Interfacing Electrical Components with the BeagleBone Black Microcontroller for a Steering Controller

---

Autonomous Bicycle Project
Semester Report | 12/18/2014
Ariam Espinal

Contact Info:
Email: Aee49@cornell.edu
Cell: (646) 338-5958
Senior, Mechanical Engineer

Course Info:
MAE 4291 – Supervised Senior Design
Section 742 - 3 credits

Table of Content

I. Introduction

The goal of this project is to adapt an old Dahon folding bicycle from Professor Ruina's lab to steer and propel itself at a constant velocity and stably remain upright. To control the steering, a Maxon F 2260 motor was mounted directly to the bicycle's turning fork. To propel the bicycle forward, a Magic Pie 2 motor hub was placed on the rear wheel. The steering motor and motor hub are controlled by a BeagleBone Black microcontroller. The BeagleBone Black executes a python script which sends a constant voltage signal to the motor hub and runs a feedback loop to control the steering motor velocity. The feedback loop begins with the microcontroller gathering data from two sensors: a Bourns 10KΩ Continuous Potentiometer, measuring the steering angle and a Microstrain Inertial Link, inertial measurement unit (IMU) measuring the lean angle of the bicycle. The voltage signal from the potentiometer is passed through a voltage differentiator in order to measure the steering angular velocity. Using the measured values from the potentiometer and the IMU, the microcontroller calculates a target steering angular velocity that will bring the upright if it is tipping over. The calculation for the target steering angular velocity is based on a controller for a simplified dynamic model of the bicycle. With a target steering angular velocity, the microcontroller then calculates a duty cycle for a pulse width modulated (PWM) signal to output to the motor based on a proportional integrating derivative (PID) controller. While repeatedly sending PWM signals to the steering motor, the microcontroller sends a constant signal to the motor hub to maintain a constant velocity on level surface. Once the microcontroller output has outputted to the motors, the loop begins again. By having this loop run fast and consistently, the bicycle should be able to correct for error and stabilize itself.

II. Electrical Components

In order to implement the control loop, all of the electrical hardware involved must be able to send and receive appropriate signals from the BeagleBone Black. To achieve this, it is key to understand the inputs and outputs of the electrical hardware as well as the kind of signals that the BeagleBone needs to send the hardware.

II. A. Microcontroller

The microcontroller used for this project is the BeagleBone Black (BeagleBone), which is running the Angstrom distribution of Linux. The BeagleBone was selected to be used, as opposed to other microcontrollers like Arduino, because of its speed and versatility. The BeagleBone has a 1GHz clock speed, it allows users to code in a variety of languages, and has multiple pin output functionalities. These pin outputs include, general purpose input-output (GPIO), pulse width modulation (PWM), analog to digital converting (ADC), and serial communication through a universal asynchronous receiver/transmitter (UART). A pin output diagram showing all of the pins on the BeagleBone can be found in Append A. The GPIO pins are used to send a voltage signal to the motor hub, PWM pins are used to control the motor output, the ADC pins are used to read the voltage on the steering potentiometer, and UART pins are used to send and receive data packets of the lean angle from the IMU. Generally all of the output pins on the BeagleBone produce 3.3V; with the exception of the ADC pins that produce 1.8V. The maximum current output from the pins is 2.5 mA. A Python script that is written onto the BeagleBone's onboard memory executes all of the necessary pin outputs and inputs. Python is used for this project because of its large online library of source code and the support there is for python on the linux platform.

II. B. Potentiometer

A Bourns 10kΩ continuous potentiometer is used to measure the steer angle. Using the BeagleBone's ADC pins, we can power and read the potentiometer's analog output. Pins P9_32 and P9_34 provide 1.8V and ground respectively to the potentiometer's 1st and 3rd terminal. Pin P9_40 reads the voltage difference between ground and the 2nd terminal. When the BeagleBone reads a voltage, its ADC scales down the 0-1.8V signal to a number between 0 and 1. This value can then be rescaled to a number between 0-360 or 0-2π that corresponds to the current steer angle. In converting the voltage from analog to digital there is a loss in resolution, which is due to the fact the BeagleBone uses a 12 bit analog to digital converter. With noise in the voltage

signal, the angle measurements range about 0.8 degrees when the potentiometer is stationary as shown in figure 1 below.

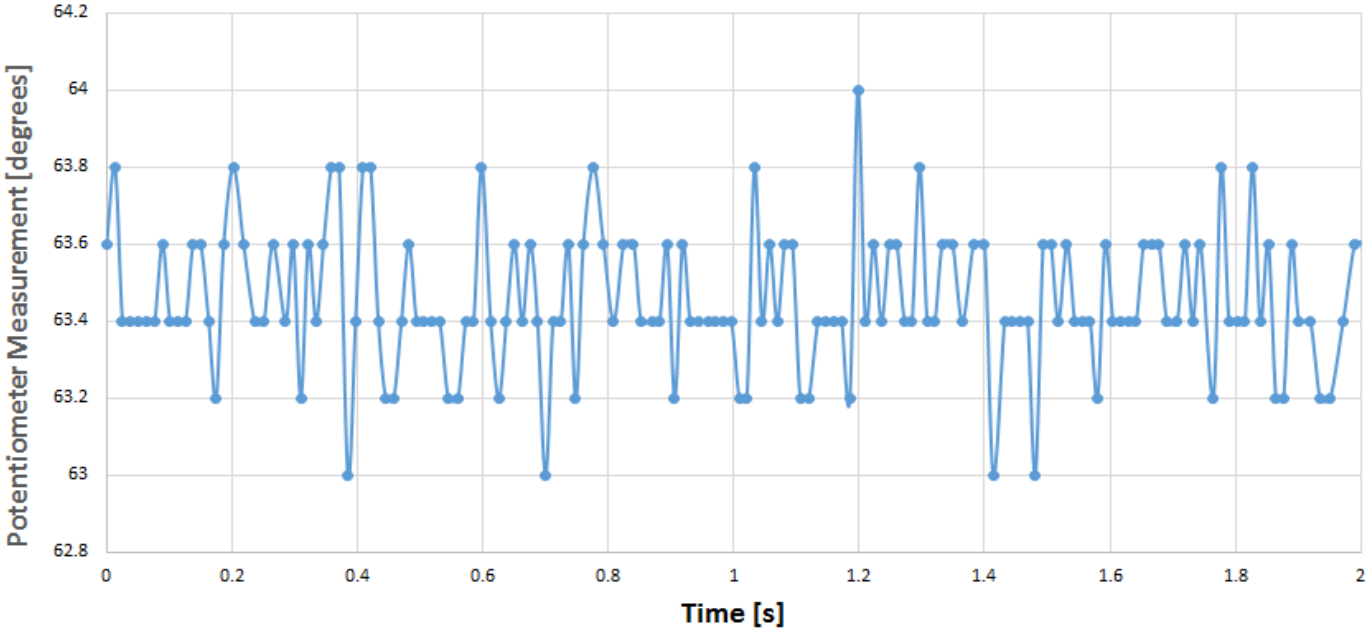


Figure 1 - The noise in the signal potentiometer signal causes the angle measurement to bounce around ± 0.4 degrees while the potentiometer remains stationary. This plot also shows that the resolution of the signal is 0.2 degrees.

II. C. Inertial Measurement Unit (IMU)

The IMU we are using is Microstrain's Inertia-Link. The IMU uses RS-232 9 pin connector for its serial communication, which the BeagleBone does not natively support, to relay bytes of data. As mentioned earlier, the BeagleBone uses a UART to do serial communication. To get around this a RS-232 Serial Micro Cape is used to bridge between the two connectors. Once connected, the BeagleBone first needs to initiate the IMU by sending two command bytes indicating what data is desired from the IMU. In this case, the desired data is the roll angle of the IMU. The IMU then sends a string of 31 bytes back to the BeagleBone that get placed on a buffer. The received bytes contain information about all of the euler angles that need to be parsed to extract the desired roll angle. The BeagleBone then takes the string of bytes off of the buffer and converts them from hexadecimal to a float number using the IEEE 754 standard. Initially this

process was done in two steps, the first sending the command byte, and second reading and parsing the received bytes. However, now with multiprocessing incorporated into the Python script, the BeagleBone can now send a command byte and read the previous returning string at the same time.

II. D. Voltage Differentiator

Normally a tachometer would be used to measure the angular velocity of the motor, but the current mechanical setup does not have the space to attach one to the motor. Instead a voltage differentiating circuit is being developed to give the BeagleBone feedback as to how fast the steering motor is turning. The circuit consists of an op amp, resistors and a capacitor connected as shown in Appendix B. The resistor and capacitor used are $680\,k\Omega$ and $0.068\ \mu F$ respectively. The op amp is powered by the BeagleBone's ADC 1.8V just like the potentiometer. The input of the differentiating circuit, V_{in} is the voltage signal that is outputted by the potentiometer. The output of the differentiating, V_{out} circuit is then connected to P9_39, which is an ADC pin on the BeagleBone. The relationship between the input and output voltage is:

$$V_{out} = -RC\frac{dV_{in}}{dt} \tag{1}$$

where R is the resistance of the resistor and C is the capacitance of the capacitor. The negative in the equation indicates that the output voltage can be lower than the ground provided by the BeagleBone. However, the BeagleBone's ADC is not made to handle negative voltage, so the ground level of the op amp was raised to 0.9V, so that the op amp's operation range would be 1.8-0V as opposed to -0.9 - 0.9V. This was done by adding two resistors of equal resistance as a voltage divider between the Beaglebone's ground and 1.8V output.

Using the relationship between the potentiometer angle, θ and the potentiometer output voltage:

$$\theta = 200V \tag{2}$$

and equation 1, a relationship between the the voltage output of the differentiator and the steering motor's angular velocity can be found. This is done by differentiating equation 2 and substituting it into equation 1:

$$\frac{d\theta}{dt} = -\frac{200}{RC}V_{out} \tag{3}$$

II. E. Motor

Previously the Yaskawa Minertia J Series Type J02 motor was being used, but it did not put out enough torque to turn the steering fork. Now a 24V 80W Maxon F 2260 motor with a gearbox is being used to assure that there is enough torque to turn the front fork. The gearbox had a 5 to 1 ratio. The majority of the BeagleBone's pins are set to output 3.3V at a max current output of 250mA, which is not enough current to power a motor. To compensate for the BeagleBone's lack of power, we are connecting the BeagleBone to a motor driver which will direct current from an external power supply to the motor. The motor driver we are using is a Pololu High-Power Motor Driver 36v20. The motor driver requires at least two 5V signals to control the current direction and PWM signal that the motor receives. Since the BeagleBone can only deliver 3.3V, a two channel opto-isolator with transistors is being used. The opto-isolator isolates the BeagleBone from the high voltage motor power supply and the transistors amplify the Beaglebone's direction and PWM signals. A schematic of the circuit designed to power the motor using the BeagleBone is in Appendix C. Using a DC power supply, we were able to successfully control the motors direction using the BeagleBone.

III. Motor Control

With all of the electrical components able to communicate with the BeagleBone, a PID for position controller was coded. At the time that the controller was coded the goal was position control as opposed to velocity control. The controller was coded using the PID control laws given from Lab 6 of MAE 3260 System Dynamics which are as follows.

$$e_k = r_k - y_k \tag{4}$$

$$q_k = q_{k-1} + \Delta t\, e_k \tag{5}$$

$$u_k = K_p[e_k + (\frac{q_k}{T_i}) + T_d(\frac{e_k - e_{k-1}}{\Delta t})] \tag{6}$$

Where:

y_k is the current steering angle
r_k is the target steer angle
e_k is the error in the steer angle
q_k is the approximate error integral
T_i is the integral control tuning parameter
T_d is the derivative tuning parameter
K_p is the proportional gain

In order to tune the controller, the BeagleBone was hard coded a target angle. The motor was then offset from the target angle and then the controller was implemented. The parameters were then tuned iteratively until the overshoot and oscillations were removed. Since there wasn't a dynamic model yet to calculate a target steer angle, the target steer angle was set to be proportional to the IMU roll angle. This allowed for the testing of the motor's response to a changing target angle. Now that the interest is angular velocity control, the code has to be altered to use velocity instead of position.

IV. Discussion

The goal of all the circuitry is to relay data back to the BeagleBone. The data is of no use if it comes back with too much noise. To some extent any electrical circuit will have some level of noise and in order to combat this filters can be implemented.

IV. A. Signal Noise

As seen in Figure 1, there is quite a bit of noise in the potentiometer signal which is also amplified by the resolution of the ADC. In order to get a better signal, a second order Butterworth filter could be applied to the signal as shown in Figure 2 below. The filter removes noise that is above a certain cutoff frequency. The drawback of filtering the data is that the signal then begins to lag behind the actual position of the motor. The cutoff frequency of the filter is determined by the sample rate of the BeagleBone and the r-value of the filter. The r-value is the ratio of the cutoff frequency to sample frequency. The BeagleBone samples the potentiometer signal at about 200 Hz, which means that any noise well over 200Hz should be filtered out. Filters also take up time to implement on top of causing the signal to lag. A Butterworth filter requires a couple of calculations that could take up a couple of milliseconds from every loop iteration. As of now we are debating on what r-value to use or if we should use a different filter altogether. Alternatively an exponential filter, which requires less math could be used to clear up the signal. The exponential filter is simple and does not take up much processor time to calculate values.

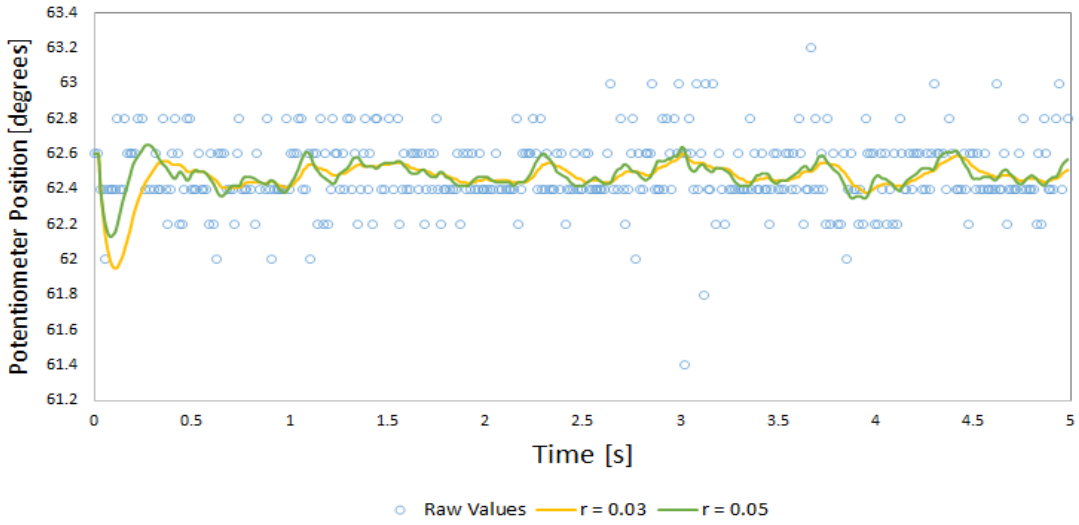


Figure 2 - A second order Butterworth filter was applied to the static potentiometer data from figure 1. The signal now bounces around 0.02 degrees which is also the resolution of the ADC.

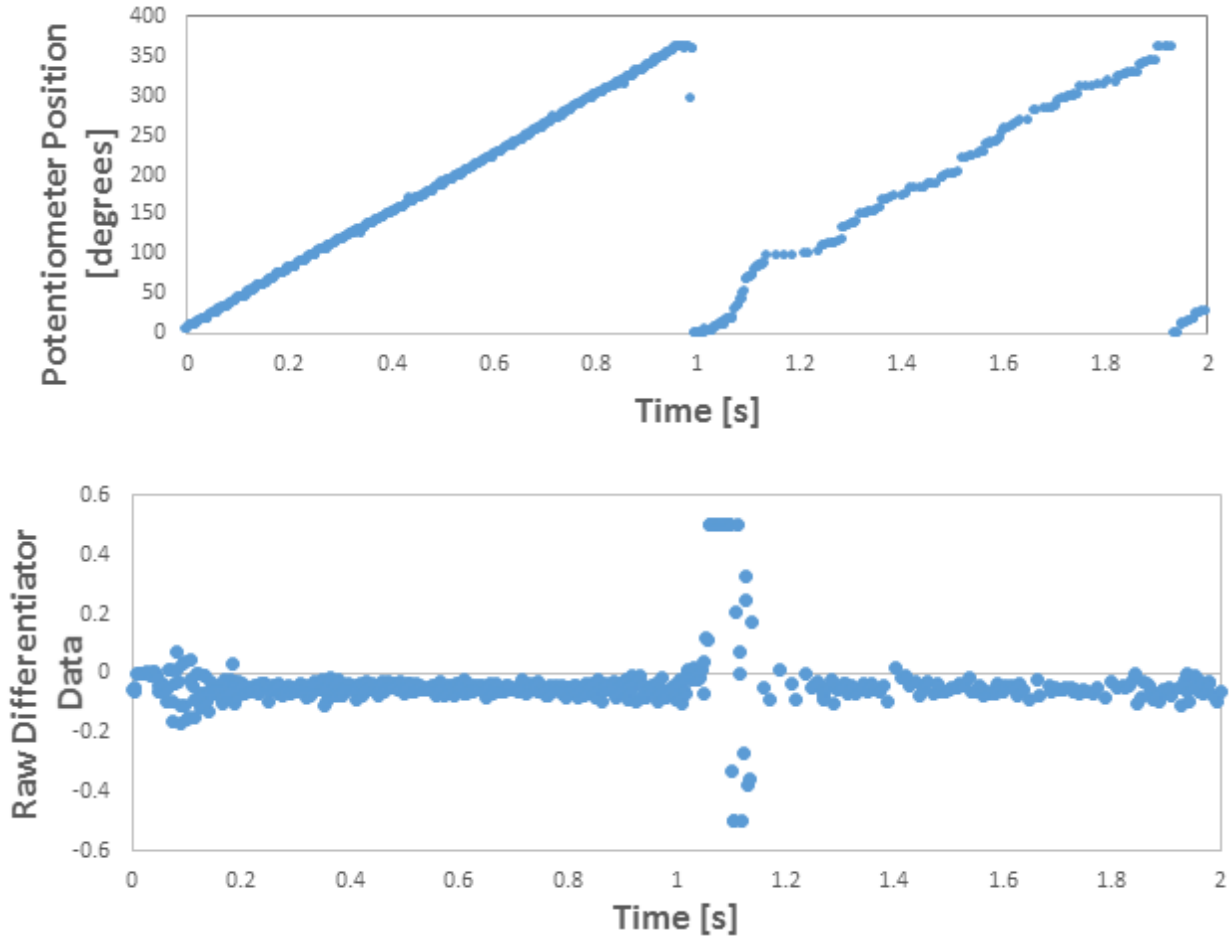


Figure 3 - The steering motor was spun at a constant rate. The top plot is of the potentiometer position and the bottom is the differentiated signal.

The signal noise of the differentiating circuit is even worse than the potentiometers. This is probably due to the fact that the differentiator amplifies the voltage signals that pass through it thus amplifying the noise. In Figure 3, the signal clusters around -0.05 but it is not clear at all what the signal actually is. With this much noise it is not possible to use the differentiator to control the angular velocity. Also there is a huge voltage spike when the potentiometer transitions from 360 degrees to 0 degrees. This occurs because the transition happens so quickly that the op amp becomes saturated. The voltage spike means that the range of operation for the steering has to be limited to between 50 and 300 degrees.

V. Conclusion

Currently all of the electronic hardware necessary for feedback control has successfully been interfaced with the BeagleBone Black. However now that all of the hardware has been connected, the quality of the signals being sent to and from the devices needs to be assured. This can be done by applying digital filters such as the Butterworth filter to the incoming signals. With the current noise level it was possible to do angular position control, but it does not seem possible yet to do the same with angular velocity due to the amplified noise in the differentiator. Once the signal noise has been reduced, an angular velocity can properly be implemented. Also moving forward, all of the electrical and mechanical components need to be mounted within the bicycle so that testing can begin within the first few weeks of the following semester.

VI. Appendix

A. BeagleBone Black Pinout

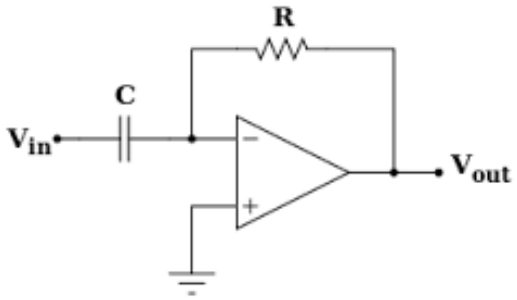
P9

GND	1	2	GND	
3.3V	3	4	3.3V	
5V Raw	5	6	5V Raw	
5V	7	8	5V	
	9	10		
Serial4 RX/GPIO0_30	11	12	GPIO1_28	
Serial4 TX/GPIO0_31	13	14	PWM1A/GPIO1_18	
GPIO1_16	15	16	PWM1B/GPIO1_19	
GPIO0_5	17	18	GPIO0_4	
GPIO0_13	19	20	GPIO0_12	
Serial2 TX/GPIO0_3	21	22	Serial2 RX/GPIO0_2	
GPIO1_17	23	24	Serial1 TX/GPIO0_15	
GPIO3_21	25	26	Serial1 RX/GPIO0_14	
GPIO3_19	27	28	GPIO3_17	
GPIO3_15	29	30	GPIO3_16	
GPIO3_14	31	32	VDD_ADC	
AIN4	33	34	GND_ADC	
AIN6	35	36	AIN5	
AIN2	37	38	AIN3	
AIN0	39	40	AIN1	
GPIO0_20	41	42	GPIO0_7	
GND	43	44	GND	
GND	45	46	GND	

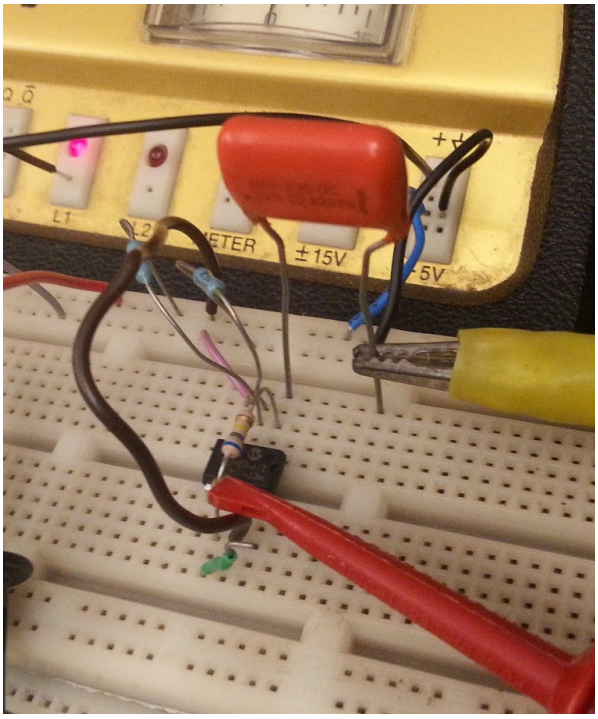
P8

GND	1	2	GND	
GPIO1_6	3	4	GPIO1_7	
GPIO1_2	5	6	GPIO1_3	
GPIO2_2	7	8	GPIO2_3	
GPIO2_5	9	10	GPIO2_4	
GPIO1_13	11	12	GPIO1_12	
PWM2B/GPIO0_23	13	14	GPIO0_26	
GPIO1_15	15	16	GPIO1_14	
GPIO0_27	17	18	GPIO2_1	
PWM2A/GPIO0_22	19	20	GPIO1_31	
GPIO1_30	21	22	GPIO1_5	
GPIO1_4	23	24	GPIO1_1	
GPIO1_0	25	26	GPIO1_29	
GPIO2_22	27	28	GPIO2_24	
GPIO2_23	29	30	GPIO2_25	
GPIO0_10	31	32	GPIO0_11	
GPIO0_9	33	34	GPIO2_17	
GPIO0_8	35	36	GPIO2_16	
Serial5 TX/GPIO2_14	37	38	Serial5 RX/GPIO2_15	
GPIO2_12	39	40	GPIO2_13	
GPIO2_10	41	42	GPIO2_11	
GPIO2_8	43	44	GPIO2_9	
GPIO2_6	45	46	GPIO2_27	

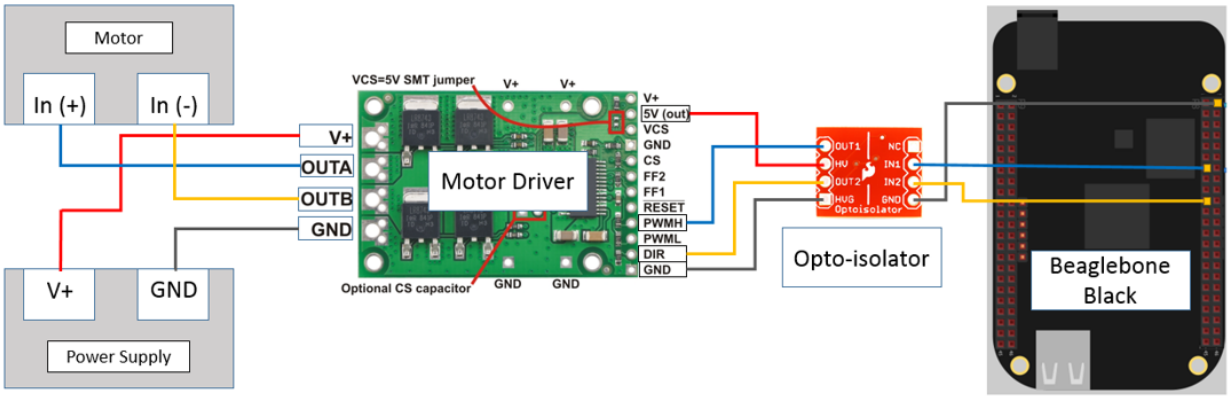
B. Differentiating Circuit Schematic



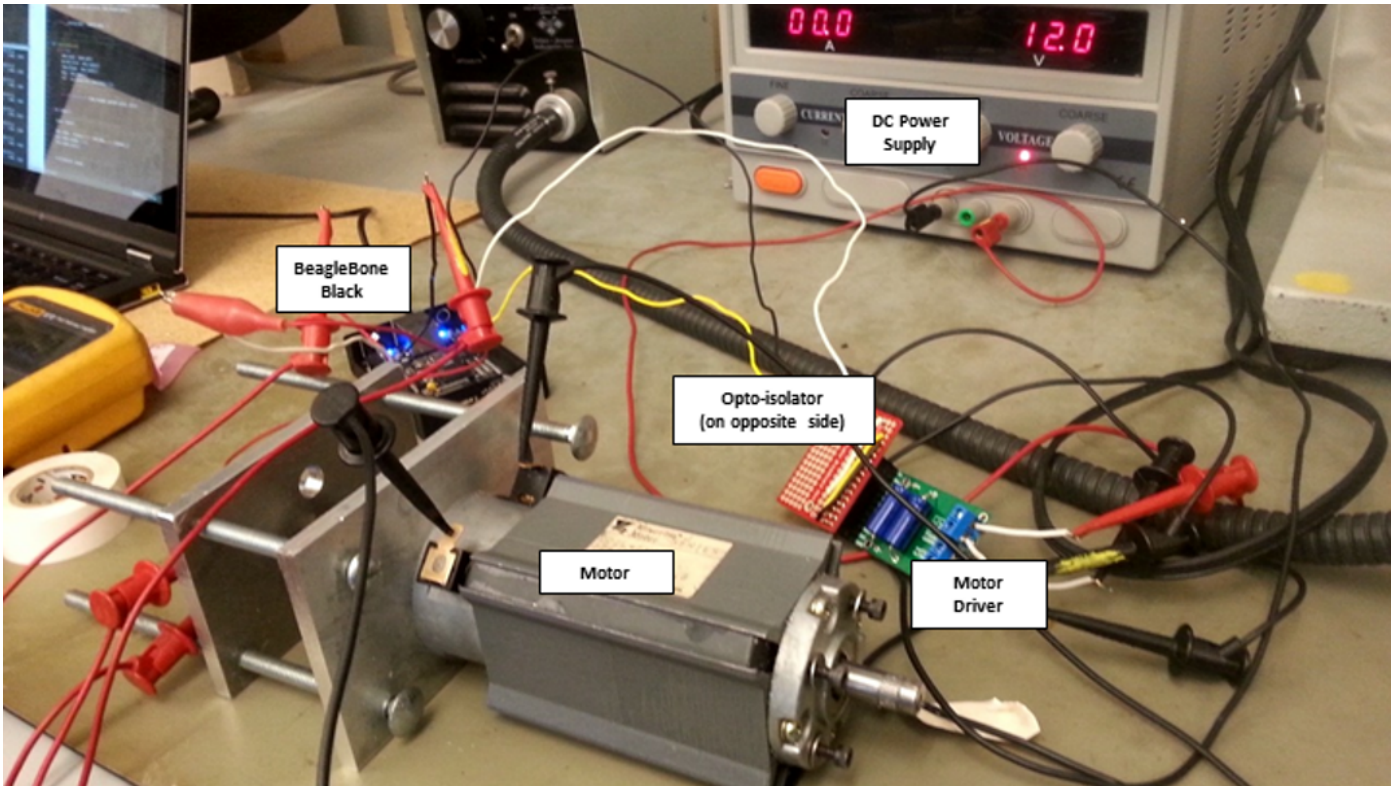
C. Differentiating Circuit



D. Schematic of Steering Circuit



E. Steering Circuit



F. Electronics Data Sheets and Web Page

- BeagleBone Black:

http://www.adafruit.com/datasheets/BBB_SRM.pdf

- Inertia-Link IMU

http://www.microstrain.com/inertial/Inertia-Link

- Bourns 10KΩ Continous Potentiometer

https://www.bourns.com/pdfs/6539.pdf

- Pololu High-Power Motor Driver 36v20 cs

http://www.pololu.com/product/1457

- Sparkfun Opto-Isolator Breakout

https://www.sparkfun.com/datasheets/Components/SMD/ild205t.pdf