5-1-2019

Pace It: Final Design Report

Dupri Grimes
Trinity University

Brian Guenther
Trinity University

Molly McCullough
Trinity University

William McElvogue
Trinity University

Luc Scranton
Trinity University

See next page for additional authors

Authors
Dupri Grimes, Brian Guenther, Molly McCullough, William McElvogue, Luc Scranton, and Samuel Studebaker

Pace It
Final Design Report

Dupri Grimes, Brian Guenther, Molly McCullough,
William McElvogue, Luc Scranton, and Samuel Studebaker

Team Advisor: Dr. Mehran Aminian
ENGR 4382

May 1, 2019

Executive Summary

The most efficient way to run a distance event is to maintain a consistent pace over the entire race, but this is difficult for runners to accomplish unaided. The Trinity University cross country team is seeking a visual training aid that can simultaneously help multiple runners running at the same pace build muscle memory in order to maintain a constant, specified pace over a specified distance, while also allowing a bystander (coach) to see how closely the runners are maintaining that pace.

The objectives of the project are to design, build, and test a pacing system for the Trinity University Cross Country and Track teams that can maintain a constant pace between 7.5 and 12 mph for at least 1600m within the $1200 budget. This report covers the overall design and success of the final prototype.

The overall design is a pacing robot that navigates the track by following one of the lane lines. The design was split into several main subsystems: motor/chassis, line-following, user interface, main controller, and power systems. The chassis is a prefabricated RC car chassis that includes a motor and gearbox that is used in our design. Motor control uses a tachometer built from a Hall effect sensor and magnet to determine motor speed in RPM. For line-following, an array of six IR sensors is used to detect the line. The user interface is a smartphone application that communicates wirelessly with the robot to allow the user to input a desired pace and distance, as well as start and stop the robot. All of this is controlled and communicated with by an FPGA.

The majority of subsystems performed as dictated in the determined criteria, however there were complications with the implementation of line following at high speeds. The robot is able to identify the line at any pace up to 12 mph, however it was successful in following the line for paces only up to 5 mph. This was because the servo motor on the remote control car that we purchased behaved in a non-linear manner, which made it difficult to control through a standard PD loop, even after we had managed to eliminate the noise from external light and rapid changes in distance from the ground. As a possible solution to this issue, we suggest the implementation of steering control motor, whether that be a higher quality metal geared servo or stepper motor. Additionally this would likely require a large front end chassis redesign in order to incorporate it into the system. Although the system does not meet the requirements for pace, we consider this a working prototype as it can follow the line at slower speeds.

1 Introduction

The most efficient way to run a distance event is to maintain a consistent pace over the entire race, although this is difficult for runners to accomplish unaided.
The Trinity University cross country team is seeking a visual training aid that can help multiple people running at the same time build muscle memory in order to maintain a constant, specified pace over a specified distance, while also allowing a bystander (coach) to see how closely the runners are maintaining that pace.

The objectives of the project are to design, build and test a pacing system for use at practice by the Trinity University Cross Country and Track teams within a $1200 budget. The system will have to maintain a constant pace, between 7.5 and 12 mph (8:00 and 5:00 minutes per mile respectively), for a distance of at least 1600 meters. The system will also feature a user interface that allows the user to select the desired speed and distance.

Additional criteria for the design include the ability to maintain a constant speed for at least 1600 meters between battery charges and be smaller than a single lane width (1.22m) in order to fit within one lane on the track. The system must also be able to safely navigate a 400m track without major variations in lane position or hitting bystanders or runners.

2 Design Overview

The design for the pacing system is a line-following robotic car that navigates the track by following the lane lines. The design adheres to the IEEE Standard Ontologies for Robots. The design for the pacing robot is considered in five major subsystem designs: motor/chassis, line-following, user interface (UI), main controller, and power system. A block diagram of the overall design is seen in Fig. 1. This diagram gives a top level look at all of the main components and how they interact.

The main body of the pacing robot contains the battery, drive and servo motors, tachometer, main controller, and wireless communication module. The array of infrared (IR) LEDs for line-following is housed in a case that extends from the front of the robot to ensure constant distance between the track and the sensors for better line-following capabilities. The case also protects the sensors from water and provides shade to limit the sunlight from affecting the sensors. The ultrasonic sensor was an additional safety measure initially considered in the design, but was not implemented in the final prototype. The main safety system is the user's ability to start/stop the pacing robot through the wireless UI and an automatic shutdown if the robot loses connection with the UI. Detailed descriptions of hardware used in each system are included in Appendix B.

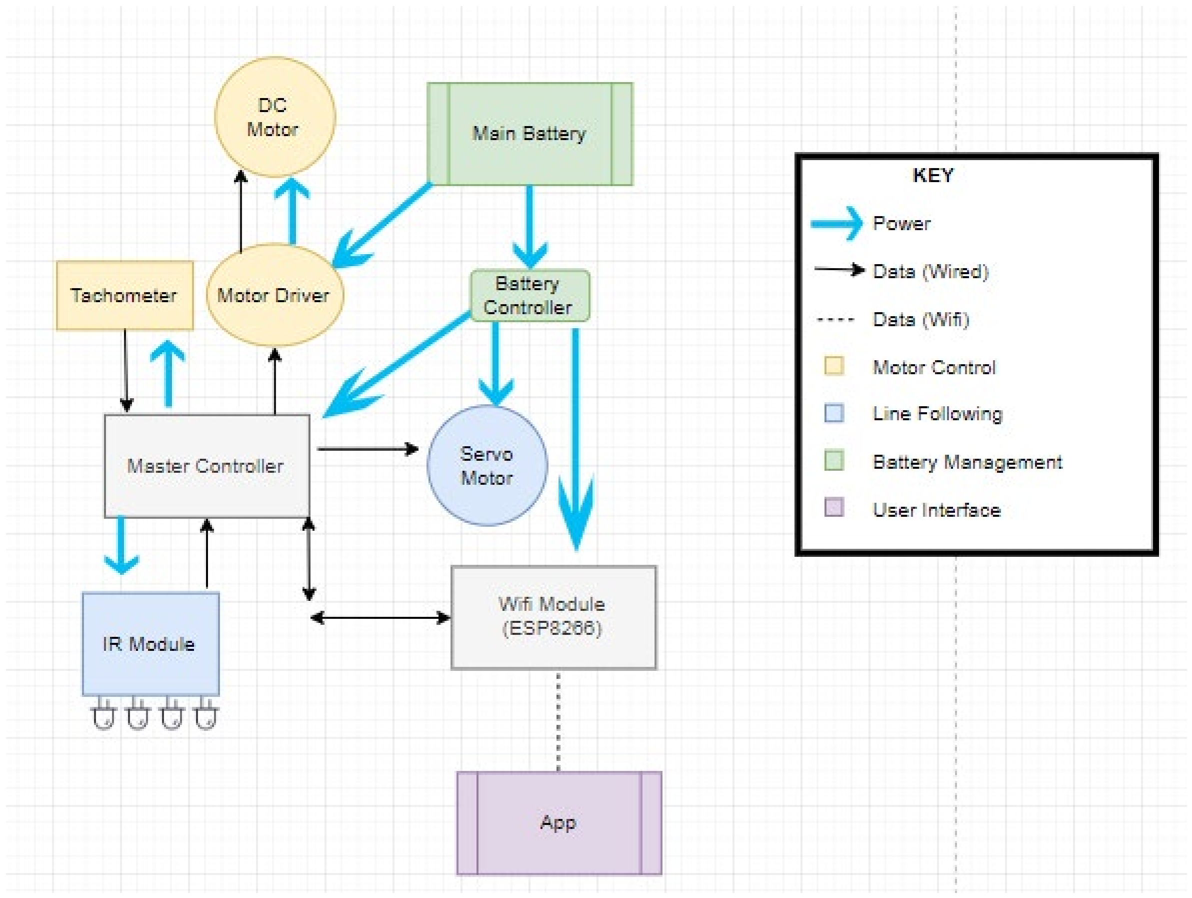


Figure 1. Final Overall Design Block Diagram

2.1 Motor/Chassis subsystem (motor driver, motors, tachometer)

The final prototype is built from the prefabricated chassis of the commercially available RC car model Helion Conquest 10MT XB. The chassis is lightweight, weatherproof, and has room for all essential components: motors, battery, controller, and wireless module for communication with the user interface. Line-following hardware was moved off the main chassis for better line detection, rather than a lack of space on the chassis.

The drive and servo motors used in the design were those included with the RC car purchase. The drive motor is rated for speeds of up to 20 mph, exceeding the maximum requirement of 12 mph per the design objectives. The gear box and electronic speed control (ESC) system provided on the car were used to interface with the motor. In traditional operation of the RC car, the ESC and servo motor communicate with the user through the handheld remote. For the pacing robot, the ESC and servo motor communicate directly with the main controller.

Drive motor speed was monitored through the use of a group-assembled tachometer consisting of a magnet and a US5881 Unipolar Hall Switch sensor (Hall Effect sensor). A magnet was attached to an exposed gear outside the motor with gorilla glue.

Each time the gear completes one revolution, the magnet passes by the Hall effect sensor once. When the sensor detects the magnet, it sends a digital low to the controller. The frequency of the signal from the Hall effect sensor is directly related to the speed of the gear, from which the speed of the car is determined through processing modules in the controller.

2.2 Main Controller

Originally, both the Arduino and Raspberry Pi were considered for the main controller. However, the main controller of the final prototype is a Cyclone IV FPGA on a DE0 Nano board. An ESP8266 was used to interface with the user's smart device via the Blynk application architecture. An FPGA was used as the controller due to its ability to run many different tasks in parallel. For example, it can process readings from all of the IR sensors while also communicating with the motor, Hall effect sensor, and servo all simultaneously.The main controller interfaces with the line-following IR array, the servo motor, through the ESC to the drive motor, and to the wireless communication module, all with GPIO pins.

The IR sensor array and servo motor are used for the line-following system. The main controller monitors which IR sensors in the array are detecting the lane line, and then uses that information to determine which direction to turn the wheels of the robot. For motor control, the main controller receives a setpoint from the user through the wireless communication module and compares it to the calculated RPM based off the Hall effect sensor signal . The controller communicates with the ESC to start the motor and monitors the current speed with feedback from the tachometer using PID control to keep the speed of the motor within 10% of the setpoint. The ESC controls the motor speed based on a specialized pulse width modulation signal typically used for servos. This signal has a 60 Hz frequency and contains a pulse ranging from 1.0 ms to 2.0 ms in length. A 1.5 ms pulse is a neutral signal and anything longer will cause the car to go forward; the speed of the car increases as the signal gets longer. A pulse less than 1.5 ms causes the car to reverse, which is not used in this project. Thus, the controller outputs a 1.5 to 2.0 ms pulse to the ESC. The main controller also cuts the signal to the motor when the user sends a stop signal. For more information on the controller and how it was programmed see Appendix B.

2.3 Line-following

The pacing robot navigates the track following a lane line with line-following control. Line-following is implemented using infrared sensors and receivers. An array of six IR sensors and emitters, along with six standalone analog to digital converters is housed in a case that extends from the front of the chassis. The housing includes two smaller wheels in front of the array, which ensure a constant distance between the sensors and the track along the entire array.

The IR sensor/receiver packaged used is the QRE1113 [B1.1]. This is a small module that contains an IR LED and IR phototransistor in one package, with four pins. The analog to digital converter used is the MCP3001 [B1.2]. These chips each convert one IR sensor reading from an analog voltage to a digital value that is transmitted to the controller independently through SPI. We chose to use six individual analog to digital converters so that we could take advantage of the parallel processing nature of controller, and we wouldn't be bottlenecked by only one line of SPI.

The IR LED emits infrared waves which are then reflected by the track back to the phototransistor. Depending on the amount of waves that are reflected back, the transistor's resistance varies, which changes a signal voltage that is transferred from an analog value to a digital value and sent to the controller . Different colored surfaces reflect different amounts of infrared light: when the sensor is above the white lines of the track it outputs different signals than when it is above the red lanes of the track. The voltage signals are transmitted to the controller, which uses that information to determine the motion of the servo motor to steer the car along the line. For a more in depth look at the QRE113 see Appendix B1.1.

The controller receives the readings from each of the IR sensors and has to decide whether the sensor is on the white lane or the red track. Due to the inconsistency of the IR readings, having one reference threshold value would not work. Each sensor received different readings from the others, and the sensors also sometimes receive different readings based on the location of the track. Thus, the controller looks at each sensor individually and averages 20 samples before evaluating the reading.. It starts off by setting a calibration value to compare new readings to. If a new reading is 5% lower than the calibration value, it sets that sensor as "white" and then replaces the calibration value with a new reading. If a new reading is 5% greater than the calibration value, it sets that sensor as "red" and then replaces the calibration value with a new reading. The value of 5% was determined after measuring the output voltage of the IR sensors on the track.

The controller then looks at how many sensors are set as white. The IR array was designed so that no more than 2 sensors would be over the track line at one time. Thus, the controller does not consider any readings when more than 2 sensors are considered white. This could happen due to a bad reading or due to other markings nearby on the track. Furthermore, the controller does not consider any readings when non-adjacent sensors are also considered white.

If one sensor or two adjacent sensors are set as white, the controller then assigns an error value based on which sensor(s) are seeing the line.

This error value is then fed into a PD loop to control the onboard servo that requires the same servo signal described in Section 2.2; a 1.5 ms pulse puts the servo facing straight, a 1.0 ms pulse turns it all the way to the left, and a 2.0 ms pulse turns it all the way to the right. It was discovered during testing that the servo was biased to the left and much smaller changes in the signal pulse were needed to turn left. In other words, a 1.4 ms pulse turned the servo further left than a 1.6 ms pulse turned it to the right. Furthermore, this meant that small changes in signal would result in a turn to the left and not to the right. Due to this effect, the PD loop places a very higher emphasis on the derivative control than on the proportional control because the proportional control had difficulty dealing with the servo bias. The derivative control is based on the change in error and placing a higher emphasis on it allows the car to react quickly to changes in error. Having strong derivative control means the system is vulnerable to noise, but this was mitigated by limiting the pulse width. Limiting the pulse width keeps the servo from steering to sharply to the side. The pulse width was limited asymmetrically around 1.5 ms because of the servo bias previously mentioned.

2.4 User Interface

The pacing robot uses a wireless user interface to allow the user to input a desired speed and distance for the pace, as well as start and stop the robot remotely. The user interface was generated using the Blynk application, which is available on both iOS and Android. An image of the interface and operating instructions can be found in Appendix A5 and A6. The application allows the user to set the speed and distance for the robot to travel as well as start and stop the robot. Additionally, it allows the user to set speed before starting operation and adjust the speed while the robot is already moving. These signals are received by the on-board ESP8266 wireless communication module and transmitted through I^2C to the main controller to be used as the set point for the motor control.

2.5 Power System

The power system consists of the battery and a separate circuit board used for distributing power to all of the other system. The main battery used for powering the device is a Venom 7.2V 6-cell 5000mAh High Power NiMH battery. This large capacity battery can provide multiple runs without recharging. A Helion 0.5A Battery Charger is used to charge the battery. Strategic cutouts were made to allow the battery to be easily charged without having to remove it from the device. There is also a spare battery, a Helion 7.2V 6-cell 2000mAh NiMH battery, that can be used as a replacement. The spare can be used if the main battery were to malfunction. It is recommended that the batteries are not switched out unless absolutely necessary, as battery replacement is a very intrusive process.

The battery directly powers the motors and the stock RC controller at 7.2V. The DE0 controller runs on 5V power and the ESP controller runs on 3.3V. The 7.2V battery could not be connected directly to these devices so a voltage regulator was needed to interface to these devices. A LM2596 Buck Voltage Regulator was used to step down the battery voltage to 5V. This 5V is used to power the IR LEDs, ESP8266 and the controller, which then powers the IR receivers and the Hall effect sensor at 3.3V. This specific voltage regulator was chosen because it has a 92% conversion efficiency. It also came packaged on a PCB, making it easy to connect to the power circuit board. The current requirements for the devices being powered by the LM2596 are shown in Table 1. The voltage regulator can source a maximum of 3A, so the maximum current requirement of 0.96A is acceptable. The main battery is rechargeable, and will go from drained to fully charged in 3 hours.

Table 1. Maximum current required for the devices connected to the LM2596 Buck Voltage Regulator

Device	# of Devices	Max Current Requirement (mA)
ESP	1	160
DE0	1	500
IR LEDs	6	300
Total	9	960

2.6 Safety Systems

The safety of the users and runners interacting with the pacing robot were an important consideration in the design process. The wireless user interface allows the user to send a stop signal to the pacing robot, to allow the robot to be stopped at any point during operation. Additionally, the ESP can send an automatic shut off signal to the controller when it loses connection with the app, so that the robot is never operating while the user is unable to control its behavior. We had previously suggested that the use of an ultrasonic sensor would create an extra level of safety within the already available application interface. We have determined this to be outside of the scope of the project and will not be implementing this system. This will require the operator of the robot to be present at all times while it is in use.

3 Prototype Tests and Results

Initial tests on the motor control and line-following systems were conducted prior to the Critical Design Review last semester.

The tests performed served as proof-of-concept work for both systems, to confirm the accuracy of the tachometer and the ability of the IR sensors to perceive a difference between different colored lines on the track. Tests of the motor control system were done using the tachometer on a basic DC motor with a spinning wheel. The motor control program was able to make the motor meet and maintain the hardcoded setpoint of 1000 RPM with little overshoot. Testing of the IR sensors showed significant differences in the outputs voltages when detecting a black surface as opposed to a white surface as seen in Appendix B1.1.

Further testing was required for all subsystems, as well as the final prototype. The testing of the pacing robot evaluated the user interface, motor/chassis, and line-following individually as well at various levels of integration as shown by Fig. 2. The number of subsystems involved in each level of testing is indicated by color and the arrows indicate the dependence of tests of higher systems on successful tests of the integrated subsystems.

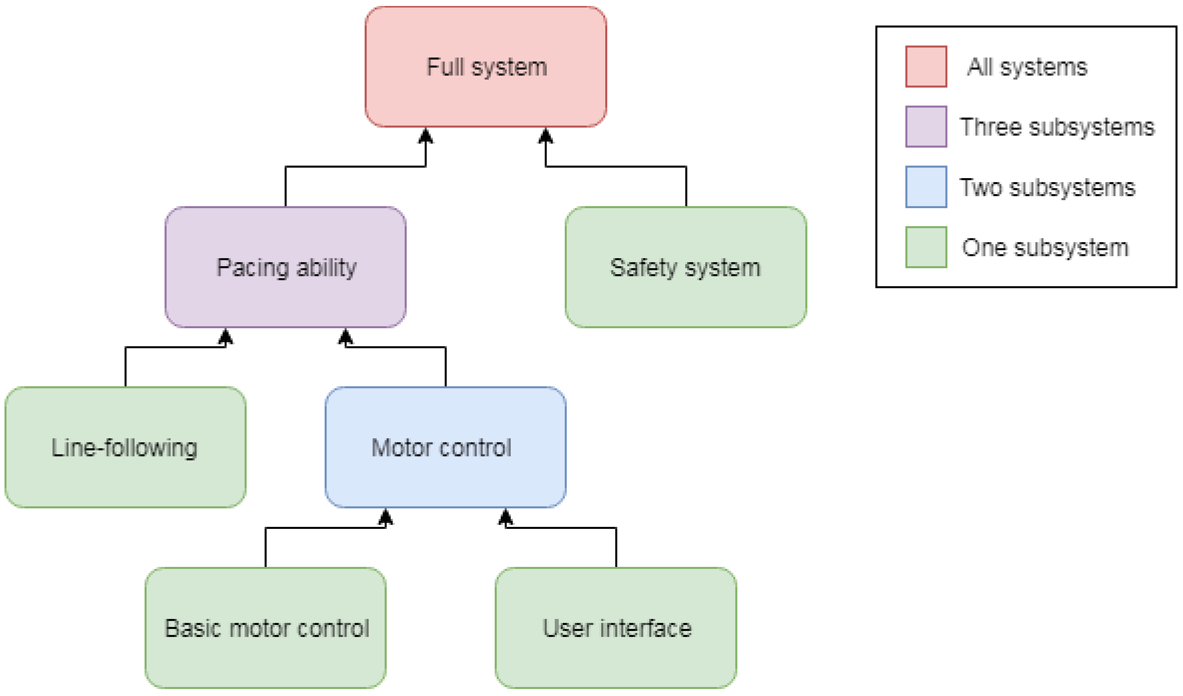


Figure 2. Subsystem testing structure.

3.1 Basic motor control testing

<u>*3.1.1 Test overview*</u>

<u>Basic Motor Control Test</u>**:** This test will prove control of the motor speed with the tachometer and hardcoded setpoint.

3.1.2 Objectives

The objectives of this test are to prove motor control using the motor and ESC of the RC car with the motor control program and tachometer setup proved in testing done last semester. This test builds on previous testing by implementing the tachometer and motor control on the actual motor in the RC car rather than an external motor used for the proof-of-concept.

3.1.3 Feature(s) evaluated

The feature evaluated in this test is the ability to change the speed of the motor using the ESC and a hardcoded setpoint with the tachometer.

3.1.4 Test scope

Motor control will be tested with hardcoded setpoints 10 RPM increments, based on the required RPM values to meet the desired range of paces (5:00-8:00 minutes per mile).

3.1.5 Test plan

This test requires the car chassis/motor, as well as the tachometer and the DE0 for both control and RPM feedback. The test setup will require the DE0 to be connected to a computer in order to manually change the setpoint, as well as a way to run the motor while the chassis remains stationary.

In order to test the motor/chassis system, the correct RPM values based on the gear to wheel ratio and the given range of paces need to be calculated. The RPM values displayed by the DE0 will also need to be recorded to determine the average pace and the amount of deviation from that pace.

The data collected from this test will include the hardcoded setpoint, the average RPM and largest deviation from the average during each test as well as the error between the average RPM and the setpoint.

3.1.6 Acceptance criteria

The motor/chassis system will be accepted if the motor control and tachometer work for the full range of setpoints with an error of less than 15% from the setpoint and variations from the average RPM value of less than 10%.

3.1.7 Test Results and Evaluation

The ratio of the number of turns of the gear onto which the magnet is attached to the wheels was determined to be approximately 2.5:1, shown in Fig. B3.3 of Appendix B. The calculation of the speed of the car was determined through this information.

Because the motor control algorithm had been tested effectively during the development of the initial prototype, the testing of the motor control and tachometer were folded into the Motor Control Test, discussed in Section 3.3.

3.2 User interface

3.2.1 Test overview

User Interface Test: This will test if the user interface can continuously alter setpoint values on the ESP8266 microcontroller. These values are then to be sent via I^2C to the motor control system.

3.2.2 Objectives

The objectives of the user interface test are to determine the ability of the wireless user interface (the Blynk smartphone application) to communicate with an ESP8266 microcontroller which then sends the RPM setpoint to the DE0 for use in motor control. The application should be designed in such a way as to show the user the current speed set point and include functionality for start and stop switches.

3.2.3 Feature(s) evaluated

The feature evaluated in this test is the ability of the user to communicate with the robot using the Blynk smartphone application.

3.2.4 Test scope

Preliminary testing were performed using the ESP8266 in conjunction with a DE0 microcontroller. Paces between 5:00 and 10:00 minutes per mile are set within the Blynk application, that are then sent to the DE0 to be displayed on the four digit seven segment display. Additional testing includes verification of the communication between the ESP8266 and the Blynk application at multiple points around the track to confirm the connection of the device.

3.2.5 Test plan

This test requires the Blynk application on a smartphone, and the ESP8266 in conjunction with other robotic control systems.

The test also requires the calculation of desired RPM based on the gear-wheel ratio and the desired pace to compare with the values determined by the ESP. The test also considered the assumption that the motor control is properly calibrated in adjacent control architecture.

RPM values calculated by the ESP will be sent to the motor control system for further processing. These values are sent using I^2C communication for use in motor control. On a stop command, the ESP8266 will send an RPM value of 0. This will cause the motor control system to stop the motor.

3.2.6 Acceptance criteria

The communication between the ESP and Blynk application must work at the maximum distance expected between the robot and user, meaning that the motor can be stopped or started at any point along the track. Additionally, the accuracy of the I^2C communication should be higher than 90%, meaning that the value sent to the motor control system is nearly always the correct RPM setpoint.

3.2.7 Test Results and Evaluation

In testing with the DE0 equipped with the four digit seven segment display, the system had 100% accuracy sending values between 5:00 and 10:00 mile paces. Once the communication system was integrated, there is an identified fault where the communication between the ESP8266 and DE0 can be halted after long periods of time and many communicated set points. In general, this occurred once out of every twenty set point changes. This is still within our acceptance criteria of 90% accuracy. Additionally, tests were performed to determine if the robot would be continuously connected at any point on the track. These tests proved successful having only one instance of noticeable user disconnect. The user interface can therefore be considered a successful sub-system. As a recommendation for future projects, additional safeguards against loss of I^2C communication could be implemented. This would come in the form of an additional start/stop bit communication or additional pin output for start and stop values. This would increase system stability in case of I^2C disconnect.

3.3 Motor control

3.3.1 Test overview

Motor Control Test: This will test the integration of the basic motor control system and wireless user interface (UI).

3.3.2 Objectives

The objectives of the motor control test are to confirm that the motor control system can be controlled with a setpoint determined by the user rather than a hardcoded setpoint and that the start/stop signals from the UI result in the expected motor behavior.

3.3.3 Feature(s) evaluated

The feature evaluated in this test is the capability of the motor to correctly read the data sent from the user and immediately change the speed of the car, start/stop the car, and update battery life.

3.3.4 Test scope

The test will run motor control for all paces available in the app, as well as the start/stop options.

3.3.5 Test plan

The setup for this test is the same as the setup for the basic motor control test but uses the wireless UI rather than the computer to communicate the setpoint. The RPM readings will still be displayed on the DE0.

The expected RPM values, as used in previous tests to determine accuracy, are needed for this test to determine how well the system works with a user setpoint, rather than a hardcoded setpoint.

Like the basic motor control test, results for this test will be collected in the form of the average RPM values as well as the largest variations from the average over tests performed for at least 1 minute. The time it takes for the system to respond to a change in the user inputs will also be recorded to determine responsiveness of the system.

3.3.6 Acceptance criteria

The RPM measurements on the DE0 must be accurate to within 15% of the expected value with variations of less than 10% during the entire test. The system must work for all setpoints and the motor must respond to start/stop signals in under 5 seconds.

3.3.7 Test Results and Evaluation

The test results have shown that the motor control was effective to about 10% error from the RPM setpoint. This meets the requirements laid out by the acceptance criteria. Additionally, the system was able to respond to start and stop signals from the user interface within 5 seconds, also meeting the acceptance criteria.

3.4 Line-following

3.4.1 Test overview

Line-following Test: This test will prove the ability of the robot to follow a lane line around the track for 400m (1 lap).

3.4.2 Objectives

The objective of the line-following test are to prove the operation of the line-following, showing that the robot can follow a lane line on the track in various light conditions at various speeds within the desired range for at least 400m.

3.4.3 Feature(s) evaluated

The feature evaluated in this test is the ability of the car to follow the lane line all the way around the track.

3.4.4 Test scope

The test will be performed over 400m at speeds close to the desired speeds as determined by the required paces.

3.4.5 Test plan

To perform the line-following test, the robot will be placed on a lane line on the track and allowed to run for 400m, using the remote control that came with the car to control speed rather than the motor control system.

The test requires the isolation of the motor control and line-following systems by using the remote control to control robot speed to determine that the line-following system can operate correctly.

The data collected from the test will include the variation in lane position of the robot on each 100 meter section of a 400 meter lap around the track, based on how much the robot oscillates over the lane line and the largest variation from the lane line during the test.

3.4.6 Acceptance criteria

The line-following system will be accepted if the robot can navigate 400 meters of the track without variations of more than the car width outside of the lane line, and that this result can be replicated in different light or weather conditions based on the functionality of the shield.

3.4.7 Test Results and Evaluation

As of 4/30/2019, the robot was able to navigate one hundred meters at a time at a 13 minute mile pace. However, these results are rather preliminary and were not found using the finalized control algorithms for line following. We do not have any more thorough recorded testing of the line following in this report because as we were doing the final tweaking of the line following control our controller broke. Our new controller does not arrive until 5/1/2019. We hope that when a new controller arrives the robot will be able to perform significantly better than this, once the perfected control algorithms are set up.

3.5 Safety system

Implementing the previously suggested ultrasonic sensor was determined to be outside of the scope of the design criteria. The start and stop functionality of the Blynk application will be the main safety system, however this will require a supervisor to maintain visual contact with the robot.

3.6 Full system testing

3.6.1 Test overview

Full System Test: The final test performed is to test the operation of the entire system.

3.6.2 Objectives

The objectives of the test are to determine if all subsystems of the robot work together when implemented to meet the criteria and constraints.

3.6.3 Feature(s) evaluated

The features evaluated in this test will be the overall capability of the robot to perform meet all criteria with all subsystems working.

3.6.4 Test scope

This test will be done over the full range of desired pace inputs, based on the previous motor control tests, as well as the start/stop functions over the required 1600 meter minimum distance.

3.6.5 Test plan

The test will be performed on the track and require the robot with all subsystems connected as well as a runner. The data collected will be the amount of time it takes the runner to complete one mile while following behind the robot with the pace set at 5:00 minute/mile and 8:00 minute/mile.

3.6.6 Acceptance criteria

The test will be considered a success if the runner is able to meet the desired paces.

3.6.7 Test Results and Evaluation

This test has not yet been performed due to insufficient success in design and implementation of some of the subsystems. Our line following subsystem was not able to sufficiently pace a runner at a 5:00 mile pace and therefore this test cannot be completed with the current prototype. See line following test section 3.4.7 for recommendations on future work. When our new controller arrives on 5/1/2019 we will be attempting this final test plan and recording the results to put in our final presentation.

4 Conclusions

The functionality of the completely integrated pacing robot can be considered inconsistent. Individually, certain subsystems performed exceptionally well. For example, the motor control and user interface met all of the requirements set out by the scope of the design project. The motor control was accurate within 10% of the RPM setpoint and was able to start/stop within 5 seconds of command from the user interface. The user interface had 100% accuracy sending values between 5:00 and 10:00 mile paces. Additionally, the integrated power system also performed extremely well. We were able to sufficiently power subsystem components as well as the electric motors for significantly longer than the minimum race time. However subsystem integration caused some issues. There were rare instances of I^2C connection loss when using the line following software on the DE0, as well as insufficiencies in the area of line-following speed.

We were successful in creating a wireless user interface that allows a user to set a desired pace and a robot that can maintain the desired pace within 10% of the RPM setpoint for a distance of 1600 meters. To conserve space, the robot is designed to fit inside one lane. The total cost of this robot is $621.24 which is well under the $1200 allowed budget. However, as of 4/30/2019, we have not had the chance to test the final control algorithm with the line-following. We were also not successful in implementing line-following at all desired paces. The line-following was only accurate at slower paces for shorter distance which did not fall into the range of goal paces. With this being said, the IR array was able to correctly communicate the location of the line to the controller which was able to process this value. However, due to the non-linearity of the servo motor increasing the complexity of the line control algorithm, we experienced many obstacles in getting the robot to actually stay on the line at fast paces. However, when our new controller arrives we plan on hopefully implementing line following at faster paces for longer distances.

Our group suggests continual implementation of the control system for line-following as well as replacing the provided servo with a more accurate servo or stepper motor. These suggestions are not feasible for us to accomplish fully in the next two weeks, but could be implemented by a future senior design group. We do believe with enough time spent nailing down the control algorithm for the line follower, the system can efficiently follow lines at higher paces than we were able to reach in testing. The main issue with our lack of line following success can be attributed to the non-linearity of the servo motor used for steering. The IR array was able to regularly find and identify the line, but we found it was difficult to control the servo using a traditional PID loop because it tended to turn stronger to the left than to the right, which had large effects on the position of the car at higher speeds.

Appendix A - Setup and Operating Instructions

A1. Operating the Device

1. Download the Blynk app from Google Play or the Apple Store to smart device. User will be required to set up a Blynk project linked to their personal account. QR code to link user to a clone of the project can be found in A7. If the blynk user changes, so does an authentication key within the ESP8266 software that will need to be changed for use.
2. Review section A5 and A6 for operation instructions and considerations for use of the Blynk app.
3. Take the top cover off of the robot so that the main controller and ESP are visible. The battery and power connections are located on the right side of the robot when facing the IR array. Plug the battery wire into the power wire as shown in section A2.
4. On the same side of the car as the battery and power wires is the electronic speed control (ESC) system with the power switch for the drive motor. Lift up the black battery cover to reach the power switch and turn the robot on.

   You should hear one beep when the ESC is turned on and a second beep to indicate the battery is sufficiently charged. If no second beep is heard, recharge the battery following in the instructions in section A3.
5. Place robot on lane line of track, with the IR array facing the desired direction of travel.
6. Select speed and distance with the Blynk app and hit SET.

   Note that the settings must be changed in order for the robot to receive the correct settings. If the speed bar does not change when SET is pressed, adjust speed and distance to dummy values, hit SET, then readjust to the correct values and hit SET again.
7. Hit START to start the robot. Note that the robot will begin moving up to 5 seconds after START is pressed. A high pitched tone will indicate that the robot is about to start moving.
8. Use STOP to stop pacing. The robot will not stop at the set distance, so STOP is the only way to stop the robot.
9. When finished pacing, turn off the ESC and unplug the battery from the power. Recharge the battery using the information in section A3.

A2. Powering the Device

To power the device, first remove the top cover of the RC car. Plug the wire labeled BATTERY into the wire labeled POWER.

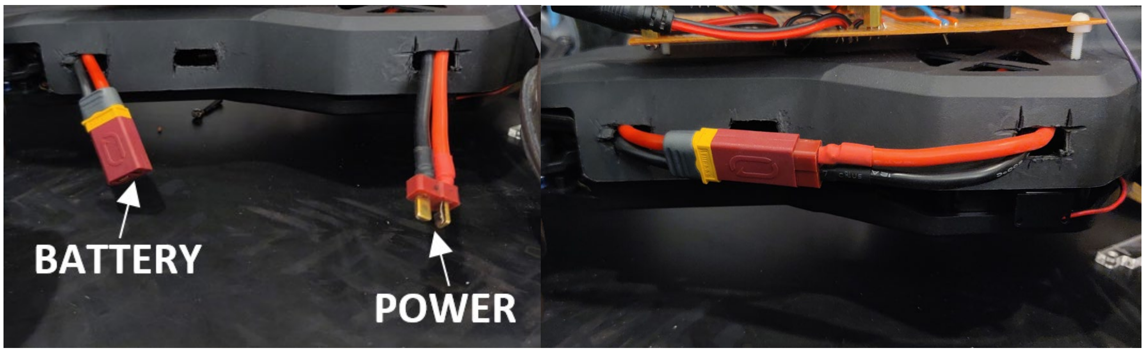


Figure A2.1. Powering the device.

A3. Charging the Device

A Helion 0.5 Amp Battery Charger is supplied. The LED Charge Indicator displays the status of the battery while it is charging. It can also be used to check the battery before runs. If the light is green while plugged into the battery, the battery is fully charged.

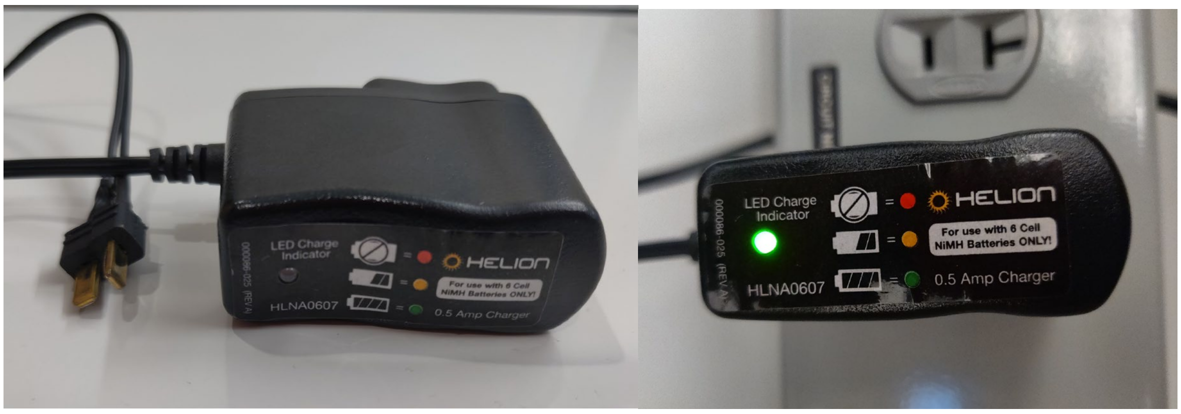


Figure A3.1. Helion 0.5 Amp Battery Charger

To charge the battery, plug the charger into the battery port.

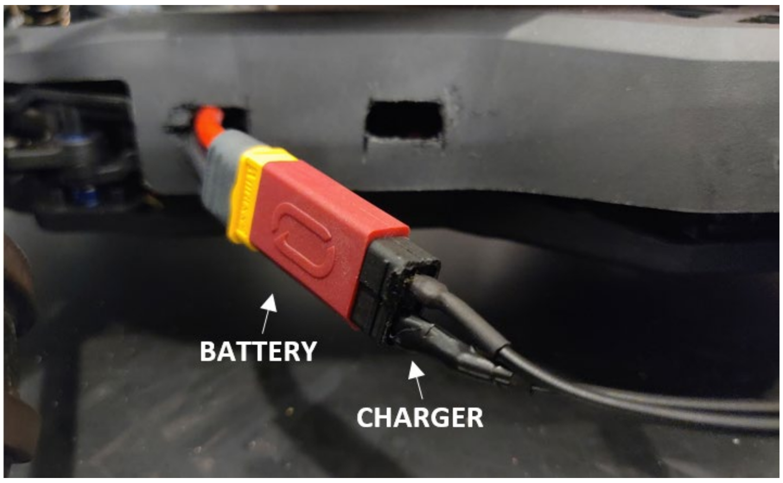


Figure A3.2. Charging the battery.

WARNING: DO NOT LEAVE THE BATTERY UNINTENDED WHILE CHARGING AND DO NOT LEAVE ON CHARGER ONCE IT HAS FINISHED CHARGING. OVERCHARING THE BATTERY CAN CAUSE OVERHEATING AND FIRE.

A4. Battery Replacement

An additional 2000mAh replacement battery is supplied. If replacement is needed, please contact the Pace-It Design Team for assistance.

A5. General Use

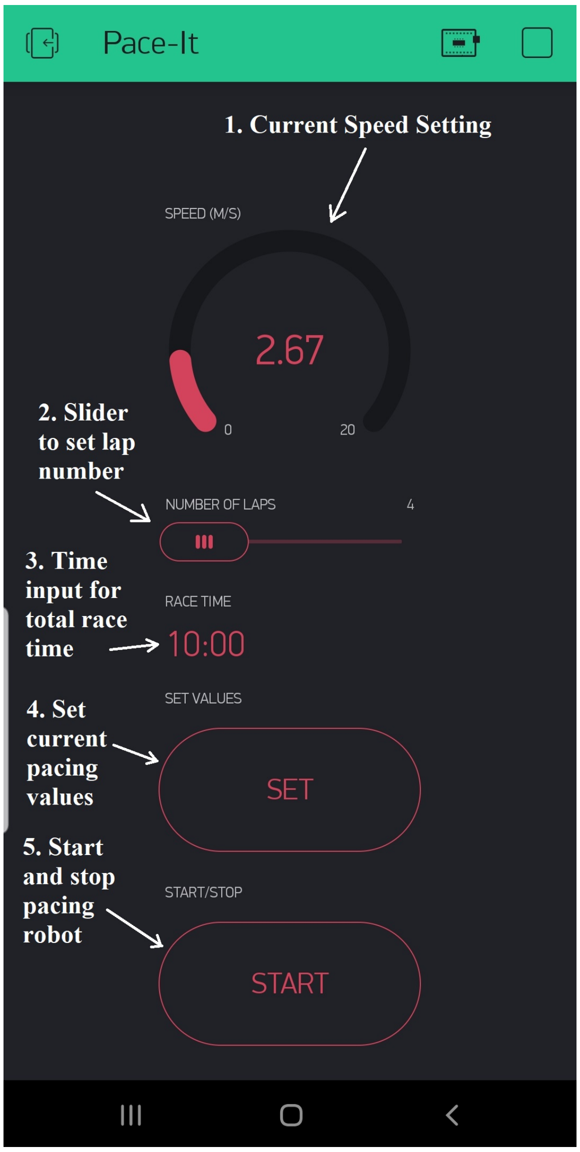

Pace-It
1. Current Speed Setting
SPEED (M/S)
2.67
0
20
2. Slider to set lap number
NUMBER OF LAPS
4
3. Time input for total race time
RACE TIME
10:00
4. Set current pacing values
SET VALUES
SET
5. Start and stop pacing robot
START/STOP
START

Figure A5.1: Normal operation user interface

1. **Current Speed Setting:** feedback showing the current speed set point of the system. Not an input, only shows the current speed, cannot alter speed setpoint using this dial.
2. **Slider to set lap number:** slider can be used to set the total lap length of the race from 4 laps to 16 laps. **DISCLAIMER:** as of now the pacing robot does not stop after the length of the race has been reached! Do not expect the robot to stop automatically. This value is used to calculate the pace of the robot
3. **Time input for total race time:** Time input for total time length of the race. Note: on android platforms the time value shows as MM:SS format, whereas on iPhone the time is shown as MM:SS:mm, make sure the pacing values are placed in the correct time location(M: minute, S: second, m: millisecond).
4. **Set current pacing values:** when pressed, uses the number of laps input and race time input to set the current setpoint of the pacing robot. Does not start or stop the robot, can be used while robot is active or inactive. If used while active, will change current set point to new set point.
5. **Start and stop pacing robot:** Toggle switch used to start and stop the robot. Also sends the current set point to the pacing robot.

A6. Special Considerations With Use

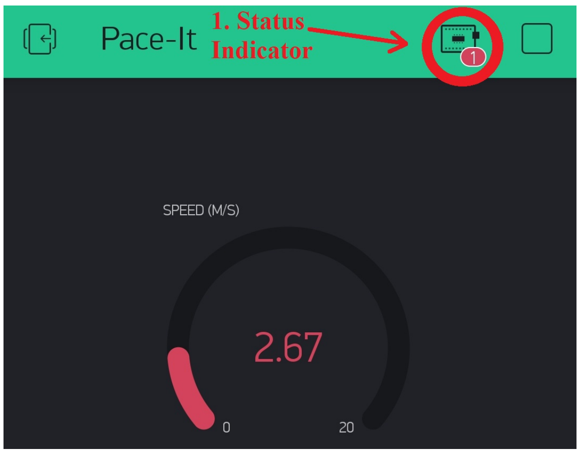


Figure A6.1: Status indicator

1. **Status Indicator:** Informs user of current status of Blynk program. Most importantly will inform the user if the application is properly connected. In general, if a red bubble with a number is seen, the device is likely disconnected and will not operate correctly. Make sure all device components are properly connected to power.

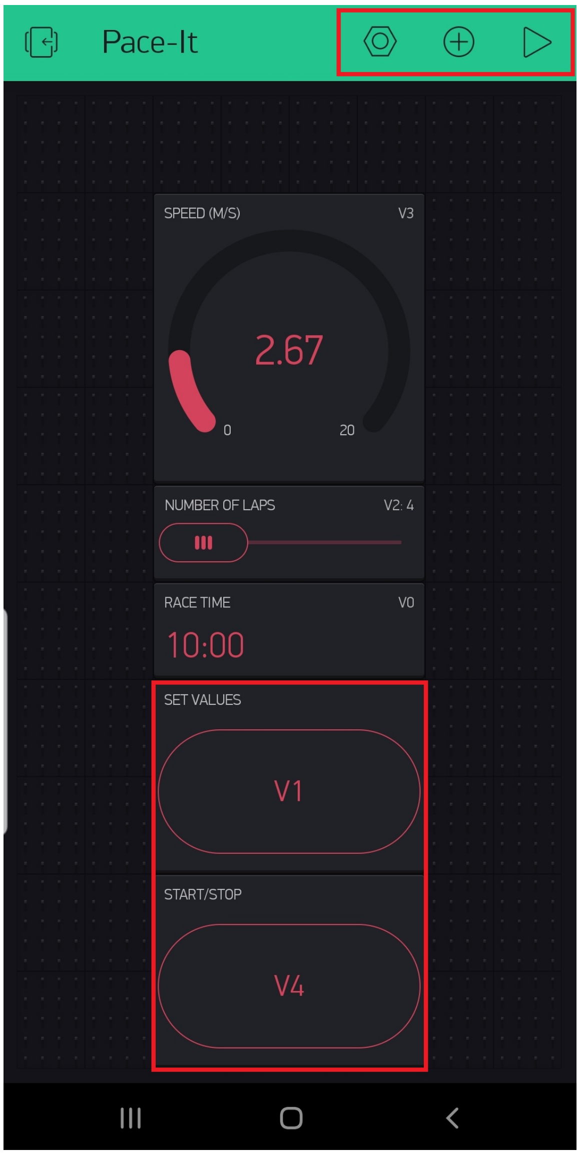
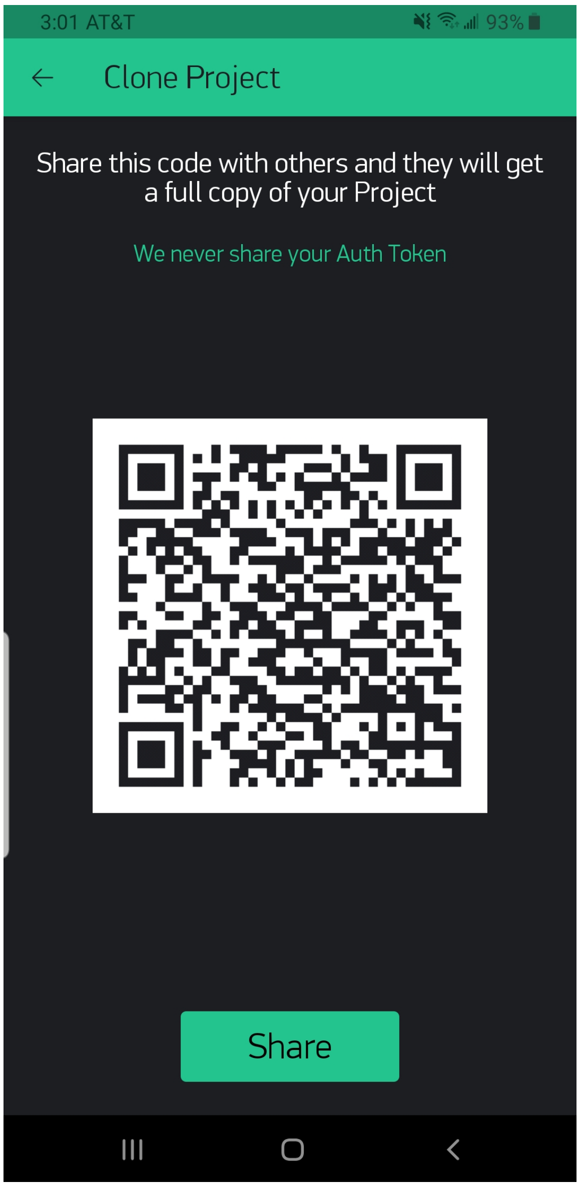


Figure A6.2: Edit mode of Blynk

2. **If the application interface looks like this picture then the application is not in run mode.** It is not advisable to alter any of the values on the screen in this mode. Press the play/triangle button in the top right corner of the screen to attempt to connect the application to the pacing car.
3. **When first using the application, change the number of laps setting and race time setting before sending the desired value.** This includes if the value is at the desired set point. The user must change these values and select set, then they can change the values to the desired set point. This allows the application to initialize with the ESP8266 before sending the setpoint.


3:01 AT&T 93%

Clone Project

Share this code with others and they will get a full copy of your Project

We never share your Auth Token

Figure A6.3: QR link to project clone

4. **Use this QR code to link to the Pace-It blynk project.** This will create your own personal version of this project connected to your Blynk account. Keep in mind this will also change the authentication token that will need to be set within the ESP8266 software.

Appendix B - Detailed Information on Design (Subsystems)

B1. IR Array Subsystem: IR sensors, ADCs, Enclosure

In order to detect the line we used an array of IR sensors. These sensors were soldered on a single project board along with resistors and Analog to digital converters (ADCs). There were a total of 6 IR sensor packages along with six ADCs. A schematic that describes the IR array used in our final design can be seen in Fig. B1.1.

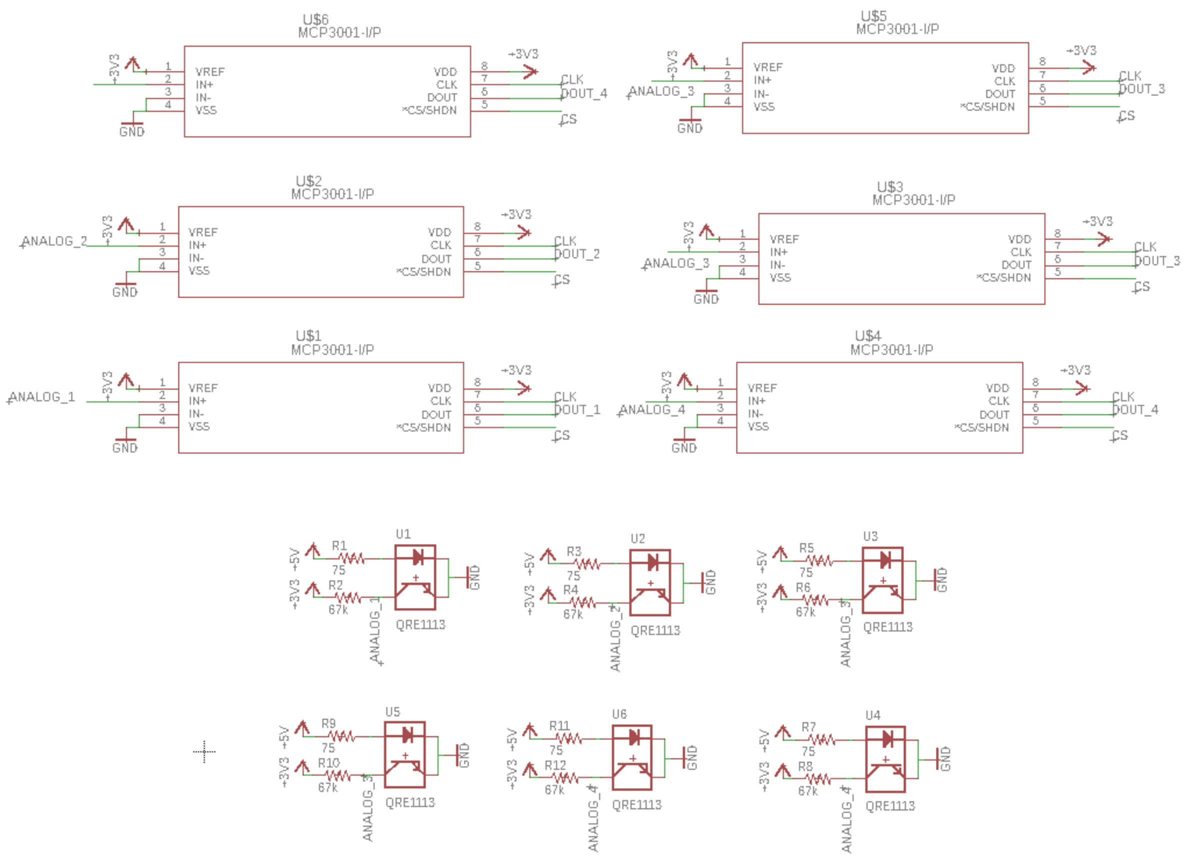


Fig. B1.1 Final Schematic of IR Array

The 3.3V, CLK, GND, and CS pins are connected to the DE0 controller through wires. All of the D_OUT pins are also connected to controller as individual inputs. The 5V and GND pins are connected to the outputs from the 5V switching regulator. For more information on how the controller uses/reads these inputs to accomplish line following see Appendix B2.

Fig. B1.2 shows the physical implementation of the schematic seen in Fig B1.1. The purple and white wires are the CS and CLK signals, with the orange and red wires powering the ADCs at 3.3V. The underside of the IR array is seen in Fig B1.3, which is the side with that faces the track as the car moves.

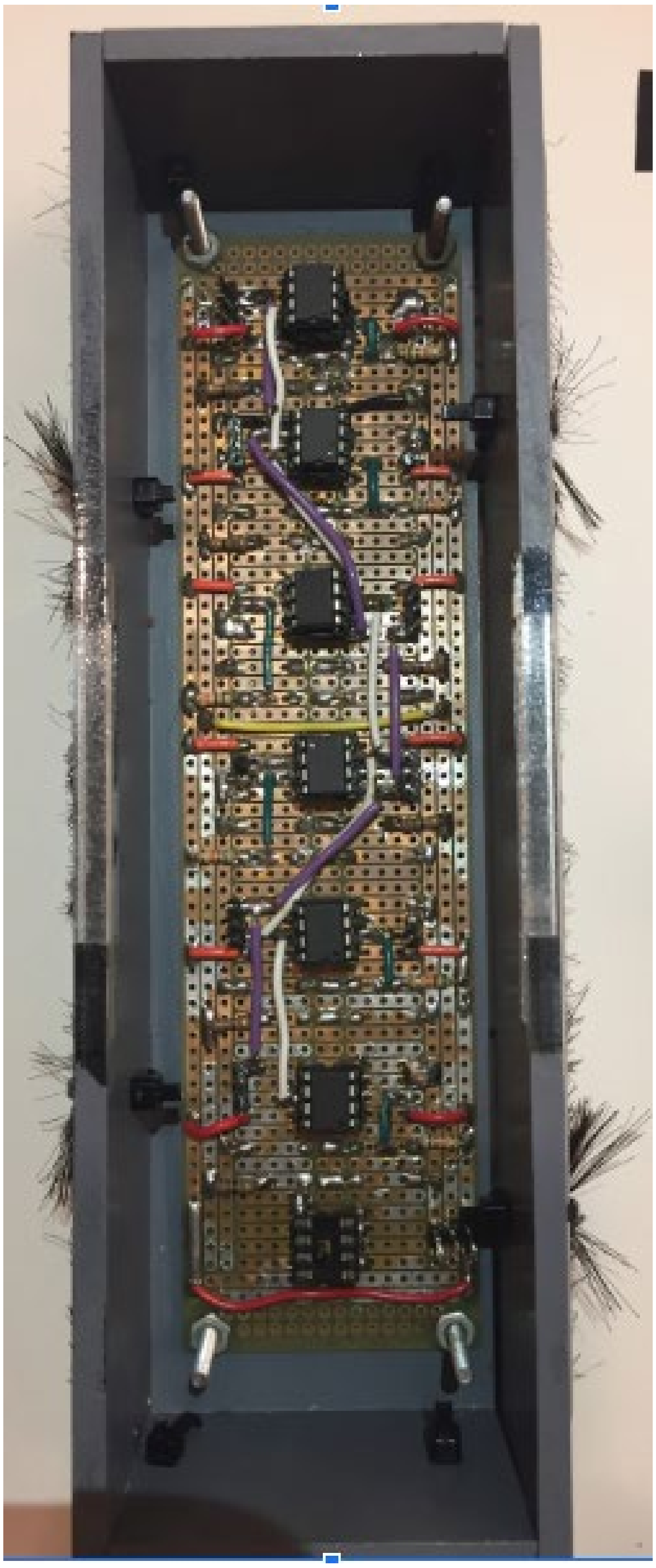


Fig. B1.2 IR Array as implemented in final design in enclosure from above.

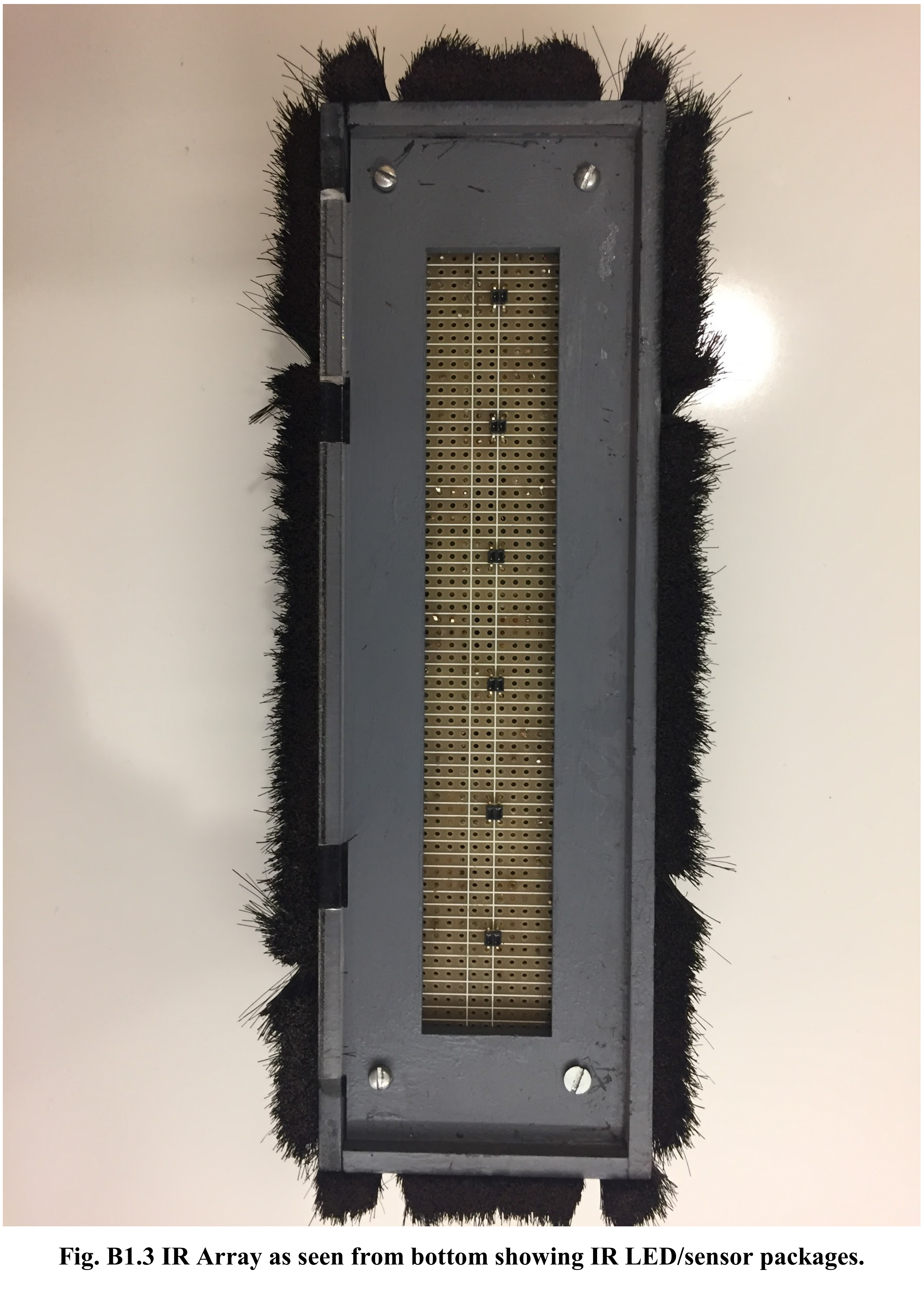


Fig. B1.3 IR Array as seen from bottom showing IR LED/sensor packages.

B1.1 IR LED/Sensor

The IR sensor used was the QRE1113. This component contains a diode that emits infrared light when it receives a certain current and forward voltage, and a phototransistor that varies its resistance from its collector to emitter based on the amount of infrared light it receives on its base. Figure B1.4 shows an array of QRE1113, where Rd and Rt are chosen based off of the ratings found in Table B1, and V_adj is changed to vary the brightness of the LED. Figure B1.5 shows the setup conditions for the initial testing of the QRE1113, and Table B1.2 shows the results of these tests.

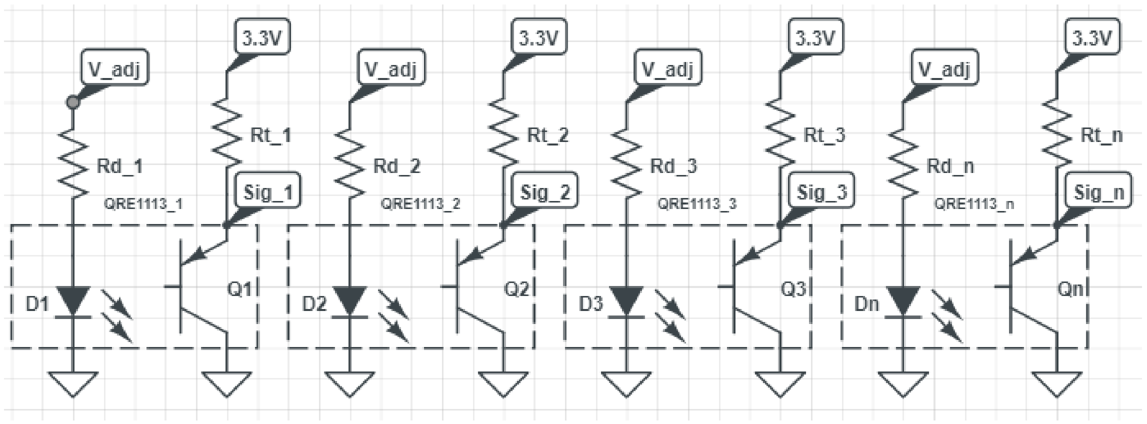


Figure B1.4 IR Array based on QRE1113 emitter/receiver package, with n packages.

Table B1.1 Max ratings for QRE1113 IR Module [1]

LED	Max Current	50 mA
	Forward Voltage	1.2 to 1.6 V
	Max Power Dissipation	75 mW
Photo Transitor	Max Current	20 mA
	Max Vce	30 V
	Max Power Dissipation	50 mW

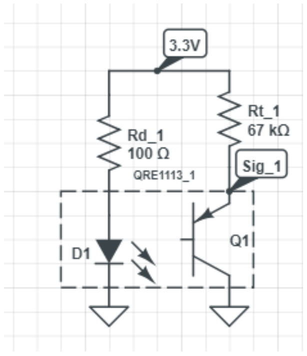


Figure B1.5 Initial testing setup of QRE1113

Table B1.2 Sig_1 when Black surface and white surface were held approximately 1 cm above QRE1113 configured as shown in Fig. A1.2.

Surface	Sig_1
White	2.1 V
Black	3.1 V

B1.2 Analog to Digital Converter

The ADC used was the MCP3001. It is a 10 bit ADC which we powered at 3.3V. It communicated with the main controller through SPI. We chose to use 6 individual ADCs instead of one 6 channel ADC so that there would not be a bottleneck on one SPI output and we could read and process the reading from each sensor at the same time. A schematic of the ADC can be seen in Fig. B1.6.

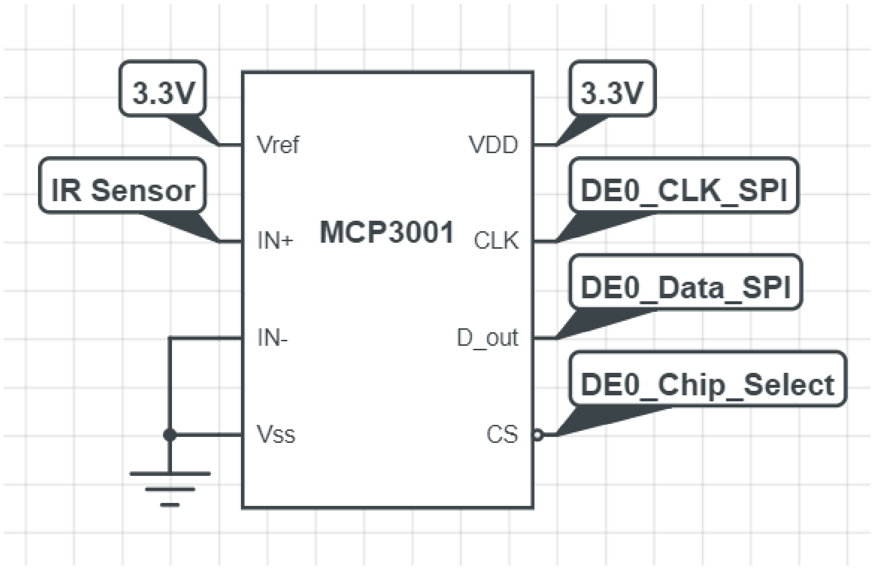


Figure B1.6 Analog to Digital Converter Schematic

B1.3 IR Array Enclosure

The IR array project board is screwed into a plastic enclosure that was designed in Fusion 360 and laser cut with the Zing laser cutter in the Trinity machine shop. A front, top, side, and isometric view of Fusion 360 drawing of the case is shown in Fig. B1.7.

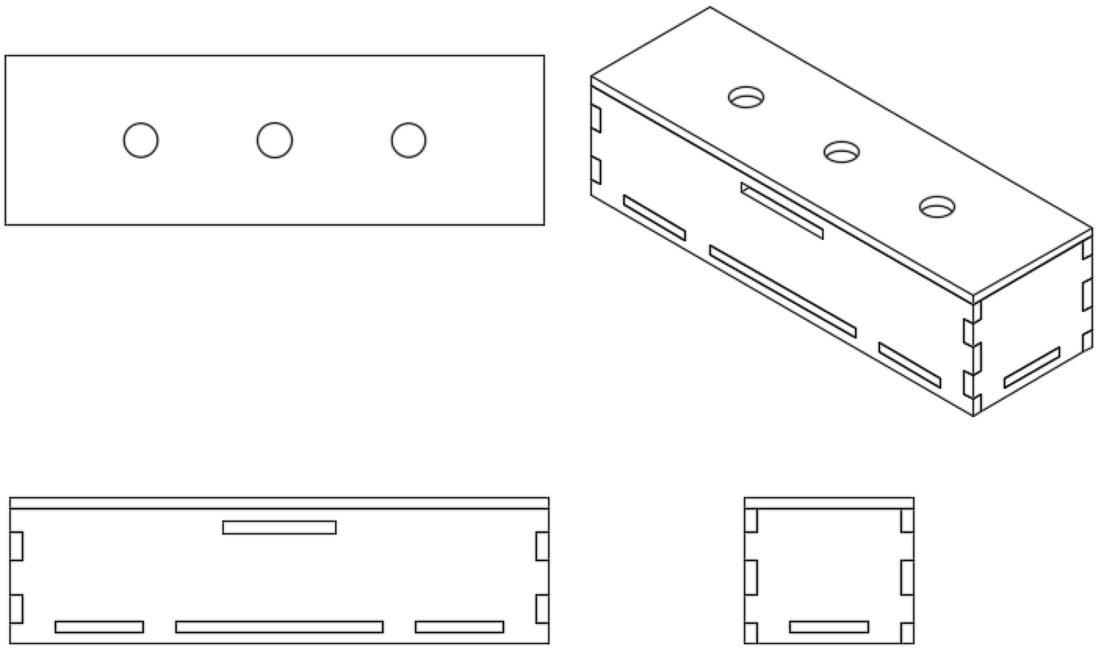


Figure B1.7 Fusion 360 drawings of the IR sensor enclosure.

Brushes are attached around the edges of the case with zip ties in order to push debris out of the way of the robot and block out external IR light from the sun. Watertight seals are used to output the wires that connect to power and the controller. A piece of formed aluminum is bolted to the front of the chassis, which has the plastic mount for the case attached to it with screws. This was done so the mount could be removed if necessary. The mount was also laser cut and assembled with acrylic. The design for the mount is shown in Fig. B1.8.

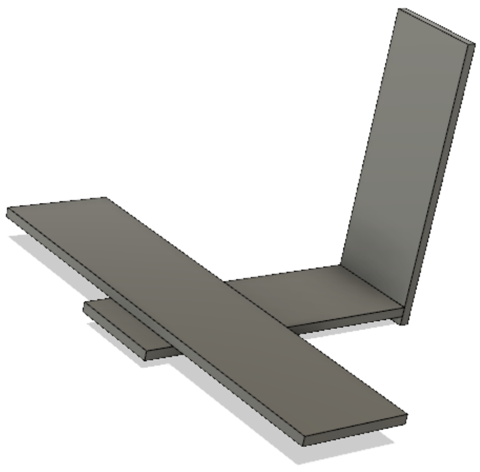


Figure B1.8 Fusion 360 image of mount used for IR sensor case.

As seen in Fig. B1.8, the case is designed to slide onto the flat bottom arm of the mount. This was done so the case can easily be taken on and off for easy access to the IR sensor array. The horizontal piece perpendicular to the bottom arm is then screwed on to hold the case in place. There are two one inch wheels that attach to on either side of the horizontal piece that keep the array a constant height from the ground. The wheels move freely so they do not affect the turning of the car. A front, top and side view of the full mount and case can be seen in Fig. B1.9-11.

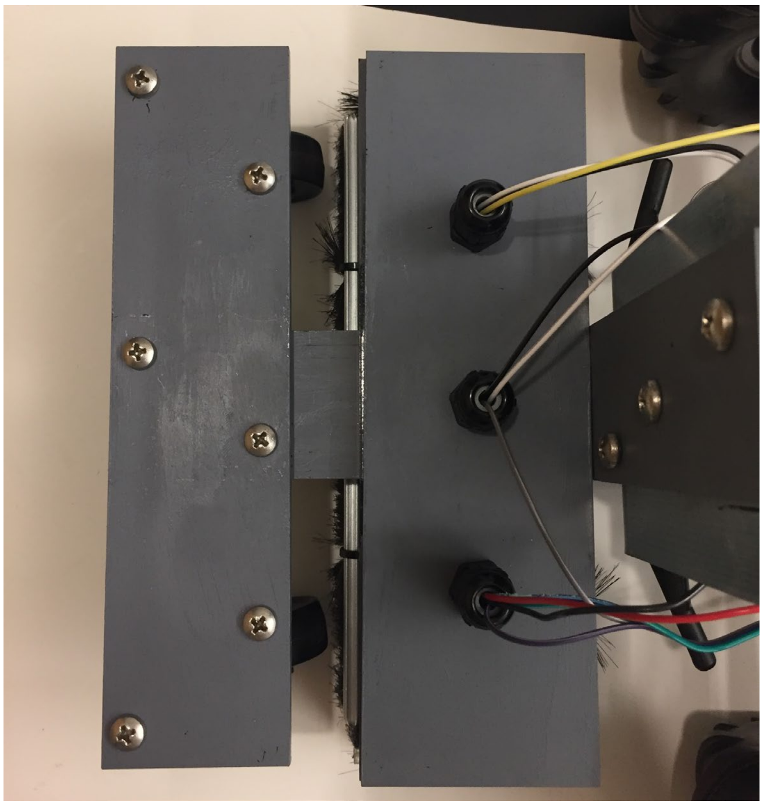


Figure B1.9 Top view of IR Array enclosure.

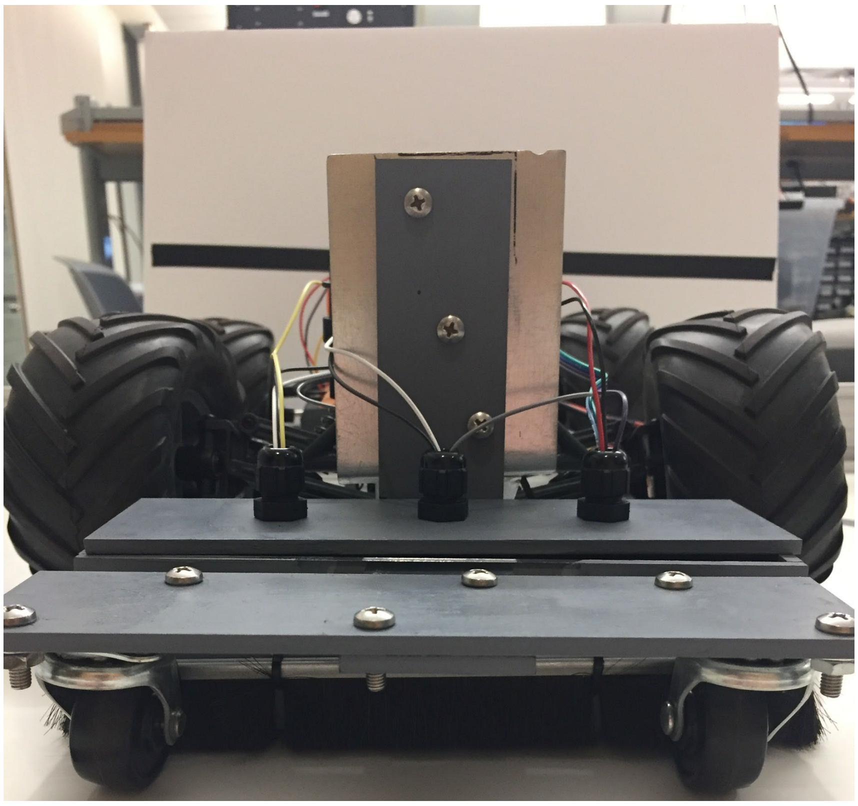


Figure B1.10 Front view of IR array enclosure.

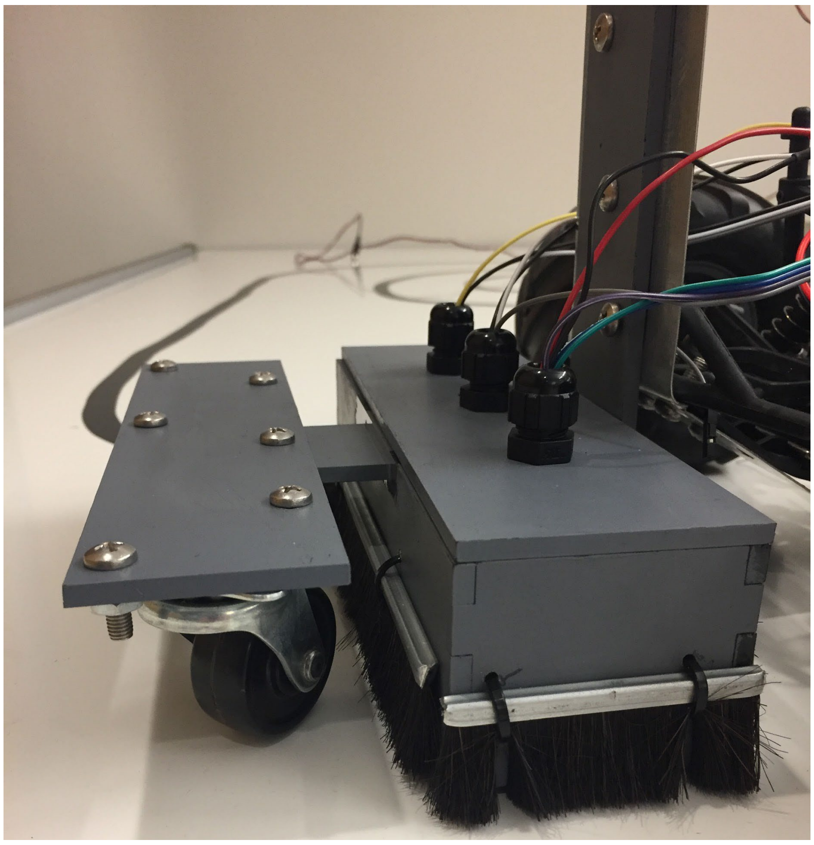


Figure B1.11 Side view of IR array enclosure.

B2. Controller

The controller used was the DE0 Nano which is a development board focused around the Altera Cyclone® IV EP4CE22F17C6N FPGA. This controller is responsible for reading the values sent through the ADCs from the IR sensors, reading the values sent from the Hall effect sensor, reading the values sent from the ESP8266, controlling the servo, and controlling the DC motor. In order to accomplish all of this it was programmed using VHDL through the Altera Quartus software. The design architecture can be broken down into many subsystems (modules), each created by a different VHDL code file in Quartus II software, and the entire system is represented by interconnected blocks. The overall design uses approximately 85% of the total available logic elements including elements used for debugging purposes. In addition to the major functional blocks listed below there are also several clock prescaler modules, multiplexers, buffers, and various debugging blocks.

B2.1 ADC

There are six individual modules used to read the SPI communications from each ADC, along with six modules used to convert process these values and set an output that describes the status of the robot on the line. These blocks can be seen in Fig. B2.1. There are also separate modules used to reset these stored values.

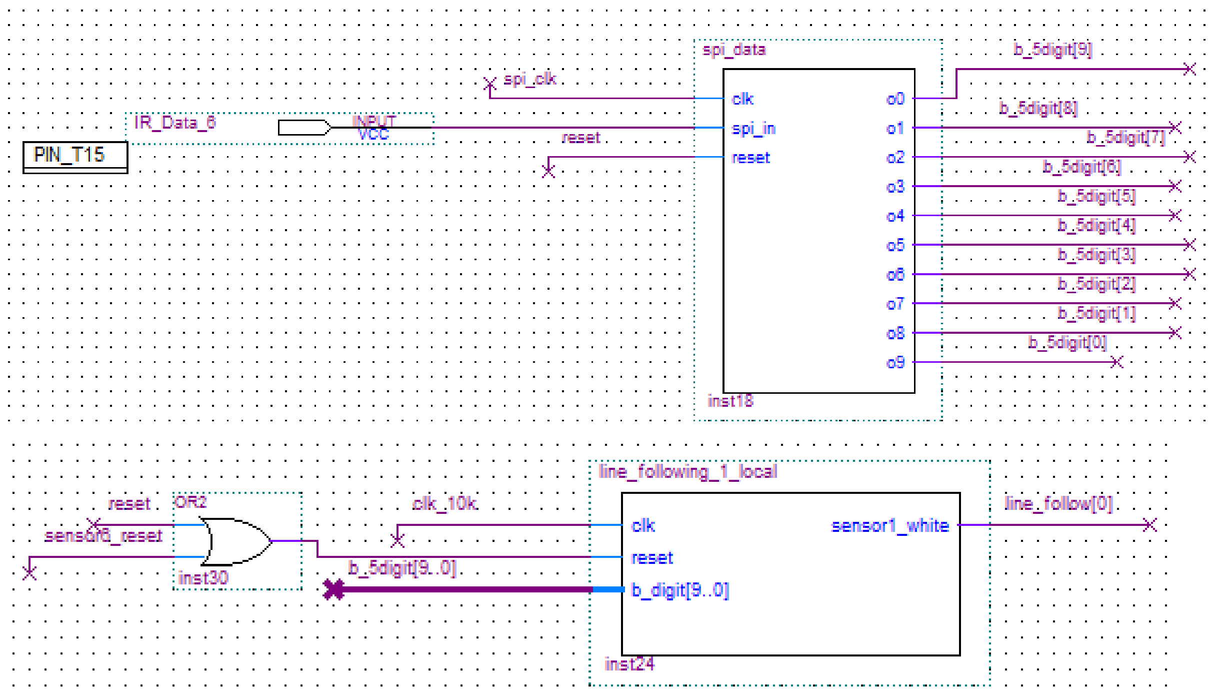


Fig. B2.1 SPI processing module (top) and line detection module (bottom).

B2.2 Servo Steering

There are two different modules used to control the servo motor that steers the robot. These can both be seen in Fig. B2.2. The duty_pid_steering block takes the outputs from the line_following blocks and processes them using proportional and derivative control to create a duty cycle value (32 bits long) that is then fed into the pwm_steering module that outputs a pulse width modulation signal that is sent from a GPIO pin to the servo motor.

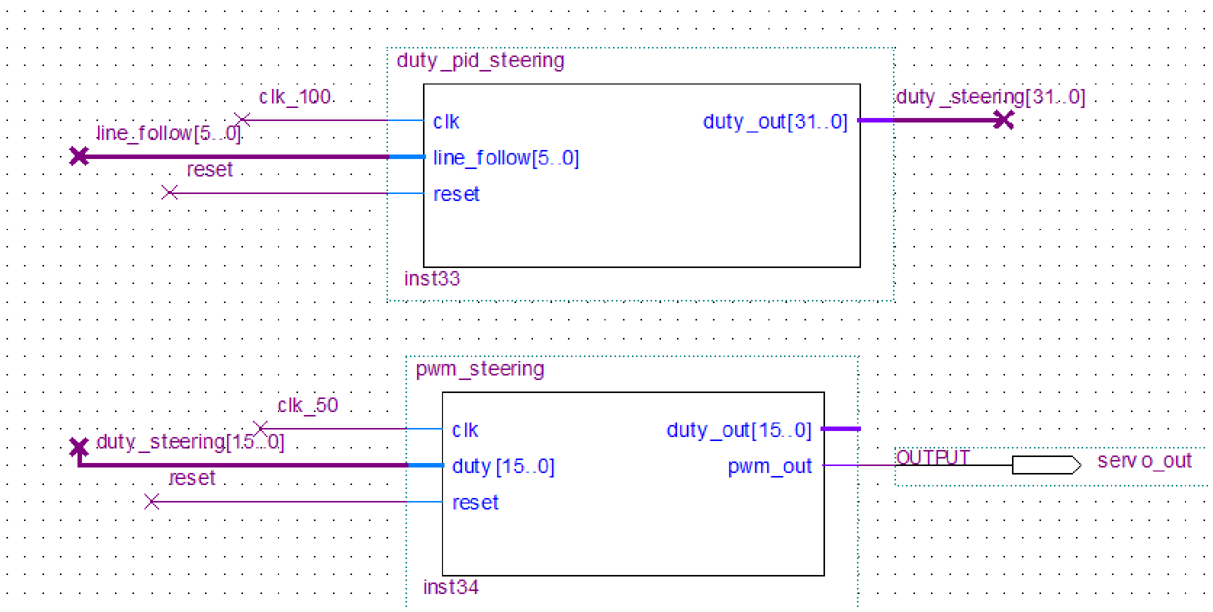


Fig. B2.2 Servo Controlling Blocks

B2.3 Communication with App

There is one module that facilitates communication between the app and the DE0. The I^{2}C block seen in Fig. B2.3 and takes the SCL and SDA signals from the ESP8266 and processes them to output the setpoint data, which is sent based on the user choosing it in the app. This is sent as data_1 (16 bits). This module also outputs addresses which are used for debugging.

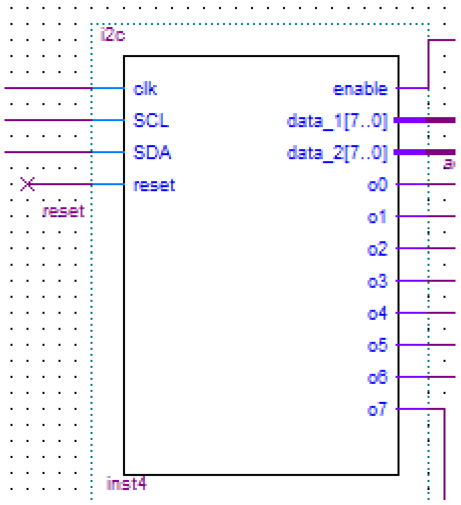


Fig. B2.3 I^{2}C decoding module, where o1-o7 are the address values used for debugging.

B2.4 Processing the Hall Effect Sensor

There is one module required for processing the signal sent by the Hall effect sensor. The motor module receives the square wave sent from the Hall effect sensor output (received through a GPIO labelled as rpm_measure). It then counts the time between magnet detections to calculates the instantaneous RPM value, which is then sent to the DC motor control blocks. This block can be seen in Fig. B2.4

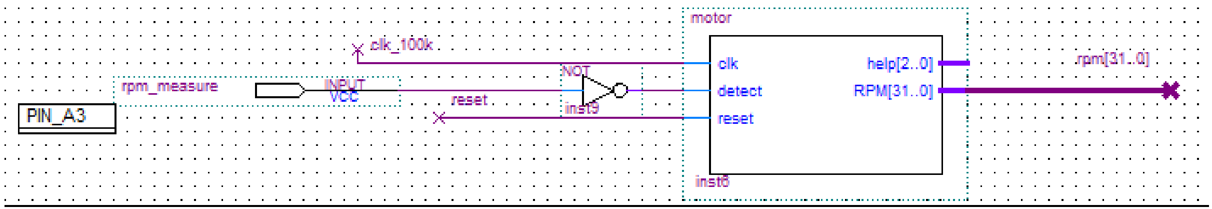


Fig. B2.4 Hall effect sensor processing module

B2.5 DC Motor Control

There are two modules that control the motor. Duty_pid_motor takes the RPM setpoint received from the I^{2}C module and compares it to the measured RPM from the Hall effect module and outputs a duty cycle value (32 bits) that is sent to the PWM module that outputs a pulse width modulation signal that is sent to the ESC through a GPIO pin. These modules can be seen in Fig. B2.3

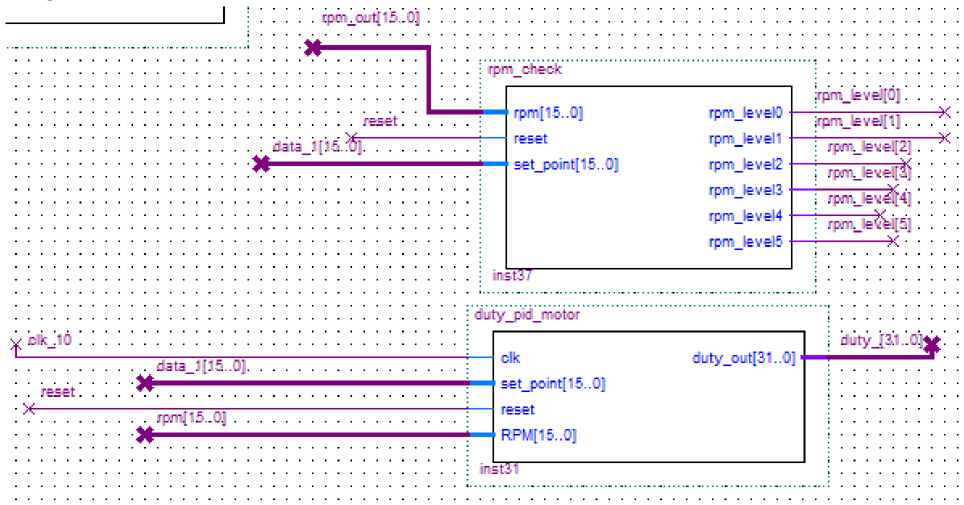


Fig B2.3 DC Motor Control, where data_1 is the data sent from the I^{2}C module

\

B3. DC Motor/Hall Effect Sensor

The DC motor that moves the robot forward and backward is controlled by an ESC that came on the remote control car, which is controlled through PWM by one digital output sent from the controller. The speed of the motor is measured using a Hall effect sensor and a magnet. The Hall effect sensor is mounted on a small project board that is placed near the back of the vehicle near an exposed gear connected the motor shaft. A small magnet was attached to this gear. The Hall effect sensor outputs a pulse every time the magnet passes it. Through this action, the controller is able to count the revolutions of the gear and determine then calculate the revolutions per minute.

B3.1 Hall Effect Sensor

The Hall effect sensor used was the US5881LUA. A schematic for the Hall effect sensor circuit can be seen in Fig. B3.1, with the actual implementation shown in Fig. B3.2.

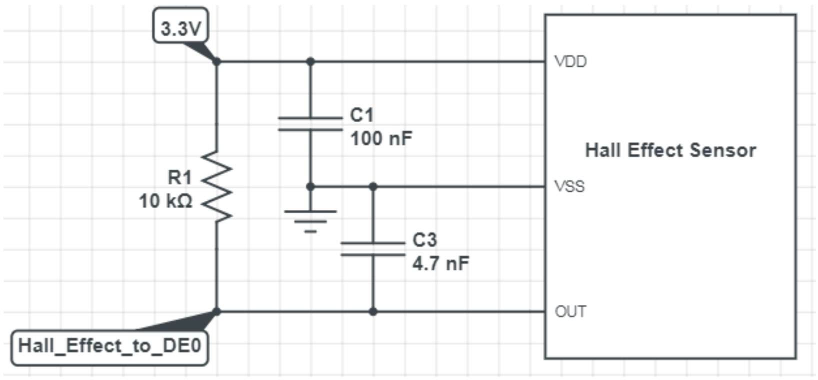


Figure B3.1 Schematic for Hall Effect Sensor

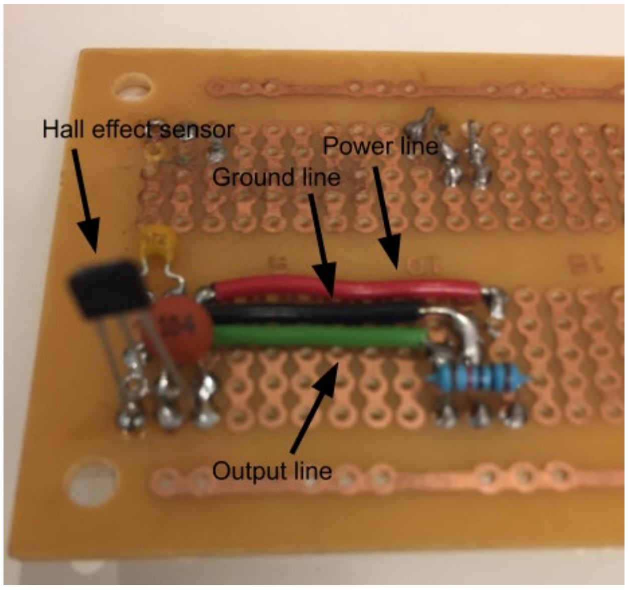


Figure B3.2 Implementation of Hall Effect Sensor circuit.

B3.2 Initial Testing of Hall Effect Sensor

An initial test showing the output of the Hall effect sensor on the oscilloscope when a magnet is passed in front of it can be seen in Fig. B3.4. The Hall effect sensor was then tested on a motor that had a built-in tachometer in order to compare RPM measurements. As seen in Fig.B3.5, the built-in tachometer outputted a sinusoidal wave with a frequency 8 times that of the Hall effect sensor. This is because the built-in tachometer is an inductive sensor with 8 magnets. Thus, this testing showed that our Hall effect sensor was successfully measuring the RPM of the motor.

In the actual RC car, it was not plausible to place a magnet on an actual car wheel and have a Hall effect sensor located nearby for measurements so it was placed on a gear connected to the motor. Since the wheel rotation was not being measured, the ratio of the turns of the gear to the turns of the wheel needed to be found in order to calculate the RPM of the gear based on the speed of the car. The group accomplished this by temporarily attaching a magnet to a wheel and holding a Hall effect sensor nearby in order to compare it to the signal from the Hall effect sensor near the magnet on the gear. The results for this test are shown in Fig. B3.6. Several tests showed that the ratio of gear rotations to wheel rotations was around 2.5.

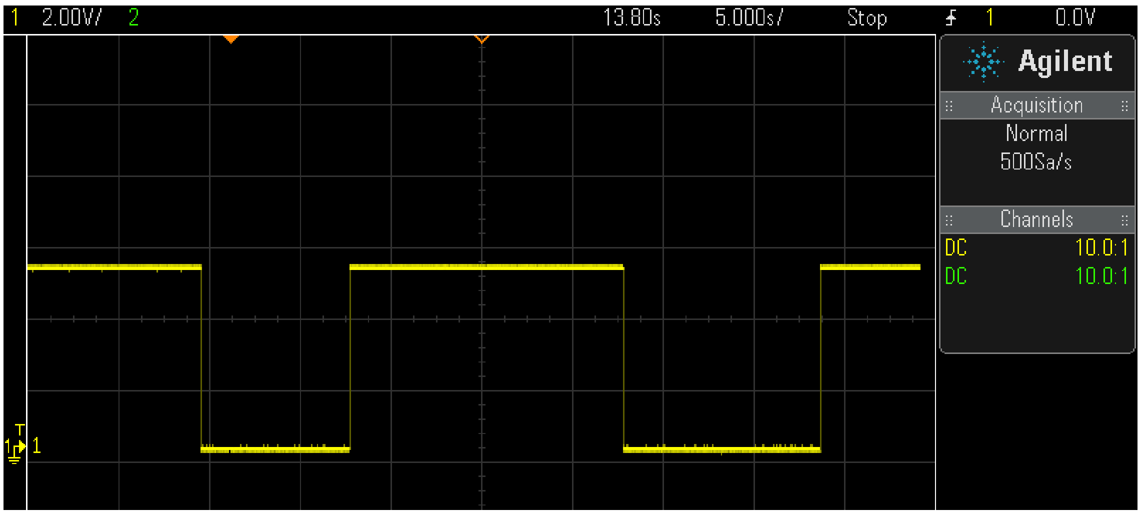


Figure B3.4 Oscilloscope image from the Hall effect sensor showing two detections.

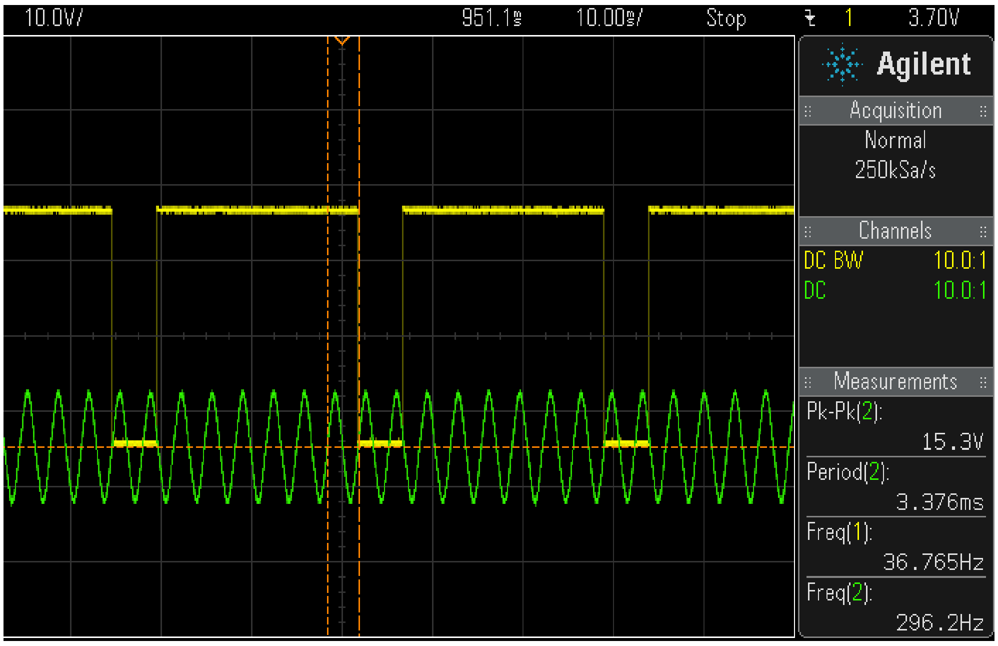


Figure B3.5 Oscilloscope image from the Hall effect sensor signal (yellow) vs built-in tachometer signal (green).

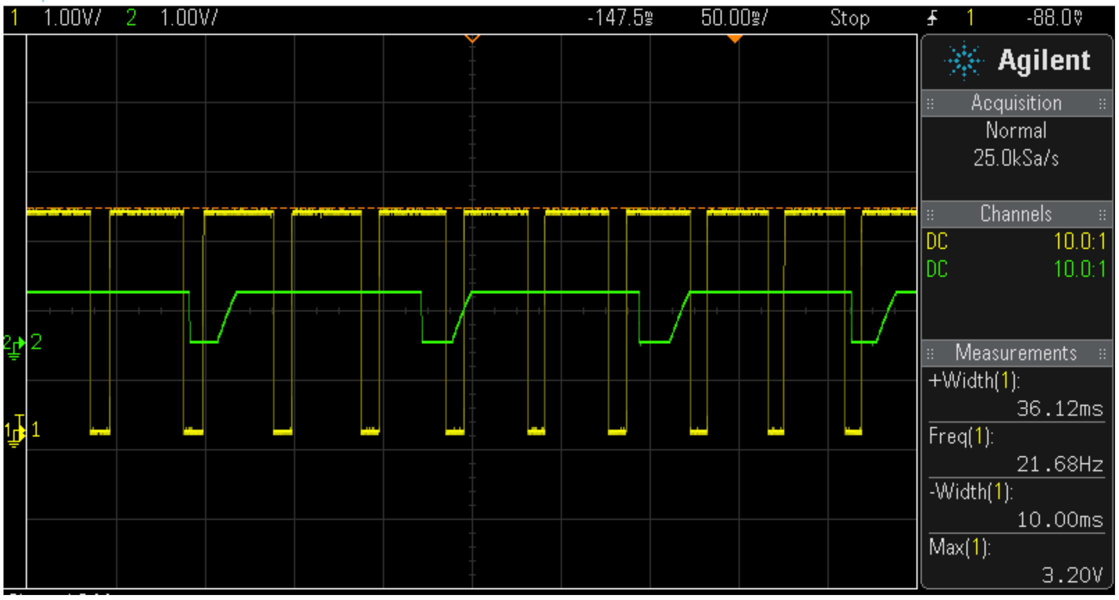


Figure B3.6 Oscilloscope image from test to determine ratio between gear rotations (yellow) and wheel rotations(green) using the permanent tachometer to measure gear rotations and a temporary tachometer of the same design to measure wheel rotations.

B4. Battery Management

A Venom 7.2V High Power 6 Cell NiMH battery was purchased to power the Pace-It device. This high capacity battery has enough power to provide multiple runs without recharge.

This is beneficial because 1) multiple recharges degrades the battery and the high capacity battery will require less charges, 2) higher capacity means that less time can be spent charging the device and more time can be spent using it, and 3) the charging of NiMH batteries have a low chance of safety risks; minimizing the number of charges required maximizes safety in this regard. The battery is pictured in Fig. B4.1

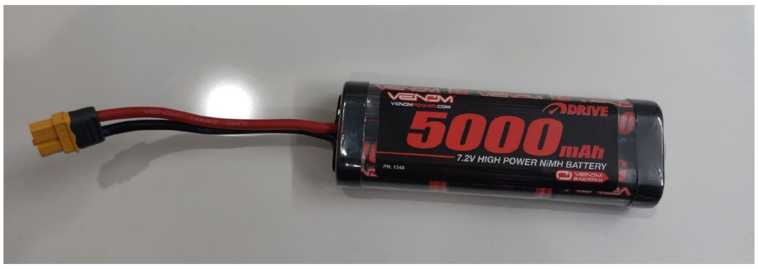


Figure B4.1. Venom 7.2V High Power 6 Cell NiMH battery.

A replacement Helion 7.2V 6 Cell 2000mAh NiMH battery is provided as a backup if complications were to occur with the main Venom battery. This battery has 40% the total capacity of the Venom battery but can still provide enough power for several laps around the track. The replacement battery is pictured in Fig. B4.2.

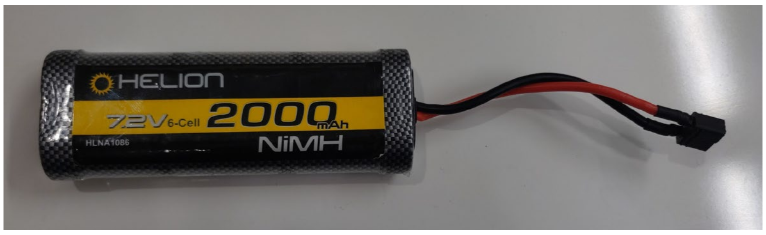


Figure B4.2. Helion 7.2V 6 Cell 2000mAh NiMH battery.

When the battery is at full charge it is at about 9V. When it is low, it goes down to about 7.5V before dying. A voltage regulator is required to supply a constant power source for all of the systems running at 5V. A LM2596 Buck Voltage Regulator takes the variable battery voltage and provides a constant 5V. The specific LM2956 used came prepackaged with the protection circuitry and a 10kΩ potentiometer for an adjustable voltage output. The circuit diagram is shown in Fig. B4.3. The LM2596 has a conversion efficiency of about 92%. The LM2596 is able to provide a constant voltage ranging from 4.5V - 40V, depending on the input voltage. A 10kΩ potentiometer was used to step the voltage down to 5V. The voltage was tested and was able to stay within ±0.01V from full to zero charge. The physical LM2596 Buck Voltage Regulator on board is pictured in Fig. B4.4.

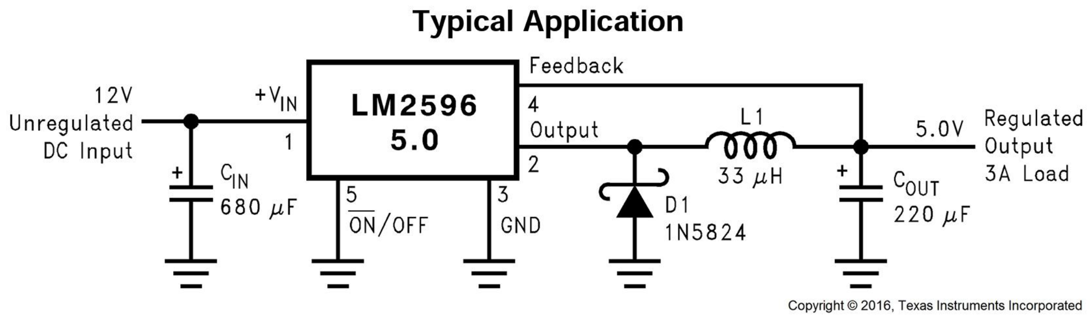


Figure B4.3. LM2596 Buck Voltage Regulator circuit diagram.

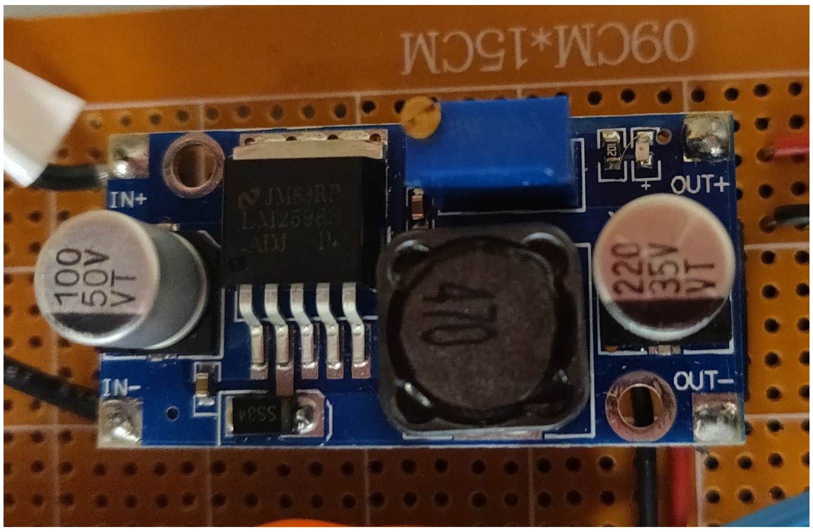


Figure B4.4. LM2596 Buck Voltage Regulator on board.

Appendix C - Total Cost

Table C1. Total Cost Broken Down by Purchase

Date	Description	Vendor	Cost	Balance Remaining
9/25/18	Opening Balance	Trinity University		$1,200.00
11/13/18	Magnets, Hall Effect Sensor	Adafruit	$26.43	$1,173.57
11/09/18	Optical Switches 10 ea	Mouser	$15.73	$1,157.84
02/05/19	Helion 2WD RTR Truck	Amazon	$169.98	$987.86
02/05/19	10-bit SPI Sgl Chl	Mouser	$24.79	$963.07
03/18/19	De0-Nano Dev Kit	Amazon	$79.00	$884.07
04/3/19	Hall Effect Sensor replacement	Mouser	$13.44	$870.63
04/2/19	Strip Brush	Zoro	$16.13	$854.50
04/2/19	Circuit board, pin headers, diodes, buck converter	Amazon	$32.74	$821.76
03/27/19	IR receiver + emitter	Mouser	$15.73	$806.03
04/8/19	Cables and connectors	Amazon	$24.28	$781.75
04/23/19	7.2V Battery	Amazon	$44.99	$736.76
04/29/19	2x De0-Nano Dev Kit replacements	Amazon	$158.00	$578.76
04/29/19	Bottom screws, nuts, glue, zip ties, acrylic, spray paint	Trinity Machine Shop	$0.00	$578.76