WobbleBot

By

Marc Backas

Phillip Lovetere

Mingrui Zhou

Final Report for ECE 445, Senior Design, Fall 2019

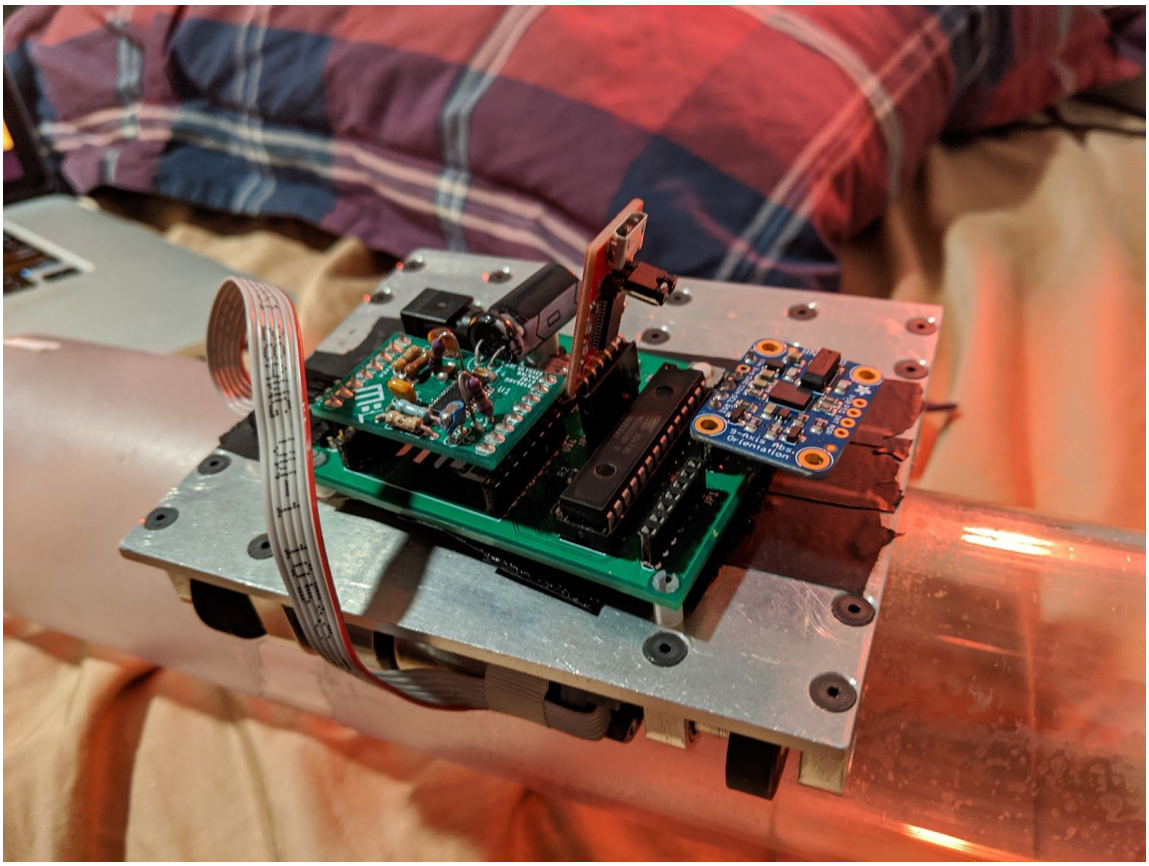
TA: David Null

4 December 2019

Project No. 25

Abstract

For our senior design project, we set out to design, build, and test a robot to balance on top of a cylinder. This is a difficult problem in controls because of the multiple dynamic bodies involved, and because of the fast dynamics of the small, yet heavy robot. By using accurate sensors, precise motor control, and fast processing, we were able to get our robot to balance like we set out to.



Contents

1. Introduction

Problem: It is a challenge to build a robot that is capable of balancing atop a cylinder. Many people studying robotics or controls would be interested in experimenting and improving upon a robotic platform like this, and it does not currently exist.

Solution: We will build such a robot in an affordable way, using hardware and software that enables precise realization of the control algorithm chosen by the researcher. The robot will measure data, process it, and actuate its motor so that it will stay on top of the cylinder. Once this project is complete, it will be an accessible development platform for those studying robotics and controls.

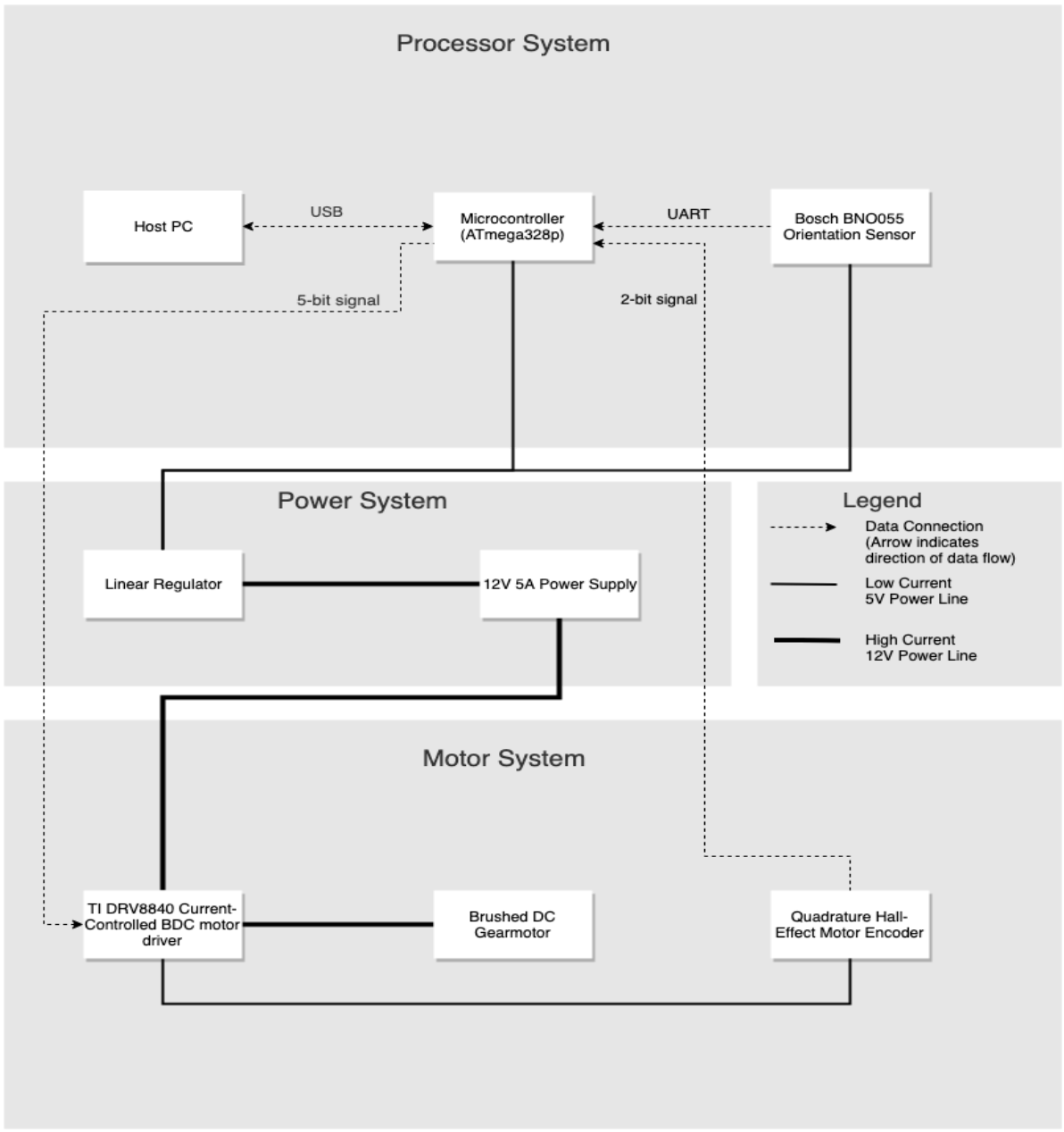
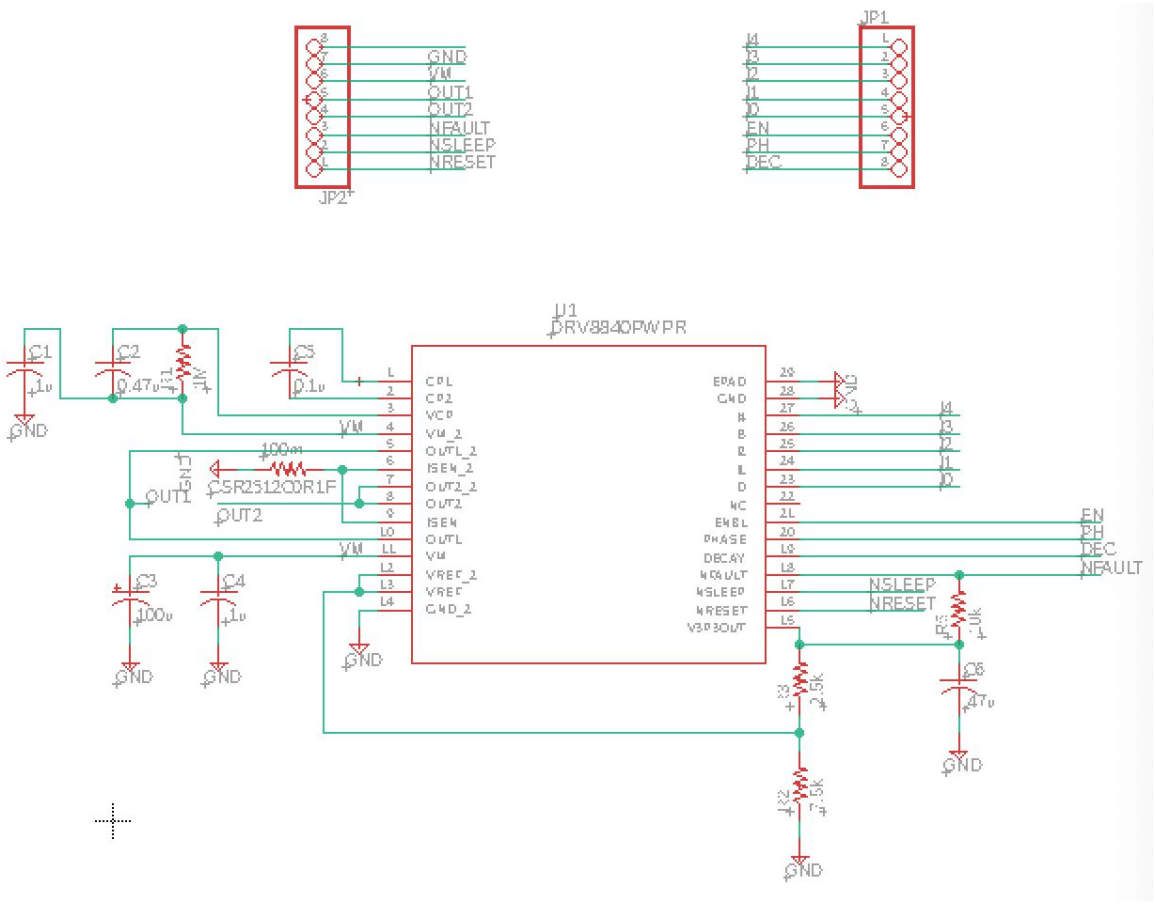
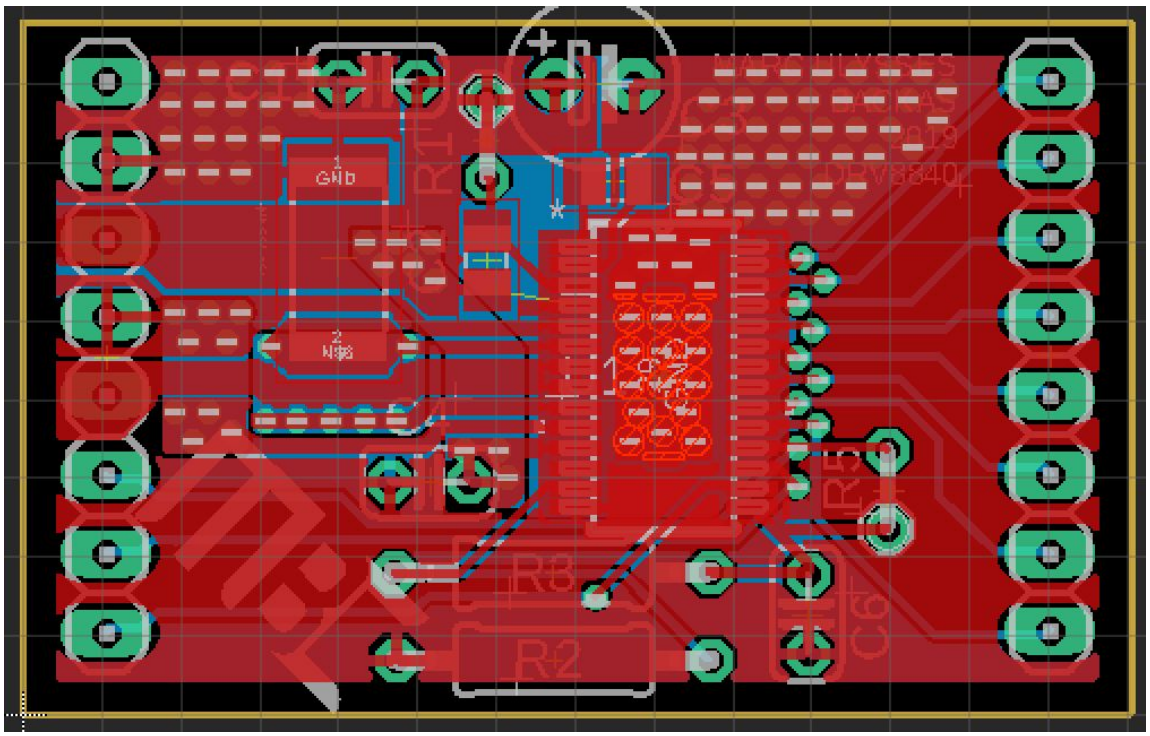


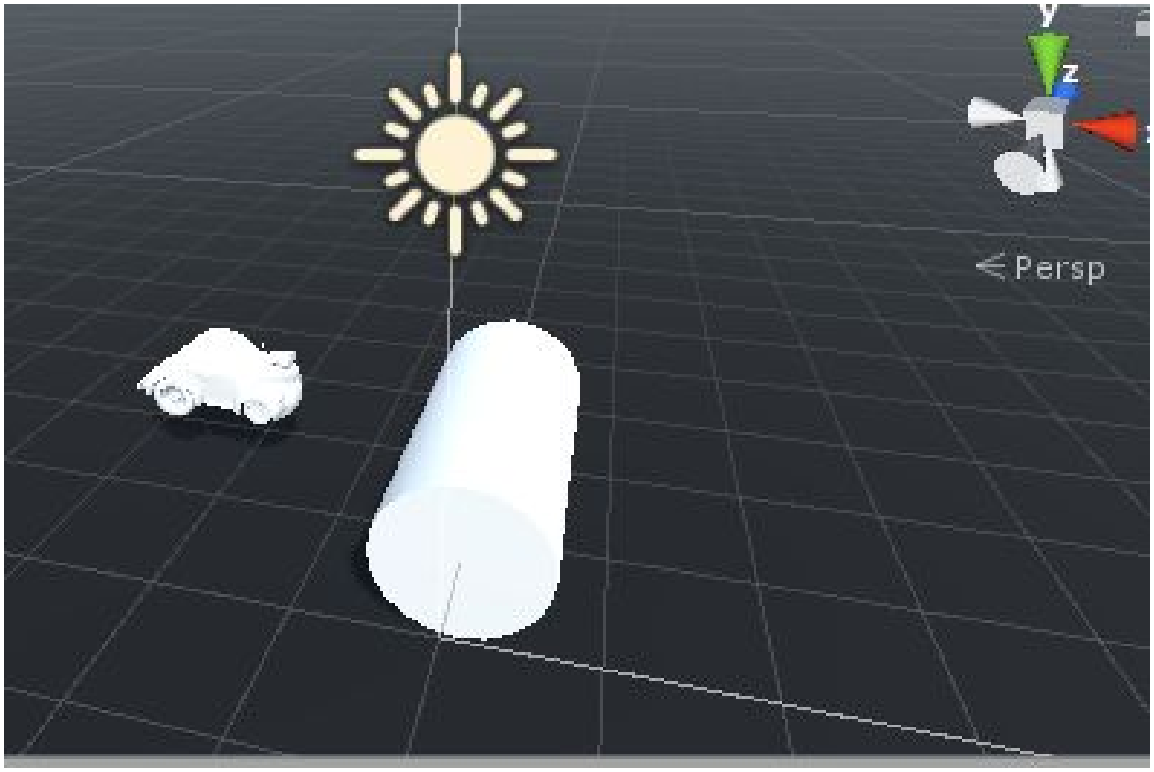
Figure 1: System Block Diagram

2 Design

A major choice in designing a robot that can balance on a cylinder is the motor. One can choose either a Brushed DC motor or a stepper motor. Stepper motors have very good positional and velocity accuracies, but suffer from lack of precise torque control. Brushed DC motors on the other hand, combined with an encoder and a feedback loop can also achieve very good positional and velocity accuracies, with the added benefit of torque control by monitoring and controlling the current through the armature windings.


JP2
GND
VM
OUT1
OUT2
NFAULT
NSLEEP
NRESET
JP1
EN
PH
DEC
U1
DRV8840PWPR
C1
C2
C3
C4
C5
C6
R1
R2
R3
100m
CSR2512CDR1F
OUT1
OUT2
VM
GND
NSLEEP
NRESET
EN
PH
DEC
NFAULT


GND
R1
R2
R3
C1
C2
C3
C4
C5
C6
DRV8840




```
1 using System.Collections;
2 using System.Collections.Generic;
3 using System.IO;
4 using UnityEngine;
5
6 public class Car_Move : MonoBehaviour
7 {
8     public GameObject obj;
9     IEnumerator OnMouseDown()
10    {
```

2.1 Power System

This system will deliver power to all other blocks in our design so that the robot remains functional. The system consists of a 12V 5A power supply and a
5V linear regulator.

2.1.1 Wall Power Supply

Supplies the power needed by all other blocks. Once it is connected to the PCB via a barrel connector, it is directly connected to the motor driver and the linear regulator through copper traces on the PCB.

2.1.2 Linear Regulator (LM1117)

Sources the stable 5V signal needed to power the MCU and orientation sensor.

2.2 Motor System

This system will exert the necessary torque, calculated by the processor system, on the drive wheel to balance the robot.

2.2.1 Motor (ROBOT ZONE 638260)

Converts current output by motor driver into torque output to the drive wheel. The torque applied will drive the robot toward stability atop the cylinder.

2.2.2 Motor Driver (TI DRV8840)

Converts the digital control signal from the MCU into a precise current used to drive the motor.

2.2.3 Motor Encoder (included with motor)

Reports the shaft position of the motor to MCU. Necessary for MCU to compute proper control signal.

2.3 Processor System

2.3.1 Microcontroller (ATmega328P)

The microcontroller (MCU) will run the control algorithm, processing data prom the sensors and generating a control signal for the motor driver.

2.3.2 Orientation Sensor (Adafruit BNO055 absolute orientation sensor)

Measures the tilt angle of the robot in the world reference frame, reporting it to the MCU. The goal of the robot is to drive this angle to zero degrees from vertical

2.4 Unity3D Physic Simulator

Unity's built-in physics engines provide components that handle the physical simulation for our projects. In our unity simulator, we set the size and weight of the cylinder and motor to be the same as our wobblebot, even the size of the wheels. All of them are in 3-d models.

2.4.1 Feature of Unity3D Simulator

- Physics engines, Physics parameters and Physics connection.
- Mock orientation sensor and motor encoder. UnityEngine includes the Gyroscope.input.package which can access the tilt angle and shaft position and assign these features to the controller.
- Functions and equations related to the movement between motor and cylinder. By changing some parameters, it will break the equations and the motor will not stay balance on the cylinder.
- MonoBehaviour includes the SystemObject(cylinder and motor) and Collections.

2.4.2 Goal of the Unity3D Simulator

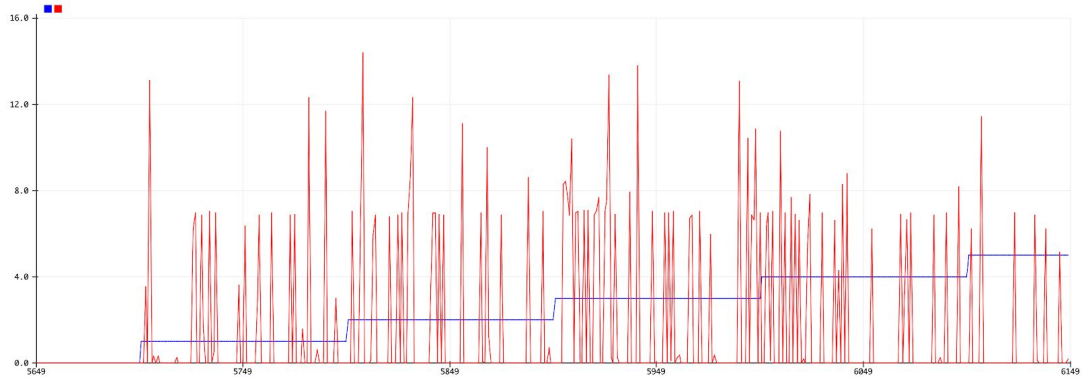
- Test the functioning requirement of the wobblebot with same data set up as the real hardware has. By changing the parameters of some features such as resistance, velocity or mass, it will differ the result of whether the motor can balance on the cylinder.
- Using the simulator, it is easier to set the mass, velocity, resistance and anything on the board and controller. It will be more convenient to test the functionality by using the simulator.

3. Design Verification

3.2 Motor System

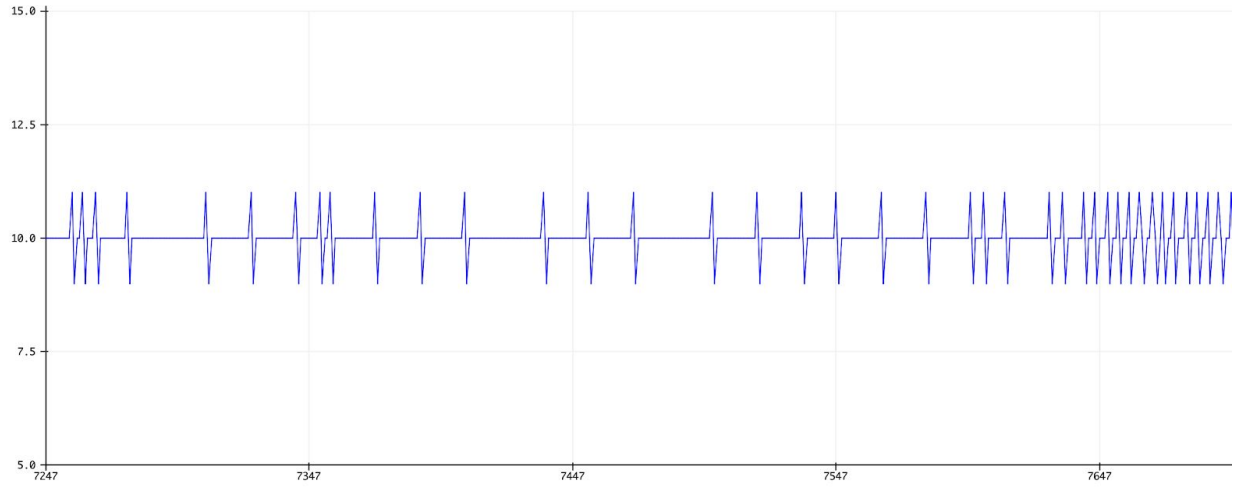
3.1.2 Motor Driver

The goal of the motor driver is to deliver a current proportional to the digital signal it receives. This requirement was tested by applying different signals to the motor driver, and measuring the voltage across a shunt resistor in series with the motor. From the results, it is unclear whether there is a correlation between the desired and actual current, as the voltage changes very rapidly. To improve this, a different motor driver IC could be tested and interfaced with a high precision 14bit DAC to provide a voltage reference for the current limiter.



3.3 Processor System

One requirement of the processor system is to operate at a period of 10 ms or 100 Hz. To verify this, the full code for the project was ran, and the amount of time to complete each loop was recorded.



Clearly, the period stays between 9 ms and 11 ms, satisfying our requirement.

4. Costs

4.1 Parts

Based on a 100k salary per group member and 8-week time window in which the project is performed, plus an estimated upper limit of $200 for every component making up the robot, this project will cost:
(100,000 * 3 group members) / ((12 months/year)/(2 months/project)) = $50,000

- $200 for the robot

= $50,200

The estimated upper-limit of $200 is based on the cost of the $50 motor and $30 orientation sensor(the two most expensive components present), combined with a fairly wide buffer for the unforseen(i.e. something breaking, etc.).

4.2 Labor

Based on a 100k salary per group member and 8-week time window in which the project is performed, plus an estimated upper limit of $200 for every component making up the robot, this project will cost:
(100,000 * 3 group members) / ((12 months/year)/(2 months/project)) = $50,000

- $200 for the robot

= $50,200

5. Conclusion

5.1 Accomplishments

In the end, we were able to create a robot that balances on top of a cylinder, so we were successful.

5.2 Uncertainties

5.3 Ethical considerations

Our robot contains hard, possibly fast moving parts which could cause injury to a person in the event of a collision. This event would go against IEEE #9 “to avoid injuring others.”[2] To avoid this, people will be kept clear of the robot’s trajectory while it is powered on, and a group member will be prepared to disconnect power rapidly if the robot begins to behave dangerously.

Our project has the potential of furthering others' knowledge in the area of control systems and robotics, but only if it is made available for use. This concerns IEEE code of ethics #5, “to improve the understanding by individuals and society of the capabilities...of conventional and emerging technologies, including intelligent systems.”[2] To abide by this, our hardware and software designs will be open source, freely available for duplication by anyone interested in the topic.

Our project team consists of three students who wish to gain valuable technical experience. Although there are many individual areas of work to be done, it is possible for one or more group members to take responsibility for the project’s completion away from the others, denying them of the experience they wish to gain. This would go against the IEEE code of ethics #10, which requires its members to “assist colleagues and co-workers in their professional development.”[2] To prevent this, our group will ensure the work to be done is shared equally, and allocated according to each member’s overall career development goals.

References

[1]U. Control Systems Instructional Laboratory at University of Illinois, "UIUC Segbot - Home", *Coecsl.ece.illinois.edu*, 2019. [Online]. Available: http://coecsl.ece.illinois.edu/segbot/segbot.html. [Accessed: 20- Sep- 2019].

[2] Ieee.org, "IEEE IEEE Code of Ethics", 2016. [Online]. Available: http://www.ieee.org/about/corporate/governance/p7-8.html. [Accessed: 9-19 Sep-2019]

Appendix A Requirement and Verification Table

Table X System Requirements and Verifications

Requirements	Verification	Verification status (Y or N)
1. Must provide 5A of current at 12V continuously without overheating 2. Output voltage must remain within 5% of 12V	1,2. Plug in power supply to wall power and attach resistive load such that current draw is 5A. Ensure voltage remains between 11.5V and 12.5V	
3. Must provide 30mA continuously while staying cooler than 125 degrees Celsius. 4. Output voltage must remain between 4.5 and 5.5V	1,2. Attach power supply voltage at input of regulator, attach resistive load such that current draw is 30mA. Measure voltage and ensure it is between 4.5 and 5.5V. Measure temperature using infrared thermometer, ensure it is below 125 degrees Celsius	
5. Must rotate at a max speed of 153rpm 6. Must provide at least 1.86 in-lbs. of torque when stalled 7. Must provide torque output proportional to the applied current 8. Must not exceed 60 degrees Celsius while operating under load	1. Connect 12V power supply to motor terminals, measure speed from encoder signals with arduino 2. Connect 1 ft. lever arm to motor shaft, measure force at end of lever with digital scale 3. Apply different currents from 0 to 5 volts and measure stalled torque. Plot data and ensure the correlation is linear. 4. Apply 12V to stalled motor, measure temperature using infrared thermometer	
9. Must output at least 5A at 12V continuously while staying below 160 degrees Celsius 10. Must output current proportional to control signal	5. Connect 12V power supply to driver, set current limit to 5A, load with motor, and measure current and voltage with DMM, and the temperature with an infrared thermometer.	

(coefficient TBD) through motor load	6. Specify all 32 current levels using Arduino, and measure output current through motor using DMM. Plot the data and find the line of best fit. Ensure the maximum error from the line is less than 5% of expected	
1. Must report shaft position within 5% accuracy at all times 2. Must report shaft position at least 100 times per second	1. Use Arduino to count pulses from encoder. Allow motor shaft to rotate 10 full times. Ensure count number is 1973. 2. Use Arduino to poll the encoder count number every 10ms. Set a GPIO pin high while the processor is idling, and low while it is busy, monitor the pin using an oscilloscope and ensure the code runs in less than 10ms each cycle.	
1. Must process sensor data and output control signal at 100Hz 2. Must communicate with orientation sensor over UART at 100Hz	1. Use MCU to set a GPIO pin high while the processor is idling, and low while it is executing control signal calculation. Monitor the pin using an oscilloscope and ensure the duty cycle of the pin output is always greater than 0. 2. Connect MCU to orientation sensor over UART and poll data from it every 10ms, ensuring the code runs smoothly using same method.	
1. Must report tilt angle with less than 3% accuracy at all times 2. Must report tilt angle at least 100 times per second	1. Build a test platform with attached inclinometer to measure actual tilt. Record tilt angle from sensor. Record data pairs into a table. Ensure the maximum error is less than 3% for all angles. 2. Program MCU to poll the sensor at 100Hz, ensure new data is available from the sensor at each poling time	