voltage. Conversely, a right turn would cause the IR sensor to be further from the ground and output a lower voltage. Both these voltages could be directly inputted into the Arduino and activate a specific snippet of code.

Several challenges arose while implementing the IR sensors. Installation was straightforward, but several extenuating factors affected the output of the sensors. Firstly, the IR sensors are extremely sensitive, and users of different weight would cause the sensor to be slightly closer to the ground, causing thresholds to vary from user to user. Secondly, the IR sensors were dependent on the surface: when the longboard was on carpet, the wheels sank slightly, and the sensor's reading was different from pavement. While the longboard typically is not used on carpet, it underlines a potential problem that could arise on other surfaces. However, for typical use on pavement, the system proved rather reliable.

2.2.8 Microcontroller

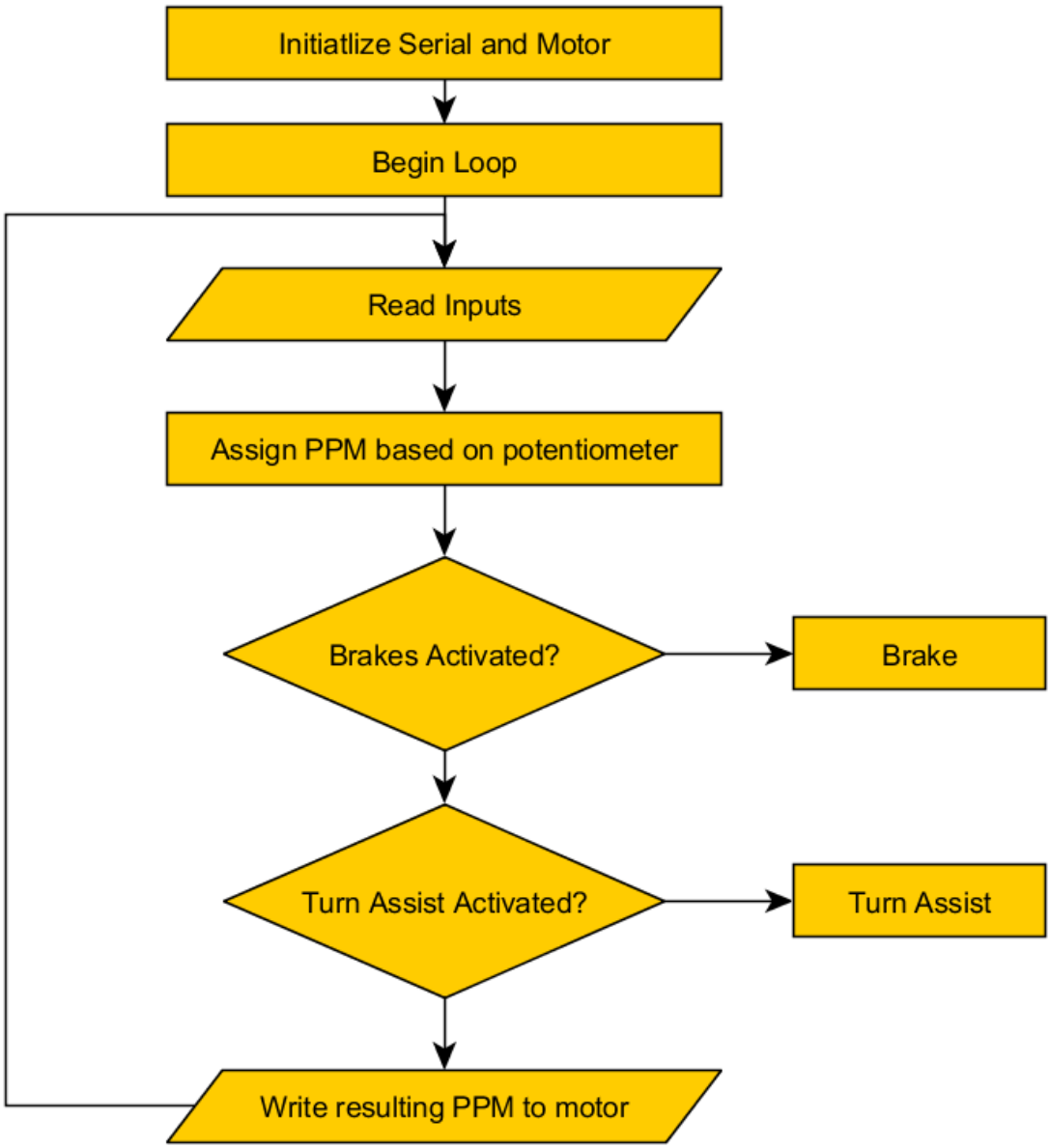

Figure 5. Flow Chart

The code begins by initializing the serial and the motor. Although serial initialization is not necessary for longboard output, it enables debugging and is an integral part of understanding the inputs and outputs flowing through the microcontroller. Once the loop begins, it starts by reading in all the inputs. There are three analog inputs: potentiometer voltage (via the low pass filter circuit), flex sensor resistance (via

the voltage divider circuit), and the IR sensor voltage. The `analogRead(pin)` returns a value between 0 and 1023, so all of the inputs had to be mapped to the appropriate voltage range. This was done using equation 2.1:

$$Voltage = ana\log \mathrm{Re}\, ad(pin)(\frac{range}{1024}) \quad (2.2.8)$$

The ranges for the potentiometer, flex sensor, and IR sensor are 0.0-3.3, 0.0-9.0, and 0.0-3.0 volts respectively. There was also one digital input into the Arduino, the digital output from the XBEE.

The first function uses only the voltage from the potentiometer to determine an initial pulse position modulation (PPM). We decided to implement four stages: deceleration, neutral, and two acceleration stages. We decided to use discrete acceleration steps because it allowed for easier testing and equally smooth acceleration. The acceleration is capped by the battery's current output, so we wanted to guarantee that regardless of the user's input on the potentiometer, it would not cut the battery off.

The next step is to check the brake conditions: the flex sensor and the button. If either is true, it enters a brake function. Not featured in the flow chart is a secondary decision within the brake function. If only the kill switch is implemented, the code will delay for just over one second before enacting the brake. This was done to accommodate more experienced users who enjoy jumping on his/her respective longboards. If the brake conditions are fulfilled after the delay, the braking will continue until the brake conditions are failed.

The last function is the turn assist function. Threshold voltages were established when installing the IR sensors. We wanted to ensure that turn assist would not be enacted with simple shaking of the board. Within the function, depending on whether the IR voltage was over the left turn threshold or below the right turn threshold, a different multiplier would be applied to the PPM and written to the motor. The PPM written inside of the function was independent of the typical PPM, as the board must return to normal operation as soon as the user exits the turn. Also included in the turn function is a check for braking, as the user may disembark or choose to brake while in the middle of a turn.

After all the functions have run, the PPM is written to the motor and the code returns to the beginning of the loop.

2.2.9 Motor controller

The motor controller served as the intermediary between the Arduino and the motor itself. The motor controller converts the DC PPM into three-phase for the brushless motor. The motor controller also allows for electronic speed control through PWM that is outputted through the Arduino. There are also many programmable options to the motor controller. We programmed the motor controller to have an immediate acceleration rather than a soft acceleration because the motor had trouble starting in this mode. Also, our code implemented an acceleration that was quick but controlled enough for the rider to handle.

2.2.10 Motor

We decided to use a brushless dc motor to run our single-wheel drive. The RC motor provided more than enough torque to run at 15mph with a load of approximately 150lbs. The motor is also significantly

lighter and more compact compared to other motors that we could have used, such as an induction motor. We had to use a motor controller that converted DC voltage to a three phase voltage source. Since the motor runs at such high RPM, we used a gearing system to bring the RPM down by a factor of 6 to 1.

2.2.11 39V Battery

We used a 39V battery that was rated for 40A. This was more than enough to run the motor and was below the rated current of both the motor and the motor controller. The battery allows for an operating range of 4-8 miles and provides a steady voltage with deviation of less than 5%.

Knowing that the desired RPM is 6000 and the kV rating is 250 kV:

$$V_{battery\ desired} = \frac{RPM_{motor}}{kV\ rating} = \frac{6000\ RPM}{250\ RPM/volt} = 24\ volts \qquad (2.2.11)$$

Now a voltage of at least 24 volts must be chosen for the battery. LiFePO4 type batteries were chosen for this project due to their high discharging current, non explosiveness, and long life cycles. LiFePO4 batteries are created by putting single cells in series, and each cell is usually rated at 3.7 V. So the desired cells of the battery S are:

$$S_{battery} = \frac{24\ v}{3.7\ v} = 6.48\ cells \qquad (2.2.11)$$

A 7 cell battery must be used for this project, which is rated at 25.9 V. The LiFePO4 battery that we have selected is rated at 25.6 V, which will fit the minimum speed requirements, even with losses due to load and wire resistance. A higher voltage battery will be considered if funding exists, and if so, a 38.4 V battery will be used in order to achieve max speeds of 20 mph.

3. Design Verification

3.1 Remote Controller

3.1.1 Joystick Potentiometer

We chose to use a joystick potentiometer as the means in which the rider controls the speed of the longboard. A joystick allows the user to easily control acceleration and deceleration with his/her thumb without having to twist a conventional potentiometer. We also decided to incorporate a two axis joystick with the x-axis disabled.

Since the XBee are rated for 3.3V, we also powered the potentiometer with a 3.3V voltage source. We tested the output of the joystick by connecting it to an oscilloscope. The lower and upper voltage bounds were 0V and 3.3V respectively. In the neutral position, the output of the joystick was measured to be around 1.6V.

The output of the y-axis is connected to one of the signal inputs of the XBee. We used a multimeter to probe the pins of the transmitter to verify that the voltage seen by the XBee was consistent with the voltage outputted by the joystick.

3.1.2 Brake Button

We decided to use a push button to brake the longboard. We also connected 3.3V to the button so that it would send a voltage high of 3.3V when pressed. The output of the button was connected to a multimeter; when pressed, the button sent a voltage of exactly 3.3V.

3.1.3 On/Off Switch

An on/off switch allows for the rider to turn off the remote to extend the battery lifetime. The multimeter shows the battery voltage when switch is turned on and no voltage when the switch is turned off.

3.1.4 Remote Battery

We tested to make sure that the remote battery was providing 12V for the remote control. We ran 2 AA batteries and 9 V battery in series. The voltage was measured with the oscilloscope to verify that it would provide power for the controller circuit.

3.1.5 Transceiver

Our group used XBees to transmit and receive signals from remote control to the microcontroller. We needed to make sure that the XBees were being supplied a rated voltage of 3.3V. Since the pins of the XBees are spaced to narrow, our group used a USB adapter and an Arduino shield to power and connect wires to the XBees. First the USB and Arduino were powered at their rated voltage of 5V and 9V respectively, and the pins powering the XBees were probed to confirm that the XBees would be receiving the right amount of voltage. Then they were mounted on the adapters, and the pins of the XBee module were measured with a voltmeter to ensure that the XBees were at 3.3V.

We probed the devices sending digital and analog signals to the transmitter were under 3.3V by using a voltmeter. Both the brake button and joystick potentiometer were measured, and they sent a voltage of 3.3V and 0-3.3V respectively.

In order to make sure that the transmitter and receiver were connected and transmitting the correct signals, we set the XBees to run at the same frequency of 30 Hz.

$$9600Baud = \frac{9600bits}{\sec} * \frac{1byte}{8bits} * \frac{packet}{40bytes} = 30Hz \quad (3.1.5)$$

Then we assigned one XBee to be the receiver and one to be the transmitter. The XBee and Arduino terminal with serial writes to ensure that the XBees were communicating correctly.

The XBees are able to work up to a distance of around 100 ft at the lowest baud rate. Since we used the third lowest baud rate, we expected the range of communication to be lower than 100ft. We tested the range by increasing the range between the transmitter and receiver until connection was lost. We tested that the XBees work reliably up to around 75ft.

3.2 Longboard

3.2.1 Micro-Controller

We tested that the Arduino Uno micro-controller was receiving the right battery voltage, we tested the power supply and VCC of the Arduino. The multimeter showed that the Arduino was being powered at around 9V, which is within the rated voltage.

Each of the signals being sent from the IR sensor, flex sensor and receiver XBee needed to be recognized correctly. The analog signals from the IR sensor, flex sensor, and XBee were connected to the analog inputs, and the digital signal from the XBee was connected to a digital pin that was programmed as an input. The Arduino serial terminal was used to verify that the digital and analog signals were being read correctly.

The analog voltage from the joystick is outputted as a PWM from the receiver XBee. In order for the PWM to be recognized as an analog input to the Arduino, we need to construct a low-pass filter with a threshold frequency lower than the operating frequency of 30Hz.

$$f_{threshold} = \frac{1}{2\pi RC} < 30Hz \quad (3.2.1)$$

Functionality of the software was tested by using the Arduino serial terminal and hard coded values. Instead of using inputs from all sensors and the XBee, we tested each component of the code individually. Then we incorporated all the inputs and tested to see that the code would function. Once we got the code to function correctly with all the different inputs, powered the longboard and calibrated the values to maximize user experience.

3.2.2 IR Sensor

The testing for the motor controller was done in two parts. First a multimeter was hooked up to the IR sensor and different distance on the longboard were measured and recorded to verify that the IR sensor could show a large enough voltage difference to signify different turns. A range of 0.65 V to 1.1 V was seen, which was sufficient. Then, the IR sensor was tested with the Arduino and the PWM was monitored using an oscilloscope. The left turn was tested to see if the PWM drops, and the right turn was tested to see if the PWM increases. Both these tests passed, and the IR sensor was determined to be working successfully.

3.2.3 Flex Sensor

The flex sensor was verified to see if the flex sensor was increasing and decreasing resistance when flexed. This was done by hooking up a multimeter and checking the resistance across the sensor when straight, bent inward, and bent outward. A resistance of about 9.2 kOhms was seen when straight, 13 kOhms when bent outward, and 8 kOhms when bent outward. Then, the flex sensor was tested on the longboard to see if the longboard would flex enough to flex the sensor. A multimeter showed that when straight, the flex sensor had a resistance of 9.2 kOhms and while a rider stood on top of the board, a resistance of 9.8 kOhms was seen.

3.2.4 Motor Controller

The motor controller test was easy. We simply hooked up the battery, motor, and Arduino, and made sure that the motor controller's self test beeps verified a working motor controller. This was signaled by two long beeps, two short beeps, followed by one long beep.

3.2.5 Motor

Motor testing was primarily done in two parts. First the motor controller, motor, and battery were connected. A hardcoded PWM was sent through the Arduino to see if rotation was confirmed in the motor. Then, the motor was tested while installed on the board to see if a minimum top speed of 13 miles per hour could be achieved. This was done by running the board at top speed and measuring the time it takes for the longboard to run 100 yards with a rider on it. Our tests showed that the motor on average was able to traverse 100 yards in less than 15 seconds, which measures at a speed of greater than 13 miles per hour.

3.2.6 Battery

The battery was tested by hooking up to a multimeter and testing to see if the voltage was 38.4 V +/- 10%. The measured voltage on average was about 39V. Then, the battery was made sure that it could successfully recharge using the charger, and this was verified by an increase in voltage seen by the multimeter. Lastly, the battery's lifetime was measured while installed on the longboard. A distance of four miles was travelled, and the voltage on the battery still held a value of about 37V.

4. Costs

4.1 Parts

Table 1 Parts Costs

Item Name	Item No.	Retail Unit Cost	Actual Unit Cost	Item Quantity	Total Item Cost
Loaded Vanguard (Complete Longboard: Flex 2, Paris or Randall 180 mm, Abec 11 80A Big Zig, Bones Super Reds)	N/A	$320.00	$80.00	1	$80.00
Great Planes Rimfire 1.60 Outrunner Brushless Motor	GPMG4795	$179.97	$179.97	1	$179.97
LiFePo4 Battery (38.4V, 9.9Ah, 380 Wh)	CU-JAS199	$398.88	$398.88	1	$398.88
Arduino Uno Revision 3	DEV-11021	$35.00	$35.00	1	$35.00
Hobbywing Platinum 60A Pro ESC	Hobbywing-Acc-HW-60A-ESC	$65.36	$65.36	1	$65.36
Smart Charger (6.0A) for 25.6V LiFePo4 Battery Pack	CH-L2596N	$78.83	$78.83	1	$78.83
Flex Sensor 4.5”	SEN-08606	$15.00	$15.00	1	$15.00
XBee Chip Antenna + Shield	WRL-08664	$38.95	$38.95	2	$77.90
Thumb Slide Joystick	CO-09426	$3.50	$3.50	1	$3.50
Mini Push Button Switch	COM-09190	$0.50	$0.50	1	$0.50
2.2 x 6.5 in. Breadboard	103-1100	$12.88	$12.88	2	$25.76
9V Battery	EN22	$1.34	$1.34	10	$13.40
Infrared Proximity Sensor Short Range	Sharp GP2D120XJ00F	$13.95	$13.95	1	$13.95
Power MOSFET P-Channel	SUP65P04-15-E3	$3.11	$3.11	2	$6.22
Power MOSFET N-Channel	FDP3672	$1.75	$1.75	2	$3.50
Power MOSFET N-Channel	IRFI540NPBF	$1.77	$1.77	2	$3.54
PCB	N/A	$40.00	$0.00	1	$0.00
Total Part Cost					**$1086.43**

4.2 Labor

Table 2 Labor Costs

Name	Hourly Rate	Overhead (2.5)	Hrs/wk	# wks	Total Hrs	Total
Daniel Moon	$40.00	$100.00	10	11	110	$11000.00
Kevin Lee	$40.00	$100.00	10	11	110	$11000.00
Leon Ko	$40.00	$100.00	10	11	110	$11000.00
Machine Shop Labor	$40.00	$100.00	20	1	20	$2000.00
Electronics Parts Labor	$40.00	$100.00	5	1	5	$500.00
Total Labor						$35500.00

4.3 Total Cost

Total Cost = Total Part Cost + Total Labor = $1086.43 + $35500 = **$36586.43**

5. Conclusion

5.1 Accomplishments

Overall we were very pleased with the results of our project. The general function test that required the board to move forward at a rate of at least 13 mph was passed. This was tested by setting a distance of 100 yards, allowing the board to accelerate to top speed, and then recording the time it took for the longboard to travel the specified distance. This basic accomplishment was exciting because it signified a successful implementation of the longboard, motor and motor controller and would eventually lead to the success of many other components.

The remote control was a component we spent a great amount of time on, and eventually the remote worked as desired. The connection to the board's microcontroller did not show a delay, and successfully transmitted signals that made the longboard accelerate, decelerate, and brake.

The kill switch function was one of the bigger safety features we installed into the board, and it was fully functional by the end of our project. It successfully stopped the board without a rider being detected, and would stop motor function if the rider were to leave the board in the middle of the ride.

The assisted turning was also successful in board. When making a left turn, the longboard would detect the turn and adjust the back left wheel to a slower speed, allowing the board to maintain speed. When making a right turn, the longboard would adjust the back left wheel to a faster speed. This module allowed for an easier riding experience and was overall a success.

Lastly, although our reverse function of the board was not achieved, the H-bridge that was designed in order to reverse the board was an accomplishment. The H-bridge was successfully designed, but due to the inefficiencies of using a two-phase H-bridge on a three-phase motor, this module was not implemented into the final design of the longboard.

5.2 Uncertainties

Although our project can be considered an overall success, there are certain uncertainties that surround the project. First, the marketability of our board can be called into question. Due to the heaviness of the battery pack, gear box, motor, and aluminum structuring, potential riders may be deterred from purchasing one of our boards. Also, the board can be considered aesthetically displeasing, which is another factor that may stop potential buyers from purchasing. Second, the overall safety of the board is a concern as well. Although the board functions perfectly under normal conditions, the safety of using the board depends heavily on the rider's skill level. This can cause the number of potential customers to drop as well as potential risks in commercializing the project. Lastly, the board's ability to last in bad weather is an uncertainty as well. Due to the battery pack being exposed, the board may simply be only used for nice weather.

5.3 Ethical considerations

With accordance to the IEEE Code of Ethics, throughout the entirety of this project, we vowed the following:

1. "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"

   In the interest of public safety, all testing of this longboard was done away from busy streets and high population density areas. The board was also designed with the safety and welfare of the rider in mind, as well as pedestrians and other vehicle riders.

4. "To be honest and realistic in stating claims or estimates based on available data"

   All claims of battery life, top speeds, and other specifications of this longboard were calculated, with the goal of obtaining the most accurate claims and simulation results possible.

7. "To seek, accept, and offer honest criticism of technical work, to acknowledge and correct errors, and to credit properly the contributions of others"

   As our group goes through the different reviews and criticisms, we accepted all comments with open minds, were upfront about our errors, and credited those who had contributed to our project outside of the group.

5.4 Future work

5.4.1 Reverse Function

A reverse function could be useful if the longboard gets into an area that it cannot turn out from. Usually, a rider would have to pick up his or her board and place it back on the ground in a more ideal spot, but with a reverse function, the rider may never have to leave the board when riding.

5.4.2 Lighter Implementation

Overall lighter and more compact implementation would be ideal for a more commercial-ready product. Reducing the size of the battery pack and installing the motor underneath the board rather than on top would make the longboard look more aesthetically pleasing and easier to carry around.

5.4.3 Regenerative Braking

Regenerative braking would increase the distance that the board could run. This recommendation would be able to take advantage of the fact that longboards usually have to make consistent stops during travel. The energy from each stop could be harvested and put back into the battery.

5.4.4 Dual Wheel Drive

Dual wheel drive would allow for active drive to be implemented, which could greatly help in sharp turns while riding. This would require two motors and bigger motor housing.

5.4.5 Miscellaneous

Other recommendations include an ergonomic controller, switches and low battery detection, as well as smoother acceleration. All of these recommendations would improve the rider's experience while on the longboard.

References

[1] "IEEE Code of Ethics." IEEE - Advancing Technology for Humanity. N.p., n.d. Web. 25 Feb. 2013.

[2] "Xbee / Xbee-PRO RF Modules." *Sparkfun*. N.p., 23 Sept. 2009. Web. 25 Feb. 2013.

[3] "Xbee Shield." *XBee Shield*. Sparkfun, n.d. Web. 26 Feb. 2013.

[4] Gammon, Nick. *Gammon*. Ed. Nick Gammon. N.p., 22 Dec. 2011. Web. 25 Feb. 2013. <http://www.gammon.com.au/forum/?id=11473>.

[5] "The Brushless Advantage for Outrunner Design Motors!" *ElectriFly*. N.p., n.d. Web. 26 Feb. 2013.

[6] “The Ultimate Guide to Lipo Batteries - Electric Motors, ESCs, Batteries, Etc. @ URC Forums." *Ultimate RC*. Ultimate RC, 14 Nov. 2010. Web. 26 Feb. 2013.

Appendix A Requirement and Verification Table

Requirements	Verifications
Longboard Battery 1. Output voltage of the battery is 38.4V +/- 10% 2. Operating range (without any regenerative braking) of 4-8 miles with 150 lb load (assume 150 lb is average body weight of rider) 3. Can be successfully recharged 4. Can be successfully recharged with motor/generator	**Longboard Battery** 1. Probe the battery terminals and confirm that the voltage falls within desired range. 2. Run board at an average speed of 15 mph with rider until battery expires 3. Send power to the battery and use a voltmeter to check if it is being charged 4. Probe the output voltage of the motor and test compatibility with motor/generator
Kill Switch 1. Ensure flex sensor can bend as much as the board with maximum load 2. Check that change in flex sensor creates a large enough resistance change 3. Measure the correct correlation between bend/weight and change in resistance/voltage	**Kill Switch** 1. Apply maximum load of around 275 lbs to the board with the flex sensor attached and confirm that the flex sensor still works 2. Attach the flex sensor to the board and add load until there is voltage change of around .1 V to find the smallest radius of curvature 3. Compare resistance from flat to resistance with loads ranging from ~50lbs to 270lbs with a step of 10lbs to derive an accurate correlation
Micro Controller 1. Ensure that ~4.8-5.2V is supplied to the Arduino via voltage divider. 2. Constant data transfer between XBee and microcontroller a. Arduino must receive the information from the potentiometer via the XBee b. Arduino must receive the information from the button via	**Micro Controller** 1. Use a voltmeter and confirm that voltage falls within desired range. 2. Wire XBee to Arduino, and connect both to PC. a. Open serial connection Send a specific signal from the potentiometer (max voltage), and print it from Arduino to PC. Compare the printed data to that transferred. Repeat for 5-10 trials to ensure accuracy b. Open serial connection Send a specific signal from the

the XBee	button (pushed) and print it from Arduino to PC. Repeat process for when button is not pushed. Repeat for 5-10 trials to ensure accuracy
3. Arduino must send appropriate PWM to motor	3. Connect Arduino to motor Run a PWM of 50% (+/- 5%) to motor. Measure RPM and check it keeps within 45-55% of max RPM. Repeat for several trials to ensure accuracy. Repeat for other PWMs (25%, 75%)
4. Constant data transfer between flex sensor and microcontroller	4. Connect flex sensor to Arduino, and connect both to PC There is no other way to measure the resistance of the flex sensor. However, it is important that the bending of the flex sensor has a corresponding change in resistance. As the sensor is bent, output the resistance to display, and check that it increases accordingly up to the max resistance of 125K Ohms
5. Code that compiles and runs properly a. Regenerative breaking	5. Individual portions of code compile/run a. Check flex sensor to see if is at 30KΩ +/- 5KΩ. If not, check whether the button has been pushed on remote (assuming 2b has been verified). If either is true, activate the code that enables regenerative breaking. While the resistance stays in the 25-35KΩ region or while the button remains pushed, regenerative breaking will remain enabled. Otherwise, disable and return to reading input. Test code by first leaving flex sensor in the appropriate range of 25-35KΩ and see whether regenerative breaking enables. Administer 5-10 trials to affirm validity. Next, test whether the code is enacted after the button is pushed for duration of 3-5 seconds. Administer 5-10 trials to affirm validity. Next, test that regenerative breaking is not

b. Acceleration	activated when the flex sensor is above 35KΩ or while button is not pushed. b. Assuming 2a has been verified, check for the signal from the XBee. Offset the incoming signal by 1.5V (the default setting for the potentiometer). Check whether the resulting signal is positive or negative, and modify the PWM accordingly. Test code by first trying several positive voltages (1.6, 1.7, 1.8…up to 3.0V) repeatedly and outputting the corresponding PWM to display. The positive voltages should net a PWM of up to 80% +/- 5%. Next test the negative voltages (0.1, 0.2, 0.3…up to 1.4V) repeatedly and outputting the corresponding PWM to display. The negative voltages should net a PWM of up to 20% +/- 5%.
Motor 1. Ensure smooth transition to generator 2. Must provide enough thrust to propel longboard (with 150lb load) at 13-20mph 3. Make sure the current capacity does not exceed boundaries (which would overheat the motors)	**Motor** 1. Run dummy code that pushes motor to become generator and vice versa 2. Calculate the torque required to run a longboard at desired speeds with load. Test amount of torque generated when running forward with max PWM. 3. Use resistor and ammeter to test current
Turn Assistance 1. Making a turn results in change in IR output 2. Change in IR output results in change in PWM	**Turn Assistance** 1. Test IR sensor independently with Arduino code. Probe the sensor outputs to ensure correct readings. 2. Ensure that code results in reduced PWM sent to motor if IR is not within normal range. Use an oscilloscope to test PWM so that changes in the IR sensor correctly correlates to the changes in PWM.

Communication 1. Communication range of 100 ft +/= 10% (2.4 GHz) 2. Ensure that 3.3V+/- 10% is supplied to XBee 3. Check that correct data is transmitted	**Communication** 1. Test that the data transferred by one XBee is received by the other XBee 2. Use voltage divider and test leads going into XBee 3. Connec t XBee to Arduino and run code that outputs data
Remote Battery 1. Check that batteries supply a total of 2.8~3.4 V for the XBee 2. Check that batteries supply enough voltage that the potentiometer will still give meaningful differences in output 3. Battery lifetime of 20 Hours	**Remote Battery** 1. Use a voltmeter to test total voltage going into XBee 2. Arrange final configuration of batteries, and connect potentiometer. Test sensitivity and change in voltage with a supply of ~2.8-3.4V 3. Test the lifetime of 2 AA batteries in series
Remote Controls 1. Ensure that potentiometer outputs a voltage of range ~0-3.0V 2. Use pull-up resistors to regulate input 3. Ensure button is registering signal	**Remote Controls** 1. Test with dummy Arduino program 2. Check that inputs does not change with turbulence 3. Connect button directly to Arduino/LED and test signal