DigSafe Technical Manual

Remy Allegro, Nathaniel Breiter, Deanna Rice, Carl Runci, Yonatan Weiner

May 6, 2021

Contents

1 How to Use this Guide

1.1 Purpose of this Guide

This document provides brief, step-by-step instructions for the development of the DigSafe Buried Cable Detection robot with the intent to assist the following year's MQP team in their next iteration. It is designed to be a general guide of all the subsystems as they stand at the end of the 2020-21 DigSafe MQP team.

1.2 Audience

This guide is intended for upcoming MQP teams with the intent of providing a base platform to assist their endeavor of the next iteration of the DigSafe Buried Cable Detection MQP.

1.3 Text Conventions

This guide uses the following conventions:

- **Bold** indicates user actions For example:
  Type **0**, then press bold: **Enter** for each of the remaining fields
- *Italic* text indicates new or important words and is also used for emphasis. For example:
  Before analyzing, *always* prepare fresh matrix.

1.4 User Attention Words

Two user attentions words appear in WPI user documentation. Each word implies a particular level of observation or action as described below:

Note: Provides information that may be of interest of help but is not critical to the use of the product.

IMPORTANT! Provides information that is necessary for proper instrument operation, accurate operating procedures, or safe use of the robot.

1.5 Related Documentation

The paper contains numerous embedded links to GitHub repositories and some eternal links, some of which are private. Members of a DigSafe MQP team can access that code by logging into GitHub, for all other users the link will appear as not available.

The followings related documents are also utilized with this system but are not located specifically with a subsystem:

- For details on ***operating the Husky A100*** chassis see the Husky A100 User Guide. Important points include battery charging, battery placement, e-stop operation, general operation, and safety guidelines.
- "***Dig Safe®*** is a not-for-profit clearinghouse that notifies participating utility companies of your plans to dig. In turn, these utilities (or their contract locating companies) respond to mark out the location of their underground facilities. Dig Safe is a free service, funded entirely by its member utility companies."

  A link to their website can be found here: DigSafe
- ***Eversource,*** our sponsor, has a mission to bring a strong commitment to providing safe, reliable and sustainable electric, gas and water service.

  A link to their website can be found here: Eversource

2 Introduction

2.1 System Overview

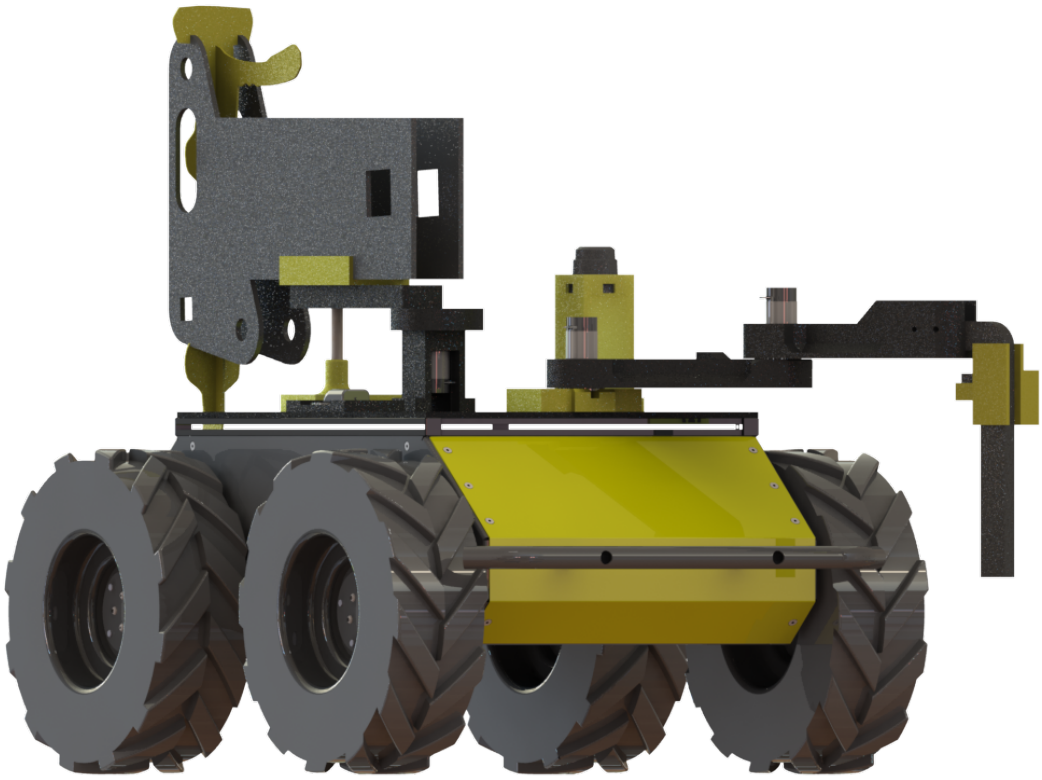


Figure 1: CAD render of DigSafe Buried Cable Detection robot May 2021

Through this project we seek to design and construct a robot which is capable of assisting Eversource and its technicians in their responsibility to mark buried power lines upon consumer request. Key to this goal is the intention to automate the parts of the cable-marking task which do not require human skills or judgement. This will allow the robot to work alongside a human operator, and free them to perform parts of a cable marking task which are more difficult and cannot be easily automated. In support of this objective of human-robot "coworking," the robot we seek to design must perform its tasks quickly and efficiently, at speeds at least equivalent to a human operator. Achieving this sort of speed would allow a technician-robot pair to complete cable marking tasks significantly faster than a single technician is able to currently, providing Eversource with expanded capacity to mark cables without requiring them to hire a larger number of technicians than they currently employ. Additionally, the development of user manuals, research of societal impacts, communication and project management, and navigating engineering projects while amidst COVID-19 will provide an opportunity to develop personally in our soft skills as an engineer.

3 Cable Detection Arm

3.1 Mechanical

The Cable Detection Arm consists of a Vivax-Metrotech vLoc3 Pro-RTK cable detection wand, a turntable system which rotates the wand across a 90 degree arc, and an Arduino-based microphone system which interprets the sound produced by the detection wand to determine the presence or absence of a cable.

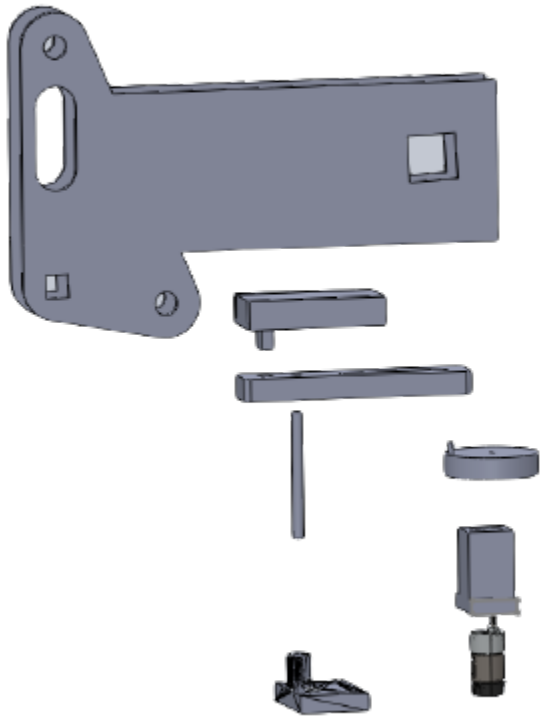


Figure 2: Exploded View of Cable Detection Wand

3.2 Detector

The Vivax-Metrotech vLoc3 Pro-RTK cable detector was supplied to us by Eversource along with charging hardware and a swappable battery. This detector provides a variety of features and operational modes, and has the capability to connect to GPS services and internet-based cloud services. The detector case also holds an additional operation manual.

3.2.1 Operational Mode

In order to work with the microphone circuit we developed, the cable detector must be in ”Omni-Peak” mode. This mode causes the detector to emit a 594Hz noise when it is not over an energized cable, and no noise when it is over one. Our microphone circuit was designed with this in mind.

3.2.2 Microphone Circuit

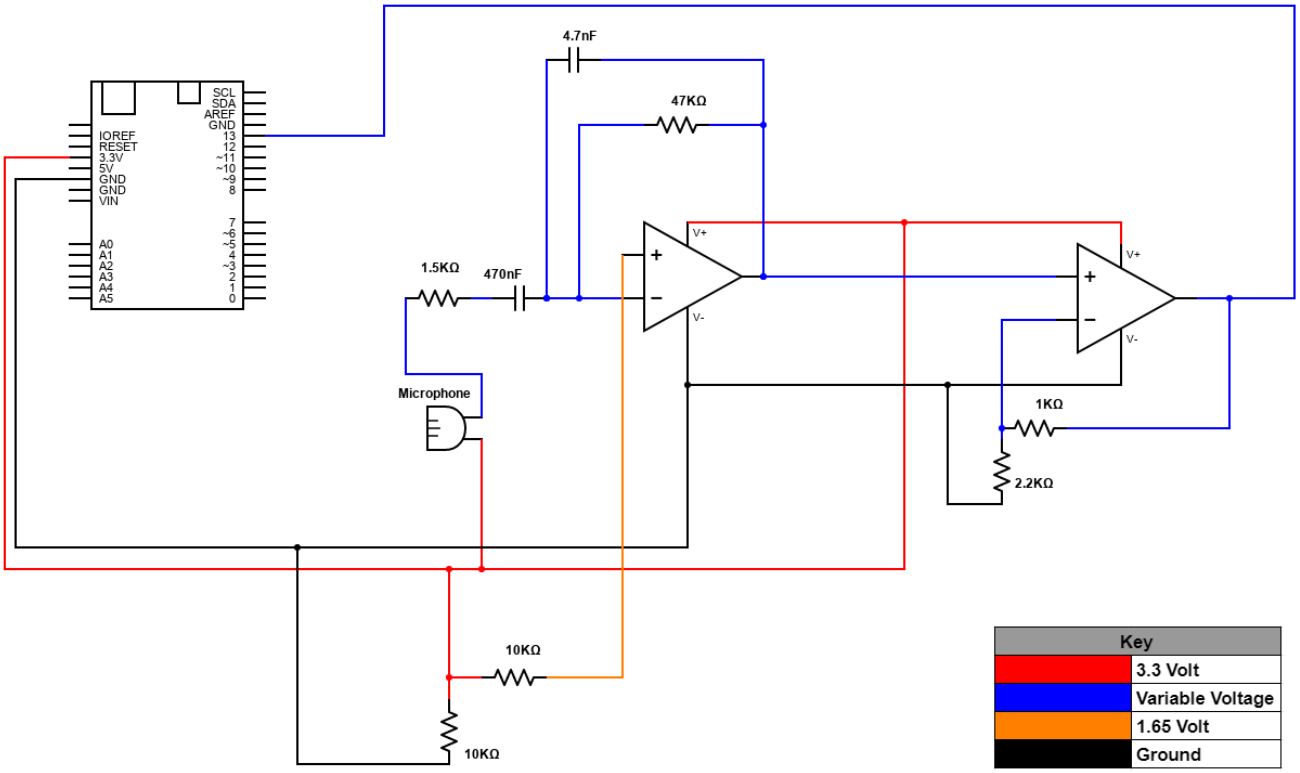


Figure 3: Circuit Diagram of the Cable Detection Sound Sensor

The circuit shown above contains two major components - a filter which isolates the 594Hz signal from the microphone, and an amplifier which scales the signal output by the filter to be from 0V to 5V, instead of from 3.3V to 5V. The microphone itself is currently mounted on a miniature breadboard within an enclosure designed to mount over the wand’s speaker.

Figure 4: Circuit and Microphone Mounted to Cable Detector

3.2.3 Codebase

The Arduino connected to the microphone circuit runs C code written in Visual Studio which runs a rolling sum of the last 50 voltage readings from the amplifier. If this sum is greater that .06V, it considers the sound to be on and publishes a uint16 ’1’ to the ”detector_msg” topic via rosserial. If this sum is less than .06V, it publishes a uint16 ’0’ instead.

4 Spray Arm

The following sections present the Spray Arm components in their state as of May 2021.The sections are broken down in the four major elements of mechanical, electrical, code base, and spray actuation.

Figure 5: Spray Arm attached to robot

4.1 Mechanical

The spray arm is a 2DoF arm with links of identical length. At present, the link lengths are decided by the print bed of a Prusa 3D printer (fitted to max X-Y capacity). The spray gun is attached at the end with a few screws pinning it into a slot.

4.1.1 Exploded View & BOM

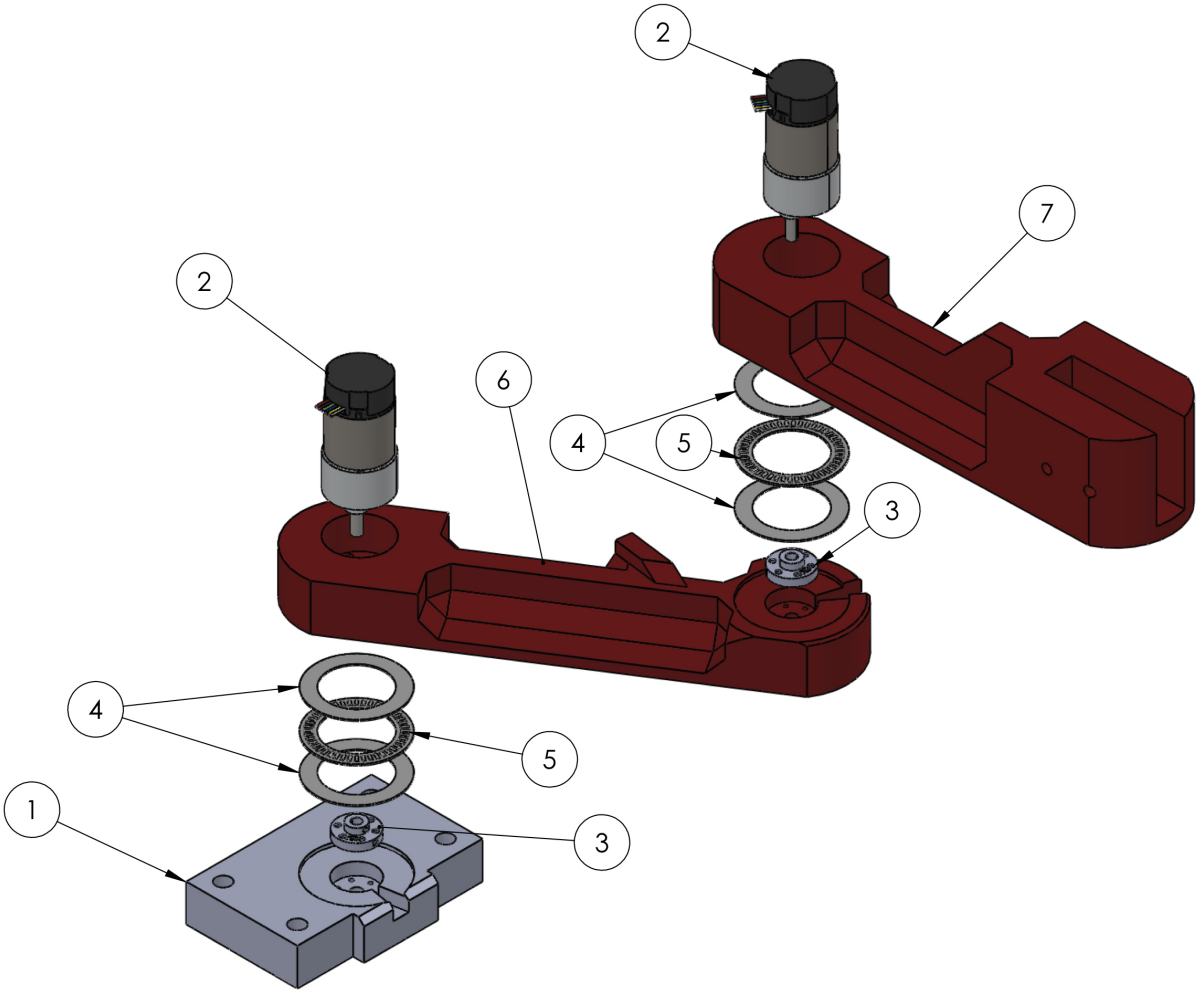


Figure 6: Exploded View of Arm Assembly

Num	Part	Quantity	Link
1	3D Printed Base	1	CAD
2	Pololu Gear Motors	2*	Pololu
3	Pololu Mounting Hub	2	Pololu
4	Thrust Bearing Washer	4	McMaster
5	Thrust Bearing	2	McMaster
6	3D Printed First Link	1	CAD
7	3D Printed Second Link	1	CAD

**The arm uses two different Pololu 37D Metal Gearmotors, a 131:1 gear ratio motor for the first link and a 70:1 motor for the second.*

4.1.2 Assembly of the Spray Arm

To duplicate, remodel, disassemble, or construct the spray arm, please see the assembly of the current spray arm.

To assemble the spray arm physical model:

1. Attach appropriate **37D motors** to *each* 3D printed link using M3 screws (Screwing farther than 3mm into the motor can impact the gearing).
2. Glue the **upper washers** to the bottom of *each* link.
   Note: We used a two part epoxy.
3. Attach the **mounting hubs** to the base and *first* link using M3 screws.
   IMPORTANT! Make sure the set screw holes are *lined up* the the notches to access them later.
4. Set a **washer and thrust bearing** into the *base and first link.*
   Neither was fastened down.
5. Put the **first link** on top of the base.
   Make sure the flat side of the motor's d-shaft is lined up with the mounting hub's set screw holes.
6. Insert both the upper and lower **set screws** using the access notches.
   Note: Loctite is highly recommended.
7. Repeat previous two steps attaching the second link to the first.
8. Mount **Limit Switches**, using plastic to plastic glue or M2 screws.
   Note: The current first link has a limit switch mount built into the print, but due to an oversight using it requires mounting the limit switch on the side.

4.2 Electrical

Control of the Spray Arm is done with a SAMD21 Dev board. Pin headers stuck in the breadboard offer easy connect/disconnect for the motors and limit switches.

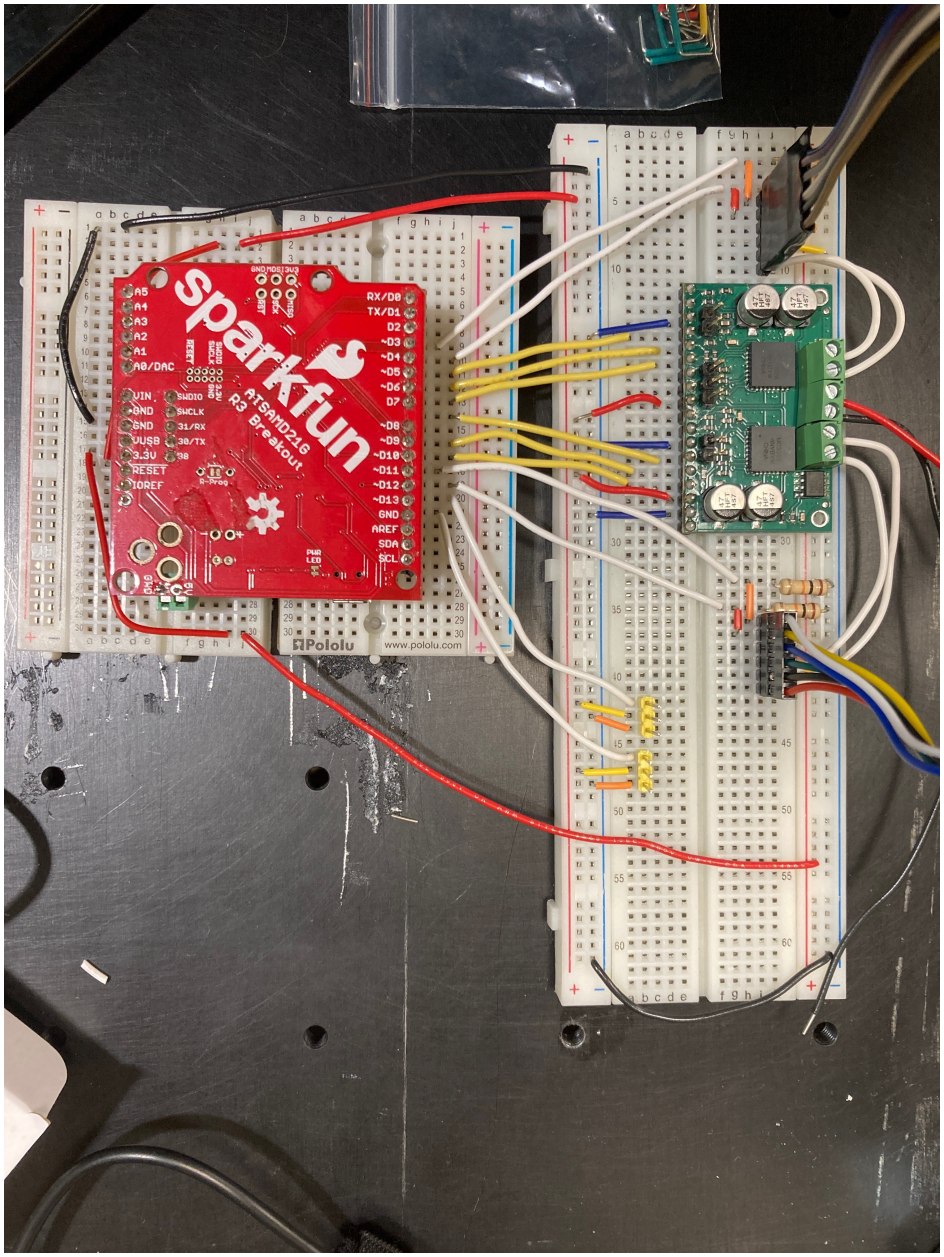


Figure 7: Spray Arm Control System

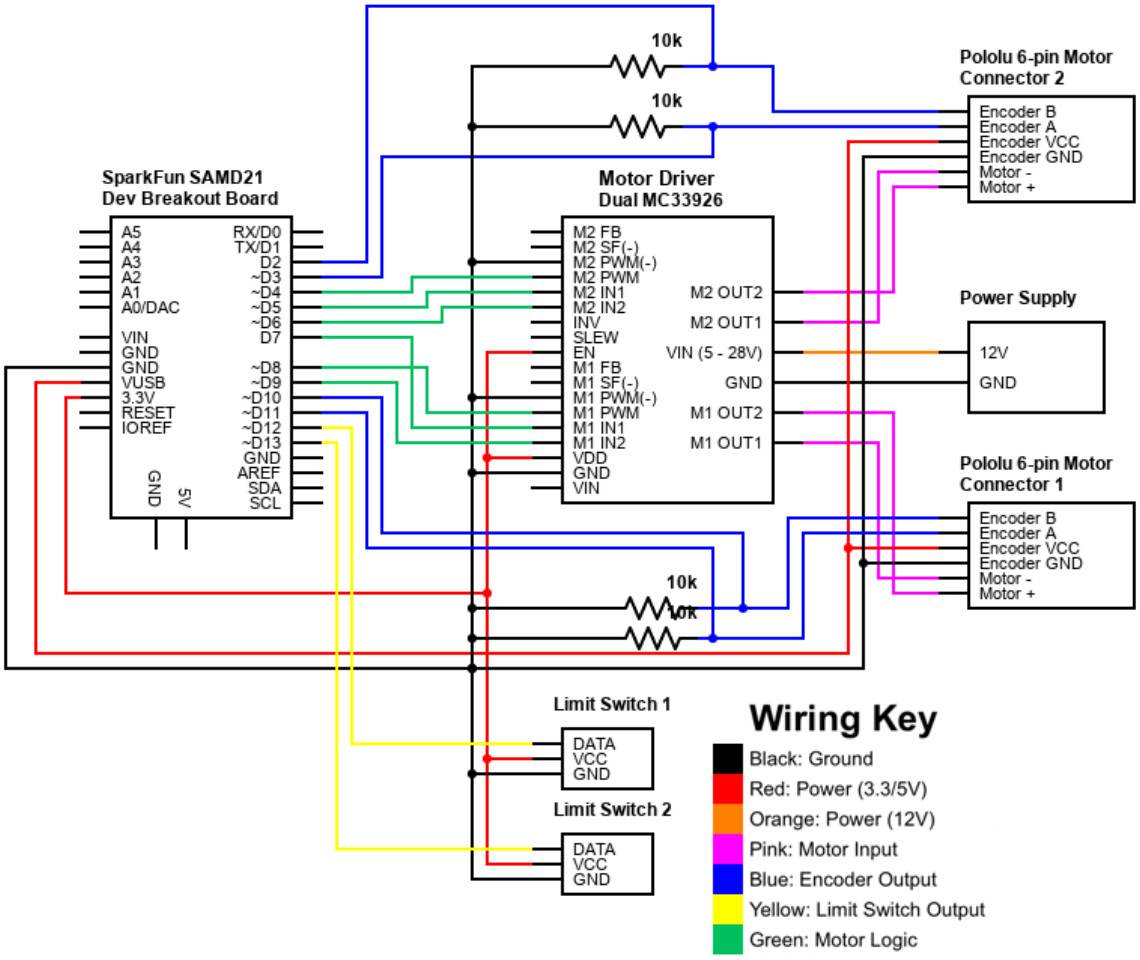


Figure 8: Spray Arm Control Wiring Diagram

4.2.1 Hardware

The following hardware was used as part of the control system.

Arduino: SparkFun SAMD21 Dev Breakout
Note: *The SparkFun SAMD21 Mini Breakout would also work and has a smaller footprint. Hookup guides for both can be found here.*

Dual H-Bridge: Dual MC33926 Motor Driver Carrier

4.2.2 Motor Connections

Each joint in the arm is directly rotated with a motor. The motors are two Pololu branded DC motor and are controlled with 12V output from the screw terminals on the MC33926. In order to control the motors:

- Each motor requires three (3) pins from the SAMD. Two (2) to indicate direction and one (1) for a PWM wave.
  Note: Make sure that one using a PWM capable pin on the SAMD
- The six(6) pin motor terminals are plugged directly into pin headers on the breadboard, as seen in Figure 8.

4.2.3 Encoders

Each motor has an inline quadrature encoder used for position control of the joint. The quadrature encoder power and output lines are also part of the six (6) pin motor terminals. To operate the encoders:

- Each encoder needs 5V power and a ground.
- The encoders have a ton of internal resistance, so a single 10K resistor tied to ground drops the output from 5V to 3.3V.
  This allows the 3.3V logic SAMD21 to read the value.

4.2.4 Limit Switches

There are two limit switches, used to calibrate the arm and prevent it crashing into the robot. We used these limit switches from Odoo. They include:

- 3.3V power
- ground
- data line

  Note: They are attached to the SAMD21 leading to a pin header for easy removal if needed.

4.3 Code Base

See the README for the ros_motor_controller GitHub repository for information regarding Arduino control of the Spray Arm.

We have also started implementing higher-level control located at the spray-control Github repository. This code is not completed, but outlines a method for sending motor control instructions from shapes to the 2DoF arm as shown in Figure 9. Additional node interactions and details are described in the `README.md` file of the related repository.

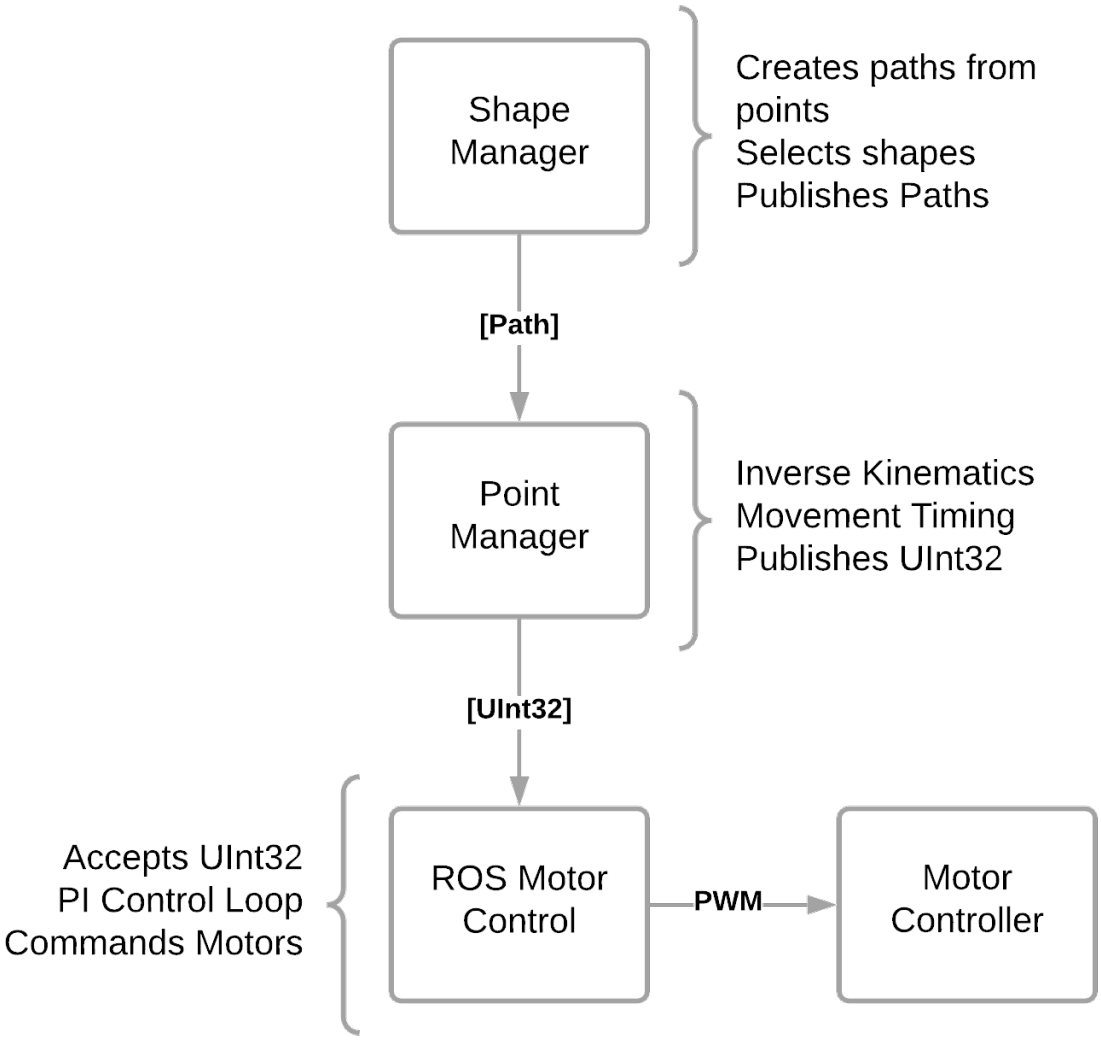


Figure 9: Spray Arm Node Interaction

4.4 Spray Actuation

The trigger is actuated with a 6V high powered servo from Odoo. This aluminum servo horn is used as well (the servo is powerful enough to destroy a plastic one). 40 degrees of travel is ideal for fully pressing and depressing the trigger.

Here is an image of the final system:

Figure 10: Spray Actuation System

5 Robot Control

5.1 Clearpath Husky Drivers

We used the Husky ROS packages, see 5.4 for more information.

5.2 Onboard Router

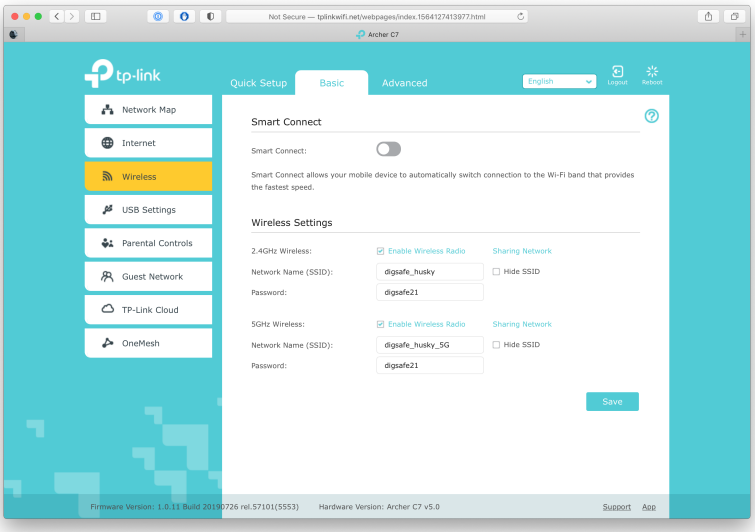
Figure 11: tp-link AC1750

The router located on board the robot is a tp-link Archer C7/AC1750. In order to connect to the router and use rosbridge v2 to communicate with the robot, you need to configure it correctly. The following section walks through some changes we made and highlights important numbers to remember.

When the router is first turned on it has the user define key parameters. These are currently configured as follows:
`Admin: DigSafeWPI`
`SSID: digsafe_husky`
`Password: digsafe21`
If the router is factory reset, you'll need to redefine these.

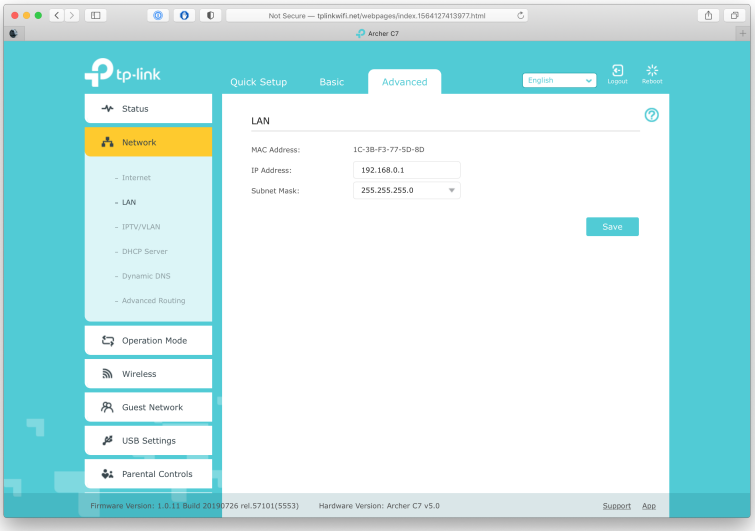
To access the router's settings, visit `tplinkwifi.net` after connecting to one of its Wi-Fi networks.

5.2.1 Configuring the local network



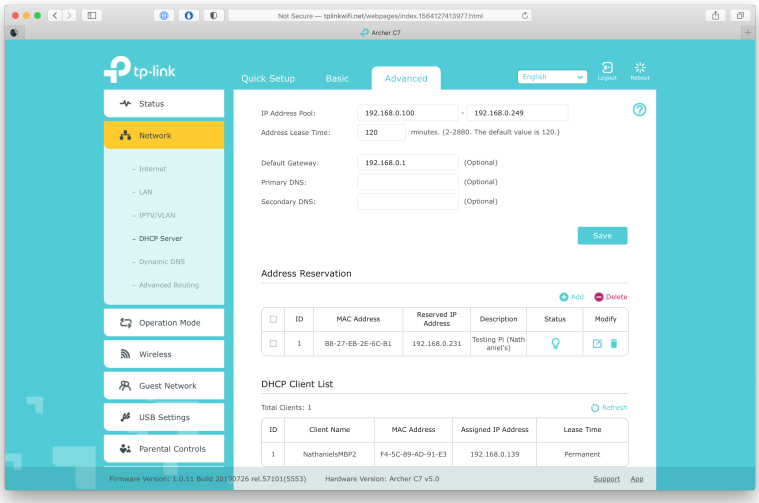
You can change the network names and passwords using this menu. It's suggested you have separate 2.4 and 5.2 GHz networks because some devices, such as a Raspberry Pi, only support 2.4.

5.2.2 Router IP



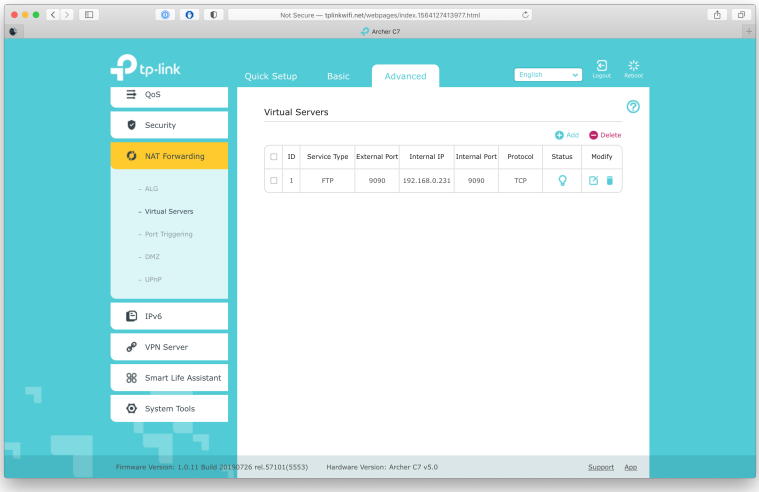
You'll want to note the IP address being used by the router.

5.2.3 Static IP



This menu is used to force a connected computer to always be assigned the same internal IP. In the screenshot above, a Raspberry Pi has been assigned `192.168.0.231` as its static internal IP. Which IP number isn't important, but you'll want a consistent IP to port forward and to connect with rosbridge. Make sure to note the exact local IP of the Jetson for use elsewhere.

5.2.4 Port Forwarding



To use rosbridge you'll also need to forward the port being used as a websocket to allow access to devices other than the Jetson. By default that's port 9090, which is being forwarded in the screen shot above.

5.3 Cellular Modem

Figure 12: Netgear Cellular Modem

For the robot we used a NETGEAR 4G LTE Modem. After purchasing a prepaid 4G capable sim card, you'll want to activate it inside the modem through the provider's website. To do so will require the ICCID located on the card and the modem's IMEI, which can be found on the bottom of the modem. Once a cell signal is received, the modem should be connected to the router with an Ethernet cable to the router's "modem" port.

5.4 Jetson TX2

The following are steps for installing Ubuntu 18.04 and all needed programs/files to get started working with the Husky A100 using ROS.

To install Ubuntu 18.04 on the Jeston TX2:

1. Download L4T 32.5
2. Download and Install Jetpack
   Note: See How to install JetPack for installing JetPack.
3. Place the **Jetson** on a flat work surface outside of the robot
4. Connect the **Jetson** to a monitor
5. Connect the **Jetson** to a mouse + keyboard combination device, or a USB hub and then a mouse and Keyboard to the USB hub
6. Run **Ubuntu 18.04** on a host computer
7. Install Nvidia SDK Manager
8. Register for an **Nvidia Developer account** if you have not already
9. Connect the **Jetson** to the host computer via a USB to micro USB cable (included with the Jetson)
10. Connect the **Jetson** to wall power using the included adapter
11. Place the **Jetson** in recovery mode by pressing and holding the REC button
12. Wait until the **Jetson** reboots to recovery mode.
    There should be a black screen on the connected monitor
13. Open the **SDK manager** on the host computer.
14. Set the **target hardware** to JetsonTX2 (it should automatically detect one connected)
15. Click **continue,** and accept the terms
16. Click **continue,** and input your Sudo password if necessary
17. Wait for the process to finish. When it is done, the Jetson should boot through the standard Ubuntu installation process.
18. Accept the licenses, and **configure** the Jetson as you would any other Linux machine
19. Once setup is complete, **run ifconfig** to find its IP address, and connect it to the internet
20. Once this process is finished, **click finish** on SDK manager
21. Run the following **command in a terminal** window on the Jetson:

    ```
    $ curl -s https://raw.githubusercontent.com/clearpathrobotics/jetson_setup/
        melodic/scripts/tx2_setup.sh | bash -s -- bash HUSKY$_$SETUP.sh
    ```

    This will install the Husky control packages on the Jetson, along with ROS and other required programs
22. Wait for the **Jetson** to reboot. After this completes, it will be ready for installation into the Husky
23. Disconnect the **Jetson** from wall power after the reboot
24. Place the **Jeston** inside of the Husky's service bay

25. Connect the **Jetson** to the husky's power supply with the power connector in the service bay
26. Connect the **Jetson** to the husky using an RS232 to USB converter to connect it the RS232 jack in the service bay
27. Using the (known) IP of the Jetson, you may now **remote into the Jetson** and control the husky.

5.5 LiDAR

The robot is currently using a RPLidar S1 for navigation. Slamtec distributes a ROS package for all of their LiDAR products that integrates well with Clearpath's navigation stack and rViz. The LiDAR is mounted on a 3D printed base that elevates it over the spray arm.
Our fork: rplidar_ros

5.6 GPS

The robot currently uses the Sparkfun GPS-RTK2 Board for GPS localization. Our implementation also made use of an antenna and ground plate, purchased through Sparkfun.
Note: This implementation requires the use of a USB-C connection.

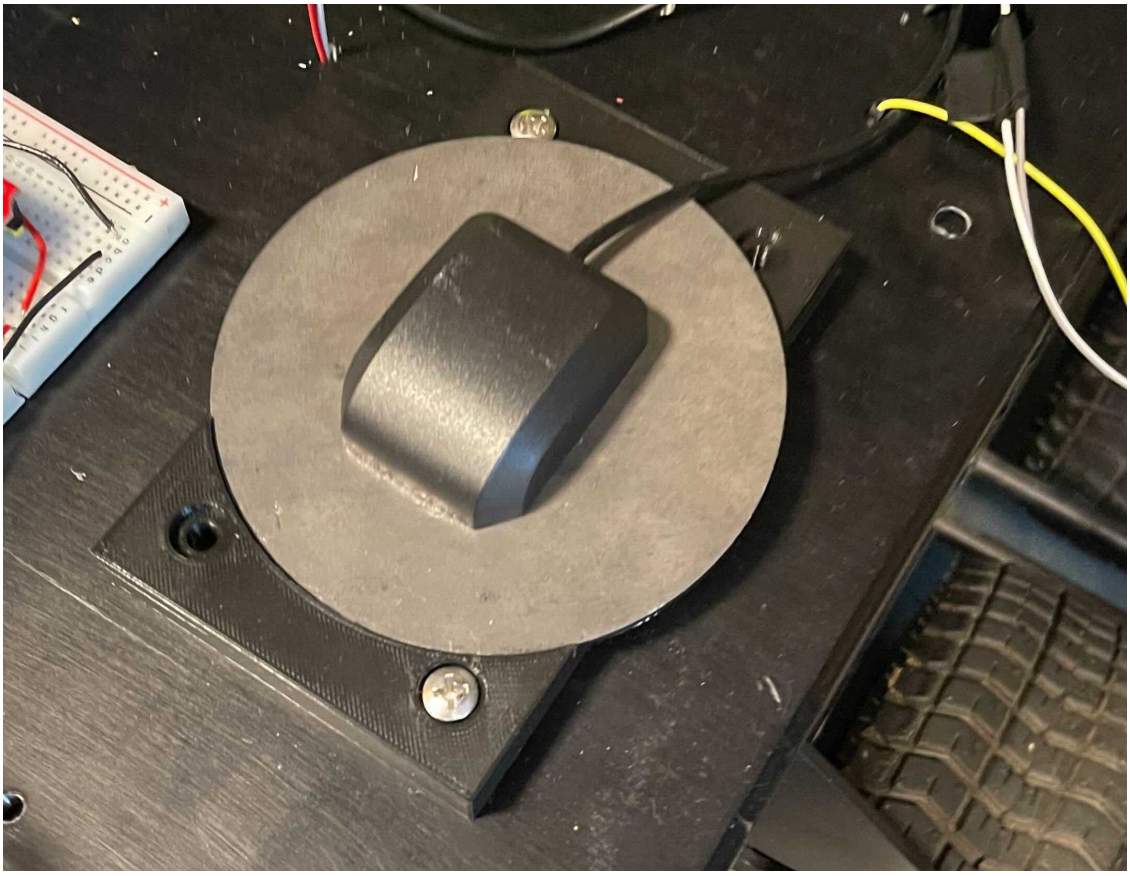


Figure 13: GPS Antenna Mounted to Robot

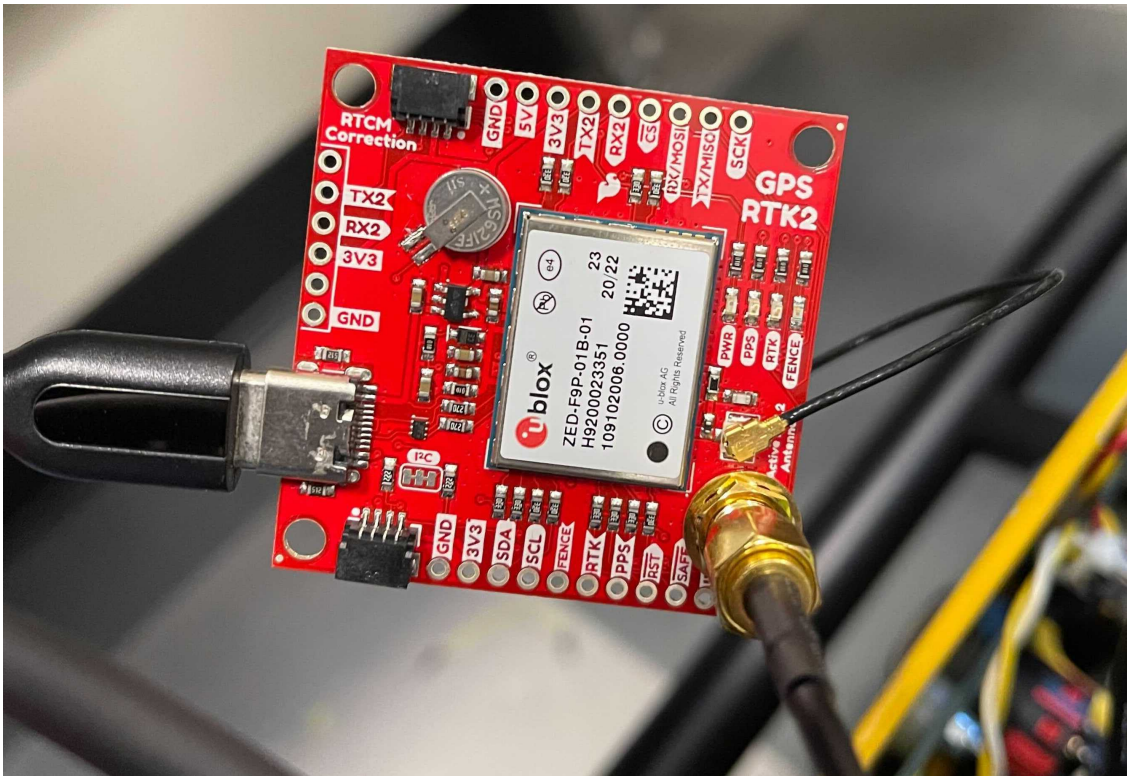
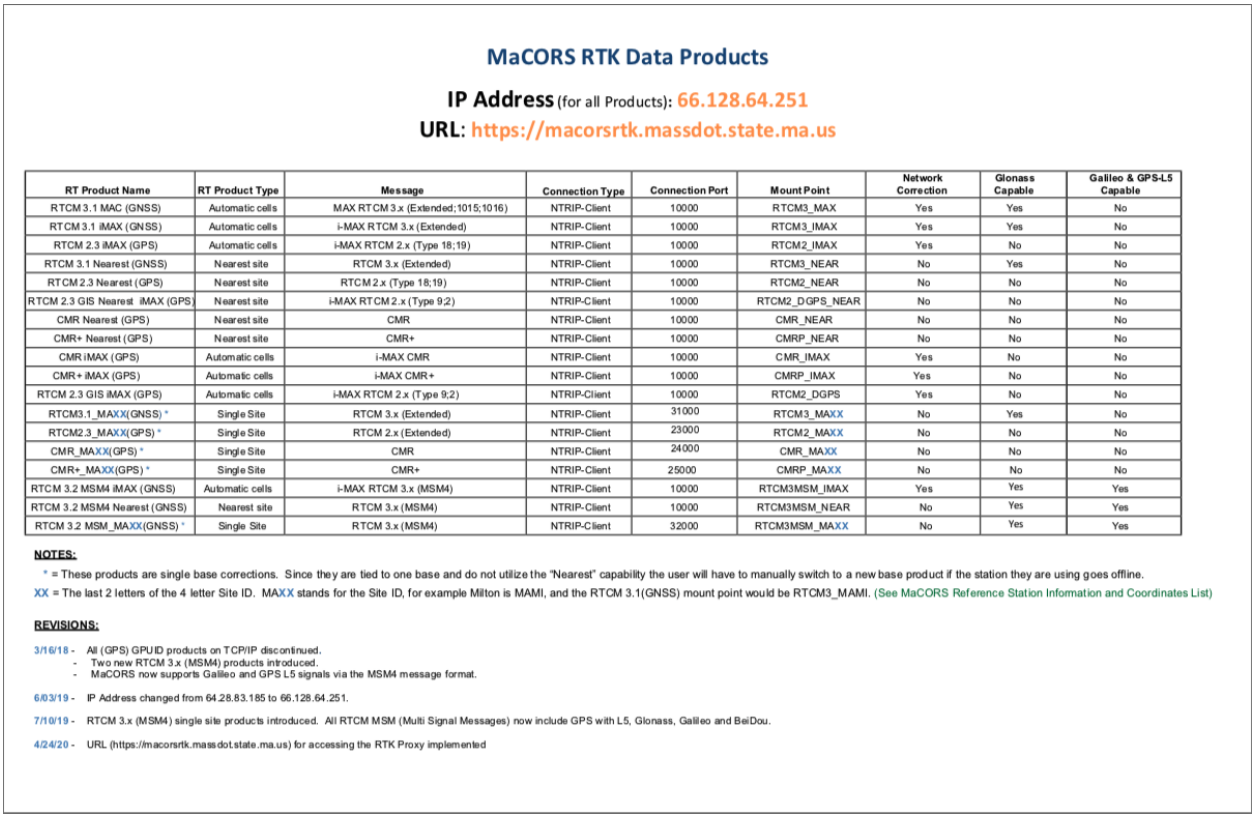


Figure 14: GPS Chip with Antenna Connected

5.6.1 Setting up for RTK

The Sparkfun GPS is set up to achieve centimeter level precision using ”real time corrections messages” or RTCM. In the state of Massachusetts, the provider of these is MassDOT through the MaCORS network. These corrections are available for free, but you’ll need to sign up for an account on their website. Alongside your account username and password, you’ll need the url, port, and mountpoint for the corrections source you want to use. That info can be found in the tables below. We found Northborough to be the closest site to WPI.



MaCORS RTK Data Products

IP Address (for all Products): 66.128.64.251

URL: https://macorsrtk.massdot.state.ma.us

RT Product Name	RT Product Type	Message	Connection Type	Connection Port	Mount Point	Network Correction	Glonass Capable	Galileo & GPS-L5 Capable
RTCM 3.1 MAC (GNSS)	Automatic cells	MAX RTCM 3.x (Extended;1015;1016)	NTRIP-Client	10000	RTCM3_MAX	Yes	Yes	No
RTCM 3.1 iMAX (GNSS)	Automatic cells	i-MAX RTCM 3.x (Extended)	NTRIP-Client	10000	RTCM3_IMAX	Yes	Yes	No
RTCM 2.3 iMAX (GPS)	Automatic cells	i-MAX RTCM 2.x (Type 18;19)	NTRIP-Client	10000	RTCM2_IMAX	Yes	No	No
RTCM 3.1 Nearest (GNSS)	Nearest site	RTCM 3.x (Extended)	NTRIP-Client	10000	RTCM3_NEAR	No	Yes	No
RTCM 2.3 Nearest (GPS)	Nearest site	RTCM 2.x (Type 18;19)	NTRIP-Client	10000	RTCM2_NEAR	No	No	No
RTCM 2.3 GIS Nearest iMAX (GPS)	Nearest site	i-MAX RTCM 2.x (Type 9;2)	NTRIP-Client	10000	RTCM2_DGPS_NEAR	No	No	No
CMR Nearest (GPS)	Nearest site	CMR	NTRIP-Client	10000	CMR_NEAR	No	No	No
CMR+ Nearest (GPS)	Nearest site	CMR+	NTRIP-Client	10000	CMRP_NEAR	No	No	No
CMR iMAX (GPS)	Automatic cells	i-MAX CMR	NTRIP-Client	10000	CMR_IMAX	Yes	No	No
CMR+ iMAX (GPS)	Automatic cells	i-MAX CMR+	NTRIP-Client	10000	CMRP_IMAX	Yes	No	No
RTCM 2.3 GIS iMAX (GPS)	Automatic cells	i-MAX RTCM 2.x (Type 9;2)	NTRIP-Client	10000	RTCM2_DGPS	Yes	No	No
RTCM3.1_MAXX(GNSS) *	Single Site	RTCM 3.x (Extended)	NTRIP-Client	31000	RTCM3_MAXX	No	Yes	No
RTCM2.3_MAXX(GPS) *	Single Site	RTCM 2.x (Extended)	NTRIP-Client	23000	RTCM2_MAXX	No	No	No
CMR_MAXX(GPS) *	Single Site	CMR	NTRIP-Client	24000	CMR_MAXX	No	No	No
CMR+_MAXX(GPS) *	Single Site	CMR+	NTRIP-Client	25000	CMRP_MAXX	No	No	No
RTCM 3.2 MSM4 iMAX (GNSS)	Automatic cells	i-MAX RTCM 3.x (MSM4)	NTRIP-Client	10000	RTCM3MSM_IMAX	Yes	Yes	Yes
RTCM 3.2 MSM4 Nearest (GNSS)	Nearest site	RTCM 3.x (MSM4)	NTRIP-Client	10000	RTCM3MSM_NEAR	No	Yes	Yes
RTCM 3.2 MSM_MAXX(GNSS) *	Single Site	RTCM 3.x (MSM4)	NTRIP-Client	32000	RTCM3MSM_MAXX	No	Yes	Yes

NOTES:

* = These products are single base corrections. Since they are tied to one base and do not utilize the "Nearest" capability the user will have to manually switch to a new base product if the station they are using goes offline.

XX = The last 2 letters of the 4 letter Site ID. MAXX stands for the Site ID, for example Milton is MAMI, and the RTCM 3.1(GNSS) mount point would be RTCM3_MAMI. (See MaCORS Reference Station Information and Coordinates List)

REVISIONS:

3/16/18 - All (GPS) GPUID products on TCP/IP discontinued.
- Two new RTCM 3.x (MSM4) products introduced.
- MaCORS now supports Galileo and GPS L5 signals via the MSM4 message format.

6/03/19 - IP Address changed from 64.28.83.185 to 66.128.64.251.

7/10/19 - RTCM 3.x (MSM4) single site products introduced. All RTCM MSM (Multi Signal Messages) now include GPS with L5, Glonass, Galileo and BeiDou.

4/24/20 - URL (https://macorsrtk.massdot.state.ma.us) for accessing the RTK Proxy implemented

Figure 15: Data Stream Options from MaCORS

MaCORS Reference Station Information and Coordinates

NAD83 (2011) Epoch 2010.00

Site ID	Site Name	Ref. Station Number	Latitude	Longitude	Ellipsoid Height (m)	Antenna Type	
MABN	Bernardston	21	42 40 11.99123 N	072 32 28.64352 W	94.903	LEIAR20	LEIM
MABT	Belchertown	22	42 16 56.08016 N	072 22 54.62401 W	120.855	LEIAR20	LEIM
MACM	Chatham	57	41 41 28.68811 N	069 58 01.53155 W	-12.946	LEIAR20	LEIM
MADA	Dartmouth	54	41 38 22.84840 N	071 01 41.39352 W	6.401	LEIAR20	LEIM
MAFA	Falmouth	55	41 37 11.07617 N	070 32 27.30426 W	-1.458	LEIAR20	LEIM
MAGS	Goshen	12	42 27 19.96844 N	072 49 56.37491 W	330.987	LEIAR20	LEIM
MAMA	Manchester	43	42 35 21.26090 N	070 47 14.34981 W	-6.303	LEIAR20	LEIM
MAMI	Milton	61	42 16 19.38615 N	071 02 55.24967 W	-17.474	LEIAR20	LEIM
MANB	Northborough	32	42 17 02.62236 N	071 39 40.52726 W	69.59	LEIAR20	LEIM
MANT	Nantucket	56	41 16 10.14706 N	070 05 00.87511 W	-15.527	LEIAR20	LEIM
MAPL	Plymouth	52	41 56 19.31493 N	070 39 18.24317 W	11.752	LEIAR20	LEIM
MASA	Salisbury	42	42 51 45.88632 N	070 53 24.94587 W	-10.292	LEIAR20	LEIM
MASB	Sturbridge	33	42 06 41.08134 N	072 05 13.98517 W	159.565	LEIAR20	LEIM
MASH	Sheffield	13	42 08 25.75403 N	073 21 51.06334 W	175.586	LEIAR20	LEIM
MATB	Tewksbury	41	42 37 48.00284 N	071 16 17.32162 W	13.188	LEIAR20	LEIM
MATU	Truro	53	41 58 51.70819 N	070 02 36.89118 W	13.348	LEIAR20	LEIM
MAWM	Westminster	31	42 33 40.62113 N	071 55 59.20751 W	317.179	LEIAR20	LEIM
MAWR	Wrentham	51	42 02 30.13370 N	071 18 15.03938 W	36.694	LEIAR20	LEIM
MAWS	West Springfield	23	42 08 14.40994 N	072 37 26.93663 W	-3.115	LEIAR20	LEIM
MAWT	Williamstown	11	42 38 21.24510 N	073 13 37.81371 W	271.513	LEIAR20	LEIM

NOTES:

- All sites offer GNSS data
- Site IDs in Green are part of the NGS National CORS Network

Figure 16: Available Reference Stations

5.6.2 Testing GPS chip on Windows

To test the GPS on its own you can use u-center.

This is a GNSS evaluation tool made by the company responsible for the internals of our Sparkfun board.

You can find a tutorial walking though the setup and connection process on Sparkfun's website.
Note: Although that tutorial is for an older board, the steps are the same.

After setting everything up, our output in the lab looked like this:

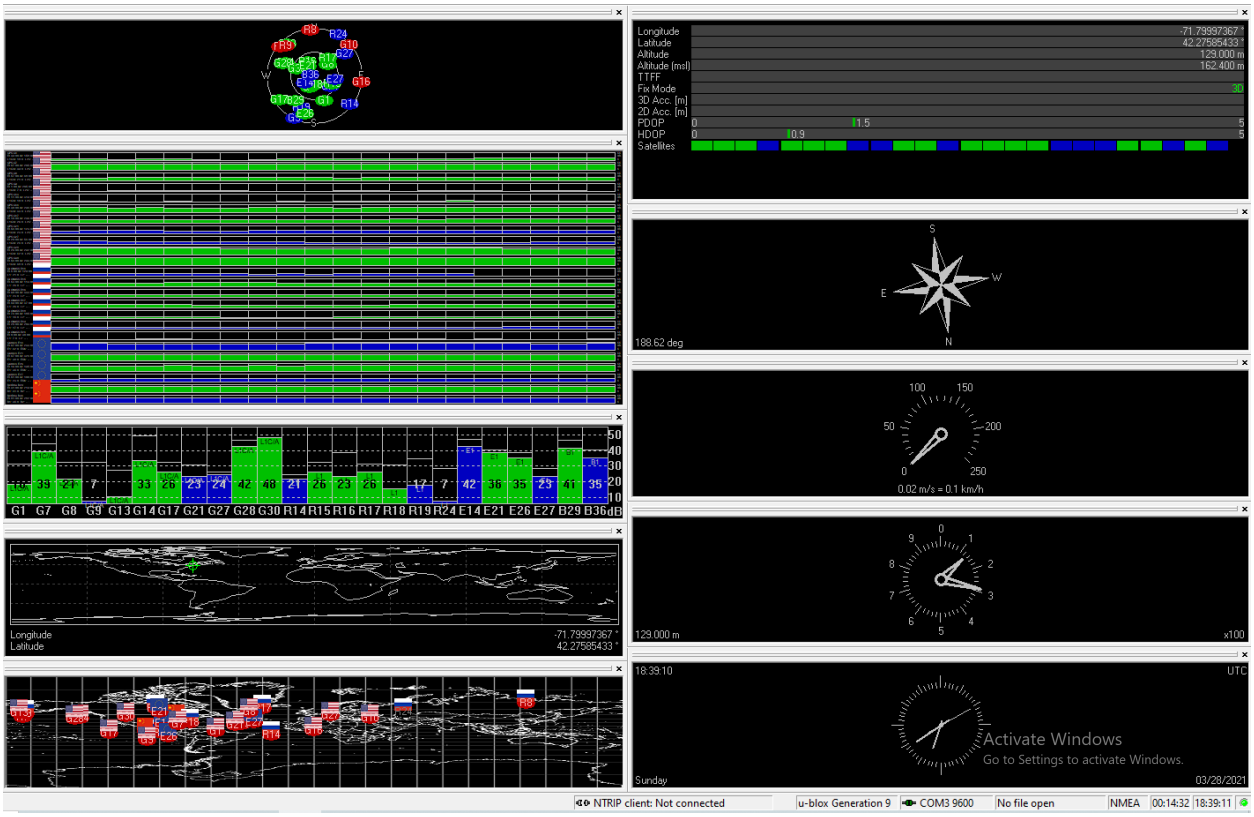


Figure 17: u-center from 85P MQP Lab

Then you can add RTCM input to take advantage of the board's RTK Capabilities.

To do this we used RTKLIB, which is a package of open-source RTK utilities.
Note: You'll also need a Serial-to-USB adapter.

Another Sparkfun Tutorial walking though the process of wiring the adapter and setting up RTKNAVI can be found here.

For comparison, our setup screens looked like this:

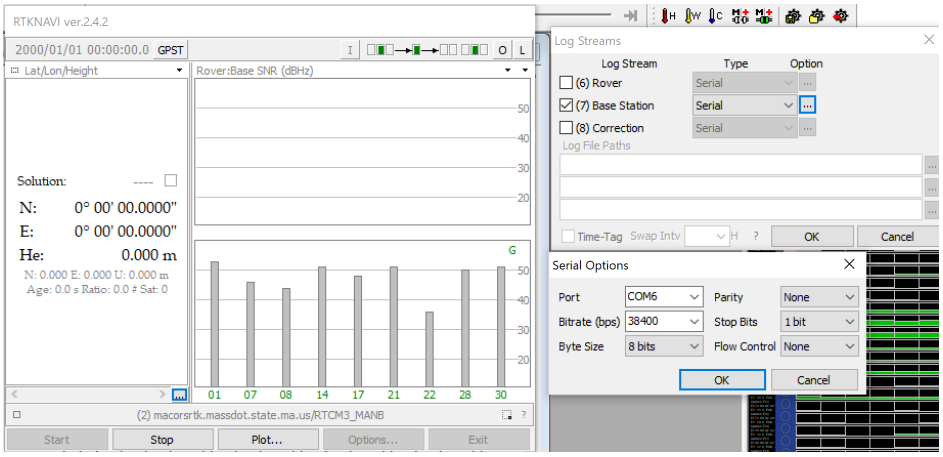


Figure 18: NTKNAVI Serial Settings

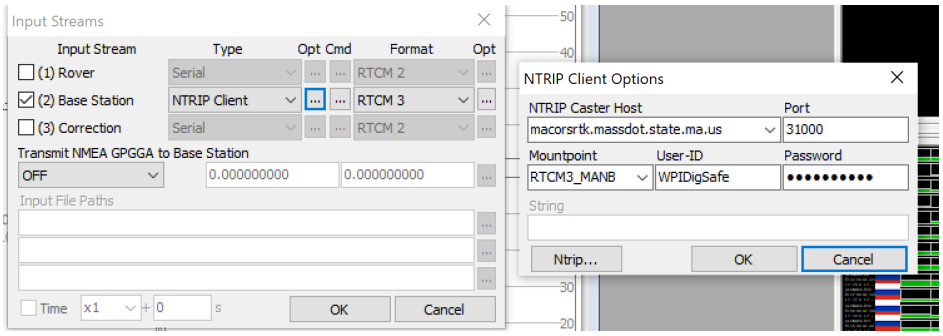


Figure 19: NTKNAVI NTRIP Settings

If the RTK light on the GPS starts flashing, it's successfully receiving RTCM3 messages.
Note: If it turns off, it has a RTK fix, although this probably won't occur in the lab

5.6.3 GPS Through ROS

The robot uses a set of ROS drivers from ros-agriculture to communicate with the GPS and pass it RTK corrections. Our forks of those ROS nodes can be found below.
ublox_f9p (GPS Driver)
rtcm_msgs (Driver Dependency)
ntrip_ros (RTCM Corrections)

That last one will require a custom launch file indicating your MaCORS account details and the corrections source you wish to use. Below you'll find our launch file for reference.

```
<?xml version="1.0" encoding="UTF-8"?>

<launch>
  <node pkg="ntrip_ros" type="ntripclient.py" name="ntrip_ros" output="screen">
          <param name="rtcm_topic" value="/rtcm"/>
          <param name="ntrip_server" value="macorsrtk.massdot.state.ma.us:31000"/>
          <param name="ntrip_user" value="username"/>
          <param name="ntrip_pass" value="password"/>
          <param name="ntrip_stream" value="RTCM3_MANB"/>
          <param name="nmea_gga"
          ↪  value="$GPGGA,202506.081,4216.550,N,07147.990,W,1,12,1.0,0.0,M,0.0,M,,*72"/>
  </node>
</launch>
```

5.7 Rosbridge

The robot uses the Rosbridge v2 protocol to communicate with the control application (which will be detailed in the following section) by sending JSON files over a WebSocket connection hosted on-board the robot. Details about how to install and run a Rosbridge server can be found on the ROS wiki.

Currently the DigSafe robot hosts its RosBridge WebSocket on port 9090. To open that server on it's own run the following command on the Jetson.

```
$ roslaunch rosbridge_server rosbridge_websocket.launch
```

5.8 Raspberry Pi 4

The robot has a Raspberry Pi 4 on board acting as a second ROS node which handles the GPS ROS packages (see 5.6.3 for details on those packages and how to run them).

To set up the Raspberry Pi 4 and using it as a second ROS node:

1. Download the 18.04.5 LTS Server image
2. Flash the **Raspberry Pi** using the Raspberry Pi Imager
3. Follow the standard Ubuntu Server setup steps
4. From the fresh installation, **run** the following **commands** to setup ROS, setup ROS master connection to the Jetson TX2, and install needed packages:

```
sudo add-apt-repository universe
sudo add-apt-repository multiverse
sudo add-apt-repository restricted
sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb_release -sc) main" >
↪  /etc/apt/sources.list.d/ros-latest.list'
sudo apt-key adv --keyserver 'hkp://keyserver.ubuntu.com:80' --recv-key
↪  C1CF6E31E6BADE8868B172B4F42ED6FBAB17C654
sudo apt update
sudo apt install ros-melodic-ros-base
echo "source /opt/ros/melodic/setup.bash" >> ~/.bashrc
echo "export ROS_MASTER_URI='http://digsafe:11311'" >> ~/.bashrc
sudo echo "192.168.0.242 digsafe" >> /etc/hosts
source ~/.bashrc
sudo apt install python-rosdep python-rosinstall python-rosinstall-generator
↪  python-wstool build-essential
```

   Note: If needed, replace the IP address with the corresponding IP address of the Jetson TX2

   The Raspberry Pi 4 is now setup and ready to communicate and run distributed ROS nodes.

6 Control Application

For a more in depth tutortial on using ArcMap see the ArcGIS Desktop 10.8 quick start guide.

6.1 ArcMap Setup

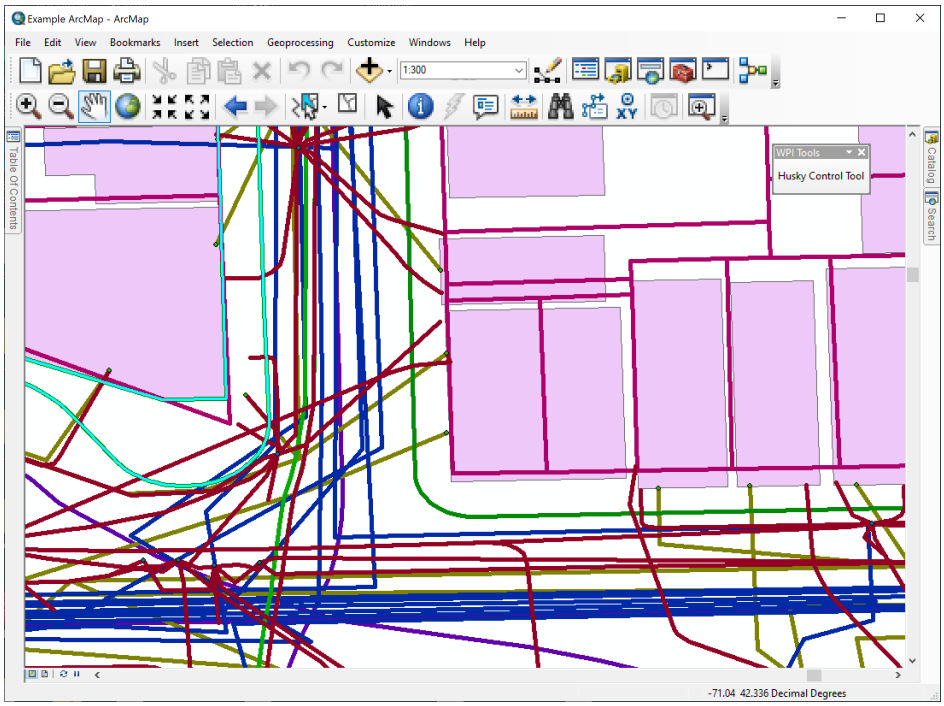


Figure 20: ArcMap Main UI

To properly use the extension, you will need to install python packages to the ArcMap python environment. To install the required packages please use the Manual Install or Pip Install sections below:

6.1.1 Manual Install

To install packages:

1. Open the **python terminal** within ArcMap (see Figure 21)

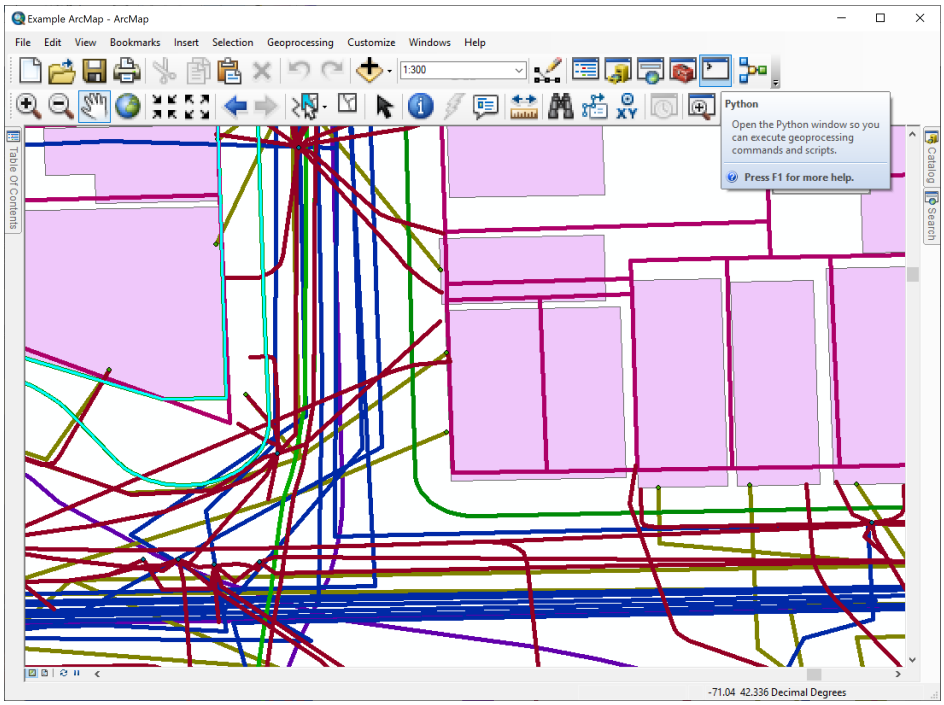


Figure 21: ArcMap Open Internal Python Terminal

2. Run the following **command** in the python terminal (Fig 22) to locate the ‘site-packages‘ directory

```
>> sys.path
```

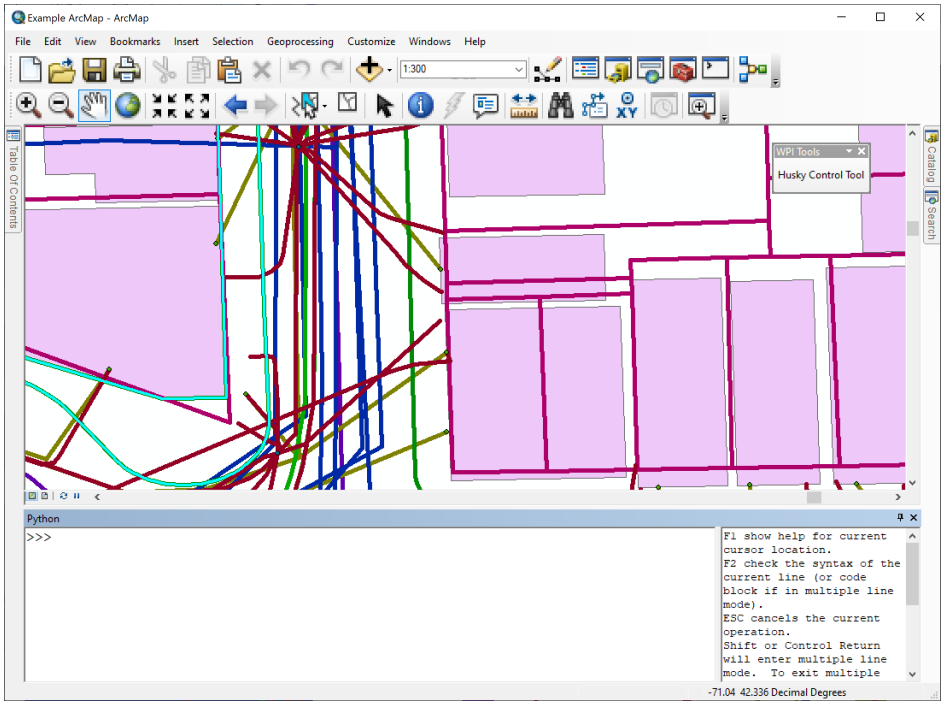


Figure 22: ArcMap Python Terminal

3. Copy and paste any needed **packages** to the 'site-packages' directory, ensuring that the top level of the package contains an `__init__.py` file

6.1.2 Pip Install

Alternatively, you can also install them using the pip package manager.
To install pip:

1. Open a **command prompt**
2. Navigate to `C:\Python27\ArcGIS10.8\Scripts` in the command prompt
3. Run the following **commands** to install the required packages at their specific versions (*These versions support Python 2.7, which is the latest version ArcMap can run*)

```
> pip.exe install autobahn==19.11.2
> pip.exe install incremental==17.5.0
> pip.exe install roslibpy==1.1.0
> pip.exe install twisted==20.3.0
```

6.2 Developing Add-Ins for ArcMap

For detailed instructions on creating a Python Add-In for ArcMap see Creating a Python add-in application extension from ESRI.

Our Add-In can be found here. The key file is located at `Install/ArcGISAddins_addin.py`.

To install the Add-In:

1. **Save** your changes to the Python file
2. Optional: change **version number** in `config.xml`
3. Run the **Python** file `makeaddin.py` to generate the extension
4. **Double-click** the newly generated **file** `ArcGISAddins.esriaddin` to install in ArcMap