

Engineering Recommendation Report: Autonomous Robot for Plant Monitoring

Akshiv Bansal
Arjun Venkatesh
Bryden Fogelman

Project Sponsors:
Dr. Saber Miresmailli
Mr. Nicolas Unick

ENPH 459
Engineering Physics
The University of British Columbia
April 8th, 2017

Project Number: 1702

Executive Summary

Ecoation Innovative Solutions (EIS) aims to revolutionize farming by providing analytics on the state of crops using their all-in-one plant scanning system, Crop Sense. Currently, EIS manually performs scans at Van Belle Nursery using a mechanical chassis that rolls on the pre-existing greenhouse rails (2.5 m apart, 1 m above the ground, and approximately 100 m in length). This report details the design and implementation of a movement system capable of traversing the greenhouse semi-autonomously to a precision of ± 1 cm. The system is able to accept user input through a joystick and can use a web app to execute scans automatically.

To function in with the wet and humid conditions of a greenhouse, all of electronics are rated to IP65 or higher while mechanical components are made from aluminum and stainless steel to avoid corrosion. The chassis is driven by a driven down the rails by a DC motor coupled to a custom built flanged wheel by a 1:1 chain drive. Stepper motors (NEMA 23 IP65 rated) are used to move Crop Sense in the horizontal and vertical directions via lead screw and timing belt. The electrical system is powered by a 24V battery pack and uses a printed circuit board (PCB) to drive the motors, read sensor inputs, and interface with a Raspberry Pi 3 (RPI) microcontroller.

The RPI is responsible for motor control, and wireless communication. On initialization, JSON configuration files are used to populate variables such as pin information and motor thresholds. Control algorithms drive the chassis down the rails, and move the stepper motors in the horizontal and vertical axes. Once the robot is in position, the RPI issues a wireless scan command to Crop Sense and waits for a success response before continuing operation. A graphical user interface (GUI) allows for remote execution of scans across a row of plants, provided the scanner is placed in the desired position along the rail and vertical distance from the plants. The software is written in python and relies on existing frameworks to assist in tasks such as wireless communication, interfacing with the pins and hosting a web server.

Further improvements to the rail drive system are needed to get to a cleaner and more finished product. These primarily include the reimagining of the flanged wheels to be made from different material, and a more careful design of the mounting brackets to allow for easier in field adjustments. The encoding and positional tracking additionally is not currently functioning, and will need some attention to allow the operator to track CropSense along the rails in software. Initially, our project scope involved the creation of a fully autonomous system able to map out a greenhouse and automatically execute scans regularly. This proved to be a more significant challenge than anticipated. While we were not able to completely implement this functionality, this report includes information that would allow someone else to finish this aspect of the project, using the groundwork laid by this project.

At this time, a functioning prototype has been built, fractionally tested at Van Belle nursery, and is being given to EIS. The project members will provide ongoing support in terms of the design done as a part of the ENPH 459 capstone course.

Table of Contents

Executive Summary	ii
Table of Contents	iii
List of Figures	vi
List of Tables	vii
List of Equations	vii
List of Abbreviations.....	viii
Introduction	1
Background.....	1
Problem Statement	2
Scope and limitations	3
Discussion	3
Stepper Motion.....	3
Existing Mechanical Design.....	3
Stepper Motors.....	4
Stepper Driver.....	4
Rail Motion	5
Mechanical Considerations.....	5
Power Transmission.....	8
DC Motor Driver.....	10
Assembly.....	10
Electrical Systems.....	10
Overview.....	10
Printed Circuit Board.....	11
Waterproofing.....	13
Software Overview.....	13
GPIO.....	13
Configuration.....	14
Wireless Communication.....	14
Position and Control.....	15
Steppers.....	16
Limit Switches.....	17
Rail Drive.....	17
Joystick Control.....	17
User Interface	17

Testing	18
Motor Drivers	18
Stepper.....	18
DC Motor.....	19
Proximity Sensor / Encoder	19
Wireless Joystick	20
Waterproofing	21
Conclusions	21
Project Deliverables.....	22
List of Deliverables	22
Financial Summary	24
Ongoing Commitments by Team Members	25
Post Exam Implementation of Manual Rail Drive.....	25
Autonomous Software and Documentation	25
Limit Switches, Setting Origin and Position Tracking.....	25
User Interface Upgrades	25
Autonomous Horizontal Traversal	25
Recommendations	26
Encoding Along Rail Direction and PID control.....	26
Debugging and Data Collection.....	27
UI Interface Expansion.....	27
Autonomous Traversal	27
Waterproof Connectors for Electrical Enclosure	28
Improvements to PCB.....	29
Rail Drive Improvements	29
Other Mechanical Improvements.....	32
Appendix A - Mechanical Drawings.....	34
Appendix B - Electrical Specifications	48
Stepper Motor	48
Stepper Driver.....	48
DC Motor	49
DC Motor Driver.....	49
Appendix C - Printed Circuit Board	51
Power Input Circuit	51
Stepper Driver Circuit	52
DC Motor Driver Circuit	53
Limit Switch Inputs.....	54
Two Channel ADC Circuit	54

DC Motor Current Sense Scaling Circuit	55
Battery Voltage Sense	55
Appendix D - Raspberry Pi Configuration Information	57
General Information.....	57
Required Packages	57
Appendix E - Software Documentation	58
Appendix F – PCB Altium Schematic.....	72
Appendix G – PCB Full Bill of Materials.....	73

List of Figures

FIGURE 1: INSIDE ONE OF THE EVERGREEN GREENHOUSES WITH RAIL SYSTEM	1
FIGURE 2: SOLIDWORKS ASSEMBLY OF COMPLETED CHASSIS WITH THE THREE AXES LABELLED.....	2
FIGURE 3: MECHANICAL CONFIGURATION OF STEPPER MOTORS AND MOUNTED CROP SENSE (BLUE BOX)	4
FIGURE 4: DRV8825 STEPPER DRIVER.....	4
FIGURE 5: RAIL SYSTEM AT VAN BELLE NURSERY	5
FIGURE 6: EXAMPLE OF FLANGED WHEELS USED IN TRAIN APPLICATIONS	5
FIGURE 7: EXAMPLE OF FLANGED WHEELS AVAILABLE FOR PURCHASE	6
FIGURE 8: TURNING THE WHEEL FROM STOCK ON THE LATHE.....	7
FIGURE 9: FLANGED WHEEL SOLIDWORKS MODEL	8
FIGURE 10: COMPLETED FLANGED WHEEL.....	8
FIGURE 11: SELECTED MOTOR - PBP45R68 DC PLANETARY GEARED MOTOR	8
FIGURE 12: EXAMPLE OF LONG RAILS, WHICH ARE NOT STRICTLY PARALLEL.....	9
FIGURE 13: SOLIDWORKS RENDER OF DC MOTOR CHAIN DRIVE	9
FIGURE 14: POLOLU G2 24V13 DC MOTOR DRIVER	10
FIGURE 15: ELECTRICAL SYSTEM OVERVIEW BLOCK DIAGRAM	11
FIGURE 16: PCB LAYOUT IN ALTIUM	12
FIGURE 17: COMPLETED PCB WITH DRIVERS AND CONNECTORS ATTACHED	12
FIGURE 18: WATERPROOF ELECTRICAL ENCLOSURE WITH PCB AND RPI INSIDE	13
FIGURE 19: EXAMPLE CONFIGURATION FILE WITH PARAMETERS FOR HORIZONTAL STEPPER MOTOR.	14
FIGURE 20: OVERVIEW OF WIRELESS SYSTEM	15
FIGURE 21: CODE MUST CONVERT FROM STEPS TO POSITION WHEN CONSIDERING MECHANICAL CHARACTERISTICS.....	16
FIGURE 22: D2VW-5L2-1HS OMRON WATERPROOF LIMIT SWITCH.....	17
FIGURE 23: FIRST REVISION OF USER INTERFACE	18
FIGURE 24: ENCODING DISK, PROXIMITY SENSORS OUTPUTS HIGH WHEN IN FRONT OF TAB AND LOW OTHERWISE	20
FIGURE 25: VOLTAGE DIVIDER CIRCUIT FOR ENCODER	20
FIGURE 26: HORIZONTAL AUTONOMOUS TRAVERSAL	26
FIGURE 27: RECOMMENDED USER INTERFACE FOR INPUTTING PATH LOCATIONS.....	28
FIGURE 28: PROXIMITY SENSOR (MCPIN-T18L-001) AND ENCODING RING	30
FIGURE 29: TOP DOWN VIEW OF CHASSIS WITH CHAIN DRIVE	31
FIGURE 30: CASTING MOLDS	32
FIGURE 31: CROSS SECTION OF ASSEMBLED FLANGED WHEEL.....	34
FIGURE 32: RECOMMENDED ROLLER CHAIN	35
FIGURE 33: RECOMMENDED ROLLER CHAIN SPROCKET.....	36
FIGURE 34: ENGINEERING DRAWING OF DRIVE SHAFT.....	37
FIGURE 35: ENGINEERING DRAWING OF RETAINING RINGS	38
FIGURE 36: ENGINEERING DRAWING OF FLANGED JOURNAL BUSHING	39
FIGURE 37: ENGINEERING DRAWING OF FLANGED WHEEL	40
FIGURE 38: ENGINEERING DRAWING OF MOTOR MOUNTING BRACKET	41
FIGURE 39: ENGINEERING DRAWING OF MOUNTING PLATE FOR ELECTRICAL ENCLOSURE	42
FIGURE 40: ENGINEERING DRAWING OF BRACKET ATTACHING CHASSIS SIDE OF ROLLER MEMBER	43
FIGURE 41: ENGINEERING DRAWING OF BRACKET ATTACHING ROLLER SIDE OF ROLLER MEMBER	44
FIGURE 42: ENGINEERING DRAWING OF ENCODING DISK.....	45
FIGURE 43: ENGINEERING DRAWING OF MOTOR ROLLER CHAIN SPROCKET	46

FIGURE 44: ENGINEERING DRAWING OF FLANGED WHEEL BRACKET	47
FIGURE 45: DC MOTOR SCHEMATIC	49
FIGURE 46: POWER INPUT CIRCUIT	51
FIGURE 47: POWER CONSUMPTION ESTIMATES	51
FIGURE 48: STEPPER DRIVER CIRCUIT	52
FIGURE 49: DC MOTOR DRIVER CIRCUIT	53
FIGURE 50: CURRENT LIMIT GRAPH FOR DC MOTOR DRIVER	53
FIGURE 51: LIMIT SWITCH INPUTS	54
FIGURE 52: TWO CHANNEL ADC CIRCUIT	54
FIGURE 53: DC MOTOR CURRENT SENSE SCALING CIRCUIT	55
FIGURE 54: BATTERY VOLTAGE SENSING VOLTAGE DIVIDER WITH LOW PASS FILTER	56

List of Tables

TABLE 1: SUMMARY OF DELIVERABLES	22
TABLE 2: FINANCIAL SUMMARY	24
TABLE 3: SUGGESTED WATERPROOF CONNECTORS FOR ELECTRICAL ENCLOSURE	29
TABLE 4: STEPPER MOTOR SPECIFICATION	48
TABLE 5: STEPPER MOTOR DRIVER SPECIFICATIONS	48
TABLE 6: DC MOTOR SPECIFICATIONS	49
TABLE 7: DC MOTOR DRIVER SPECIFICATIONS	49
TABLE 8: LOGIN INFORMATION	57
TABLE 9: INSTALLED PACKAGES	57

List of Equations

EQUATION 1: STEPS TO VERTICAL MOTION	16
EQUATION 2: STEPS TO HORIZONTAL MOTION	16
EQUATION 3: STEPPER DRIVE FREQUENCY	19
EQUATION 4: DC MOTOR PWM	19

List of Abbreviations

EIS	Ecoation Innovative Solutions Inc.
RPi	Raspberry Pi 3
GPIO	General Purpose Input Output
PCB	Printed Circuit Board
DC Motor	Direct Current Motor
NEMA	National Electrical Manufacturers Association
IP Rating	Ingress Protection Rating
OpAmp	Operational Amplifier
SPST Switch	Single Pole Single Throw Switch
SPI	Serial Peripheral Interface
IC	Integrated Circuit
PWM	Pulse Width Modulation
UDP Socket	User Datagram Protocol Socket
AP	Access Point
GUI	Graphical User Interface
JSON	JavaScript Object Notation
CAD	Canadian Dollars
USD	United States Dollars
USB	Universal Serial Bus

Introduction

Background

The sponsor, Ecoation Innovative Solutions Inc. (EIS), is a two-year-old agriculture/data science startup that have developed a proprietary system, Crop Sense. It is an automated plant monitoring and inspection system used to convey information about plant health before any visible symptoms arise.

Currently, EIS is working with Van Belle Nursery to help facilitate the growth and development of their evergreen saplings. The plants are housed in a 100-m long greenhouse that is subject to a wet, humid, and dirty environment. To expedite the scanning process of the numerous plants, EIS has developed a mechanical chassis that can traverse across an existing system of rails used for watering and transporting plants inside the nursery. The current system must be manually pushed along the rails and uses non-waterproof steppers for movement in the horizontal and vertical direction.



Figure 1: Inside one of the evergreen greenhouses with rail system

Problem Statement

The main objective of our project is to build a system that makes it simpler for an operator to move Crop Sense around the greenhouse. To this end, we aimed to have autonomous movement in all three dimensions, horizontal, vertical, and down the rails.

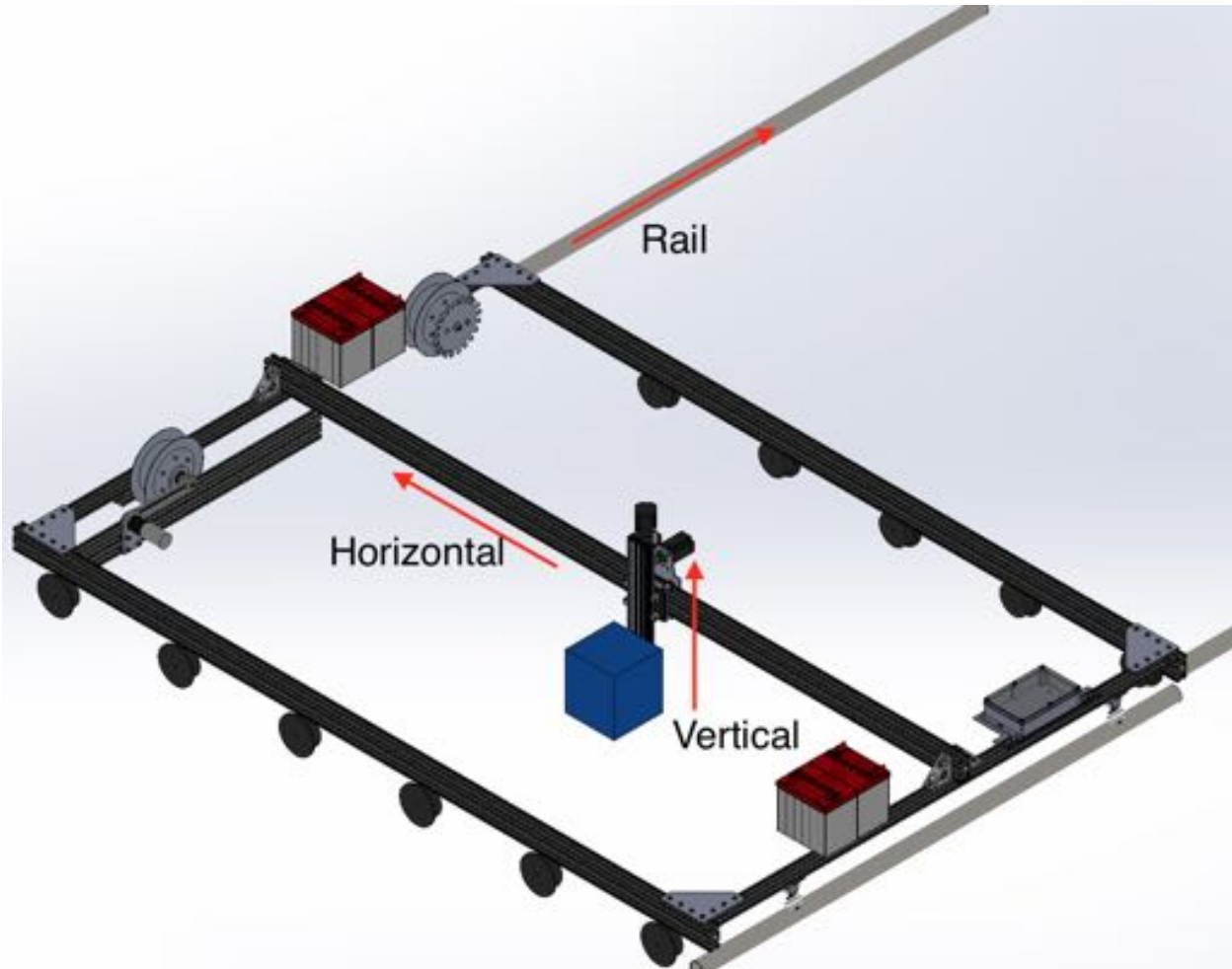


Figure 2: Solidworks assembly of completed chassis with the three axes labelled

Ultimately, the project aims to have a system capable of reliably and automatically acquiring data in field through manipulation of Crop Sense.

Scope and limitations

Our initial scope aimed to tackle all aspects of the stated problem, and produce a system with a long service life (>5 years) that could be easily modified. To achieve these goals, we set ourselves to have:

- ± 1 cm accuracy in movement in all 3 dimensions
- Ability to operate in the harsh environment of a greenhouse
- A system that can receive commands remotely and execute them automatically

After careful consideration and through the course of the project we arrived at a scaled back scope because of time constraints. We now are going to deliver a system that has:

- Fully waterproofed electronics
- Prototyped rail-drive, with untested accuracy
- Manual traversal in horizontal and vertical direction using joystick
- Receive commands remotely and can communicate with Crop Sense
- Web server with user interface to select traversal path in the horizontal direction

This change in scope allows us to still deliver a project that is useful to our sponsors even though it is not the full-scale solution that we had originally planned.

Discussion

Stepper Motion

Existing Mechanical Design

The existing stepper motor platform allowed us to focus entirely on the electrical aspects of the stepper motors. Our sponsors had a lead screw setup for motion in the vertical direction and a belt drive setup for motion in the horizontal direction. The system also has mounting points for stepper motors in the NEMA 23 configuration. This allowed us to select motors that matched our transmission requirement and fit the existing footprint. The bulk of the mounting was done through flexible elements to prevent issues with alignment.



Figure 3: Mechanical configuration of stepper motors and mounted Crop Sense (blue box)

Stepper Motors

The existing stepper motors used by the sponsor were not waterproof and could not tolerate the wet/humid conditions of a greenhouse for extended periods of time. New IP65 rated stepper motors are selected to replace the existing ones. We decided to go with the PrimoPal PHP57S82-410-IP65 stepper motor since it meets our required torque, frame size (NEMA 23), and waterproof rating at an affordable price. Appendix B outlines the key specifications of this stepper motor.

Stepper Driver

We selected the Pololu DRV8825 stepper motor driver (Figure 4) to drive our stepper motors. It was chosen for its compact size and ability to drive the required 1.0 A per phase for the selected stepper motors. Appendix B outlines the key specifications of the driver and Appendix C shows a schematic of the connections for the stepper driver on the PCB.



Figure 4: DRV8825 stepper driver

Rail Motion

One of our primary goals is to create a system that can be manipulated in all 3 dimensions. One of these dimensions is down the greenhouse rails. To accomplish this motion, we designed and fabricated an addition to the existing chassis.



Figure 5: Rail system at Van Belle Nursery

Mechanical Considerations

The primary mechanical challenges we must solve with the rail drive are: deflections along long rails, and a relatively large/heavy chassis only supported on rails. In effect, the chassis must be able to tolerate deflections on the rails but not fall off the rails in the process. A compromise that we came up with was constraining one side of the chassis while having the other side of the chassis be unconstrained. To do this we used flanged wheels on the side that was driven by the motor and rollers on the unconstrained side.

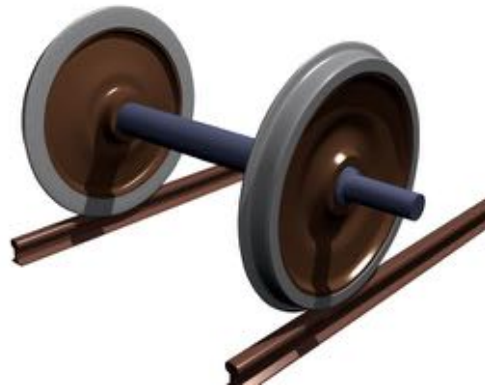
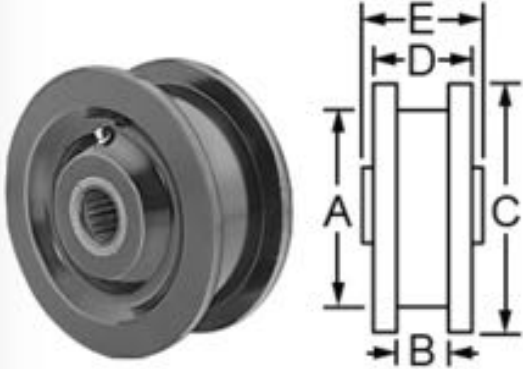


Figure 6: Example of flanged wheels used in train applications

To design the flanged wheels, we looked for a wheel type that allowed us to have a wheel width equal to the rail diameter. In this case that meant having a width of 2 inches. We first tried to simply order wheels that met this specification. Unfortunately, there were no off the shelf solutions that met our needs. There are many companies that sell wheels of this kind (McMaster Carr, Hamilton Castors, National Castors, etc.) but they all proved to be too expensive because this a part that is made to order and not stocked.

Double-Flanged Track Wheel
 6" x 2-1/8" Tread, for 1" Axle, 3200 lb Capacity



Each In stock
 \$171.16 Each
 22495T53

ADD TO ORDER

Wheel	
Diameter (A)	6"
Width (B)	2 1/8"
Flange Diameter (C)	7 5/16"
Overall Width (D)	3"
Axle Diameter	1"
Hub Length (E)	3 9/16"
Capacity Each	3,200 lbs.
Additional Specifications	Double-Flange Wheels

Figure 7: Example of flanged wheels available for purchase

Instead of ordering the wheel from these manufacturers we decided to take the base components of the wheel design and fabricate our own wheel. We chose to make the wheel out of untreated aluminum primarily because we got this material for free. We turned it down to our specifications of 2 inches wide with a 6-inch diameter.



Figure 8: Turning the wheel from stock on the lathe

Next, we made aluminum flanges, which were 8.5 inches in diameter. The diameter makes it so the chassis would have to move more than an inch off the rails to hop off the track. This is relatively unlikely given the configuration.

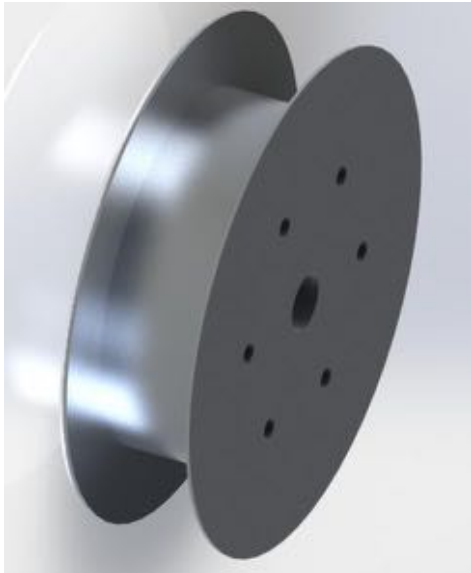


Figure 9: Flanged wheel Solidworks model



Figure 10: Completed flanged wheel

After the fabrication of the wheels, we attached the flanges by bolting them to the wheel. This was sufficient because the exact alignment of the flanges was not a concern for us.

Power Transmission

Once we had a wheel, we needed to select a motor to get it moving. This was done in the project proposal stage, and we selected a motor based on the torque and speed requirements of a 6-inch diameter wheel. We chose the PBP45R68 DC motor from PrimoPal, because its rated torque is greater than our required torque and the motor is able to provide that torque at an acceptable speed during long traverse. The long traverse consideration was something that came from the possibility of needing to move the chassis from the end of the rail back to the beginning without taking any data.

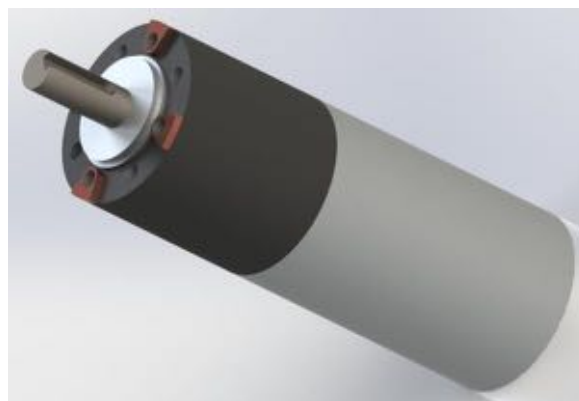


Figure 11: Selected Motor - PBP45R68 DC Planetary Geared Motor



Figure 12: Example of long rails, which are not strictly parallel

To simplify the electrical and mechanical systems we designed a single wheeled drive. This means that only one of the four wheels is providing power. This was determined to be acceptable by the sponsor given the stiffness of their chassis. To deliver power to the chassis there is a 1-1 chain drive going between the motor mount and the flanged wheel.



Figure 13: Solidworks render of DC motor chain drive

DC Motor Driver

We decided to use the G2 24v13 DC motor driver from Pololu (shown in Figure 13). It can operate at the desired 24V and can output up to 13A, which more than enough to drive the selected DC motor (4.4 A stall current). Technical specifications are shown in Appendix B and wiring schematics are shown in Appendix C.



Figure 14: Pololu G2 24v13 DC motor driver

Assembly

In order to put all the components together, we used brackets cut on the water jet to interface with the T-slotted 80/20 aluminum chassis provided. Further details of the mechanical interfaces can be found in Appendix A.

The shaft is held inside the flanged wheel with an interference fit between the flange and shaft. This is another inspiration taken from trains, as this is a common technique employed in the creation of "wheel sets". The other critical aspect is the attachment of the chain sprocket and the encoding wheel. Both objects are bolted onto the shafts. The shafts are tapped on the end to accept M-10 bolt, this allows for easy testing and replacement of those components.

Electrical Systems

Overview

The movement system is powered by a 24V lead acid battery pack (selected by sponsor). A switched mode DC/DC converter is used to provide a 5V line to power the RPi and low voltage sensors. Figure 14 shows a block diagram of the general electrical system and connections. A PCB is used to interface between the RPi, sensors, motors, and power supply in a compact and easy to connect format.

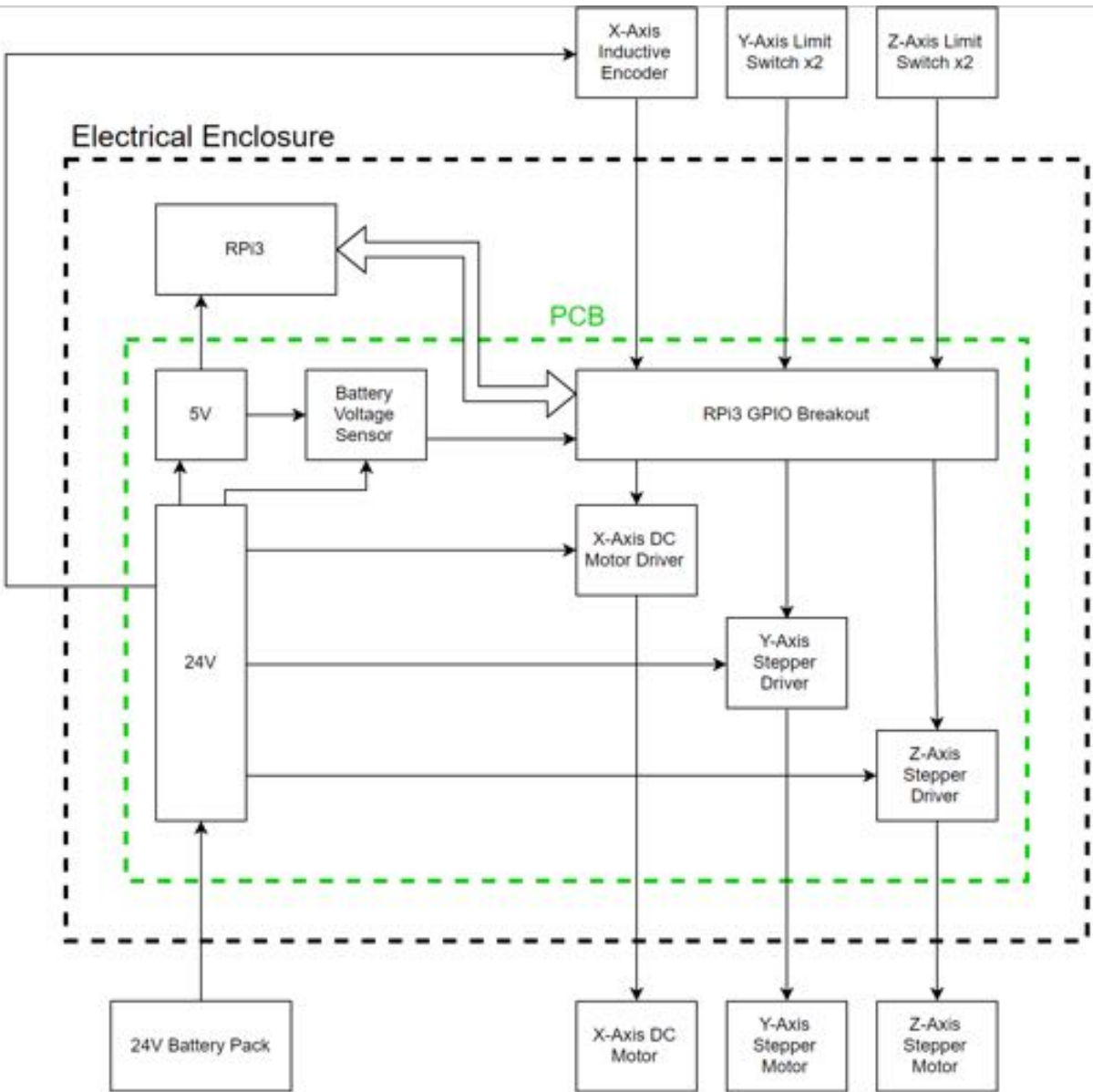


Figure 15: Electrical system overview block diagram

Printed Circuit Board

The PCB houses the drivers for the steppers and DC motor, filtering circuits for sensors, and provides overcurrent, overvoltage, and reverse polarity protection from the power supply. A 40-pin ribbon connector is used to connect to the GPIO pins of the RPi and Molex connectors are used to connect to the motors, sensors, and power supply. Figure 15 shows the board layout and Figure 16 shows the completed board with all the drivers and connectors plugged in. Detailed schematics of all the sub-circuits are shown in Appendix F and a detailed list of PCB components with item labels and Digikey links are provided in Appendix G.

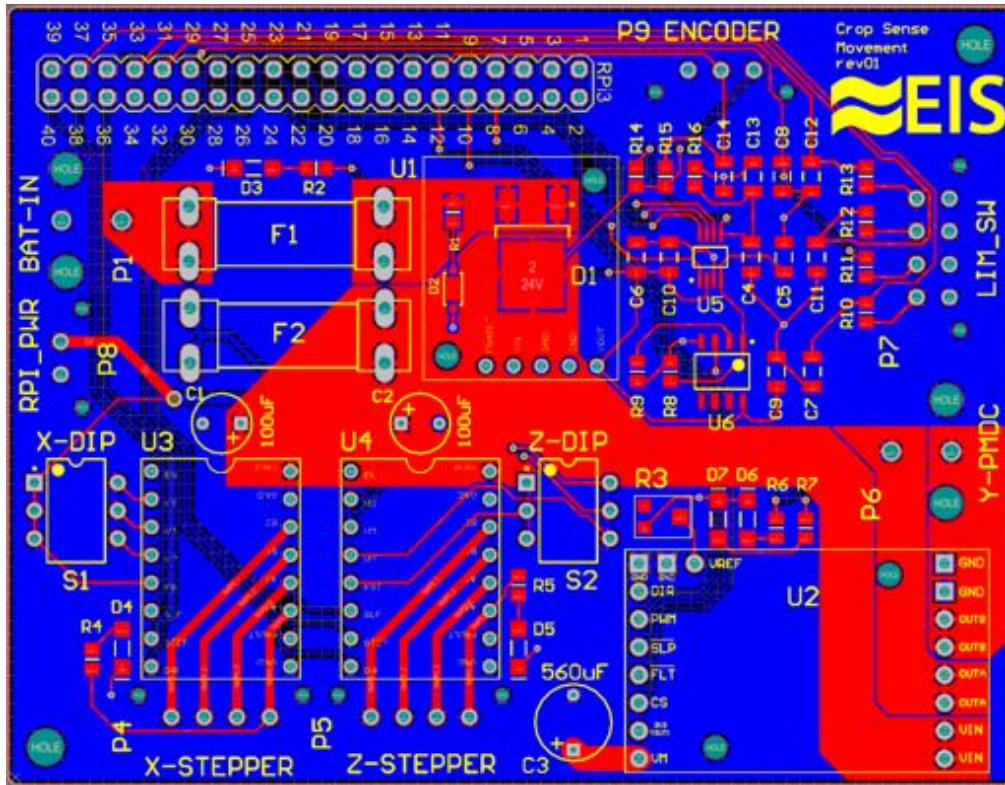


Figure 16: PCB layout in Altium

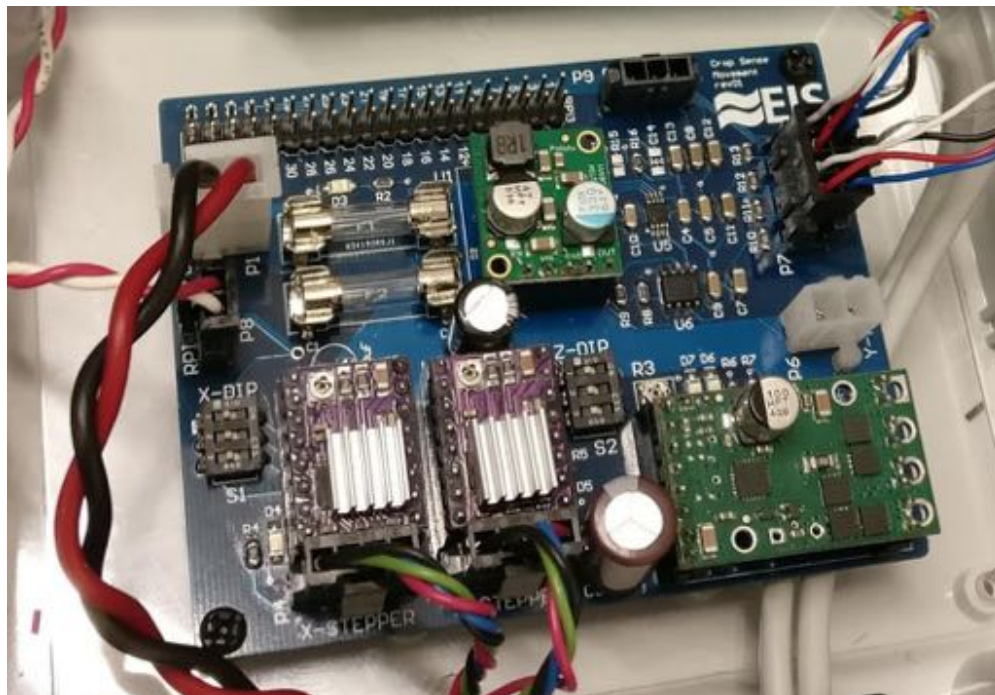


Figure 17: Completed PCB with drivers and connectors attached

Waterproofing

To keep all the electrical components dry, an IP65 rated electrical enclosure is used to house the RPi and PCB (shown in Figure 17). Waterproof cable glands were used for any wiring that needed to exit the enclosure (to plug into limit switches and motors). Additionally, the stepper motors and external sensors were specified to be IP65 rated or higher.



Figure 18: Waterproof electrical enclosure with PCB and RPi inside

Software Overview

The robot uses a RPi as the main controller for ease of use and wireless capabilities. Python was chosen as the primary programming language given the numerous available packages for extended functionality. The only disadvantage to using python was that it does not perform as well as C for timing critical applications. It was determined in the proposal stage that there are no timing critical applications that python cannot handle.

The entire architecture of the system is designed to be adaptable and modular. Using python modules, the developer can change the implementation of a function without requiring code to be written in other modules if the system conforms to the same requirements. The file structure used is outlined in Appendix E.

The following sections briefly discuss some of the smaller modules. Refer to Appendix E for code documentation.

GPIO

For GPIO interfacing, both WiringPi and RPi.GPIO python packages are used. The list below summarizes how each package is used:

- RPi.GPIO: pin input/output and interrupts
- WiringPi: hardware PWM

A custom wrapper module is used to add and improve functionality to the above packages. This implementation is also modular so that it prevents large scale code rewriting in the future.

Configuration

Many parameters defining the system are subject to change. To prevent the user from continuously modifying code, configuration files are used to contain system specific parameters. During startup, the configurations are loaded into the framework and the parameters initialized. A JSON format was chosen for the configuration because this file structure is easily parsed using python and it is commonly used in web applications.

```
"name": "stepper_x",
"params": {
  "x_min_pos": 0,
  "x_max_pos": 100,
  "left_dir": 0,
  "right_dir": 1,
  "x_micro_step": 1,
  "x_step_angle": 1.8,
  "x_npm_min": 125,
  "x_npm_max": 200,
  "x_npm_optimal": 150,
  "pitch_diameter": 19.1
},
```

Figure 19: Example configuration file with parameters for horizontal stepper motor.

Since the configuration parameters need to be accessed across multiple files and reading from a file is slow, “configuration modules” are used. Python modules are cached so once the module is imported we do not redundantly read from the files.

Wireless Communication

Crop Sense is responsible for collecting data from plants while a Raspberry Pi 3 microcontroller is responsible for movement and control. Crop Sense is required to wait for the RPi to communicate that a scan can be started. Once the scan is complete, Crop Sense sends a command to the RPi to inform it to continue traversing to the next scanning location. A simple UDP connection between the RPi and Crop Sense enables this master-slave style communication between the two systems.

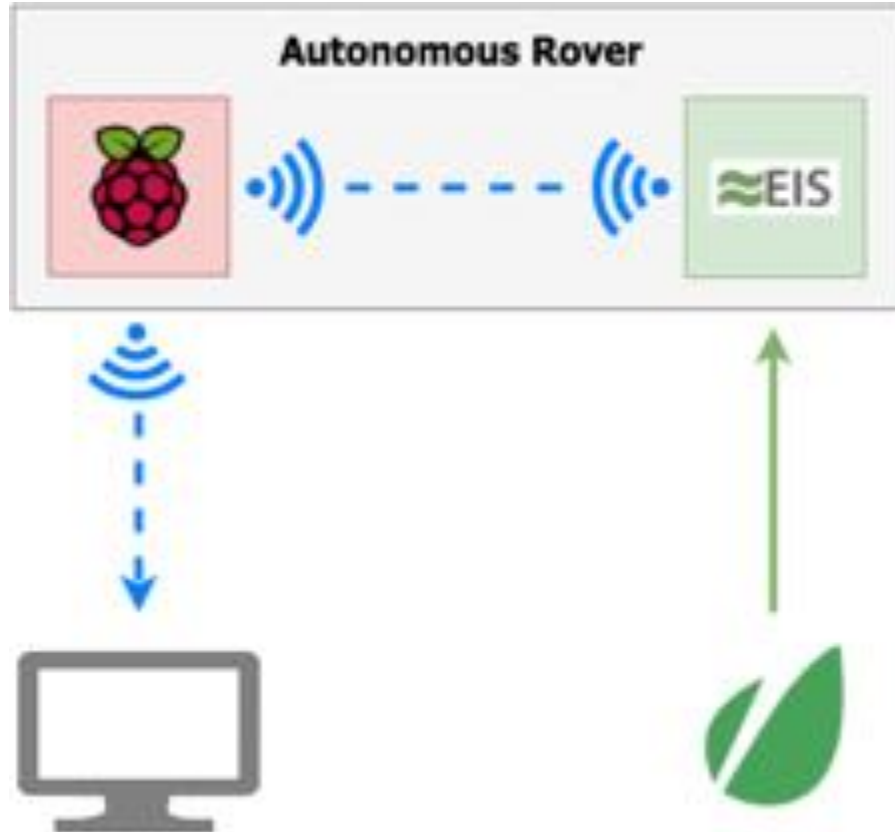


Figure 20: Overview of wireless system

During runtime, the user must be able to connect to the RPi and either access the console or in future implementations, connect to a web application. For the user to connect wirelessly, the RPi is configured in access point mode (AP). By enabling the RPi in AP mode the user can modify configuration files, run scripts and debug wirelessly. When in AP mode, the RPi is no longer able to connect to any other wireless networks, which inhibited development. We made scripts to switch the RPi between AP mode and client mode.

Position and Control

The expected use case for path traversal is as follows: the user supplies a location they want to traverse to (i.e. distance to scan location), and then the system moves to that location. In order to traverse to the scanning locations, the following is required:

1. Motor control
2. System level awareness of its current position
3. End of movement detection
4. Ability to set an absolute origin

The current location of the robot is stored in an external module and is updated inside the stepper and motor movement code. The position can easily be accessed from multiple points in the code.

Steppers

For the stepper motors, functions have been created so that a user can specify either the number of steps to take or the position to move to. The user is also required to specify the speed and direction. Movement in the vertical direction is characterized by the following equation.

Equation 1: Steps to Vertical Motion

$$steps = \frac{distance \times steps_per_revolution}{lead}$$

Movement in the horizontal direction is characterized by the following equation.

Equation 2: Steps to Horizontal Motion

$$steps = \frac{distance}{\pi \times pitch_diameter \times steps_per_rev}$$



Figure 21: Code must convert from steps to position when considering mechanical characteristics (such as a pulley or lead screw).

Once the number of steps has been determined, a square wave is generated from the RPi with a frequency proportional to the desired output speed. The optimal speed was determined using manual control and is elaborated on in the Testing section.

Limit Switches

The stepper motors also incorporate limit switches to prevent movement in unwanted areas which are triggered using interrupts. These limit switches are also used to define the origin of the robot in the vertical and horizontal direction; at initialization, the stepper motors are moved until they are in contact with the limit switches.



Figure 22: D2VW-5L2-1HS Omron Waterproof Limit Switch

Rail Drive

To control the DC motor, a PWM output must be sent from the RPi to the motor driver. Currently, there is functionality for controlling the motor by specifying the speed and direction. To determine the position traversed along the rail, an encoder must be incorporated. Once an encoder is installed, PID algorithms can be used to traverse to specified locations. Code has been written for both reading from an encoder and PID control but it has not been tested.

Joystick Control

In order to preserve the same functionality previously available to EIS, a joystick is configured to move the motors both wired and wirelessly. Using an external python package, Pygame, we were able to control the movement of the stepper motors in the horizontal and vertical direction using a joystick.

User Interface

A GUI hosted as a web application accessible on mobile devices is used to generate a path for horizontal traversal. We built the user interface using Node and Express (JavaScript, jQuery, HTML, CSS). The user can select which tray locations they want to scan and which direction they want to scan in.



Figure 23: First revision of user interface

Once the user selects a traversal direction, a JSON object is created and a python script is run containing the JSON object as a parameter. A simple path is then generated that contains the positions that the system will traverse to.

Testing

Motor Drivers

Stepper

To test the functionality of the DRV8825 stepper drivers, we set up an experiment on a protoboard. First, we had to set the current limiter built into the driver. Following the [documentation](#) we could properly limit the current to the allowed limit of our stepper motor. Once this was done, we experimented generating a square wave from the RPi. Once we could accurately create a square wave, we wired the stepper motor up to the driver. By increasing the frequency of the input wave, we can increase the speed that the motor produces steps. By specifying a speed, we can determine the frequency using the following equation.

Equation 3: Stepper Drive Frequency

$$f_{\text{step}} (\mu\text{steps / second}) = \frac{v \left(\frac{\text{rotations}}{\text{minute}} \right) \times 360 \left(\frac{\text{degrees}}{\text{rotation}} \right) \times n_m \left(\frac{\mu\text{steps}}{\text{step}} \right)}{60 \left(\frac{\text{seconds}}{\text{minute}} \right) \times \theta_{\text{step}} \left(\frac{\text{degrees}}{\text{step}} \right)}$$

Testing with the steppers, we determined that setting a speed of 150 RPM produced the most optimal stepper movement. We also found during our testing that the output frequency wave produced by the RPi was generally 10% less the expected. We decided this was not a concern during testing.

DC Motor

To use the motor driver, we tested the PWM output of the RPi. The motor driver contains a range of acceptable input frequencies between 20-100 kHz. We found that there was a trade-off between the duty cycle range and possible frequency set given in the RPi documentation

Equation 4: DC Motor PWM

$$pwmFreq = \frac{19.2e6}{pwmClock \times pwmRange}$$

By setting the clock divisor to 4 and the range to 100 we could achieve a 50kHz signal, the optimal frequency for our motor driver.

Proximity Sensor / Encoder

Using a readily available proximity sensor, we machined a circular encoding wheel (Figure 22) to test measuring position along the rails.



Figure 24: Encoding disk, proximity sensors outputs high when in front of tab and low otherwise

We realized that the output voltage was too high for the RPi once we had tested the encoder. We created the following voltage divider to set the output logic level of the encoder to around 3V.

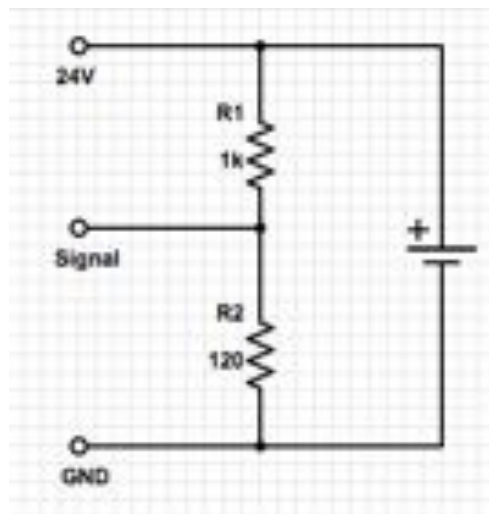


Figure 25: Voltage divider circuit for encoder

Interrupts are used to measure the number of encoder ticks. We found that when moving too fast the encoder would measure high all the time but when moving slow enough we could measure position with decent accuracy. We decided the best plan of action would be to buy a proximity sensor that could measure at a faster frequency.

Wireless Joystick

To test wirelessly sending commands to the RPi, the joystick was set up to communicate wirelessly by attaching it to a laptop and connecting to the Wi-Fi hosted by the RPi. We found during testing that there was a latency issue with communication causing the stepper to “jam” every once in a while. When testing

using a wired connection, we found that movement was extremely smooth. As a result, we decided to use a wired joystick.

Waterproofing

During our first day of testing in the field we noticed immediate condensation on the lid of the electrical enclosure. This observation confirms that the environment is very humid and that the electrical enclosure will need to be well sealed. The newly selected IP65 rated stepper motors seemed to work normally in the humid environment.

Conclusions

Holistically, our solution is capable of delivering on the goal of autonomous $\pm 1\text{cm}$ motion inside a greenhouse, but based on our current level of testing we can conclude that further improvements are needed to get the system to meet the goal. There are several aspects of the design and execution that we think work well and equally as many that require improvement.

The motion in the horizontal and vertical direction works well and can be controlled manually through a joystick. The motion is smooth and requires no physical interventions. This system fully meets the given design requirements.

The rail drive has the greatest room for improvement. The motor can drive the wheel despite its size and weight. As currently built, the wheels raise the chassis too high. This interferes with supports inside the greenhouse. A redesign of the mounting of the wheels was carried out and now these improvements must be fabricated. Additionally, the wheels could be made from a different material so as to have a greater amount of friction and be lighter. The rail drive is also missing a method of getting feedback on its current position. Simply counting the revolutions in software is not accurate enough so an encoding disk must be attached to the passive wheel. Lastly, we determined the motor should not be mounted in a direct drive configuration, and we switched to 1:1 chain drive as an alternate method of power transmission.

The software deliverables are incomplete and there are aspects of the autonomous control that can be improved. The system can currently move manually with a joystick, wireless communicate and be configured using files as required. Movement along the rails is not tuned largely because of a lack of testing time, so all the controller gains must still be found and inputted. The autonomous portion of the project proved too complex and thus only our recommendations and a bare bones implementation are being sent to the project sponsor.

The electrical systems of the system work well and are all waterproof. The components had no trouble when tested in the wet and humid greenhouse. Making components waterproof, proved to be a significant challenge. There are minor improvements that could be made to improve the efficiency of electrical systems detailed in our recommendations. This part of our project is well tested.

Generally, there is a plan to improve all the aspects of the project that fall short. The largest issues that came up during our project originated from a lack of testing. These come from either testing being too burdensome to take on, or testing requiring other parts of the project being completed. The lack of ability to work asynchronously and independently exacerbated the problems presented by having a very large test chassis, designed for a very specific environment, located far away.

Project Deliverables

List of Deliverables

Table 1: Summary of Deliverables

1	Original Deliverable	Autonomous control and movement of EIS Crop-Sense in 3 axes.
	New Deliverable	<ul style="list-style-type: none"> • Semi-autonomous movement of EIS Crop-Sense in horizontal direction with manual movement in vertical and rail direction. • Mechanical schematics and electrical infrastructure to move along 3rd axis (rails).
	Reason for Change	Retrofitting the 2 existing directions of movement to be waterproof and designing the rail drive took longer than expected. The rail drive also required a late redesign which blocked testing along the rails.
	Current State	<ul style="list-style-type: none"> • Semi-autonomous and manual movement in the two pre-existing directions of movement (horizontal and vertical) with upgraded waterproof parts. • User can use a wired joystick to move the stepper motors to the start position of the row of plants and then press a start button to autonomously scan a row of plants. • User needs to manually push the chassis along the rails on the pre-existing roller wheels between rows of plants. • The full mechanical design of the rail drive is complete and the wheels have been manufactured. Brackets need to be waterjet cut, chain sprockets need to be purchased or cut and the drive needs to be assembled. • The electrical system has the infrastructure in place (motor driver, connectors, DC motor) to incorporate rail drive.
2	Original Deliverable	Wirelessly communicate with EIS Crop Sense via Raspberry Pi 3 to send and receive scan and move commands.
	Current State	Can currently communicate wirelessly with EIS Crop Sense to send and receive commands.

3	Original Deliverable	Travel down plant rows with an accuracy of ± 1 cm.
	Current State	<ul style="list-style-type: none"> ● Open loop position tracking in the horizontal and vertical directions with the stepper motors show a reasonably high degree of positional accuracy. ● Official testing of accuracy was not conducted due to lack of time. ● Positional tracking along rails was not achieved due to delays in mechanical design, however electrical and (untested) software infrastructure exists for future testing.
4	Original Deliverable	Program a path of movement depending on the greenhouse/farm setup.
	New Deliverable	Program scan locations for a single row of plants using graphical user interface via a web app.
	Reason for Change	Unexpected complexity of task and lack of time due to delays in design. Needed to reduce the scope to something more manageable.
	Current State	Program scan locations for a single row of plants using graphical user interface via a web app.
5	Original Deliverable	Endure a humid/wet environment.
	Current State	<p>All electrical installations on the chassis are waterproof. Pre-existing motors and electrical system were not designed to endure humid/wet environment therefore the following steps were taken to meet this deliverable:</p> <ul style="list-style-type: none"> ● Electrical system was designed to fit inside an IP65 rated waterproof enclosure. ● Waterproof cable glands, a waterproof USB port, and IP65 rated on off switch are used with the electrical enclosure to keep wiring connections waterproof. ● Additional waterproofing silicon can be applied to joints if need. ● All new sensors (mechanical limit switches and inductive proximity switch used for encoding) are IP65 or IP67 rated. ● New waterproof IP65 rated stepper motors were sourced, installed and integrated with new electrical system.

Financial Summary

The costs displayed in this table include taxes. All amounts are listed in Canadian Dollars (CAD).

Table 2: Financial Summary

Part	Unit Price	Quantity	Cost (CAD)	Purchaser
RPI3	54.00	1	54.00	Akshiv
DC Motor (PBP45R68)	67.00	1	67.00	EIS
Stepper Motors (PHP57S82-410-IP65)	67.00	2	135.00	EIS
Freight and Transaction Charges on Motors	-	-	67.00	EIS
Waterjet Time	1.15	~60	69.00	Project Lab
Aluminum/Steel Sheet Stock	?	-	-	EIS/Project Lab
PCB	9.53	5 (min. order)	47.65	Arjun
PCB Components (Appendix C)	-	-	44.85	Project Lab
Aluminum Billet Stock	?	1	0.00	PHAS Machine Shop (Donation)
Journal Bushings (6338K436)	2.90	4	11.60	Project Lab
Fasteners (Various)	-	-	-	Project Lab
Stepper Motor Driver	12.88	3	38.64	Bryden
DC Motor Driver	60.93	1	60.93	Bryden
DC-DC Converter	22.4	1	22.4	Bryden
Microsoft Sidewinder Joystick	~30.00	1	0.00	Akshiv
Wires	~30.00	-	30.00	Project Lab
Limit Switches	12.21	4	48.83	EIS
Waterproof Enclosure	46.30	1	46.30	EIS

Cable Glands	2.90	3	8.70	EIS
TOTAL			\$ 697.90	

Ongoing Commitments by Team Members

Post Exam Implementation of Manual Rail Drive

Akshiv after April 24th and before April 28th

The idea with this is to get a functioning rail drive delivered that can be installed and tested at Van Belle by EIS. The critical remaining tasks are related to the shaft, because the sponsors have the ability to do the bulk of the remainder fabrication as detailed in our recommendation, the shaft is the place where we can have the greatest value add for the least amount of work.

- Press-fit new shaft into flange - Machining: Hydraulic press
- New drive shaft - Machining: Lathe
 - Grooves for retaining rings
 - Drill and tap both shafts

Autonomous Software and Documentation

Bryden and Arjun will visit the Van Belle on April 13th/14th to complete the following tasks.

Limit Switches, Setting Origin and Position Tracking

Testing the limit switches and make sure it is possible to set the origin using the limit switches. If setting the origin works as desired then position tracking should be a given but will be tested to confirm.

User Interface Upgrades

The user interface will need to include an option to use manual control (run the previously created python script) as well as a “kill switch” button to stop the program in case of an emergency. These elements will be added to the user interface. The only current unknown is how to implement the kill switch on the python side but the sponsors have already previously implemented this and will be supporting development.

Autonomous Horizontal Traversal

Using the path generated by the user interface, a simple control loop will be constructed to support the autonomous behavior seen in Figure 24. The sponsor has requested a simple autonomous traversal in the

horizontal direction where the system will move across and stop at locations specified from the user interface.

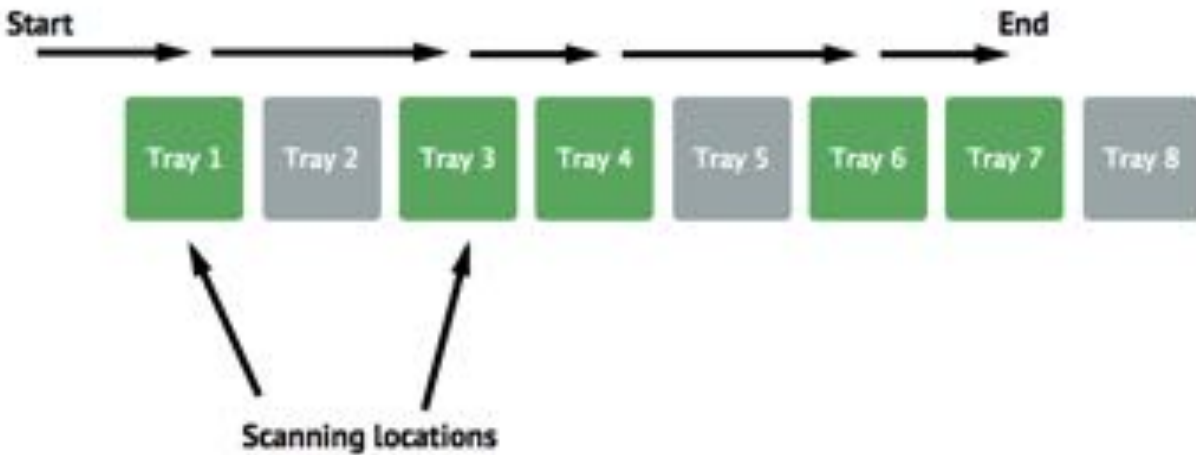


Figure 26: Horizontal autonomous traversal

Recommendations

As mentioned previously, delays in mechanical fabrication and electrical design has blocked development and testing of many software aspects of the robot. A part of these recommendations includes portions of software that have been implemented but not tested.

Encoding Along Rail Direction and PID control

The software for encoding using interrupts has been tested on a setup bench but not in the field. Whether the encoder is suitable for actual implementation must further be researched. The system should be tested to determine if any step counts are missed and the encoder can readily calculate the position the system has travelled.

Once the encoder functionality has been proven, PID control can be incorporated into the system. Currently, there is a PID function in "motor_y.py" that can be used as a base for future implementations including PID control. The PID control has not yet been tested and would be required by any future implementers. Expected issues with the PID control include the following:

1. Integral gain will grow too fast. As a result, an "I_threshold" and "I_max" parameter have been incorporated into the PID control. The integral threshold will prevent the integral gain from having any effect until the system is within the position specified. The max integral parameter will prevent the integral gain from growing too large causing large amounts of overshoot.
2. The large momentum of the chassis will cause difficulty with tuning. A large enough time threshold for how long the chassis should be within the target area should be included to prevent premature triggering of completion.

3. Tuning will be slow. A PID tuner example file has been created to help demonstrate how tuning can quickly be performed.

Debugging and Data Collection

All testing has currently been successful using examples files provided within the software package. When implementation becomes more complicated and intricate, the simple example files may not be robust enough. Creating a simple debug flag to print or save to file variable values may come in handy. When testing and tuning PID it is recommended to explore modifying the original source code to include more verbose debugging information.

UI Interface Expansion

The following features could be implemented into the user interface:

1. Motor Control
2. Feature to input full autonomous path.
3. Statistics and debugging information (i.e. battery level, run time of last scan, scans completed, etc....)

The above features can be incorporated using the existing architecture if required.

Autonomous Traversal

In order for the system to be fully autonomous, the system must be able to generate a path based on user input. A control loop incorporating the wireless communication, PID control and the generated path must then be completed. Prospective algorithms for path generation and path traversal are given below.

Suggested algorithm for path generation:

1. User inputs the start, end, scanning and obstacle locations. The user can input the data via the user interface or JSON configuration file. An example can be seen in Figure 25 where the user can place the scanning locations on the grid system. No matter the
2. The algorithm assigns preference of traversal based on the following priority:
 - a. Position from start location
 - b. Same position along rails.
 - c. Distance from previous node.
3. Ordered nodes placed in queue.

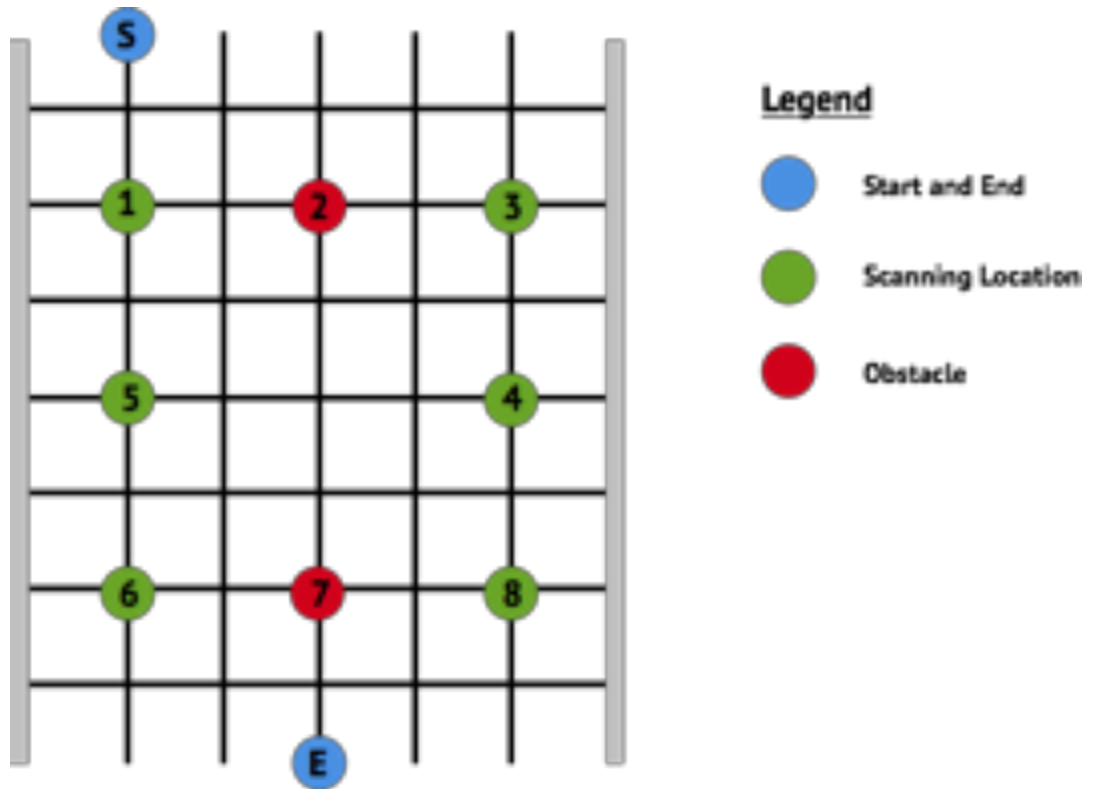


Figure 27: Recommended user interface for inputting path locations. Numbers represent traversal number after path generation

Next is a suggested implementation for the path traversal:

1. Generate path and initialize system. Enter control loop.
2. Dequeue item from path, get node information.
3. Retrieve location and pass parameters to motor control code.
 - a. If obstacle, set location to avoid obstacle before traversing.
4. Verify position has been obtained.
 - a. If scanning location send wireless command to Crop Sense and wait for response
5. Repeat step 2-4 until path is empty.
6. Return to start location.

The above should provide a basic outline for writing the autonomous control loop and path generation algorithms. A separate module has been created called traversal which can contain the path generation and Node objects (an example of an implementation of a node object is contained in the source code).

Waterproof Connectors for Electrical Enclosure

Currently the wiring harnesses for motors and limit switches exit the electrical enclosure via waterproof cable glands. First, these cable glands are not completely watertight, and may allow for humid air to enter the electrical enclosure through tiny gaps between the rubber gasket the wire bundles. These glands were selected for this project because of their low cost but they are inherently not a good solution for long

term waterproofing. These cable glands also make the wiring harnesses permanently attached to the electrical enclosure (unless the glands are removed as well).

An alternate solution to the cable glands are waterproof circular pin connectors such as the TE AMP Connectors. Table X shows a suggested set of plugs, receptacles, gaskets and boots.

Table 3: Suggested Waterproof Connectors for Electrical Enclosure

Name	Qty	Unit Price (CAD)	Extended Price (CAD)
Receptacle	2	\$4.80	\$9.60
Plug	2	\$7.85	\$15.70
Boot	2	\$13.77	\$27.54
Gasket	2	\$1.90	\$3.80
		TOTAL	\$56.64

Improvements to PCB

1. The spacing between some of the plug-in components (motor drivers primarily) was tight, making it difficult to solder in large capacitors and place fuses. The second revision of the board should have greater clearance for the larger components.
2. The inductive proximity switch used for encoding requires a 24V supply and therefore outputs a 24V logic level. The board does not have any logic level shifting to step the 24V signal down to 3.3V therefore an external resistor voltage divider will need to be soldered to the wires connecting to the board from this sensor. The second revision of the board should include a level shifting circuit to accommodate this.
3. Revision 1 of the board only has two mounting holes at diagonally opposite corners. This resulted in unwanted flexing of the board when plugging in drivers and connectors. The second revision should include mounting holes on all 4 corners to resolve this.

Rail Drive Improvements

The design of the rail drive is significantly improved from the one initially designed and tested. Unfortunately, due to time constraints we were unable to build these new additions and improvements. The best way forward is to fabricate, assemble and test the rail-drive.

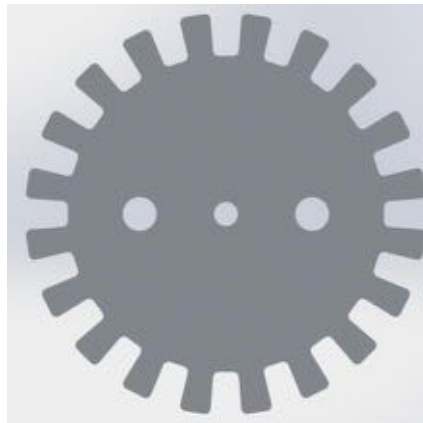
In terms fabrication of the improved rail drive components, these specific actions must be taken before Akshiv's ongoing commitments can be fulfilled. Once these tasks are complete the rail drive can be assembled and installed.

- 2 new flanges - Waterjetting
 - Attaching to existing wheel with minimal modification to the wheel
 - Reuse existing mounting holes in wheel
- 4 wheel brackets - Waterjetting
 - Press-fitting journal bushing into mounts
- Motor bracket - Waterjetting
- 2 sprockets - Waterjetting
- Create chain to connect sprockets - Machining: Hand Tools

The primary improvement would be the encoding system for the rail drive. This includes both the purchasing of a IP rated encoder and the mounting of the encoding disk. We recommend using the [MCPIN-T18L-001](#) inductive proximity switch. It is a IP67 rated component and it can operate at a frequency of up to 700 Hz, which is significantly higher than the 160Hz required for 1 cm precision. We have a rotary encoding disk that we fabricated for testing but perhaps a different one will be required once the sensor is correctly mounted.



Figure 28: Proximity Sensor ([MCPIN-T18L-001](#)) and encoding ring



The other set improvements are related to the assembly and incorporation of the rail drive into the chassis. Since we will not be able to incorporate the rail drive into the chassis, there will need to be work done to assemble the final solution. The work here is slightly more nebulous, but primarily consists of assembling the following together and incorporating them into the chassis: sprockets, encoding disk, extra 80/20 members, new flanged wheel, and DC motor. These components involve no additional design but the final assembly must be performed by a different group, using our SolidWorks models as a guide.



Figure 29: Top down view of chassis with chain drive

To improve the rail drive at a deeper level we recommend changing the flanged wheels. The primary concern is the weight of the wheel and the friction between the wheel and the rails, given that it is made from aluminum.

If the implementation is going to stay at the small scale, we suggest going with a custom molded rubber wheel. A cast can be made by 3D printing components of the cast and assembling them. With this cast pour in rubber compound and allow it to set. Once you have the wheel, drill out the centre shaft hole and finish the edge on the lathe. Lastly, continue to use aluminum flanges, and bolt the flanges on the rubber wheel. The wheel will provide greater friction and is significantly lighter than aluminum. This change will be especially important if the motion down the rail is not sufficiently smooth in our current transmission configuration.

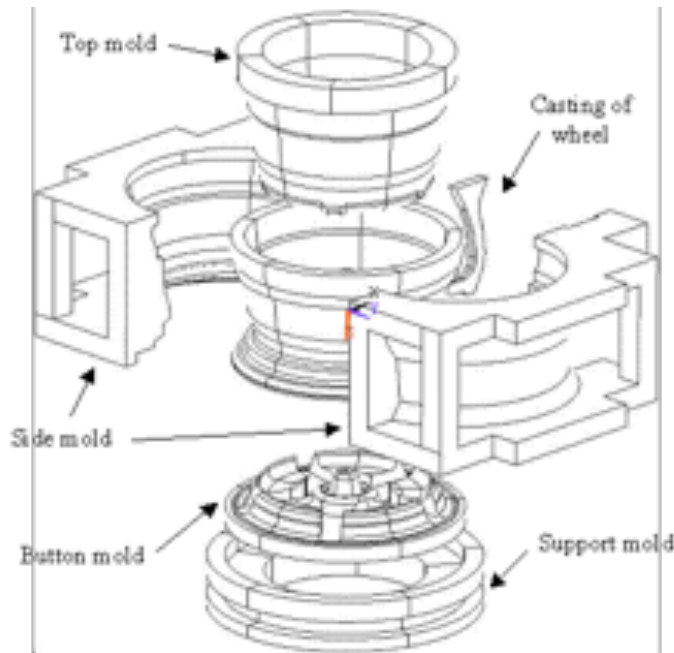


Figure 30: Casting Molds - "Computer Simulation of Casting Process of Aluminum Wheels – A Case Study" Proceedings of the IMECHE Part B Journal of Engineering Manufacture, Vol. 220, No. 2, February, 2006, p. 203-211

If the production of this solution needs to scale up, we recommend outsourcing the part to third party. McMaster Carr has the wheel that is closest to our design, but National Castors and Hamilton Castors are both cheaper especially if purchasing in bulk. All of these manufacturers make wheels are sufficient for our purposes.

Other Mechanical Improvements

A substantial change is needed in the limit switch mounting plates. This must be redesigned to allow for easier installation and adjustment. The horizontal limit switches are designed correctly, they simply need to be recut with aluminum instead of steel to prevent rusting. The vertical limit switch design needs to be changed such that the limit switches are mounted in place at the limits and the bumpers move with the chassis and run into the stationary switches arresting movement.

Many of the attachments made to the 80/20 were done using regular nuts rather than t-nuts. A way of making the chassis more adaptable is to replace our conventional nuts with t-nuts. Also, in places where normal nuts remain, there needs to either be Loctite added or the nuts replaced with nylock nuts. Currently, there are too many connection at risk of falling out because of nuts coming loose from mechanical vibrations.

Lastly, the DC motor is not currently waterproof. While there are no sensitive electronics inside and the motor will work fine in water, it is better for a longer service life to have a waterproofed motor. The coils are at risk of corrosion, if the motor stays wet for long periods of time. The simplest solution to waterproofing, is to fill the motor cavity with oil. This way there is no room for water to make its way

inside the motor coils. This is acceptable because the motor will not be getting so hot as to become damaged by insufficient ventilation. Another potential solution is to coat the motor coils with a water repellent. This avoids the issue of overheating because air can still pass over the coils but it is a more expensive option. Lastly, one could also seal the motor using silicon. This is simplest solution, but it has an even greater risk of overheating, although this is likely tolerable in our application.

Appendix A - Mechanical Drawings

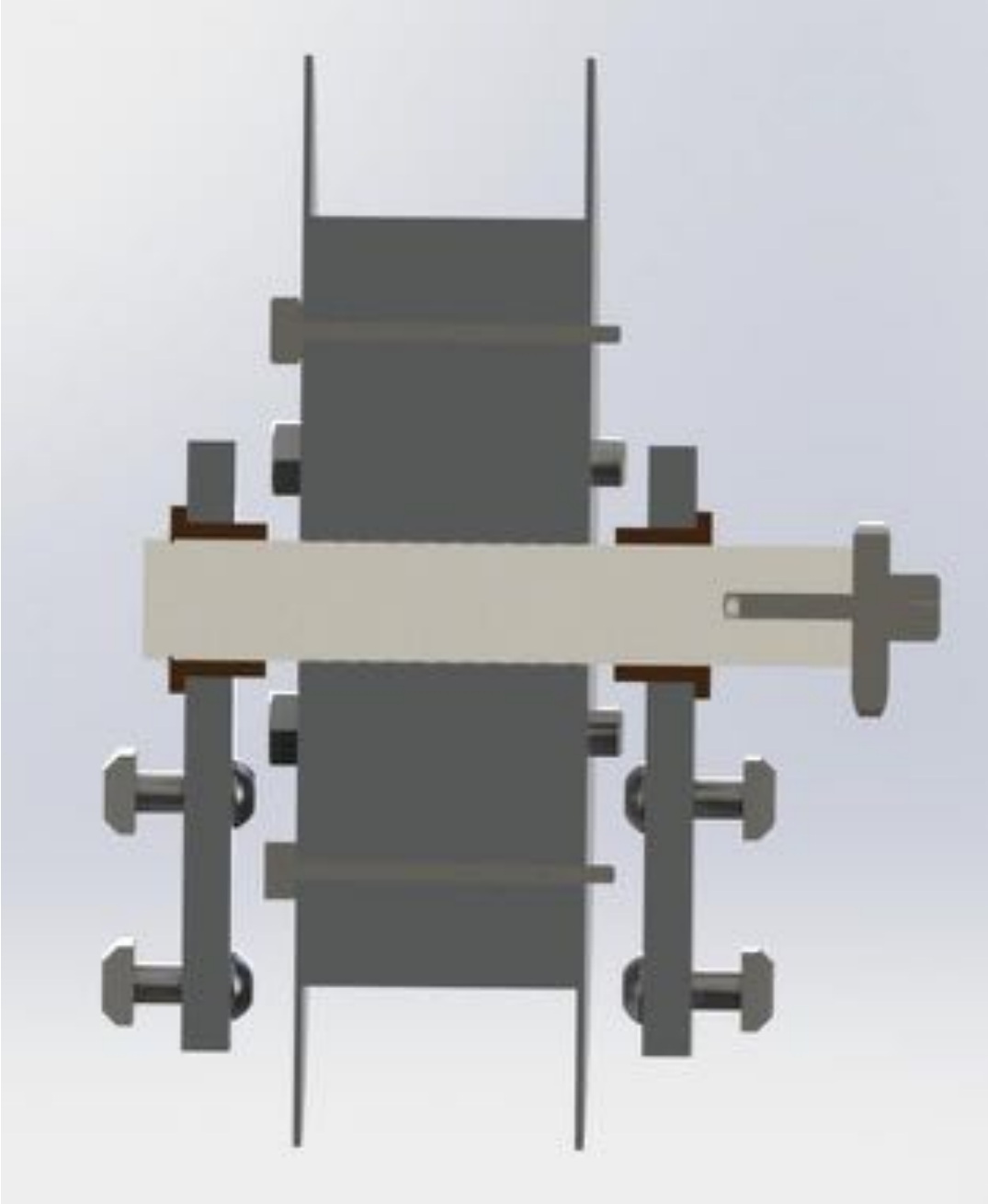


Figure 31: Cross section of Assembled Flanged Wheel

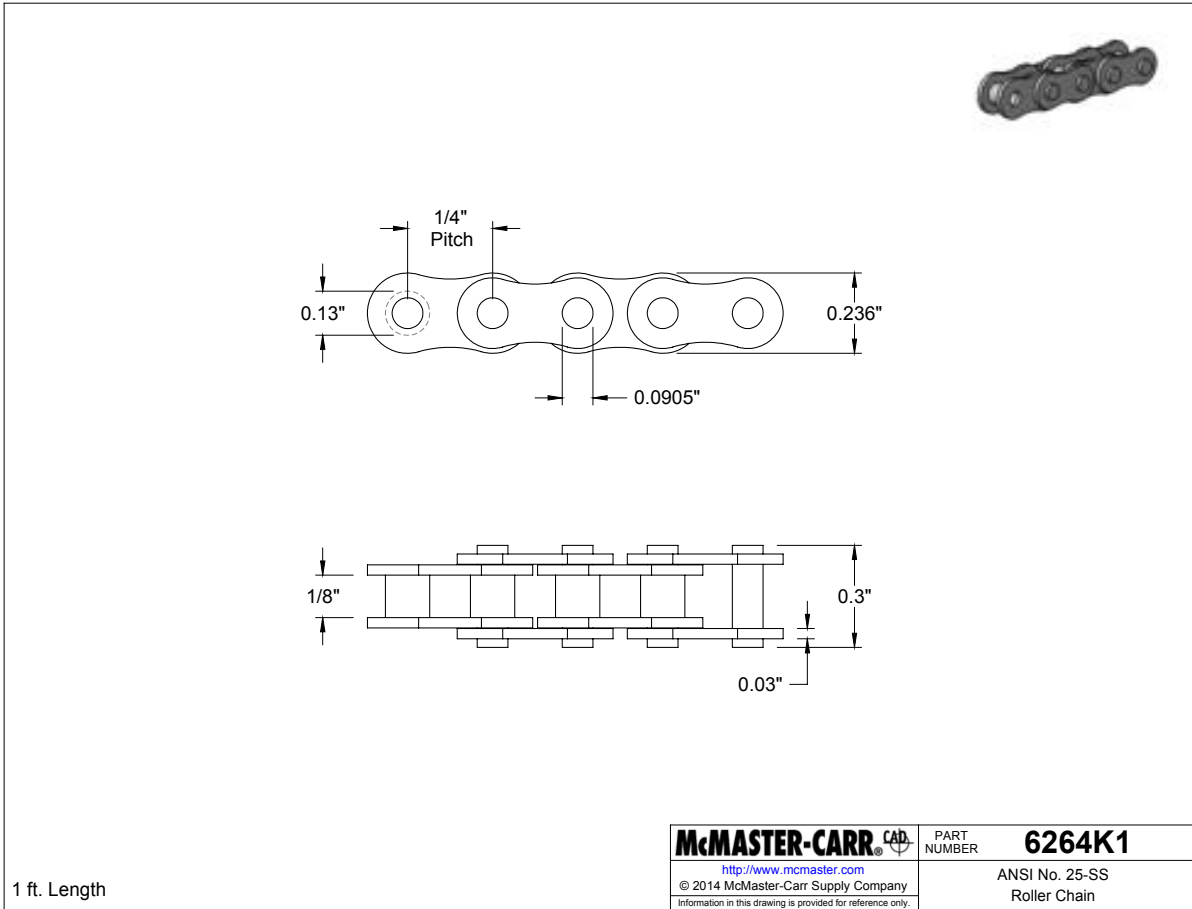


Figure 32: Recommended Roller Chain

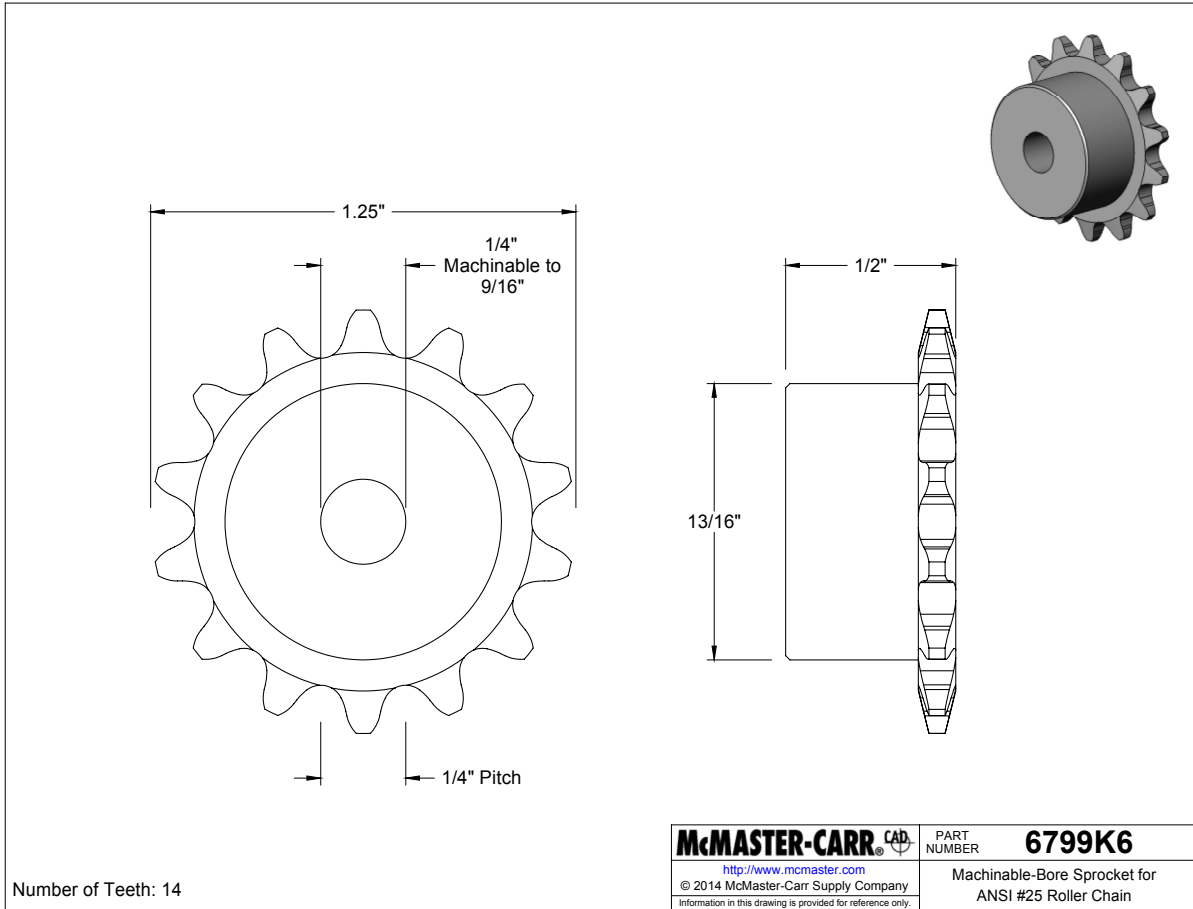


Figure 33: Recommended Roller Chain Sprocket

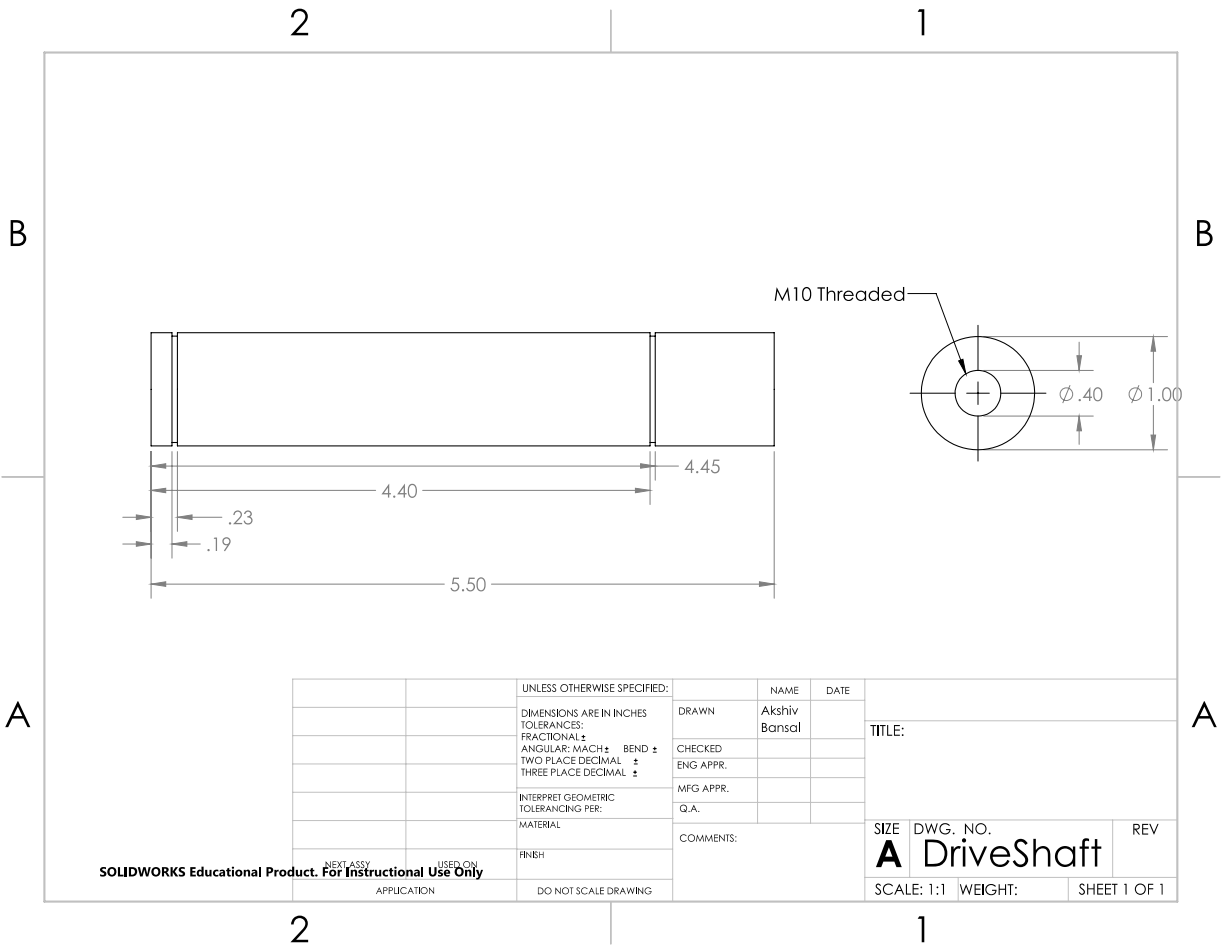


Figure 34: Engineering Drawing of Drive Shaft

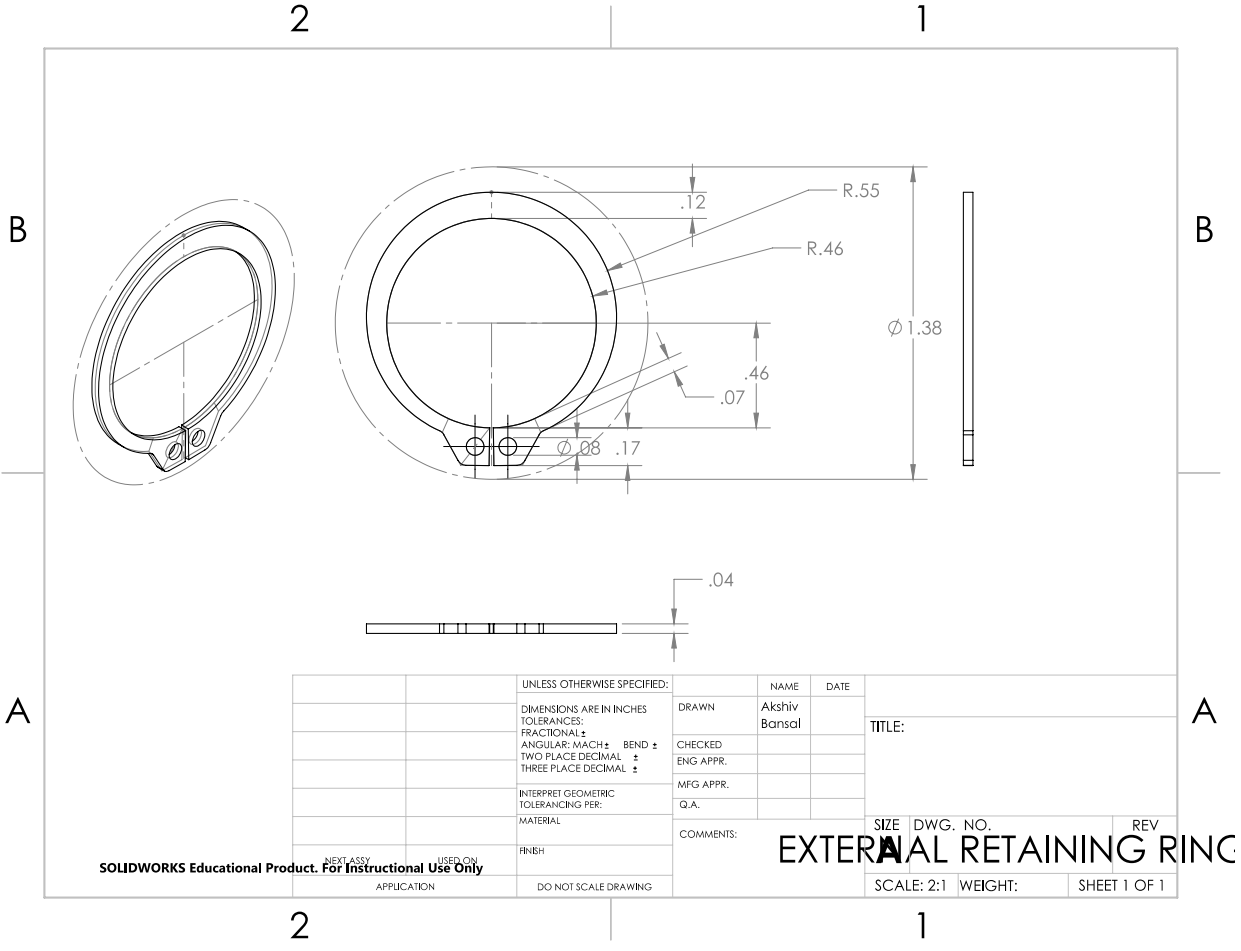


Figure 35: Engineering Drawing of Retaining Rings

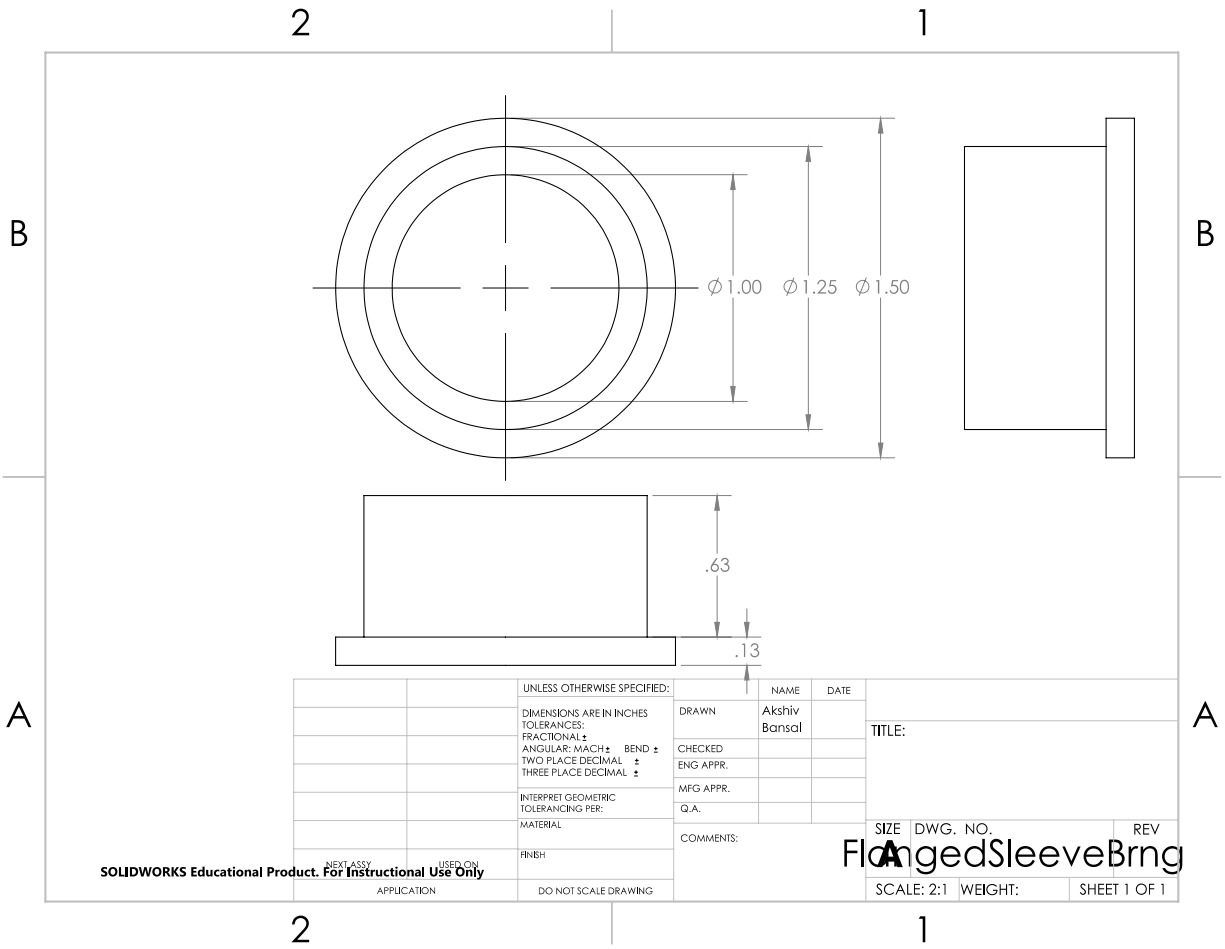


Figure 36: Engineering Drawing of Flanged Journal Bushing

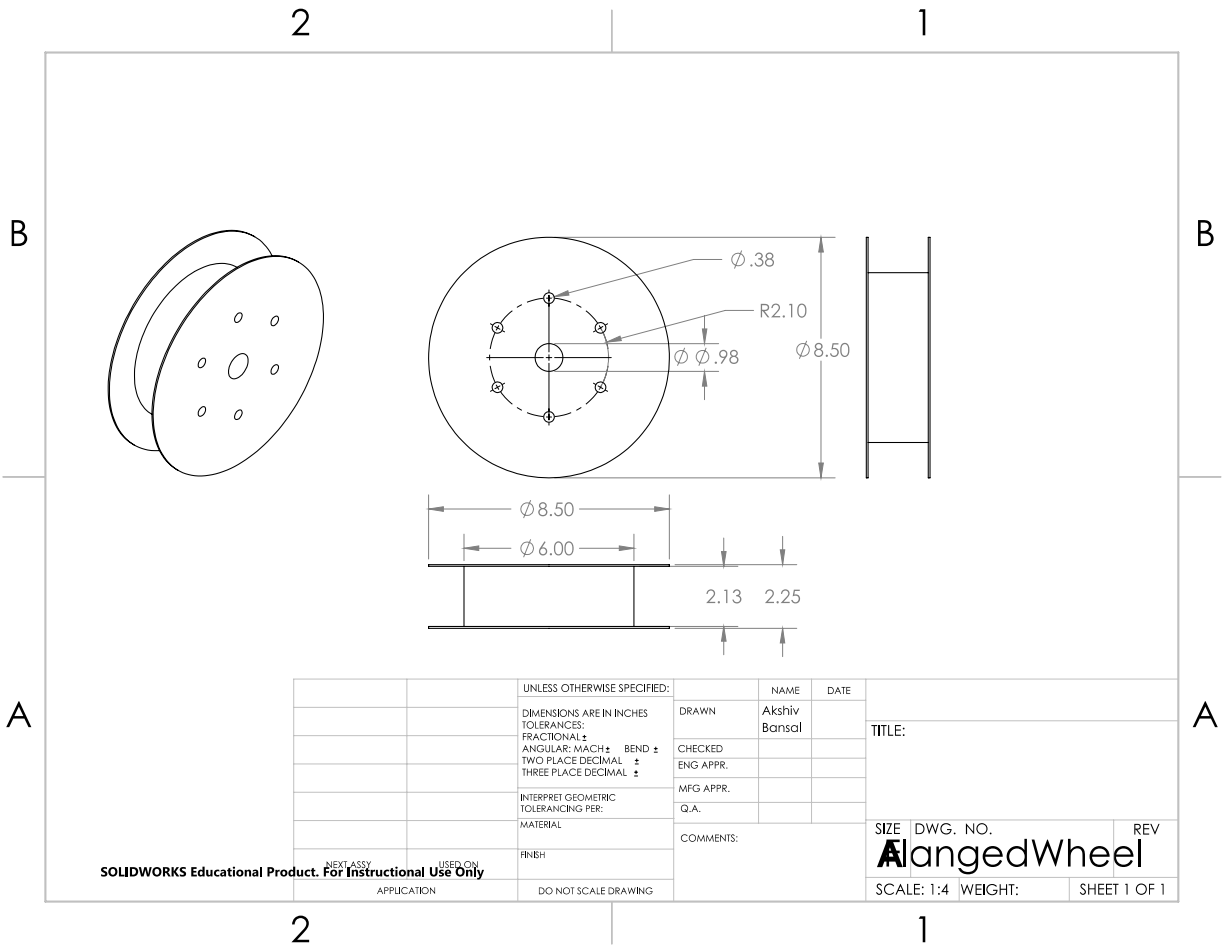


Figure 37: Engineering Drawing of Flanged Wheel

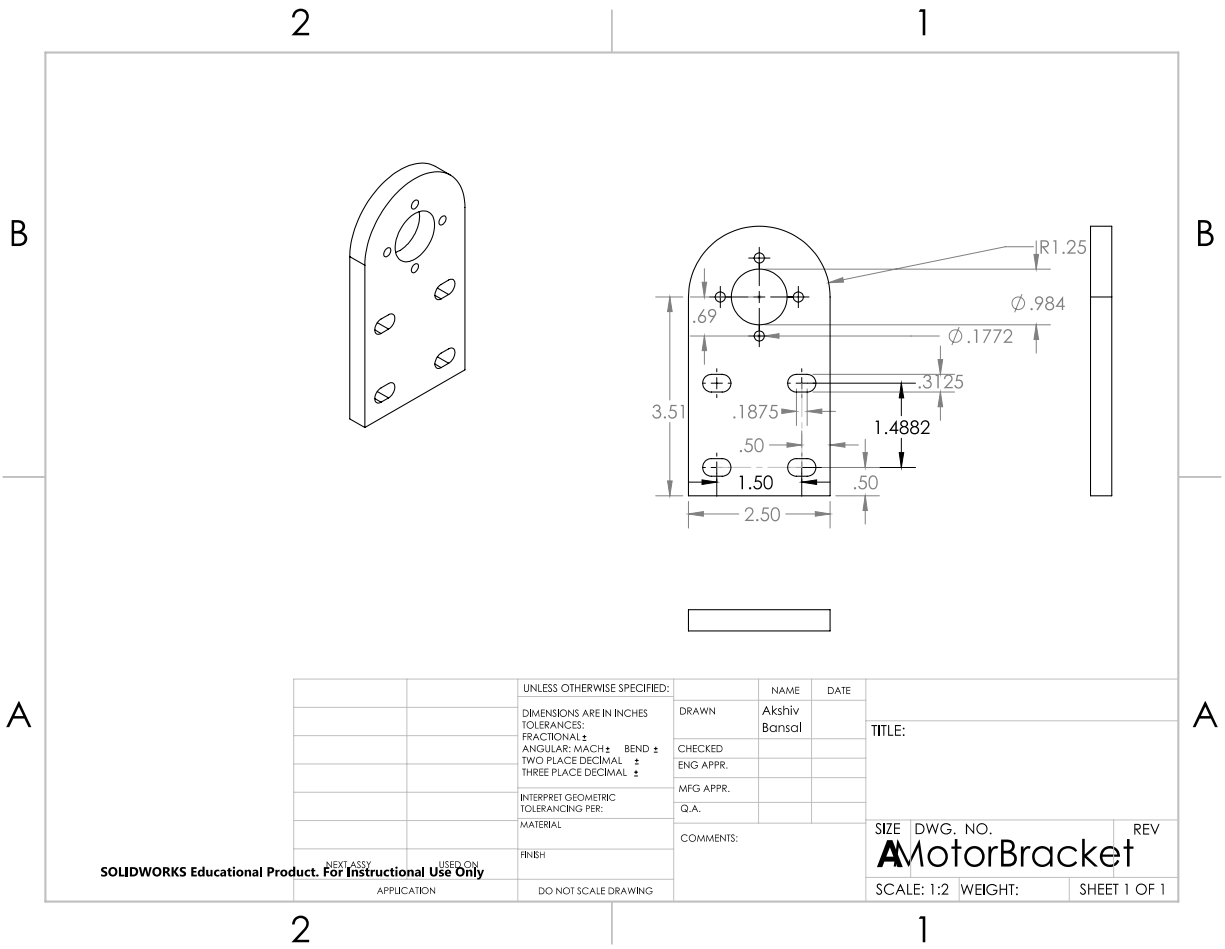


Figure 38: Engineering Drawing of Motor Mounting Bracket

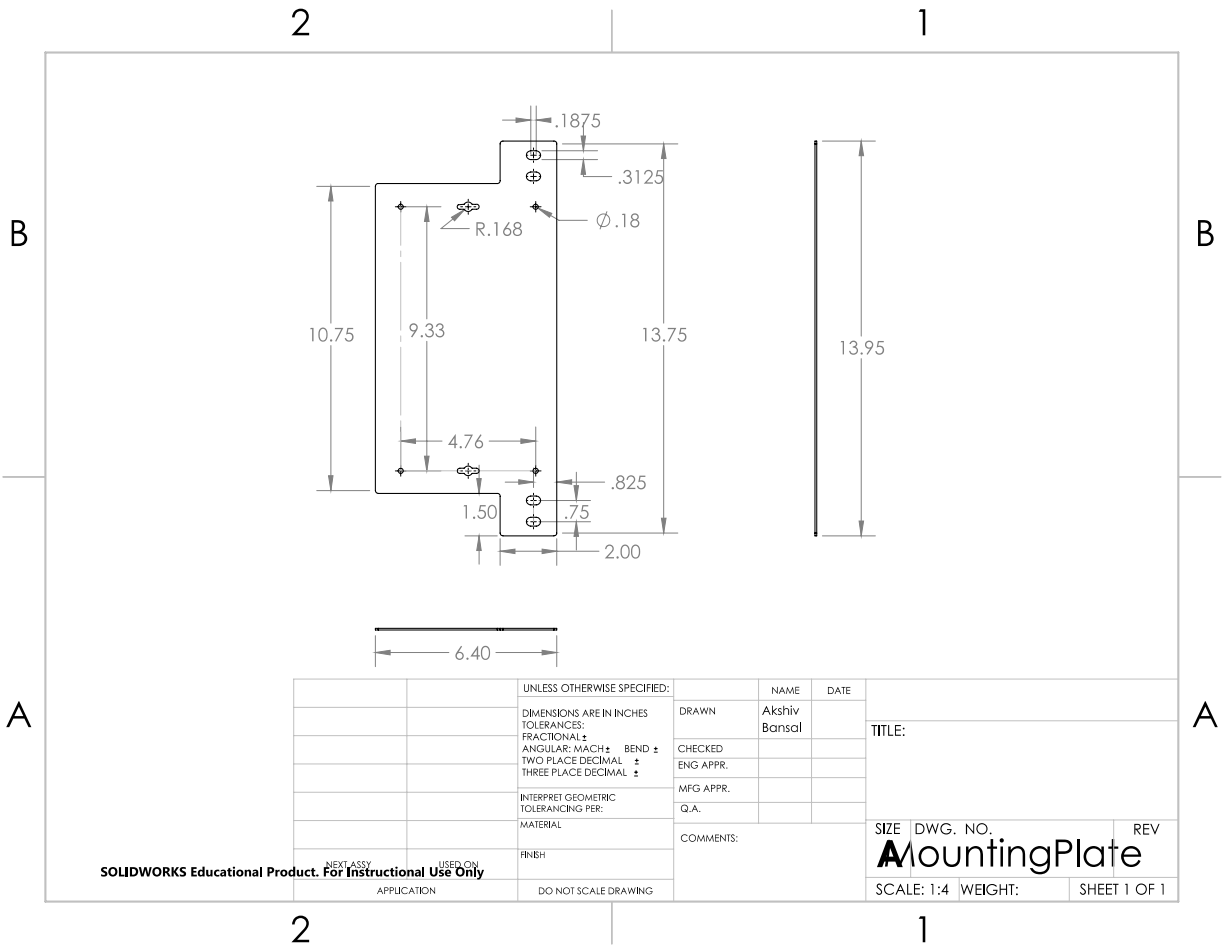


Figure 39: Engineering Drawing of Mounting Plate for Electrical Enclosure

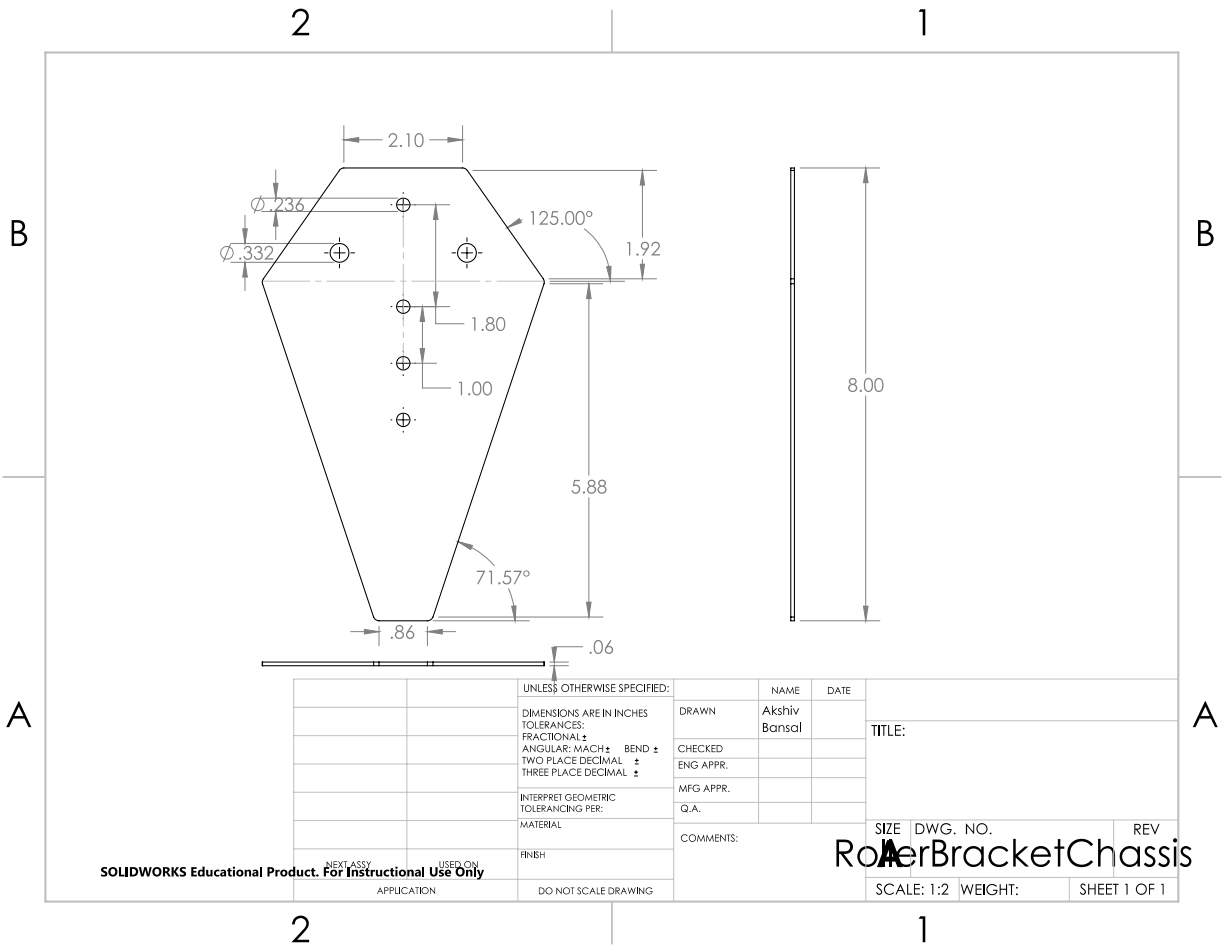


Figure 40: Engineering Drawing of Bracket Attaching Chassis Side of Roller Member

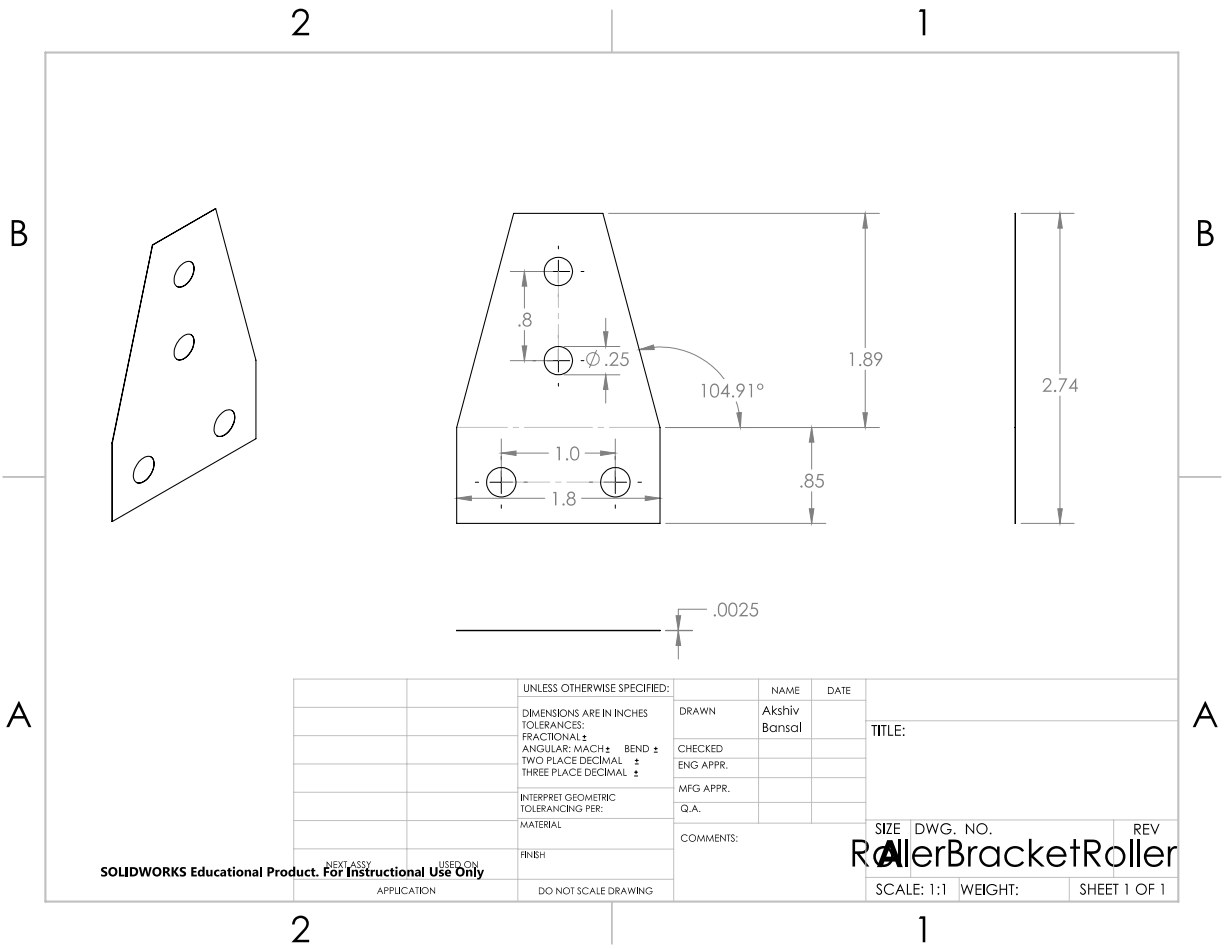


Figure 41: Engineering Drawing of Bracket Attaching Roller Side of Roller Member

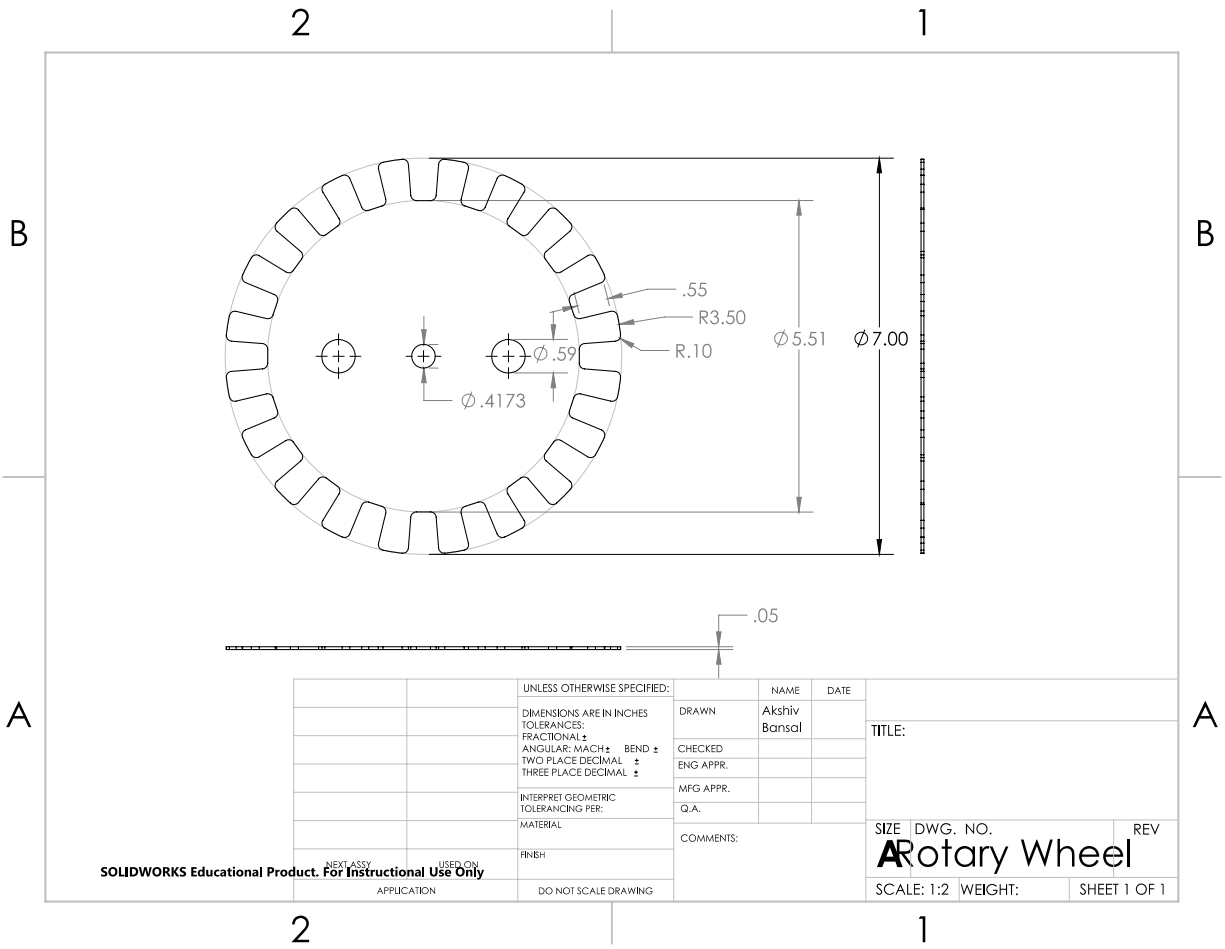


Figure 42: Engineering Drawing of Encoding Disk

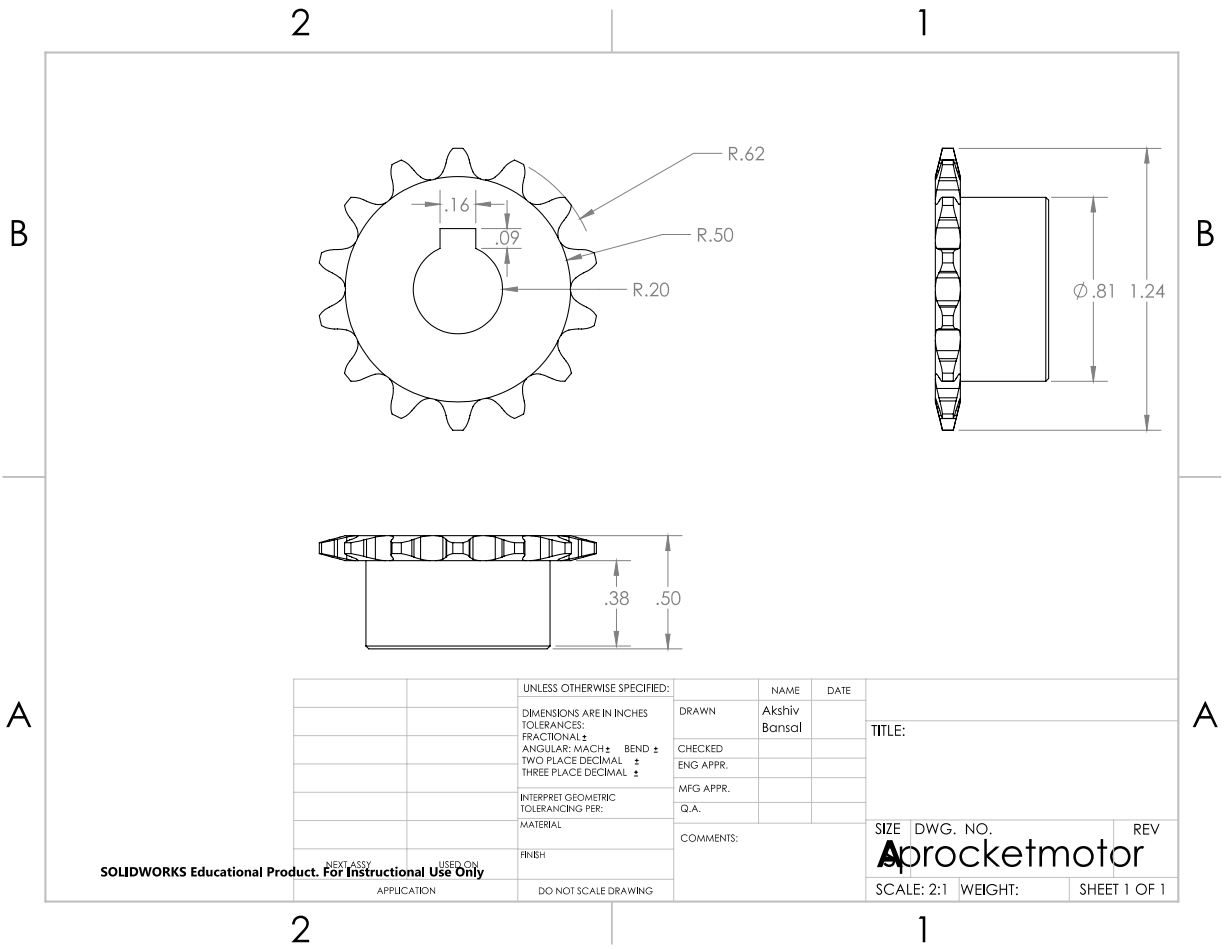


Figure 43: Engineering Drawing of Motor Roller Chain Sprocket

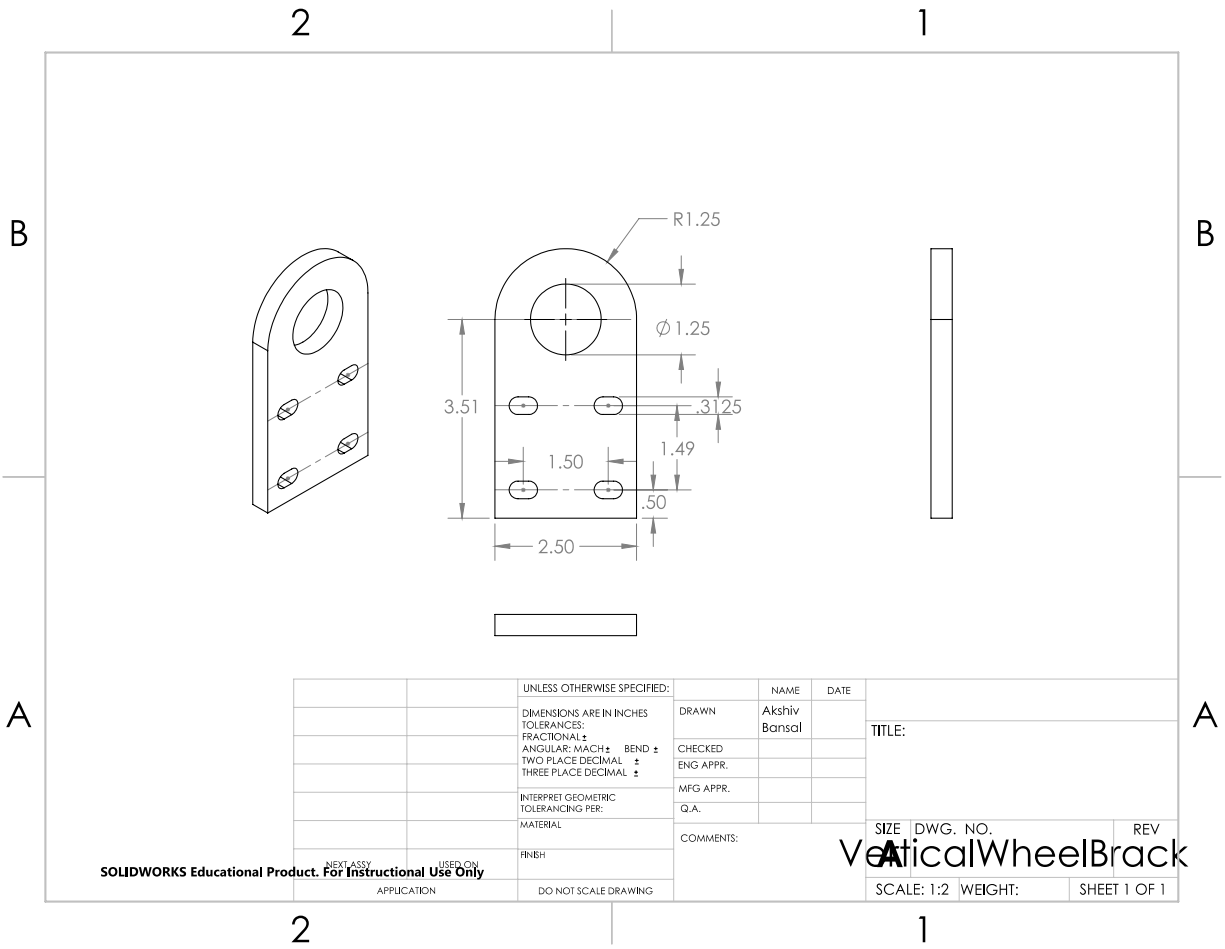


Figure 44: Engineering Drawing of Flanged Wheel Bracket

Appendix B - Electrical Specifications

Stepper Motor

Table 4: Stepper Motor Specification

Specification	Value
Manufacturer	PrimoPal
Part Number	PHP57S82-410-IP65
Rated Phase Current	1.00 A
Torque	170 oz.-in
Step Angle	1.8°
IP Rating	IP65
Type	4 Wire Bipolar
Frame Size	NEMA 23
Unit Cost (USD)	\$50.00

Stepper Driver

Table 5: Stepper Motor Driver Specifications

Specification	Value
Manufacturer	Texas Instruments/Pololu
Part Number	DRV8825
Operation Voltage	8.2 V - 45 V
Max Output Current	1.5 A - 2.2 A
Step Resolutions	1, 1/2, 1/8, 1/16, 1/32
Digital Interfacing	3.3 V or 5 V

Unit Cost (USD) \$8.95

DC Motor

Table 6: DC Motor Specifications

Gear motor Model	PBP45R68-230/PTP45S100-XXX-D8K27 (Output shaft: Ø10mm, Len/27mm, Keyway)
Gear Ratio	19
Transmission Stage	2
Body Length L (mm)	45.6
No-load Speed (rpm)	158
Rated Speed (rpm)	126
Rated Torque (N.m)	1.1
Max. Momentary Admissible Load (N.m)	12

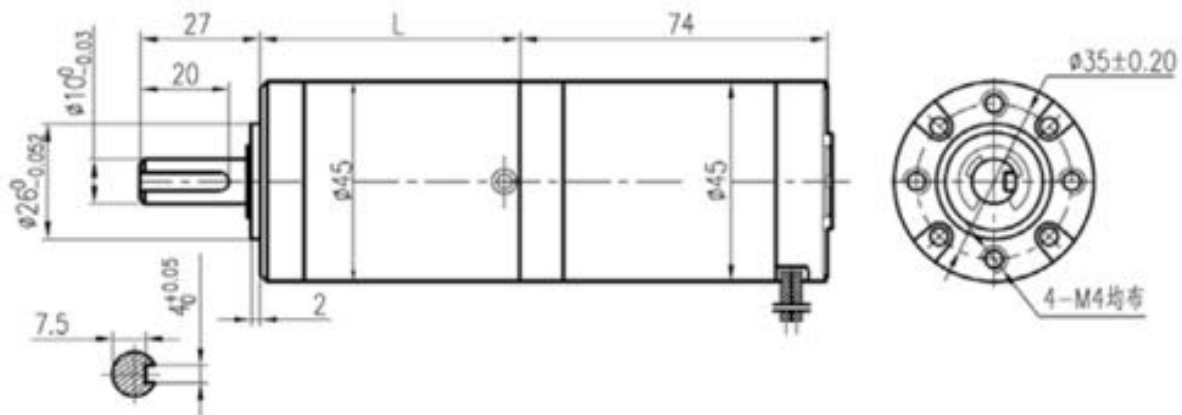


Figure 45: DC Motor Schematic

DC Motor Driver

Table 7: DC Motor Driver Specifications

Specification	Value
---------------	-------

Manufacturer	Pololu
Part Number	G2 24v13
Operation Voltage	6.5 V - 40 V
Max Output Current	13 A
Digital Interfacing	1.8 V to 5.5 V
Unit Cost (USD)	\$29.95

Appendix C - Printed Circuit Board

The following provides brief technical explanations of the circuits on the PCB. Additionally, Appendix F shows the full Altium schematics and Appendix G provides a full BoM for the components on the board. The schematics of the board can also be found on GitHub here:

<https://github.com/enph459/altium-drawings>

Power Input Circuit

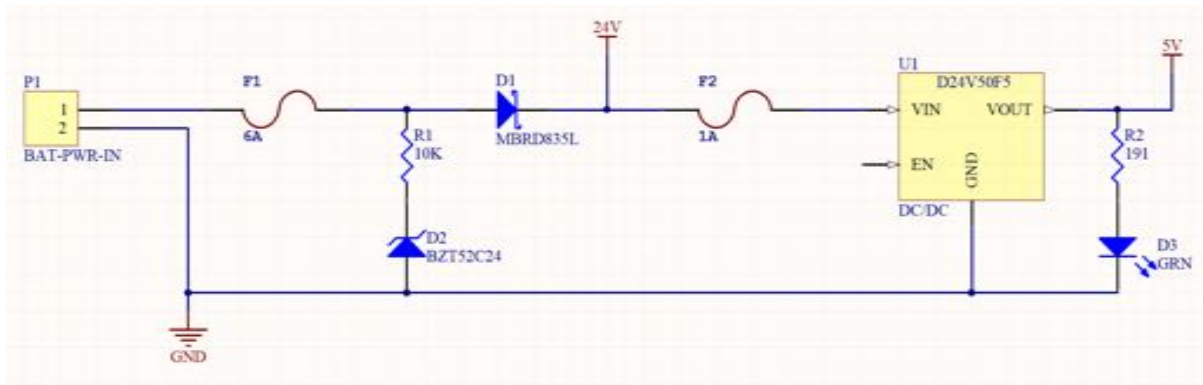


Figure 46: Power input circuit

P1 is Molex Mini Fit connector (rated up to 9A) for the 24V battery input. The main fuse, F1, is sized according to Figure 32 and conservatively rounded down to 6A.

Device	Voltage (V)	Current (A)	Power (W)	Quantity	Total Power (W)
LV Bus					
RPi3	5	2.5	12.5	1	12.5
LV Totals		2.5			12.5
HV Bus					
LV Bus			12.5	1	12.5
Stepper Motor	24	1	24	2	48
DC Motor	24	4.4	105.6	1	105.6
Grand Totals		6.9			166.1

Figure 47: Power consumption estimates

U1, the [DC/DC Converter](#), steps the 24V battery input down to 5V to power the RPi and low voltage. The low voltage circuits fuse F2 is sized as follows:

- Converter rated at 5A out at 5V = 25W
- Input voltage = 24V
- Input current max = 25W/24V = 1A

D2 is a Zener diode for overvoltage protection and D1 is a Schottky diode for reverse polarity protection. D3 is a green LED which lights up when the board is powered.

Stepper Driver Circuit

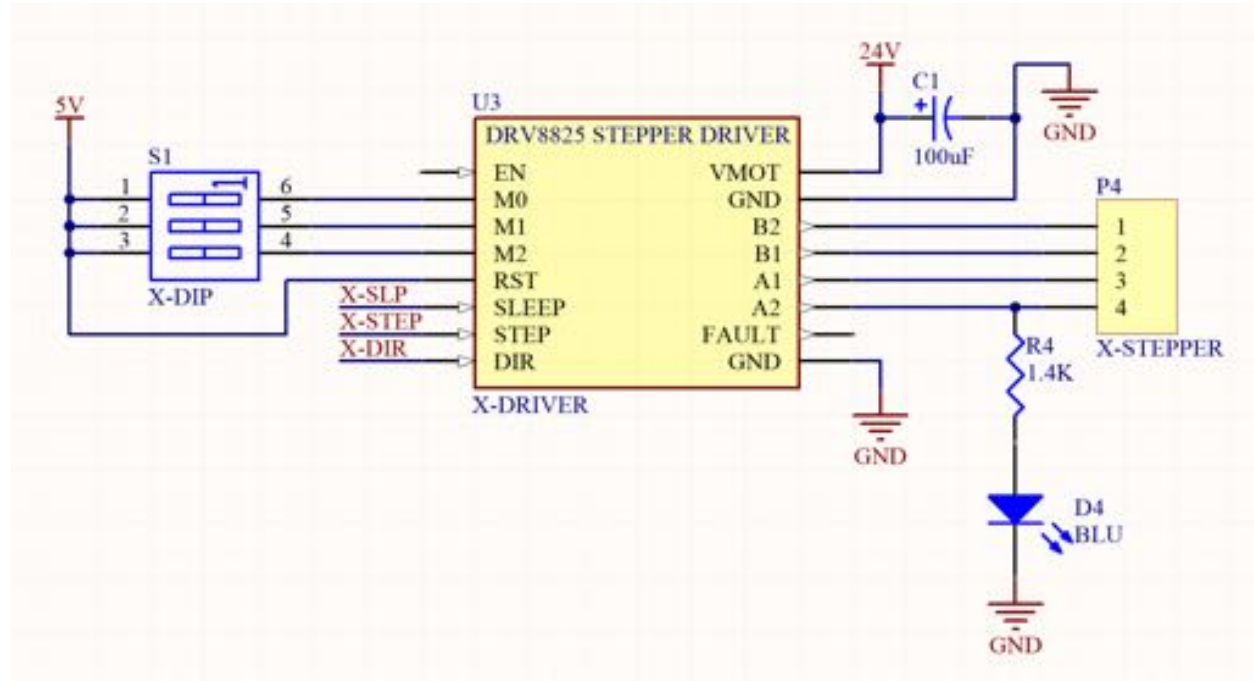


Figure 48: Stepper driver circuit

The [Stepper Driver](#) plugs into the board via female header pins. S1 is a set of 3 SPST switches used to select step resolution using M0, M1, M2 (see truth table in stepper driver data sheet). SLP, STEP and DIR pins are outputs from RPi. C1 is decoupling capacitor as recommended in the data sheet. P4 is a Molex MicroFit Connector used to connect the outputs of the driver to the stepper motor and is rated up to 5A per pin. D4 is a blue LED that lights up when the driver is driving a motor.

DC Motor Driver Circuit

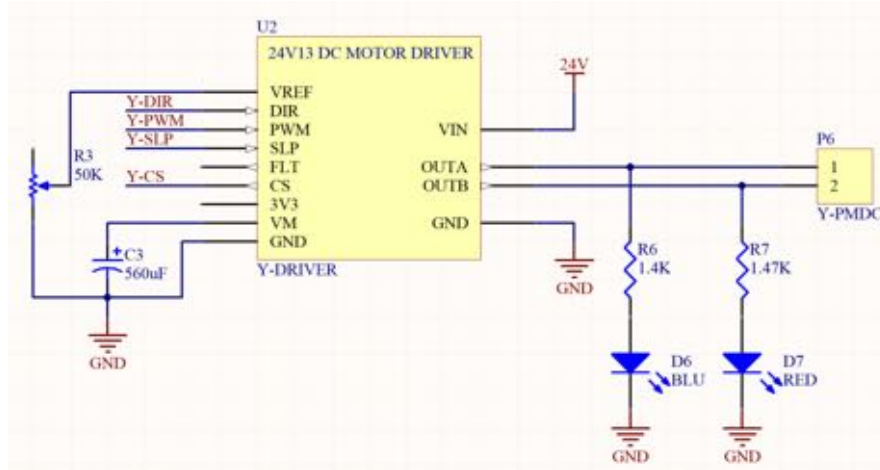


Figure 49: DC motor driver circuit

The [PMDC Motor Driver](#) plugs into female header pins on the board. C3 is a large decoupling capacitor as recommended by driver datasheet. P8 is Molex Mini Fit Connector and is rated up to 9A but the motor only draws a max of 4.4A. R3 is a potentiometer used to limit the motor driver current output and its value is selected based on the graph in Figure 50. A value of approximately 20K is suggested to limit motor current to approximately 5A. Y-CS is the analog output from driver that is proportional to current drawn by motor and follows the relation $V = 50\text{mV} + 40\text{mV/A}$.

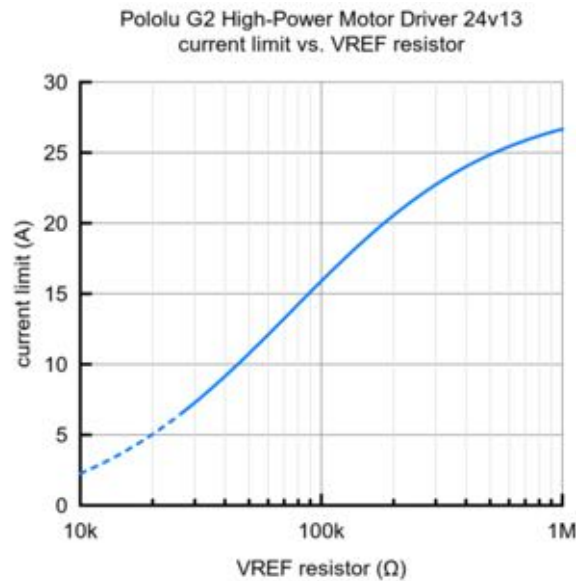


Figure 50: Current limit graph for DC Motor Driver

Limit Switch Inputs

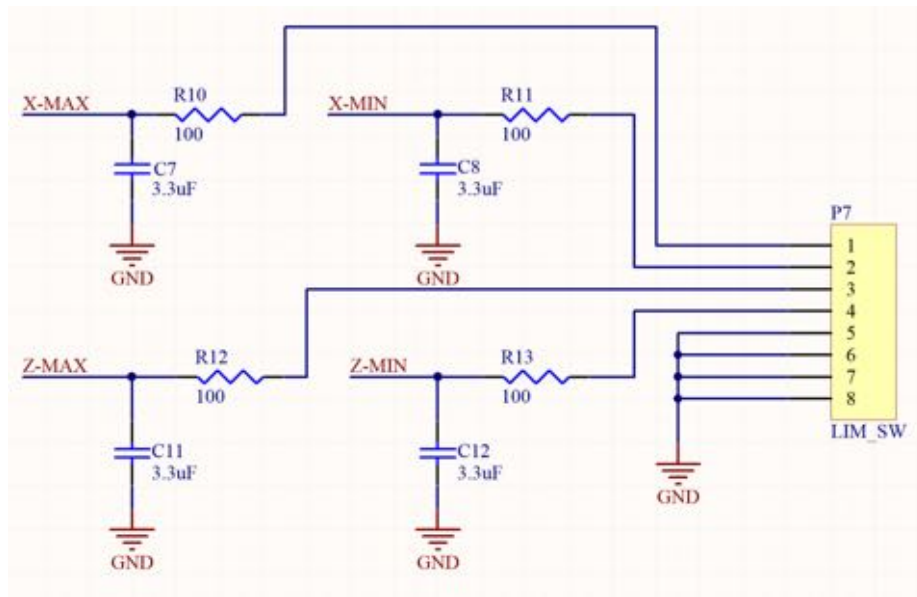


Figure 51: Limit switch inputs

P9 is 4x2 Molex MicroFit Connector and RC values have been selected to have low pass cutoff frequency of $1/(100 \times 3.3\mu\text{F}) = 3 \text{ kHz}$.

Two Channel ADC Circuit

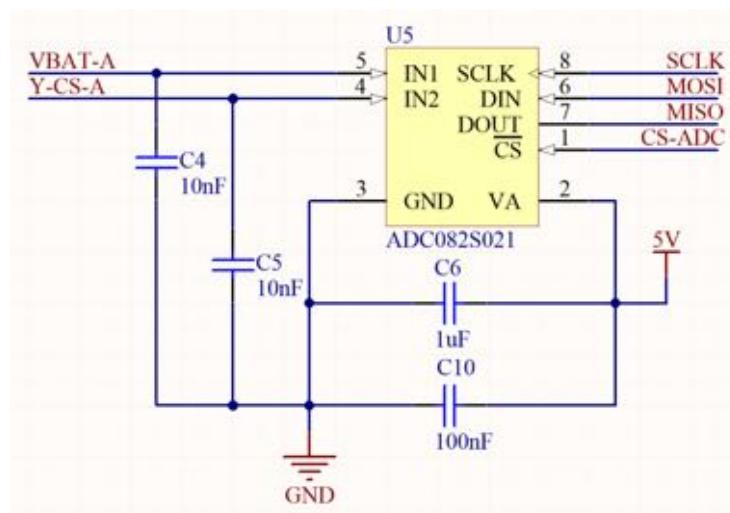


Figure 52: Two channel ADC circuit

Used to send analog battery voltage and DC motor current to RPi via SPI. C6 and C10 are recommended decoupling capacitors from datasheet. Analog input range is 0V to 5V.

DC Motor Current Sense Scaling Circuit

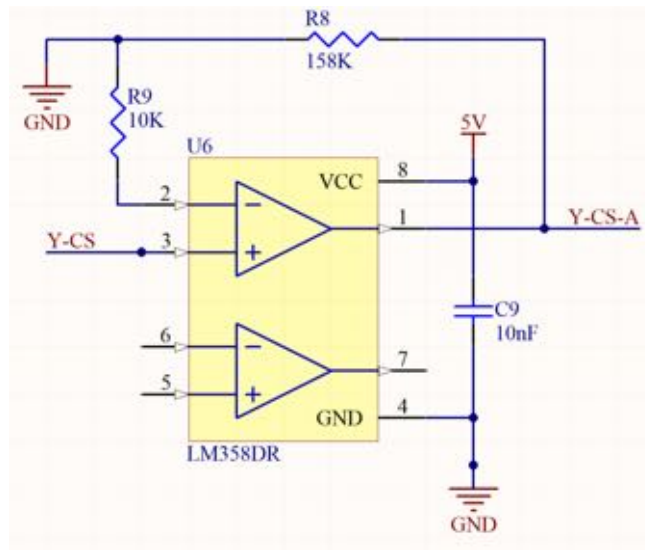


Figure 53: DC Motor current sense scaling circuit

This circuit is used to scale the analog signal proportional to the DC motor current from the DC motor driver to the 0V to 5V input range of the ADC. C9 is the recommended decoupling capacitor. The calculations used to select R8 and R9 are shown below:

- Y-CS output from driver follows: $V_{cs} = 50\text{mV} + 40\text{mV/A}$
- Motor stall current: 4.4A \rightarrow 226 mV output
- Conservative upper limit: 6A \rightarrow 290mV \rightarrow round up \rightarrow 300mV
 - Want to scale to ADC input range of 0V to 5V
 - Gain required: $5\text{V}/0.3\text{mV} = 16.7$
- Non-Inverting Amp:
 - $V_{out}/V_{in} = 1 + R_{10}/R_9$
 - R9 selected as 10K \rightarrow R10 = 158K
 - $1 + 158/10 = 16.8$
- Y-CS-A is scaled current sense output which goes to ADC input

Battery Voltage Sense

Voltage divider with low pass filter to scale 24V to 5V ADC input. Low Pass Cutoff = $1/((38.3\text{K} \parallel 10\text{K}) * 1\mu\text{F})$
= 20 Hz.

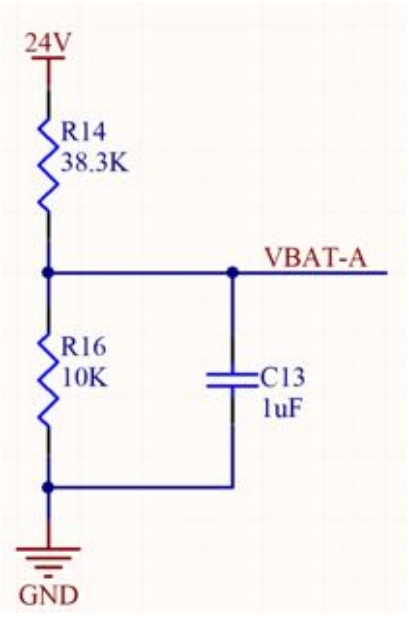


Figure 54: Battery voltage sensing voltage divider with low pass filter

Appendix D - Raspberry Pi Configuration Information

General Information

Table 8: Login information

	Name	Password
Wi-Fi	cropsensewifi	vanbelle
SSH (Wi-Fi)	pi@169.254.157.86	cropsense
SSH (Ethernet)	pi@169.254.200.49	cropsense

Tutorial followed to setup access point mode:

<http://elinux.org/RPI-Wireless-Hotspot>

Tutorial followed to switch between access point mode and client mode:

<https://hydrosysblog.wordpress.com/2016/08/07/rpi3-switch-between-wifi-ap-and-client/>

Required Packages

Note, node packages can be installed by running npm install inside root directory (contained within package.json).

Table 9: Installed packages

Package Name	Purpose	Tutorial/Source
Node	Web Server	https://blog.wia.io/installing-node-js-on-a-raspberry-pi-3
Express	Web Server	https://expressjs.com/
RPi.GPIO	GPIO	https://pypi.python.org/pypi/RPi.GPIO

WiringPi GPIO <https://github.com/WiringPi/WiringPi-Python>

Pygame Joystick <https://www.pygame.org/>

Appendix E - Software Documentation

EIS Van Belle Software Documentation

Getting Started

First the `CONF_DIRECTORY` must be properly set within `cs_vanbelle_robot.conf.conf.py`. The configuration files already contain default parameters and directory but can be modified directly.

```
import cs_vanbelle_robot # import modules from package
import cs_vanbelle_robot.setup

# sets pins and loads configuration files
setup.init()
# do things!
```

For manual joystick control, run the script `run.py`. Refer to `cs_vanbelle_robot.example` for example files.

Directory/Structure

The package itself for the robot is called `cs_vanbelle_robot`. Contained within the package are the following sub-packages and folders.

- cs_vanbelle_robot - Main Package
 - setup.py - Initialization Module
 - main.py - NOT IMPLEMENTED

- manual.py - Manual Joystick Control
- packages
 - comm - Communication Module
 - udp_client.py
 - conf - Configuration Module
 - config
 - *config modules live here*
 - conf.py
 - controls - Motor Control and Position Module
 - stepper_x.py
 - motor_y.py
 - stepper_z.py
 - position.py
 - gpio - GPIO Module
 - gpio.py
 - helpers - Helper Function Module
 - helpers.py
 - joystick - Joystick Module
 - sidewinder.py
 - traversal - NOT IMPLEMENTED
- archives - Old Files
- configuration_files - Configuration JSON Files
- examples - Example Files
- scripts - Helper Scripts
- vanbelle_ui - User Interface

Van Belle Robot API Reference

Initialization Module

This module is used to initialize all the pins as well as set the origin. Import this module and run `init` (and/or `define_position`) at the start of your script. This import is only required for the parent file and to be run once.

```
def init():
    Initialize pins, interrupts and pwm.

def define_position():
    Set X, Y and Z position to zero.
```

Main Control Loop

This file was designed to hold the autonomous control loop and it will eventually contain the autonomous horizontal movement control loop.

Manual Joystick Control

To control the joystick manually, import this file and call the function "run". The intended use for these functions is to remain inside *manual.py*.

```
def run(include_wifi = False, fixed_speed = False, position_control = False,
        start_from_UI= False):
    Main function for joystick control.
    :param include_wifi: Use wireless joystick (requires external script).
    :param fixed_speed: CURRENTLY NOT IMPLEMENTED. Use a fixed speed for smoother control
    oppose to a variable speed.
    :param position_control: CURRENTLY NOT IMPLEMENTED. Specify position instead of speed
    with joystick.
    :param start_from_ui: CURRENTLY NOT IMPLMENTED
```

The scripts for wireless joystick control are contained within the Communication Module.

Communication Module

Comm, or the communications module, involves any files that relate to communications such as UDP/TCP connection. The following files are contained within the comm directory:

- `external_joystick_client.py`
- `remote_joystick_server.py`
- `udp_client.py`

The file `udp_client` is primarily used to communicate with Crop Sense. When a scanning location has been reached, the function `udp_client.send` will be called. To see `udp_client` in use, refer to `communication_example.py`.

Note, `external_joystick_client.py` must be ran from the external device.

udp_client.py

```
def send(data, handle_recieved, host="169.254.235.239", port=9999, debug=False):
    Send the data using a UDP connection. Received data is parsed and handled using the
    passed function handle_recieved.
    :param data: the data to send to host
    :param handle_recieved: function that takes one argument, the received data
    :param host: the host IP address
    :param port: the host port number
    :param debug: if true will display sent and received data
```

Configuration Module

Used to load configuration files containing parameter and pin values. The parameters are loaded in from JSON files and stored as variables within a module contained in `conf.config` in order to prevent unnecessary file opening (files are only loaded once on `setup.init()`).

Values from the configuration files are accessed in the following way:

```
from ..conf.configs import config_x

param1 = config_x.PARAM1
param2 = config_x.PARAM2
# etc ...
```

Example JSON file:

```
{
  "name": "stepper_x",
  "params": {
    "x_min_pos": 0,
    "x_max_pos": 100,
    "left_dir": 0,
    "right_dir": 1,
    "x_micro_step": 1,
    "x_step_angle": 1.8,
    "x_npm_min": 125,
    "x_npm_max": 200,
    "x_npm_optimal": 150,
    "pitch_diameter": 19.1
  },
  "pins":
  [{
    "name": "x_min",
    "pin": 31,
    "mode": "input",
    "description": ""
  },
  {
    "name": "x_max",
    "pin": 33,
    "mode": "input",
    "description": ""
  },
  {
    "name": "x_dir",
    "pin": 3,
    "mode": "output",
    "description": ""
  },
  {
    "name": "x_step",
    "pin": 5,
```

```

        "mode": "output",
        "description": ""
    },
    {
        "name": "x_slp",
        "pin": 7,
        "mode": "output",
        "description": ""
    }
]
}

```

Example config module:

```

from ...conf import conf

# get properties
properties = conf.load('stepper_x')

# standard params
LEFT      = properties['left_dir']
RIGHT     = properties['right_dir']
MIN_POS   = properties['x_min_pos']
MAX_POS   = properties['x_max_pos']

# x-axis params
PITCH_DIAM = properties['pitch_diameter']

# stepper specific parameters
MICRO_STEP = properties['x_micro_step']
STEP_ANGLE = properties['x_step_angle']
NPM_MIN    = properties['x_npm_min']
NPM_MAX    = properties['x_npm_max']
OPTIMAL_NPM = properties['x_npm_optimal']

# get list of pins
pins      = properties['pins']

# pins

```

```
slp_pin    = properties['x_slp']
dir_pin    = properties['x_dir']
out_pin    = properties['x_step']
left_ref_pin = properties['x_max']
right_ref_pin = properties['x_min']
```

conf.py

While other functions exist in side *conf.py*, this is the only function used globally.

```
def load(filename, module=None):
```

```
    Load the JSON configuration file and parse the result.
```

```
    :param filename: name of conf file to open
```

```
    :param module: sub-folder within configuration directory.
```

```
    :return: Parsed configuration files as dictionary.
```

Motor and Position Module

Contains functions to move in X (horizontal), Y (rail) and Z (vertical) directions. Also contains a python module that is used to keep track of the absolute X, Y, Z position.

stepper_x.py

```
def limit(loc):
```

```
    Interrupt callback, disabled motors when limit switch is hit.
```

```
    :param loc: either the left or right
```

```
def enable():
```

```
    Enables stepper.
```

```
def disable():
```

```
    Disables stepper.
```

```
def get_steps(distance):
```

```
    Get steps based on distance.
```



```

:param distance: in millimeters
:return: number of steps required to get to target distance

def get_distance(steps):
    Get distance based on steps.
    :param steps:
    :return: distance traversed based on number of steps, in millimeters.

def move_x(distance, speed, direction, debug=False):
    Move in the horizontal direction.
    :param distance: in millimeters.
    :param speed: in RPM.
    :param direction: Left or Right
    :param debug: motors disabled when debug is True

def take_steps(steps, n_rpm, direction):
    Generate a square wave and take N steps. Call this function to move "manually".
    :param steps:
    :param n_rpm: speed
    :param direction: Left or Right

def zero(debug=False):
    Move in X until reaching limit. Sets position in X to zero.
    :param debug: print location

```

stepper_z.py

Same functions as stepper_x but directions are now either Top or Bottom. A separate file was created in case modifications were made to mechanical movement in the X and also for increased readability in code.

motor_y.py

```

def enable():
    Enables stepper.

def disable():

```

Disables stepper.

```
def move_y(distance, direction, debug=False):  
Move along the rails to a specified position.  
:param distance: in millimeters.  
:param direction: Forward or Backward  
:param debug: motors disabled when debug is True
```

```
def move(speed, direction):  
Move motor in Y direction.  
:param speed: Value from 0 to 100.  
:param direction: Forward or Backward.
```

NOTE: `motor_y` contains code implemented for PID and encoder counting that was not fully tested. Use these functions as a potential skeleton for future development.

position.py

Position module contains global variables X, Y and Z to keep track of the current location of the robot. These values are updated in side of `stepper_x`, `motor_y` and `stepper_z`.

```
# Globals to keep track of position  
# UNITS IN MILLIMETERS  
X = 0  
Y = 0  
Z = 0
```

The values can be accessed by first importing the module the doing the following:

```
import position  
  
position.X  
position.Y  
position.Z
```

Functions utilize `convert units` helper function. The developer is responsible for making sure the units are consistent.

```
def update_x(distance, direction, pre_units='mm', post_units='mm'):
    Updates global variable for X position (horizontal).
    :param distance:
    :param direction: 1 or 0
    :param pre_units: Incoming units.
    :param post_units: Outgoing units.

def update_y(distance, direction, pre_units='mm', post_units='mm'):
    Updates global variable for Y position (rail).
    :param distance:
    :param direction: 1 or 0
    :param pre_units: Incoming units.
    :param post_units: Outgoing units.

def update_z(distance, direction, pre_units='mm', post_units='mm'):
    Updates global variable for Z position (vertical).
    :param distance:
    :param direction: 1 or 0
    :param pre_units: Incoming units.
    :param post_units: Outgoing units.

def set_x(pos):
    Sets X position. (Overrides current value)

def set_y(pos):
    Sets Y position. (Overrides current value)

def set_z(pos):
    Sets Z position. (Overrides current value)

def get_x():
    Returns X position.

def get_y():
    Returns Y position.

def get_z():
    Returns Z position.
```

```
def print_pos(pre_units='mm', post_units='mm'):
    Prints current position.
    :param pre_units: Incoming units.
    :param post_units: Outgoing units.
```

GPIO Module

Custom function wrapper was created to separate GPIO code from installed packages as the currently used packages (RPi.GPIO and WiringPi) may update/change/or lose support. In order to prevent an entire code rewrite this module was created.

gpio.py

The current pinout is using the standard GPIO board output. For the WiringPi PWM, the pin number follows the WiringPi numbering standard. For more information visit: <https://pinout.xyz/>.

```
def set_pins(pins):
    Set multiple pins.
    :param pins: list containing pin object (dictionary with pin, mode and optional name
    key, values).

def set_pin(pin, mode, name=None):
    Set pin.
    :param pin: Pin number, using GPIO board pinout (TODO link)
    :param mode: Mode to set GPIO pin.
                    To set as input: 'input' or 0
                    To set as output: 'output' or 1
                    To set as pwm: 2
    :param name: The name of the pin.

def read(pin):
    Read value of pin.
    :param pin: Pin number.

def write(pin, value):
    Write value to pin.
```

```

:param pin: Pin number.
:param value: 0 or 1.

def get_pin_mode(pin):
Gets the current mode that a pin has been set too.
:param pin: Pin number
:return: Mode, either 0, 1, 2

def configured_pins():
Displays all currently configured pins.

def reset_pin(pin):
Resets a pin.
:param pin: Pin Number.

def shutdown():
Cleans up GPIO, resets all pins.

def add_interrupt(pin, edge_type, callback=None, bouncetime=10, name=None):
Add interrupt to pin with threaded callback.
:param pin: Pin number.
:param edge_type: Edge to detect event on, either rising, falling or both.
:param callback: Function to call when event has been detected.
:param bouncetime: Switch debouncing in milliseconds.
:param name: interrupt name.

def remove_interrupt(pin):
Removes interrupt set on pin.
:param pin: Pin number.

def event_detected(pin):
Returns a value depending on an interrupt event occurring.
:param pin: Pin number.
:return: 1 if event detected, otherwise 0.

def get_interrupts():
Displays all configured interrupts.

```

```
def pwm_init(pwm_pin, pwm_mode):
    Initialize hardware PWM.
    :param pwm_pin: Pin number.
    :param pwm_mode: Pin mode.

def pwm_set(pwm_pin, duty):
    Set duty cycle of PWM.
    :param pwm_pin: Pin number.
    :param duty: Duty cycle, between 0 and 100.

def pwm_stop(pwm_pin):
    Disable PWM.
    :param pwm_pin: Pin number.
```

Helper Function Module

Contains simple helper methods. Other functions can be included here if needed.

helpers.py

```
def disable_all():
    Disable all motors.

def error(text, name='Error', ):
    Handle error. Will raise exception
    :param text: Error message.
    :param name: Name of error.

def convert_units(distance, pre_units, post_units):
    Convert from pre_units to post_units.
```

Joystick Module

Pygame (<https://www.pygame.org/>) was used to incorporate the joystick for manual control. Create a new module for each joystick used. Refer to Pygame documentation where required.

sidewinder.py

For the sidewinder joystick: <https://www.amazon.com/Microsoft-E13-00001-Sidewinder-Joystick-USB/dp/B00006BA0Q>

```
def joystick_init(print_name=True):
    Initialize Joystick.

def get_joystick_values(joystick):
    Get values from the joystick
    :param joystick:
    :return: Values for axis1, axis2, axis3, btn0, btn1 and btn2.
```

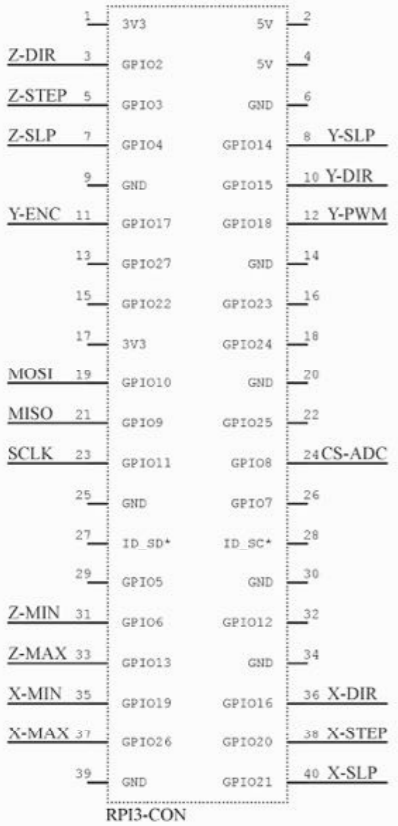
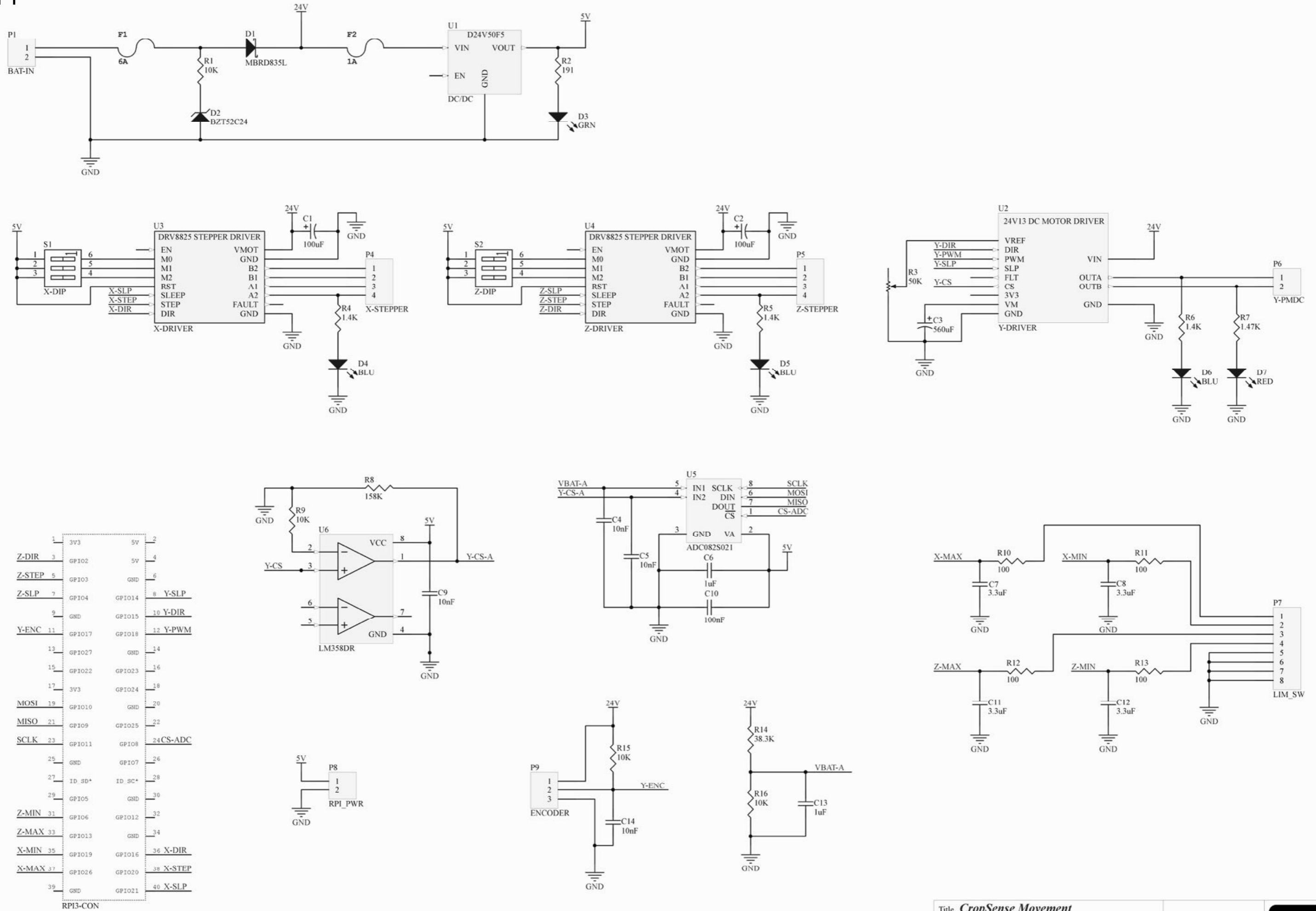
Scripts

There are 2 scripts contained within the scripts directory.

- `AP_to_client.sh`
- `client_to_AP.sh`

AP_to_client.sh and *client_to_AP.sh* are used to switch the RPi between hosting a Wi-Fi network (access point mode or AP) and being able to connect to Wi-Fi.

Appendix F - PCB Altium Schematic



Appendix G - PCB Full Bill of Materials

Label	Description	Manufacturer	Manufacturer Part Number	Digikey Part Number	Unit Price (CAD)	QTY	Extended Price (CAD)	Link	Notes
D1	DIODE ZENER 24V 500MW SOD123	Diodes Incorporated	BZT52C24-7-F	BZT52C24-FDICT-ND	\$0.29	1	\$0.29	http://www.digikey.ca/product	Power
D2	DIODE SCHOTTKY 35V 8A DPAK	ON Semiconductor	MBRD835LT4G	MBRD835LT4GOSCT-ND	\$0.96	1	\$0.96	https://www.digikey.ca/produ	Power
F1, F2	Fuse Clip 10A 250V 1 Circuit Cartridge PCB	Littlefuse	01110501Z	F4189-ND	\$0.18	4	\$0.72	http://www.digikey.ca/product	Power
F1	FUSE GLASS 6A 125VAC 5X20MM	Eaton	BK1/GMA-6-R	283-3275-ND	\$0.50	5	\$2.50	http://www.digikey.ca/product	Power
F2	FUSE GLASS 1A 250VAC 5X20MM	Bel Fuse Inc.	5MF 1-R	507-1261-ND	\$0.28	5	\$1.38	http://www.digikey.ca/product	Power
D3	LED GREEN CLEAR 1206 SMD	Lite-On Inc.	LTST-C150GKT	160-1169-1-ND	\$0.48	1	\$0.48	https://www.digikey.ca/produ	Power
P1, P6	CONN HEADER 2POS 4.2MM VERT TIN	Molex, LLC	39299023	WM3853-ND	\$0.72	2	\$1.44	http://www.digikey.ca/product	Battery In, PMDC Out
P1, P6	CONN RECEPT 2POS VERT DUAL	Molex, LLC	39013022	WM1021-ND	\$0.44	2	\$0.88	http://www.digikey.ca/scripts/l	RPi Power
P8	MICROFIT 3.0 SR VERT TH PEG 15AU	Molex, LLC	43650-0228	WM10658-ND	\$1.22	1	\$1.22	http://www.digikey.ca/product	RPi Power
P8	MICROFIT 3.0 RECEPT. SR 2CKT GW	Molex, LLC	436450208	WM11226-ND	\$0.52	1	\$0.52	http://www.digikey.ca/product	Power, RPi CON,
C1, C2	CAP ALUM 100UF 20% 63V RADIAL	Panasonic	EEU-EB1J101	P13137-ND	\$0.67	2	\$1.34	http://www.digikey.ca/product	Stepper Driver
S1, S2	3 Pos DIP SPST 100MA 20V	CTS Electrocomponents	210-3MS	CT2103MS-ND	\$0.69	2	\$1.38	http://www.digikey.ca/product	Stepper Driver
P4, P5	MICROFIT 3.0 SR VERT TH PEG 15AU	Molex, LLC	436500428	WM10669-ND	\$1.94	2	\$3.88	http://www.digikey.ca/scripts/l	Stepper Driver
P4, P6	CONN RECEPT 4POS 3MM SINGLE ROW	Molex, LLC	436450400	WM1847-ND	\$0.61	2	\$1.22	http://www.digikey.ca/product	Stepper Driver
P7	CONN HEADER 8POS 3MM VERT TIN	Molex, LLC	430450812	WM1792-ND	\$2.45	1	\$2.45	https://www.digikey.ca/produ	Limit Switch
P7	CONN RECEPT 8POS 3MM VERT DUAL	Molex, LLC	43025-0800	WM1786-ND	\$0.69	1	\$0.69	https://www.digikey.ca/produ	Limit Switch
P9	CONN HEADER 3POS 3MM VERT TIN	Molex, LLC	436500315	WM1918-ND	\$1.39	1	\$1.39	https://www.digikey.ca/produ	Encoder
P9	MICROFIT 3.0 RECEPTACLE SGL ROW	Molex, LLC	43645-0308	WM11251-ND	\$0.62	1	\$0.62	http://www.digikey.ca/scripts/l	Encoder
C3	CAP ALUM 560UF 20% 63V RADIAL	Nichicon	UHE1J561MHD	493-1648-ND	\$1.43	1	\$1.43	http://www.digikey.ca/product	DC Motor Driver
R3	TRIMMER 50K OHM 0.1W SMD	Bourns Inc.	TC33X-2-503E	TC33X-503ECT-ND	\$0.42	1	\$0.42	https://www.digikey.ca/produ	DC Motor Driver
D4,D5,D6	LED BLUE CLEAR 1206 SMD	Lite-On Inc.	LTST-C150TBKT	160-1643-1-ND	\$0.55	3	\$1.65	https://www.digikey.ca/produ	DC Motor Driver
D7	LED RED CLEAR 1206 SMD	Lite-On Inc.	LTST-C150KRKT	160-1405-1-ND	\$0.54	1	\$0.54	https://www.digikey.ca/produ	DC Motor Driver
U5	IC ADC 8BIT 2CH 200KSPS 8VSSOP	Texas Instruments	ADC082S021CIMM/NOPB	ADC082S021CIMM/NOPB	\$3.43	1	\$3.43	http://www.digikey.ca/product	ADC
U6	IC OPAMP GP 700KHZ 8SOIC	Texas Instruments	LM358DR	296-1014-1-ND	\$0.71	1	\$0.71	http://www.digikey.ca/product	OpAmp
RPI3 CON	40 Position Cable Assembly Rectangular Socket to	Assmann WSW Compone	H3AAH-4006G	H3AAH-4006G-ND	\$2.56	1	\$2.56	http://www.digikey.ca/scripts/l	Passive
C4, C5, C9, C14	CAP CER 10000PF 50V X7R 1206	Yageo	CC1206KRX7R9BB103	311-1174-1-ND	\$0.20	4	\$0.80	http://www.digikey.ca/scripts/l	Passive
C6, C13	CAP CER 1UF 50V X7R 1206	Murata Electronics North	GRM31MR71H105KA88L	490-6527-1-ND	\$0.27	2	\$0.54	http://www.digikey.ca/product	Passive
C7, C8, C11, C12	CAP CER 3.3UF 25V X7R 1206	Samsung Electro-Mechan	CL31B335KAHVPNE	1276-6860-1-ND	\$0.35	4	\$1.40	http://www.digikey.ca/scripts/l	Passive
C10	CAP CER 0.1UF 50V X7R 1206	KEMET	C1206C104K5RAC7867	399-1249-1-ND	\$0.14	1	\$0.14	http://www.digikey.ca/product	Passive
R1, R9, R15, R16	RES SMD 10K OHM 5% 1/8W 0805	Stackpole Electronics Inc.	RMCF0805JT10K0	RMCF0805JT10K0CT-ND	\$0.02	10	\$0.23	http://www.digikey.ca/product	Passive
R2	RES SMD 191 OHM 1% 1/8W 0805	Yageo	RC0805FR-07191RL	311-191CRCT-ND	\$0.03	10	\$0.29	http://www.digikey.ca/product	Passive
R4, R5, R6	RES SMD 1.4K OHM 1% 1/8W 0805	Yageo	RC0805FR-071K4L	311-1.40KCRCT-ND	\$0.03	10	\$0.29	http://www.digikey.ca/scripts/l	Passive
R7	RES SMD 1.47K OHM 1% 1/8W 0805	Yageo	RC0805FR-071K47L	311-1.47KCRCT-ND	\$0.03	10	\$0.29	http://www.digikey.ca/product	Passive
R8	RES SMD 158K OHM 1% 1/8W 0805	Yageo	RC0805FR-07158KL	311-158KCRCT-ND	\$0.03	10	\$0.29	http://www.digikey.ca/product	Passive
R10, R11, R12, R13	RES SMD 100 OHM 1% 1/8W 0805	Yageo	RC0805FR-07100RL	311-100CRCT-ND	\$0.03	10	\$0.29	http://www.digikey.ca/scripts/l	Passive
R14	RES SMD 38.3K OHM 1% 1/8W 0805	Yageo	RC0805FR-0738K3L	311-38.3KCRCT-ND	\$0.03	10	\$0.29	http://www.digikey.ca/product	Passive
Molex Crimp MicroFit	CONN TERM FEMALE 20-24AWG TIN	Molex, LLC	43030-0001	WM1837CT-ND	\$0.15	30	\$4.58	http://www.digikey.ca/product	Crimp
Molex Crimp MiniFit	CONN TERM FEMALE 16AWG TIN CRIMP	Molex, LLC	39000077	WM3112CT-ND	\$0.13	10	\$1.32	http://www.digikey.ca/scripts/l	Crimp
TOTAL							\$44.85		