

THE VICON POSITIONING SYSTEM

A Dummy Guide On Using VICON Positional System and Pixhawk6C Flight Controller For Indoor Flight Localisation

LAST UPDATED

9th May 2026

By Matthew Tam & Nura Efendic — The dummies
MSc Students, Autonomous Vehicle Dynamics and Control, Cranfield
University
Matthewchtam@gmail.com

Awaiting final reviewing: Grace Elisha Mwesigwa, Luis Leandro De Paula
PhD Students, Centre for Autonomous and Cyber-physical Systems, Cranfield
University

Contents

1	Pre word	2
2	The Big Picture	3
3	Operating VICON	5
3.1	The Hardware	5
3.2	Turn on Vicon cameras and Host PC	5
3.3	Create a Camera Mask	7
3.4	How to turn on the lights of Flight Arena	8
3.5	Calibrate the Cameras	8
3.6	Set Origin	12
3.7	Defining New Objects	14
3.8	Finishing	19
4	Ground Control Station Setup	20
4.1	Hardware Required	20
4.2	Setting Up Conda Virtual Environment	20
4.3	Downloading VICON SDK	21
4.4	Downloading pyvicon	27
4.5	Merging VICON SDK into pyvicon	28
4.6	Installing Dependencies Packages	30
4.7	Setting up pyvicon	30
4.8	Installing Mavproxy & Mavlink	31
4.9	Network setup	32
5	Drone Setup	35
5.1	Setting up the ESP8266	36
5.2	MissionPlanner Configuration	39
5.3	Mavproxy Telemetry	42
5.4	IR Markers	43
5.5	Final Intergration	43
5.6	Finishing	46
6	Concluding Remark	47

1 Pre word

It is to my astonishment when I first found out during my Group Design Project (GDP) that our most advanced flight arena in Cranfield does not have an actual written guide on using one of the most, if not, THE most valuable system within the flight arena, the VICON positioning system. For the duration of my GDP I've been asking around and seems like each PhD Student have their own way of connecting to VICON, which had led to a very incoherent and beginner unfriendly approach in using VICON when starting out. Hence, the creation of this guide! This guide's content is sourced from words of mouth from previous GDP member and PhD students, as well as some heuristic struggle from yours truly. By the end of this guide, the goal is that future cohorts from the MSc programme can figure out how to setup the Vicon Positioning System, setting up the Ground Control Station, setting up the drone, and implement custom pathplanning without doing much of the painful debugging progress or asking around.

I would like to give a special thanks to Grace Elisha, Pol Padilla Moure, Luis Paula, and Ugurcan Celik for teaching me how this system works.

For any non-dummies that understand this system much better than yours truly, please contact me from the email above for technicality refinement/comments/abuse.



Figure 1: Harry's 1st Vicon Calibration Wand

2 The Big Picture

So... what is the Vicon Positioning System? It is a system that captures the drone's position by using an assortment of Infrared Red (IR) cameras that track reflective IR markers mounted on the drone. The arrangement of cameras inside the flight arena, mounted on the ceiling, captures the visual feed and is then forwarded to a D-Link switch, which aggregates the camera feeds and sends them onwards to the Vicon Host PC. Inside the Host PC, the Vicon Tracker App computes the compiled visual footage and calculate the UAV's position, finally broadcasting the positional pose as a continuous string data.

There are many ways you can receive this continuous string of data from the Host PC. You can either do it wirelessly or wired. For this guide, we will focus on doing this wirelessly via an ethernet port.

You will need to setup a Ground Control Station (GCS) that serves as the receiving end of this ethernet connection, this is done in a Linux running laptop here. The GCS will reparse the raw positional data received into MAVLINK protocol messages, a protocol which majority of drone Flight Controller (FC) uses and understands. Once parsed into the correct protocol, we transmit these data to the drone wirelessly via WiFi.

Once the drone receives this parsed positional data, it could be applied for different purposes. In this guide, we will briefly mention how to use this for localisation, path following. The overall architecture is visualised in 2:



Figure 2: A high level flow diagram on the communication for localisation.

The wireless telemetry between the GCS and Pixhawk FC uses an ESP8266 WiFi module, jumped directly into the telemetry port of a Pixhawk6C FC, as seen in figure 3

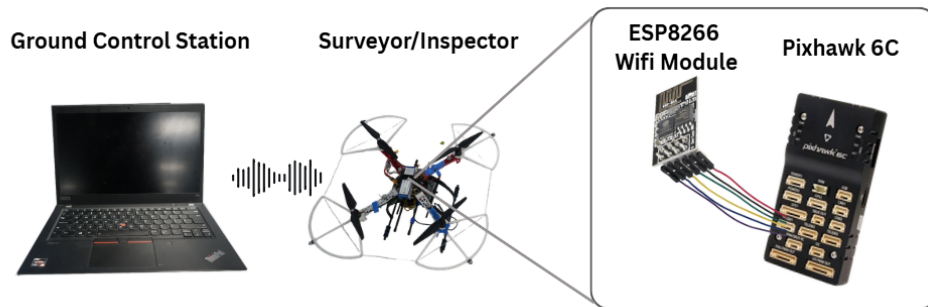


Figure 3: Figure on The Wireless Communication Method between GCS and Drone.

The parsing from raw Vicon data to MAVLink is achieved using a combination of MavProxy (A command line driven Ground Control Station middleware developed by

Ardupilot), the Vicon Software Development Kit (SDK) from Vicon, and “PyVicon”, a Python3 script written by MathGoron. Together, these software components convert the broadcasted Vicon data into Mavlink in the form of GPS coordinates. This setup is mentioned in great detail in section 4. A figure illustrating the software integration flow as described above is shown below in figure 4:

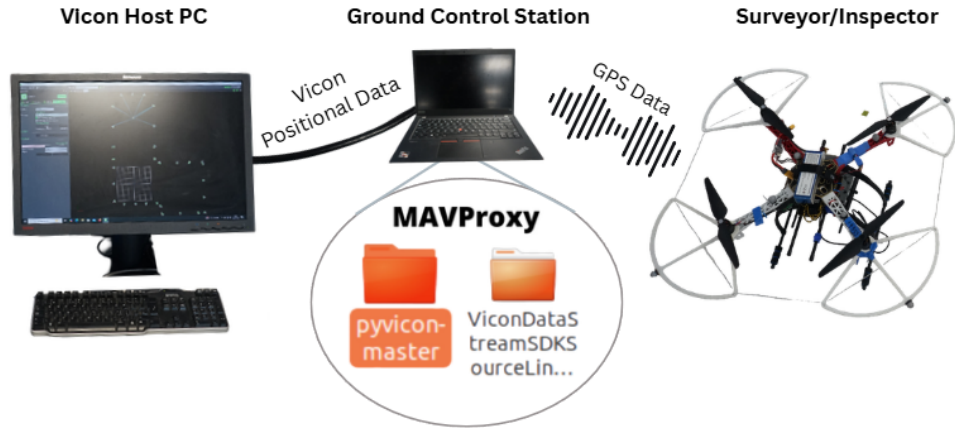


Figure 4: Figure on the Software Integration in GCS.

In the next section in 3, we will begin the guide with a straight forward procedure on operating the Vicon Positioning System inside the Cranfield Flight Arena.

This space intentionally left blank

3 Operating VICON

3.1 The Hardware

From within the Cranfield flight arena, this is the current hardware setup for the VICON positioning system. Image 5 is your reference for the name I uses throughout this guide.

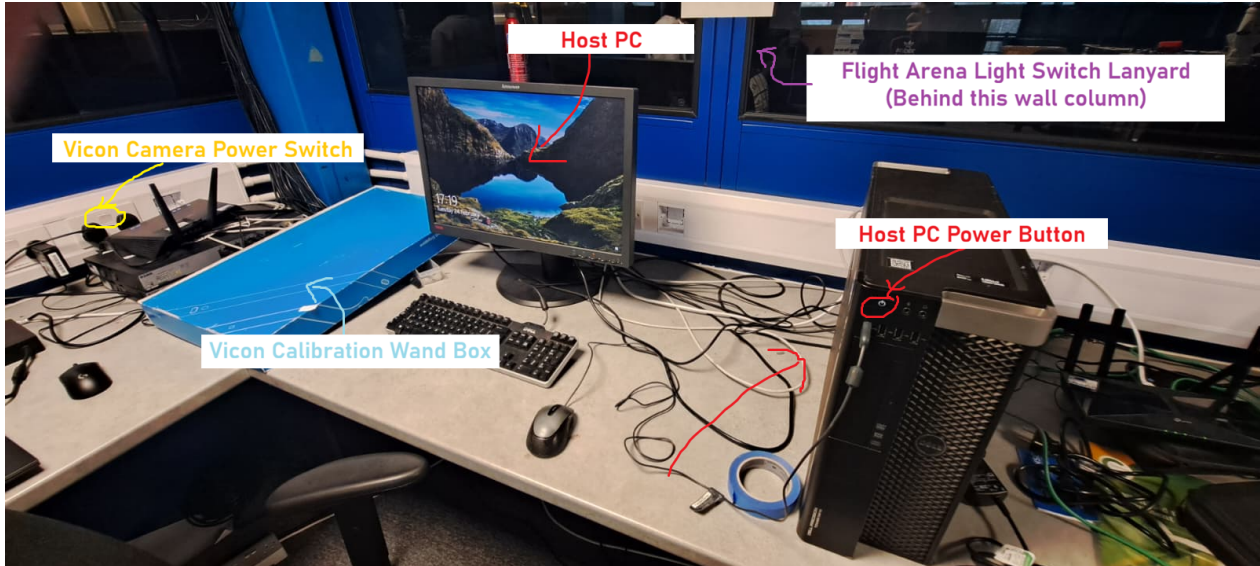


Figure 5: Overall VICON Hardware Architecture

3.2 Turn on Vicon cameras and Host PC

To use the Vicon, firstly, on the corner of the wall, switch on both of the VICON Camera Power Switch. **Wait 20 minutes for the Camera to warmup.** Meanwhile, turn on the VICON Host PC by pressing the Host PC Power Button. Log into the computer, when asked for the password, it is "**Cranfield01**". Once logged in, launch the "Vicon Tracker 3.10.0 x64" app.

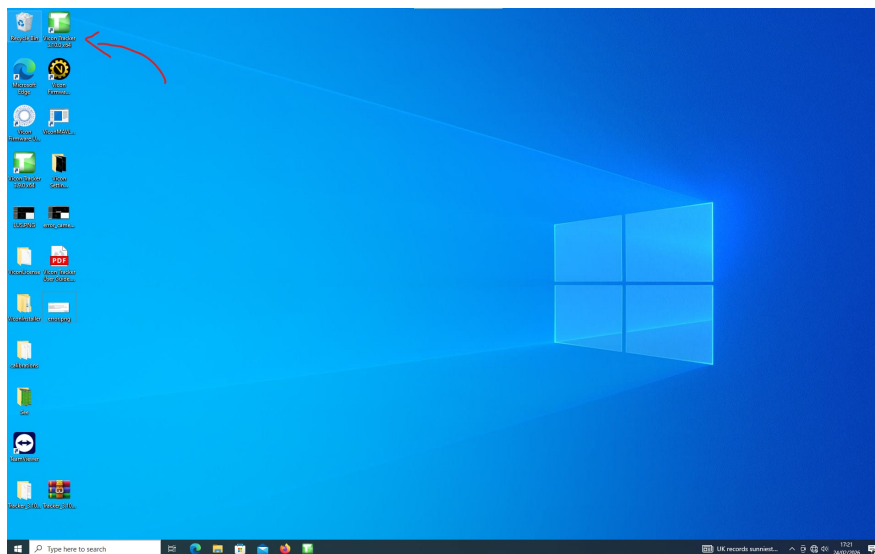


Figure 6: Image of desktop and the app's corresponding location.

App will launch, focus on the left side of the app as seen in figure 7, and navigate to the *SYSTEM* tab. After 20 minutes of warmup, verify that all cameras under *Vicon Camera* are good and without yellow or red crosses.

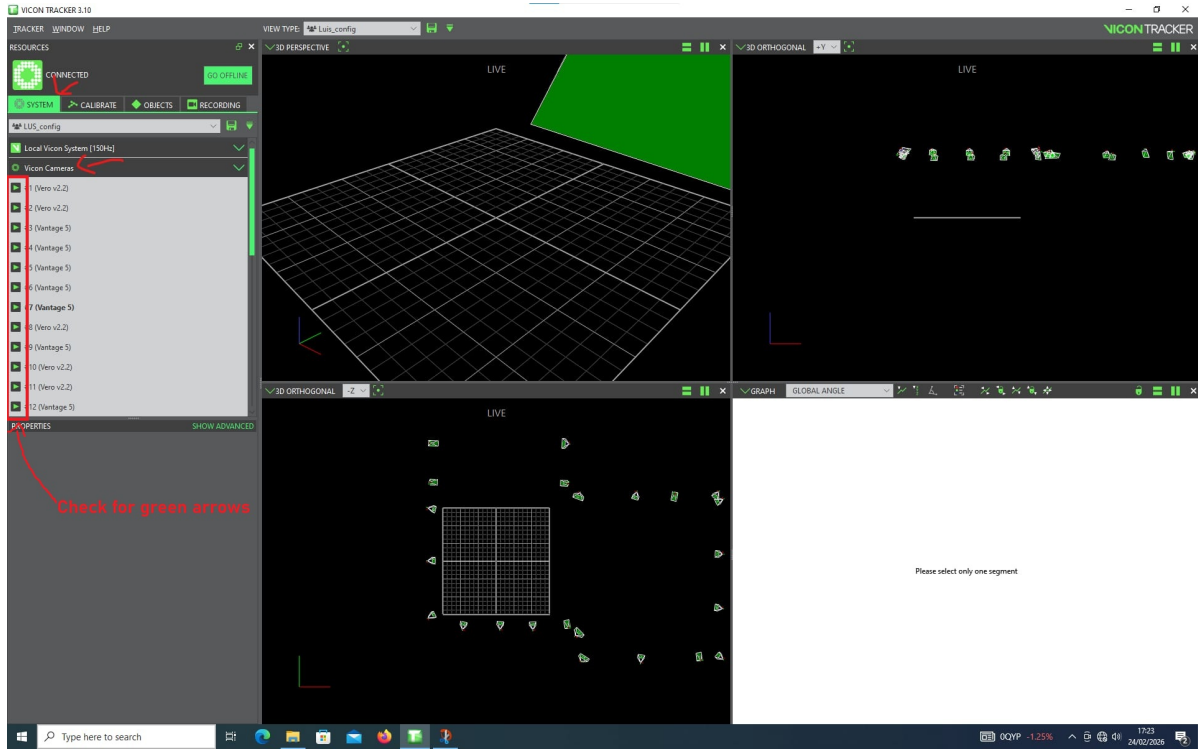


Figure 7: Image of "SYSTEM TAB" and verification of camera after warmup.

If cameras are not ready, click on *Local Vicon System [150Hz]* → *PROPERTIES* → *REBOOT ALL*, as seen in figure 8 below.

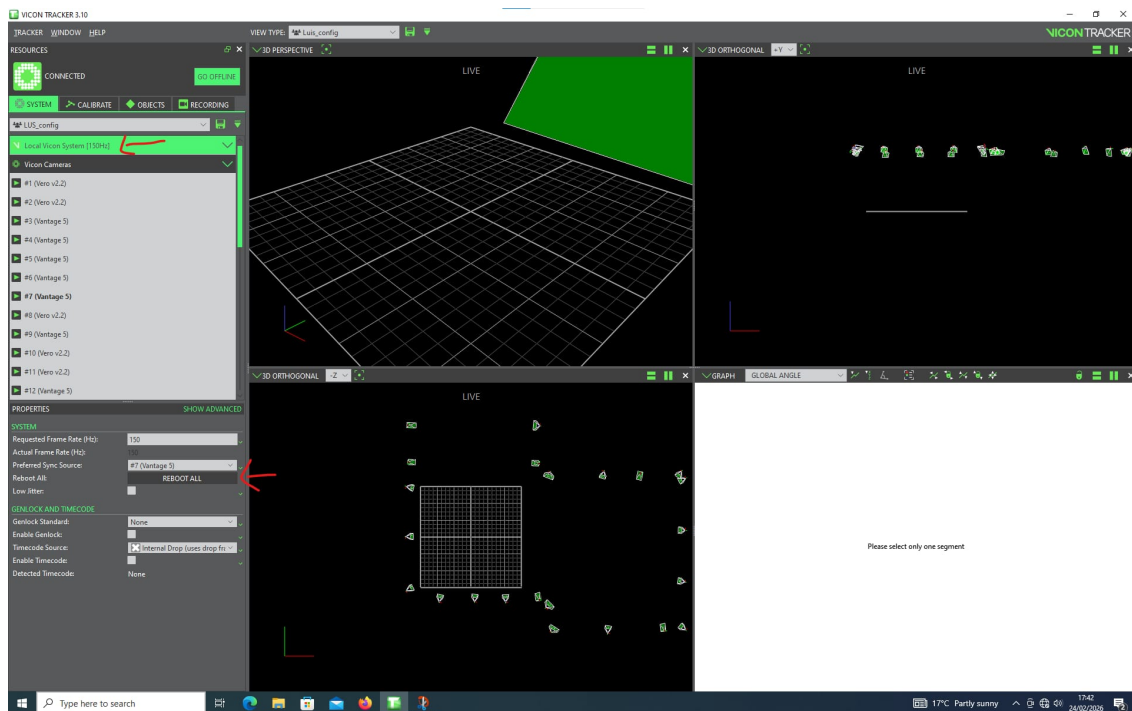


Figure 8: Image of *SYSTEM* tab.

3.3 Create a Camera Mask

Navigate to the *CALIBRATE* tab next to *SYSTEM* tab. Create a camera mask by pressing *START* under *CREATE CAMERA MASK* as seen in figure 9. This masks takes a snapshot of the environment and filters out any IR reflection it sees from the environment.¹

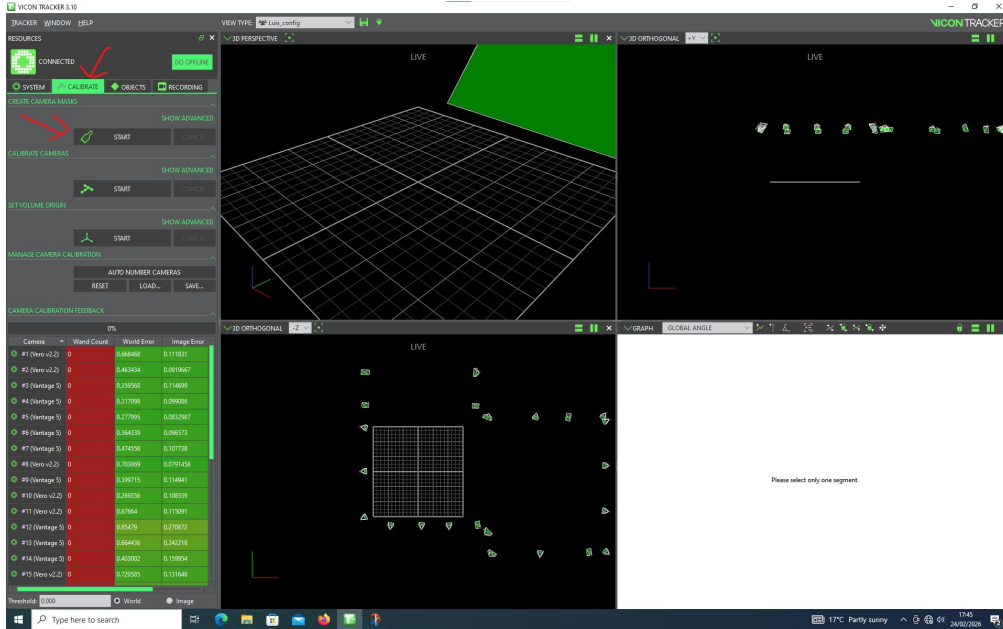


Figure 9: Image of *CALIBRATE* tab and Camera Masks button.

The blue pixels highlighted on each individual camera views are the IR reflection the camera spotted from within the environment. Once the blue dots stops changing much, press *STOP*. (If any cameras have complete blue screens, that means something is wrong and a camera reboot/power flushing the cameras is needed.)

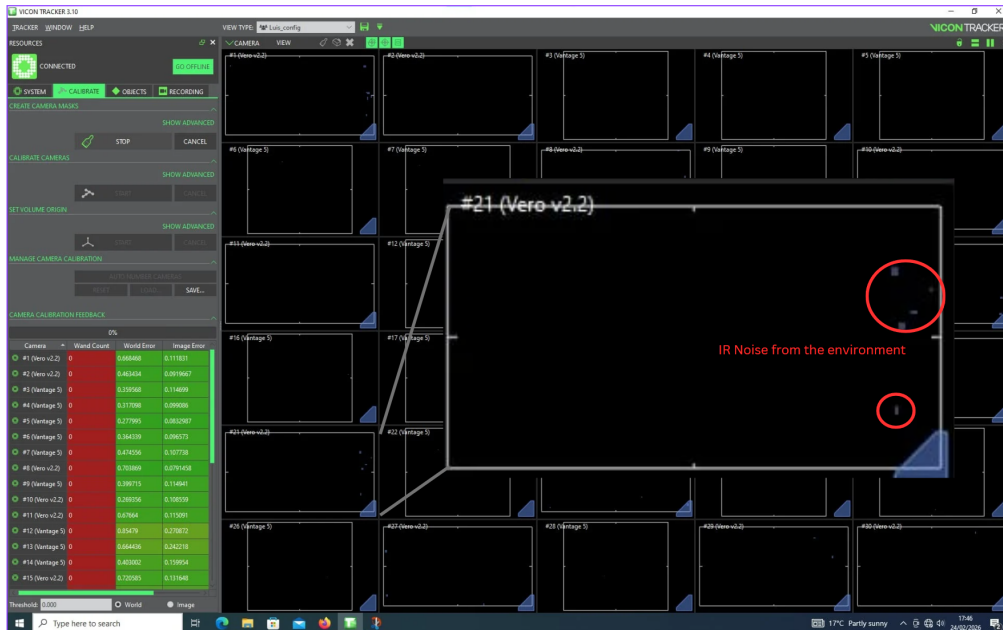
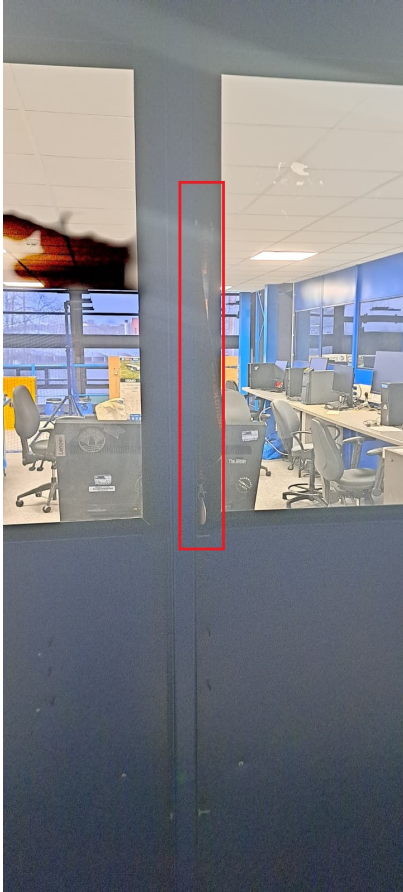


Figure 10: Image of Camera Mask being created and the corresponding *STOP* button.

¹If your testing have any obstacles, put them in physically before creating a camera mask. Also, make sure you turn on the flight arena lights!

3.4 How to turn on the lights of Flight Arena

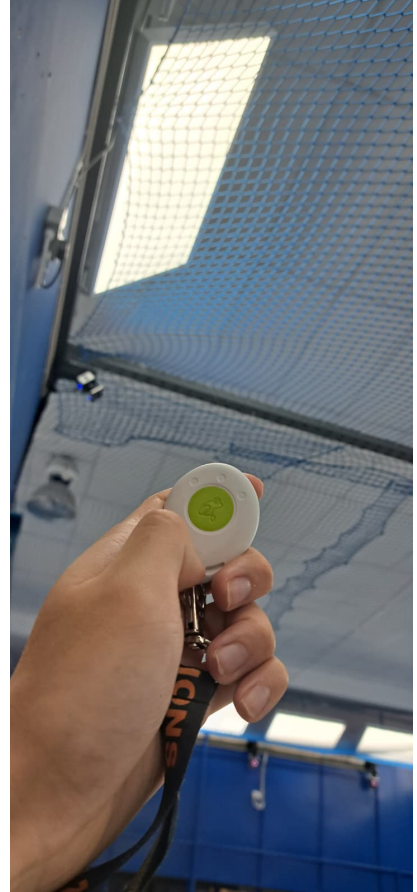
To turn on the lights, physically enter your biological body into the flight arena, walk towards the wall that faces the Host PC. Find the light switch lanyard (Yes a lanyard, not a button), aim it towards the lights on top, click the yellow button.



(a) Location of Flight Arena Light Switch Lanyard.



(b) Aim the button towards the ceiling and depress the button.



(c) Lights turns on! Hooray!

Figure 11: Image of how to turn on the flight arena lights

3.5 Calibrate the Cameras

Next, the cameras need to be calibrated. Under the category *CALIBRATE CAMERAS*, seen in figure 12, click *START*. The computer will verbally respond with "Calibration started".

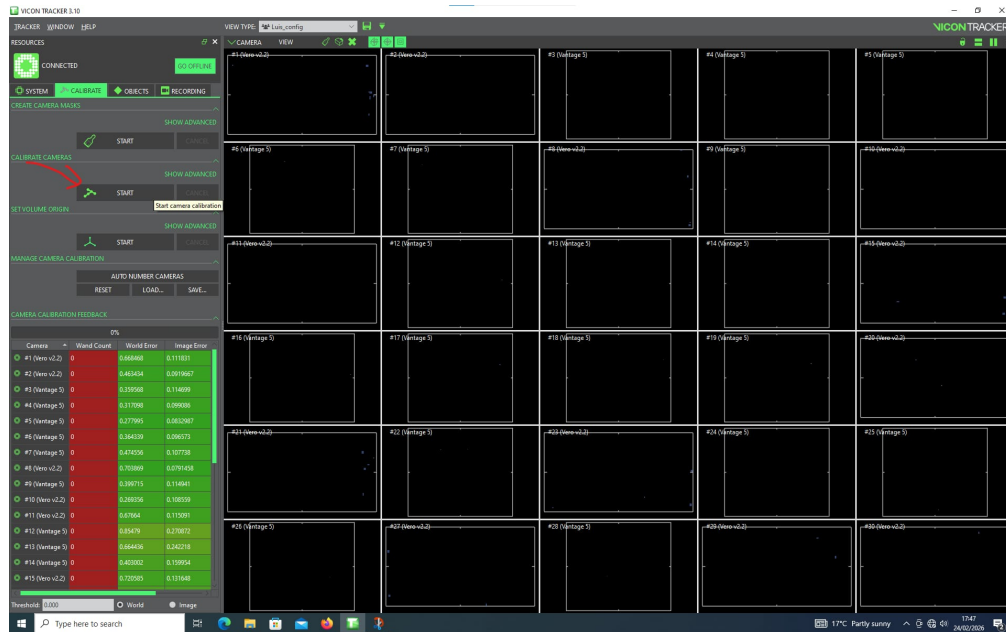


Figure 12: Image of the *CALIBRATION CAMERAS's START* button.

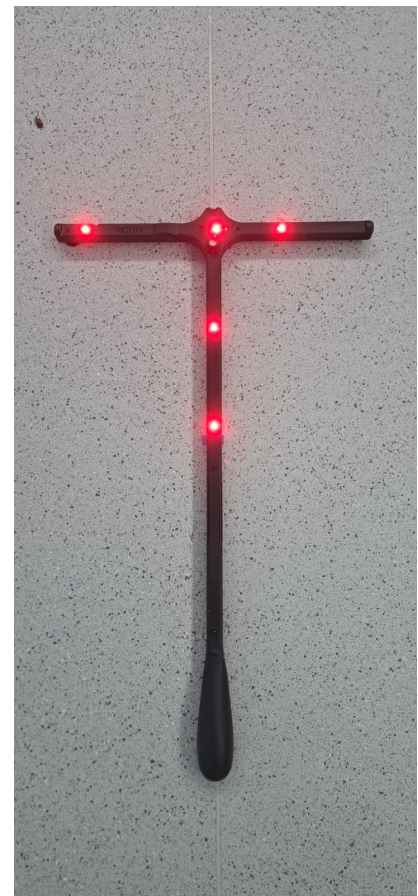
Search for the Vicon Calibration Wand. Find the LED switch as seen in figure 13 and switch it on, red lights should come on.



(a) The VICON Calibration Wand and its spawn point.



(b) The Wand and the LED light switch.



(c) Lights turns on! Hooray!

Figure 13: Image of how to turn on the VICON Calibration Wand

Biomechanically, leave the Host PC and walk into the flight arena and sweep your arm in a cyclical pattern towards the cameras, an example of such interpretive dance can be seen below in figure 8.



Figure 14: Images of dance moves used calibrate the cameras.

Repeat this sweeping motion in front of all the cameras within the flight arena. To determine if a specific camera has been calibrated (or when you can stop sweeping in front of it), you can visually see the LED changes from red/pink to green for the small cameras, and red to green for the big cameras. (For big cameras, you can see a screen on the camera that shows the progress bar of the calibration, see figure 15). Repeat your interpretive dance until all cameras are calibrated. Visually, all camera would turn from static green to static red if every camera is calibrated.

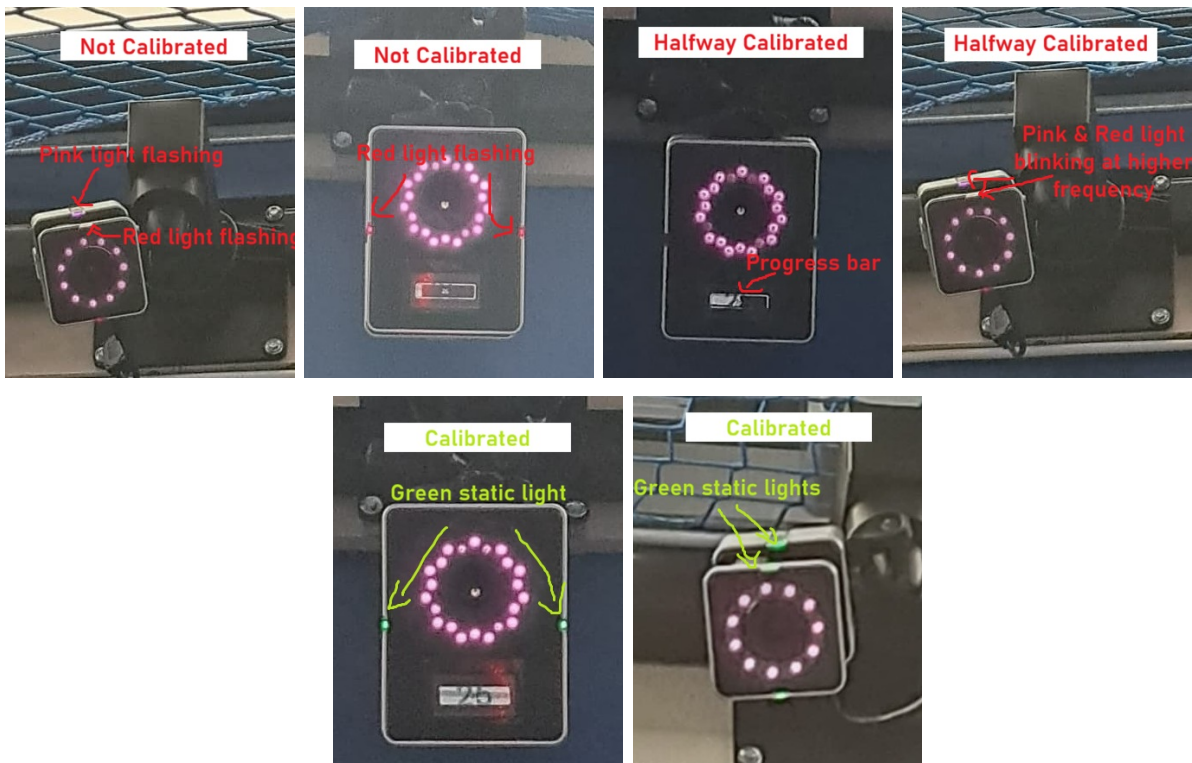


Figure 15: Images of camera stage's of calibration and it's visual indicators.

Return to the PC, focus on the bottom left of your screen on *CAMERA CALIBRATION FEEDBACK*, a progress bar will show the calculation of the poses from the Tracker, shown in figure 16 below.

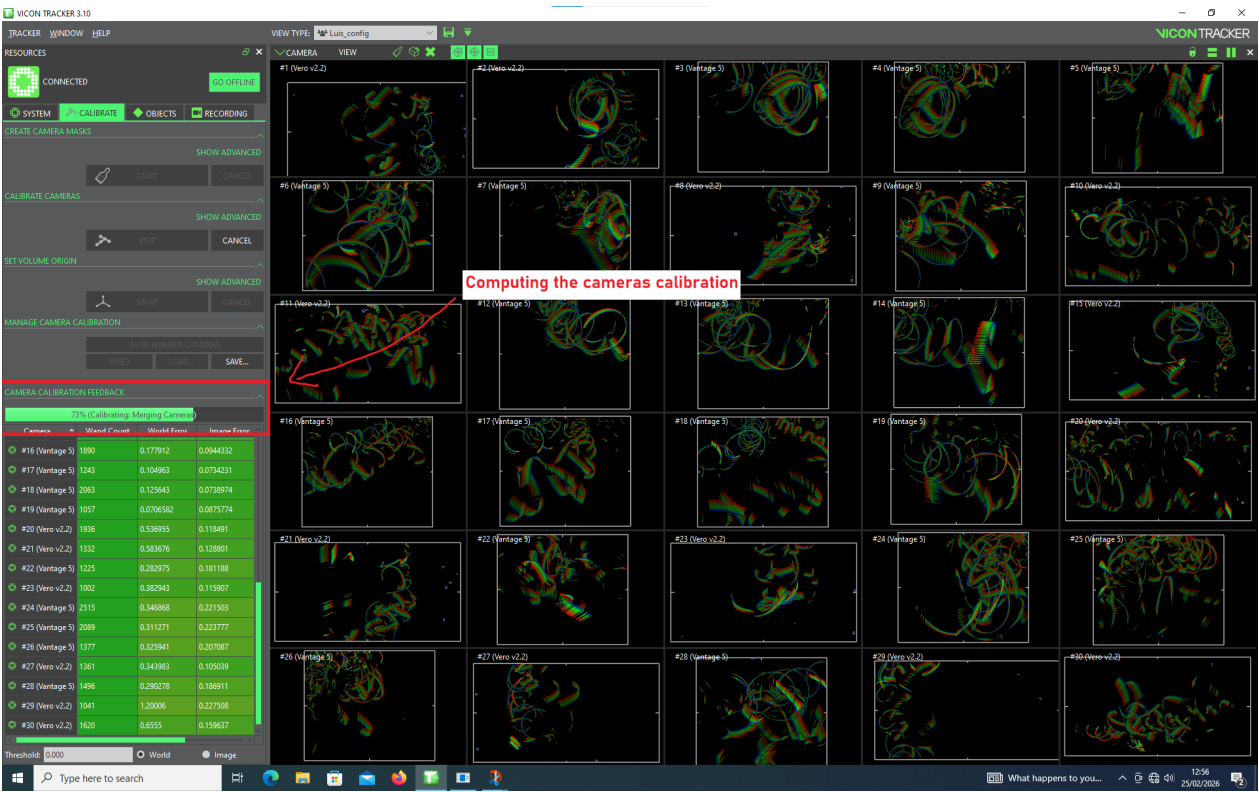


Figure 16: Image of feedback computation after physical camera calibration

Once the calculation progress bar is filled and therefore completed, check the *World Error* and *Image Error* from all the cameras, If you see any orange or red tiles from the table as shown in figure 17, you will need to repeat your calibration step.

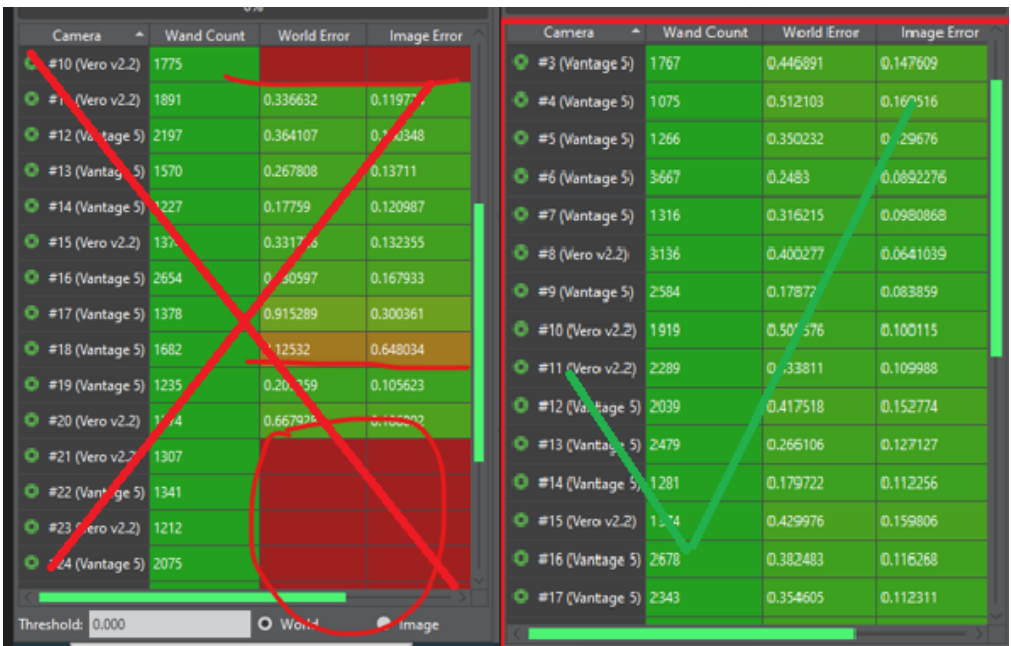


Figure 17: Comparison figure of bad vs good camera calibration

3.6 Set Origin

Next, you will need to set the origin of your coordinate system. This essentially defines your $x y z$ axis for your inertial positioning system. Without pressing anything on the PC, physically go back into the flight arena with the Vicon Calibration Wand. Place your wand flat on the ground on where you want to defined your origins are.

For default, I suggest you place them on the blue tapes in the intersectional middle of the flight arena, as seen in figure 18



(a) Default location for origin, located in the middle intersectional area.

(b) Closed up location, identifiable with 2 blue tapes and pencil markings.

(c) Overall wand placement location for origin. Located in the corner of flight arena.

Figure 18: Conventional origin location placement for Vicon Calibration Wand.

Noting that the direction convention for the Vicon Calibration Wand is that bottom of handle faces West, left handle faces North (Refer to figure 19). With the best of your ability, align it to the orthogonal landscape of the flight arena (The silicone sealant on the floor is a good orthogonal reference, align the wand with it). This is imperative as your Vicon Calibration Wand's West is what the Ardupilot/flight controller determines as it's environment West. If your West is badly oriented, you will just have a non-orthogonal axis as your inertial coordinate frame, for which, your path-planning person will sure thank you a lot for :).

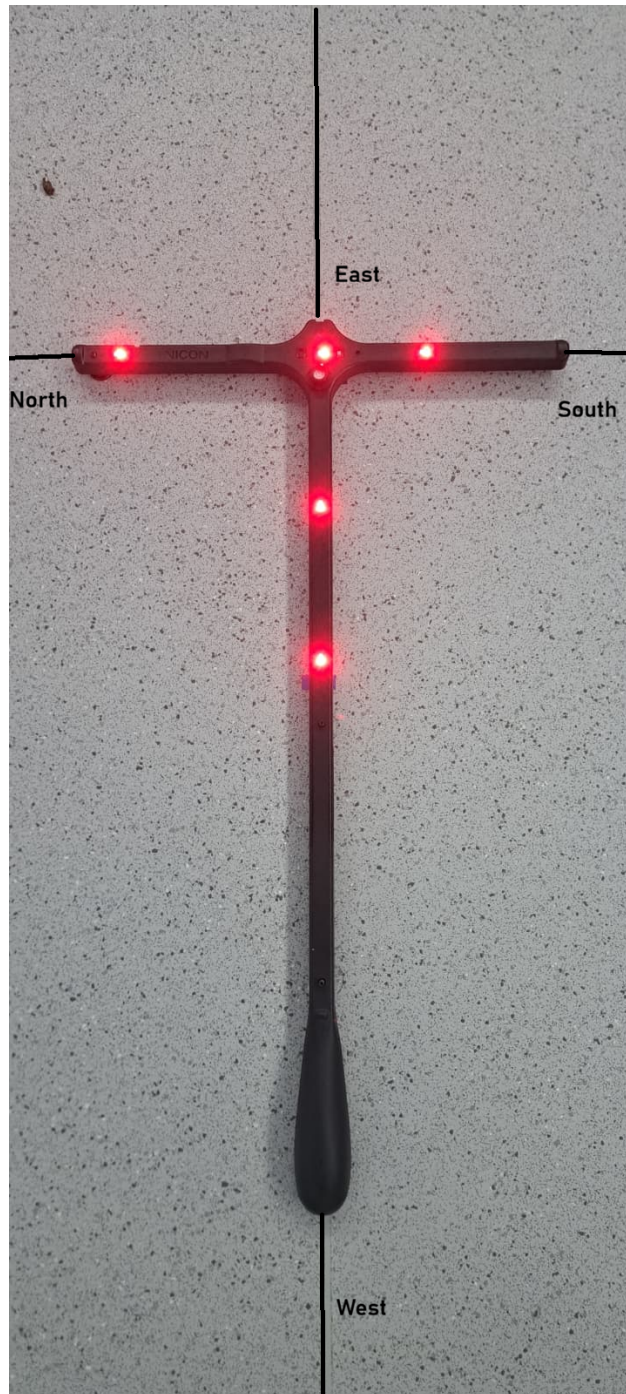


Figure 19: Direction conventions for the Vicon Calibration Wand.

Leave the wand on the ground, return to the PC and navigation to the *SET VOLUME ORIGIN*, click *START* (figure 20), the computer will respond verbally with "Origin set".

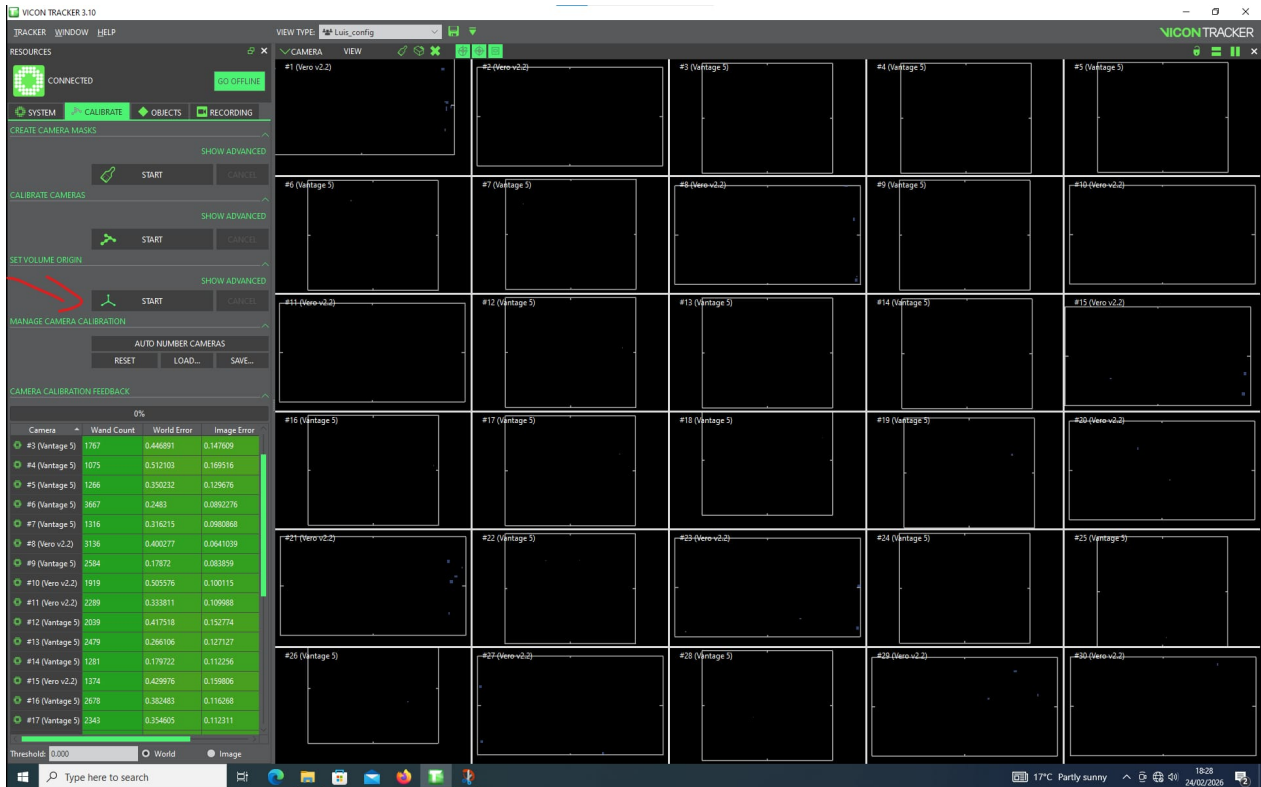


Figure 20: Image of Set Volume Origin button's location.

3.7 Defining New Objects

The way objects are recognized in the VICON algorithm is that it remembers the assortment of reflective markers placed on your drone. This works by defining your markers as *Object*.²

To make a new Object, you first need to glue the IR markers on your drone (4+ markers as a minimum empirical requirement). An example of which is seen below in figure 21.



Figure 21: Example Image of IR Markers Placement

²For my lovely GDP group, our drone is already defined as an object by yours truly and is called INSP!

Once this is done, physically put your drone into the flight arena. Align the front (North) of your drone to the origin's North. Pick up your wand and return to the host PC.

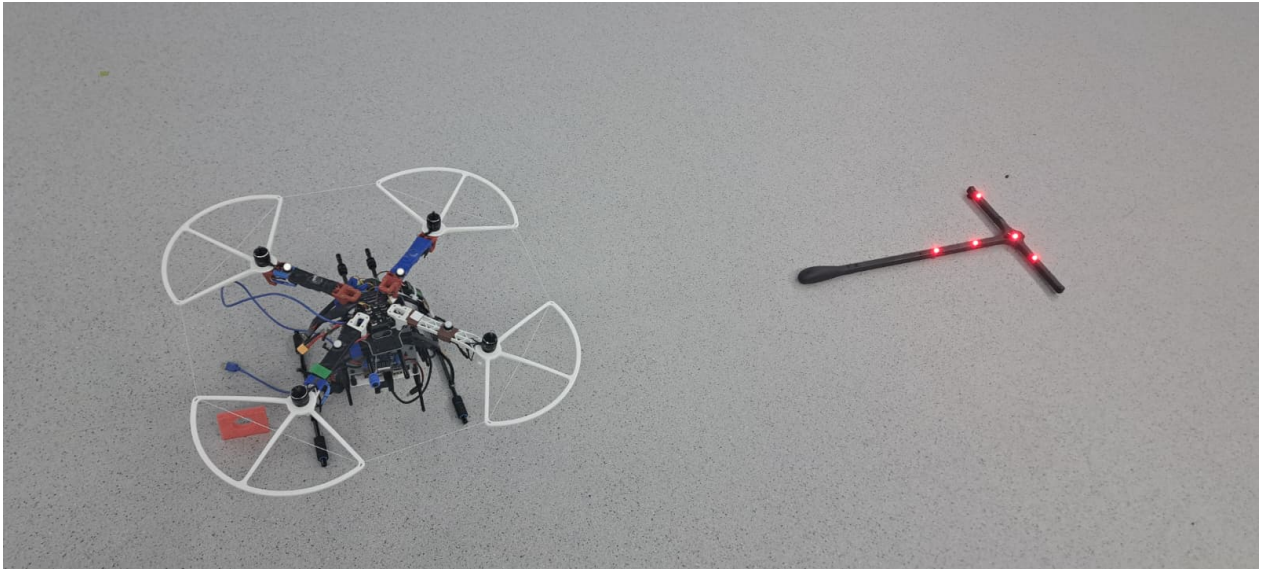


Figure 22: Image of drone with the Vicon Calibration Wand in it's origin setting position.

On the PC's viewport, click the *OBJECT* tab. (If you are in my GDP group, select INSP as the the drone.

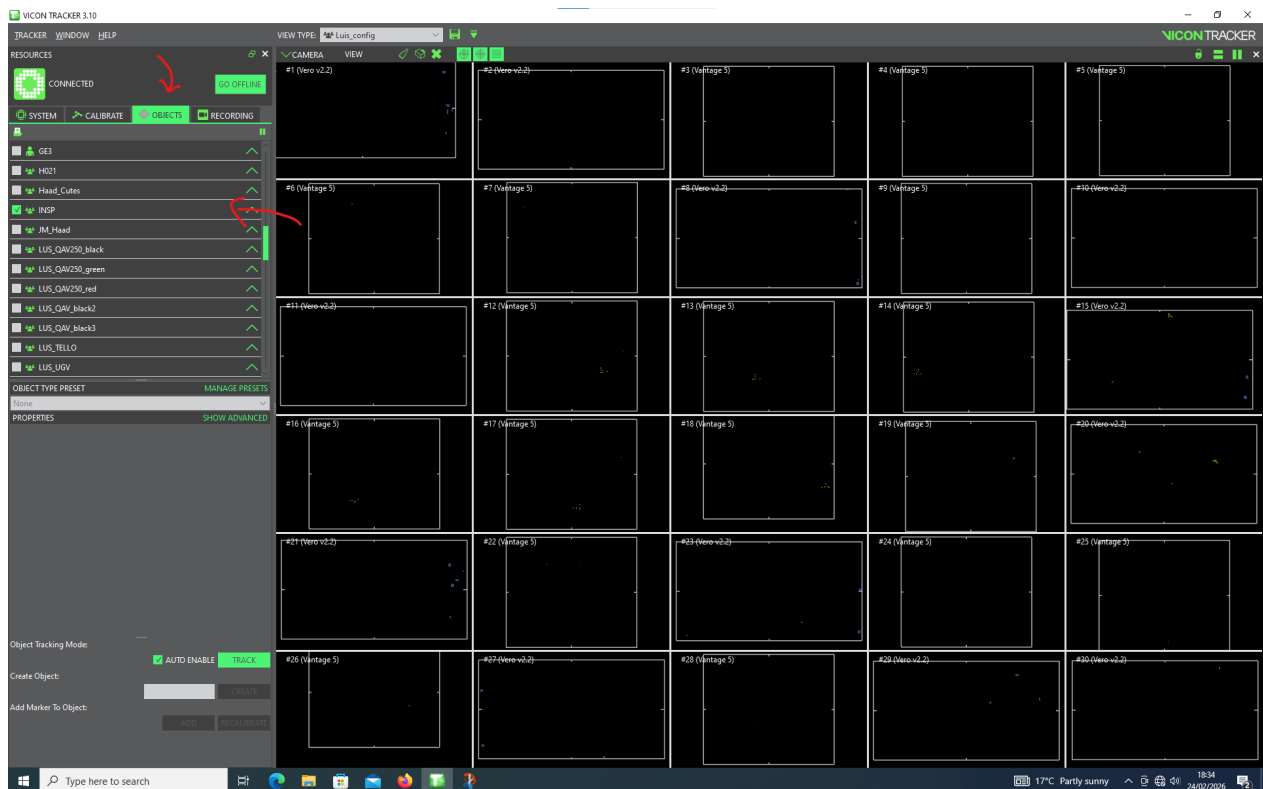


Figure 23: Enter Caption

You need to select the markers in the view port on the right side of the screen. (If you can't see the markers, navigation top left of the view port, and choose *3D ORTHOGONAL* to *-Z*, as seen in figure 24)

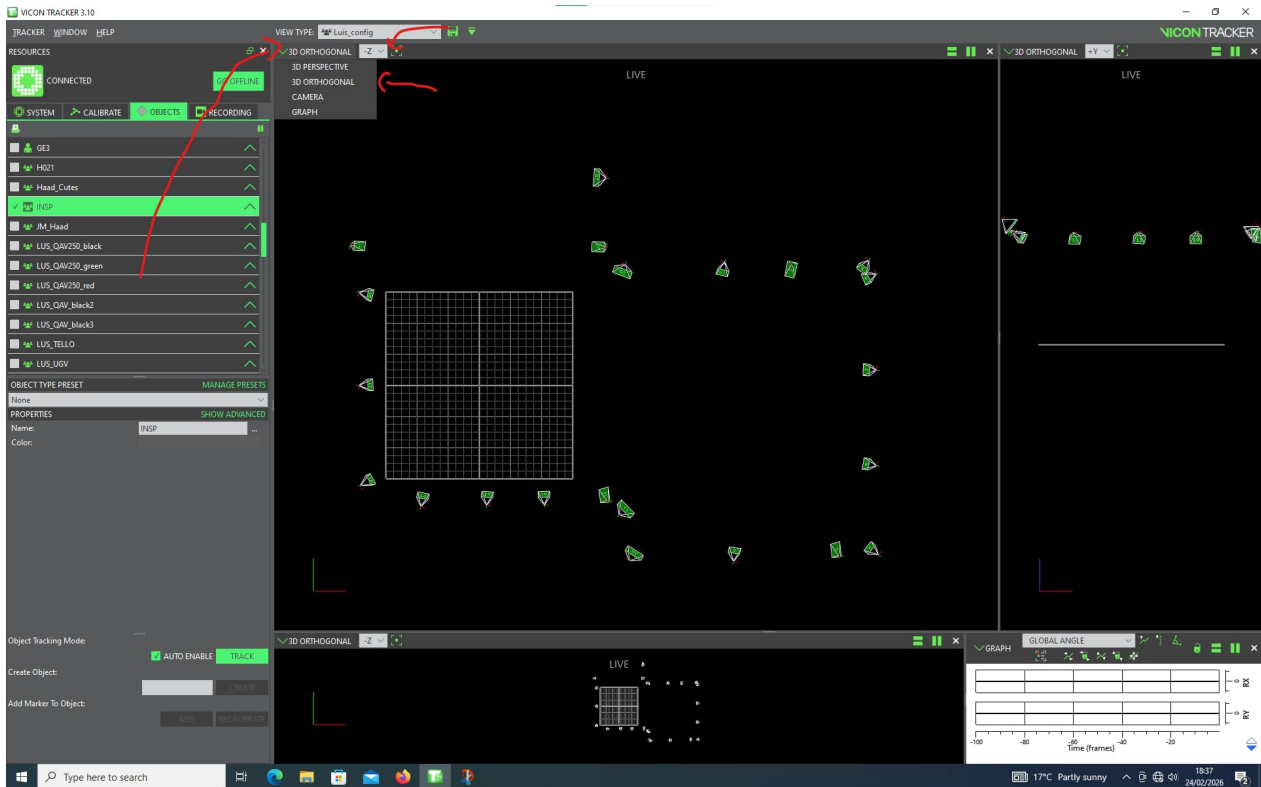


Figure 24: Image of viewport settings

To navigate the viewport, right click to zoom in, middle click to pan, alt-click to select across the points you see within the viewport. You should see your assortments of IR markers in the viewport in the expected location, similar to figure 25.

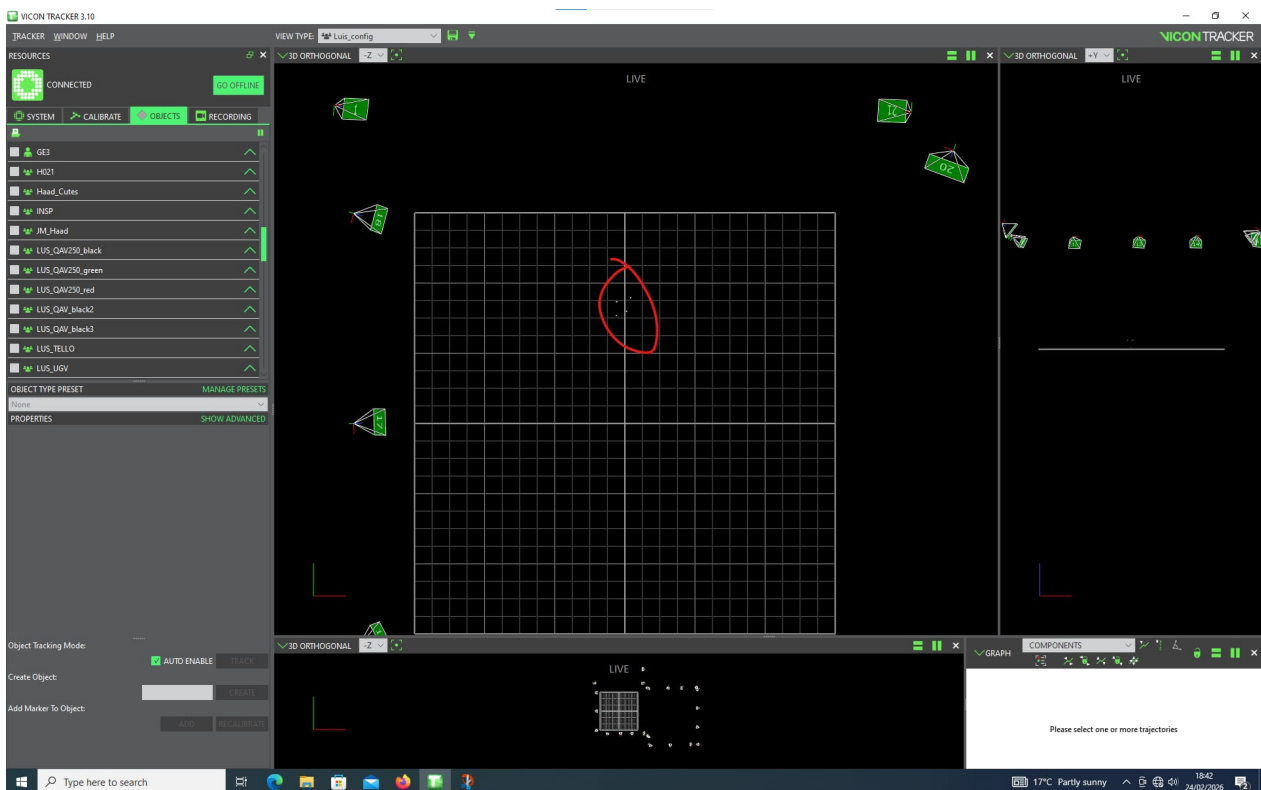


Figure 25: Image of IR markers spotted in viewport.

Select the points by alt-clicking across them. It should show what cameras is currently tracking them, like in figure 26.

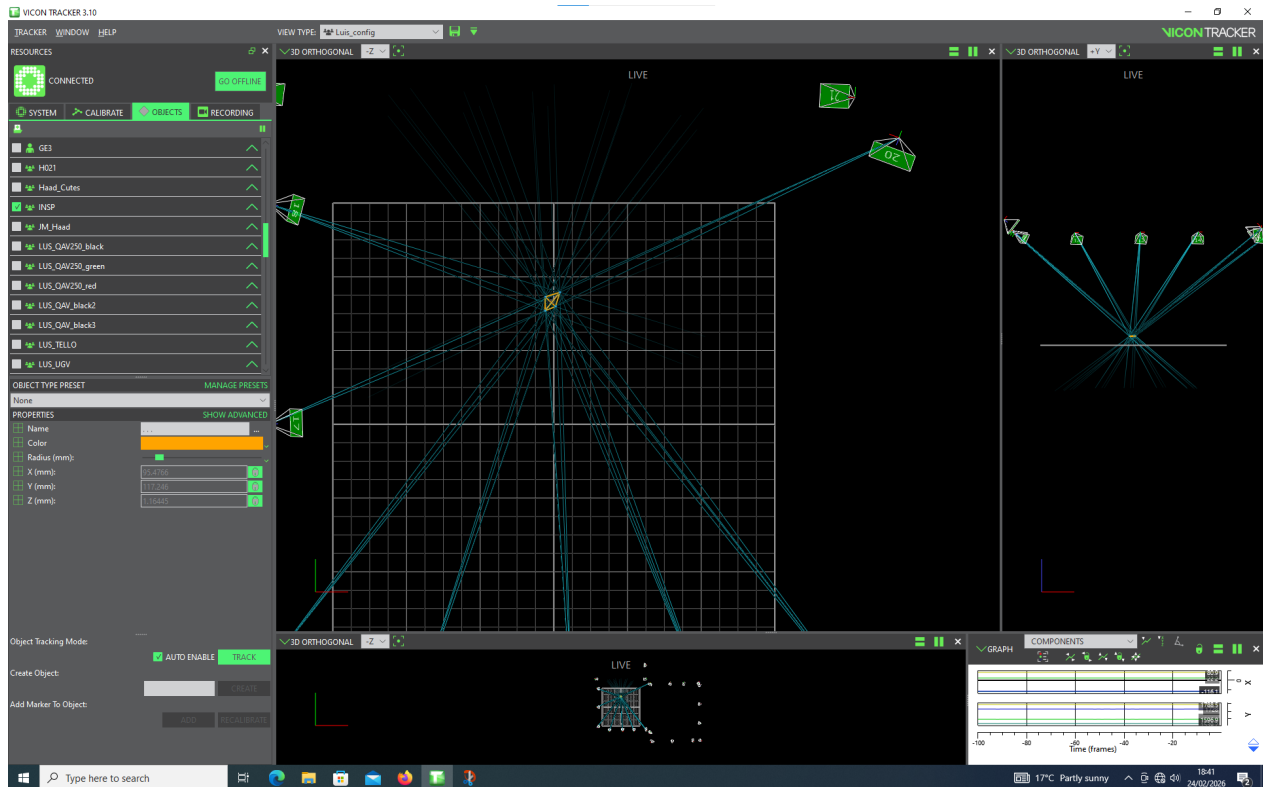


Figure 26: Image of IR Markers being selected.

Align your focus to the bottom left of the screen, seen in figure 27. Type down the name of your object, and the *CREATE* button should be enabled like in figure 28. Press *CREATE*, click *SHARED*.

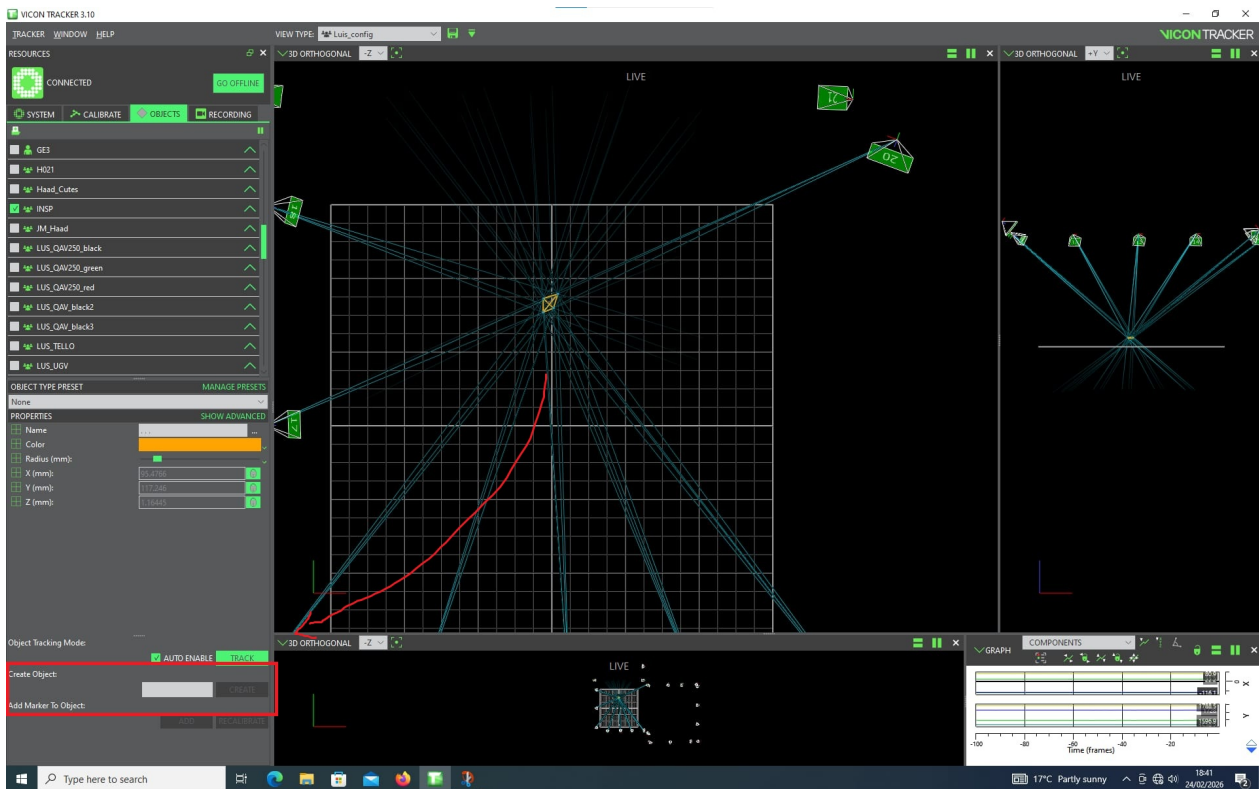


Figure 27: Image of the TRACK button

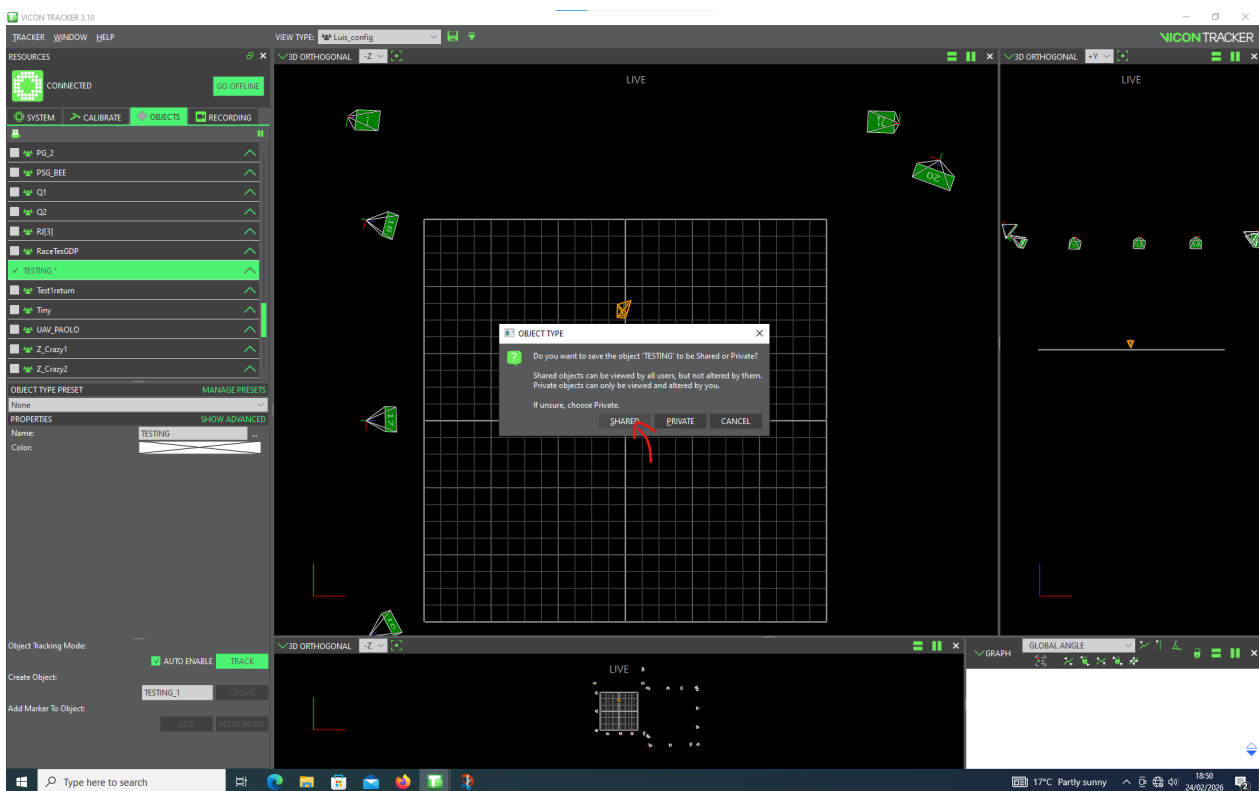


Figure 28: Image of creating object

From the list of objects, search for your newly defined object, and save to the list as shown in figure 29.

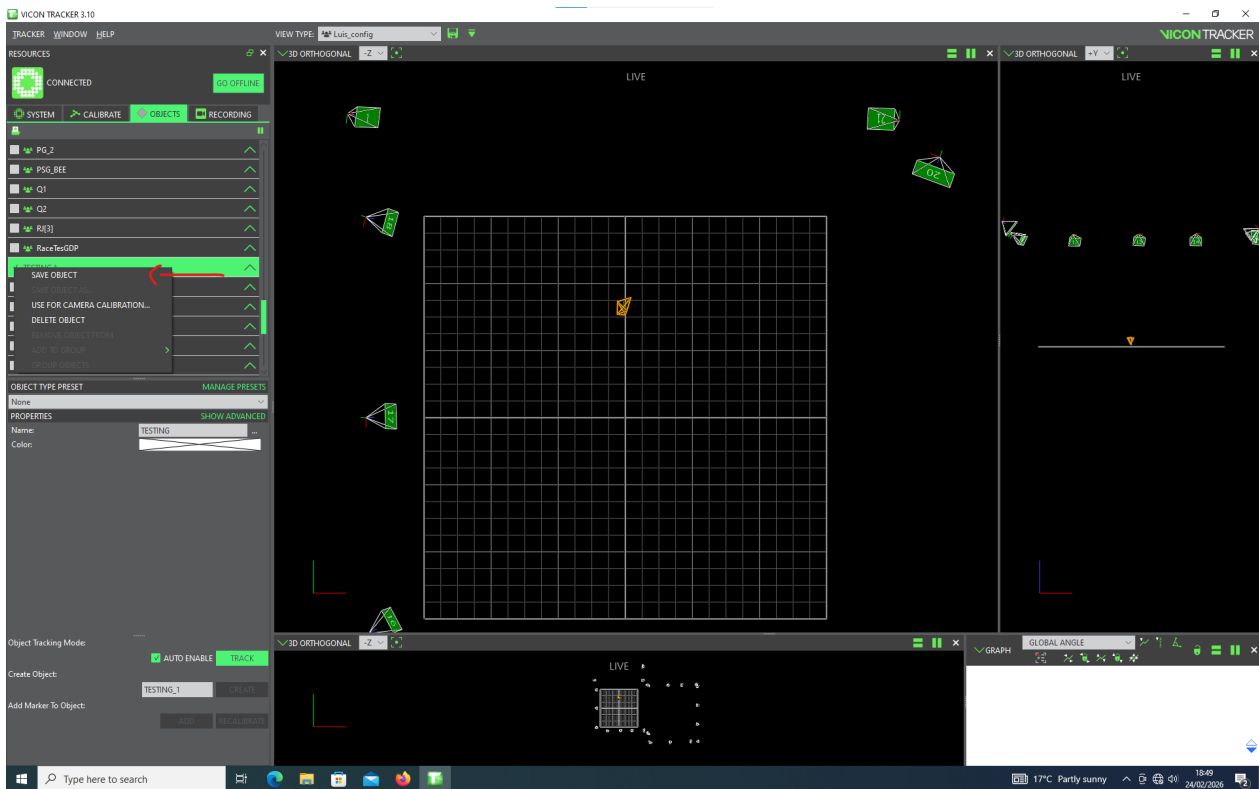


Figure 29: Image of saving from object list.

From this, your object is now integrated within the list of objects created by everyone else in the lab! I have been advised that it is good practice to have your object name to be around 4 letters, although I have seen names longer than 4. Luis will also thank you if you name your drones beginning with Z, so that it does not bring the ordering of the drones into a chaotic list.

3.8 Finishing

Ensure you pack your Wand back into the box and return where you found it, sign out of the computer, remove all the obstacle and drones placed inside the flight arena, switch off the flight arena lights, turn off the Vicon Camera Power Switch.

From this point, the Vicon Server should be tracking your now defined object! This information can be broadcasted through Ethernet cables and to your Ground Control Station via *MAVPROXY* & *MAVLINK*, this is most likely your next step. This will be reflected in next section 4.

4 Ground Control Station Setup

Now, in this section we first set up the Ground Control Station (GCS) as seen above in figure 4. This section goes over the installation of VICON SDK, Pyvicon, MavProxy/Mavlink, and some corresponding package dependencies. This section is prone to some debugging against my best wishes, and everything is not really Plug N Play - Some resiliences is required!

4.1 Hardware Required

Please liaise with your corresponding supervisor/client and prepare a Linux running laptop with both Wifi and Ethernet Capability ^{3 4 5}, a router (in the flight arena), and an ethernet cable (in the flight arena), as seen in figure 30.



Figure 30: Hardware Requirement

4.2 Setting Up Conda Virtual Environment

Before we start setting up the softwares inside the virtual environment, let's set up a Virtual Environment (Venv). This step is to segregate your files you are going to install here away from the rest of the computer. This ensures that the package we download does not influence any other projects you have going on in your computer. (For example, you have 2 projects which requires 2 different version of a package, leading to a conflict). The use of venv is an industry practice, so if you are starting out (Like me), make sure to use venv! To install the virtual environment, I use miniconda, a guide on installing can be found at <https://www.anaconda.com/docs/getting-started/miniconda/install/linux-install> Once you have installed miniconda, Open a new virtual environment in python 3.10 named *VICON* by typing:

```
conda create -n VICON python=3.10
```

³Note that when running Linux, some laptop's wifi card does not support Ubuntu 18.04, which you will need to rebuild the laptop to 20.04.

⁴Don't use Window Subsystem for Linux (WSL), it is a big hassle to figure out port communication permissions.

⁵Dual-booting your laptop into Linux/Window configuration is best if you only have a Window. **Warnings however, dual-booting is intrinsically an "easy to mess up" process with catastrophic consequences. Some statistics for you, 2/12 students in my MSc cohort wiped out all their personal data while dual-booting their personal computer, so please backup all your data before dual-booting and make sure you know what you're doing!!!!**

```
(base) inspection@inspection-ThinkStation-P320:~$ conda create -n VICON python=3.10
```

Figure 31: Setting up the virtual environment for Vicon.

When prompted, type *y* to confirm the package installed. Once installed, activate the environment by pressing:

```
conda activate VICON
```

It should look something like figure 32 below:

```
Proceed ([y]/n)? y

Preparing transaction: done
Verifying transaction: done
Executing transaction: done
#
# To activate this environment, use
#
#     $ conda activate VICON
#
# To deactivate an active environment, use
#
#     $ conda deactivate

(base) inspection@inspection-ThinkStation-P320:~$ conda activate VICON
(VICON) inspection@inspection-ThinkStation-P320:~$
```

Figure 32: Activating Virtual Environment

4.3 Downloading VICON SDK

Once we have set up the Conda environment, let's move on to downloading the VICON SDK now as aforementioned in section 2. Visit

<https://www.vicon.com/software/datastream-sdk/>, the website looks like figure 33:

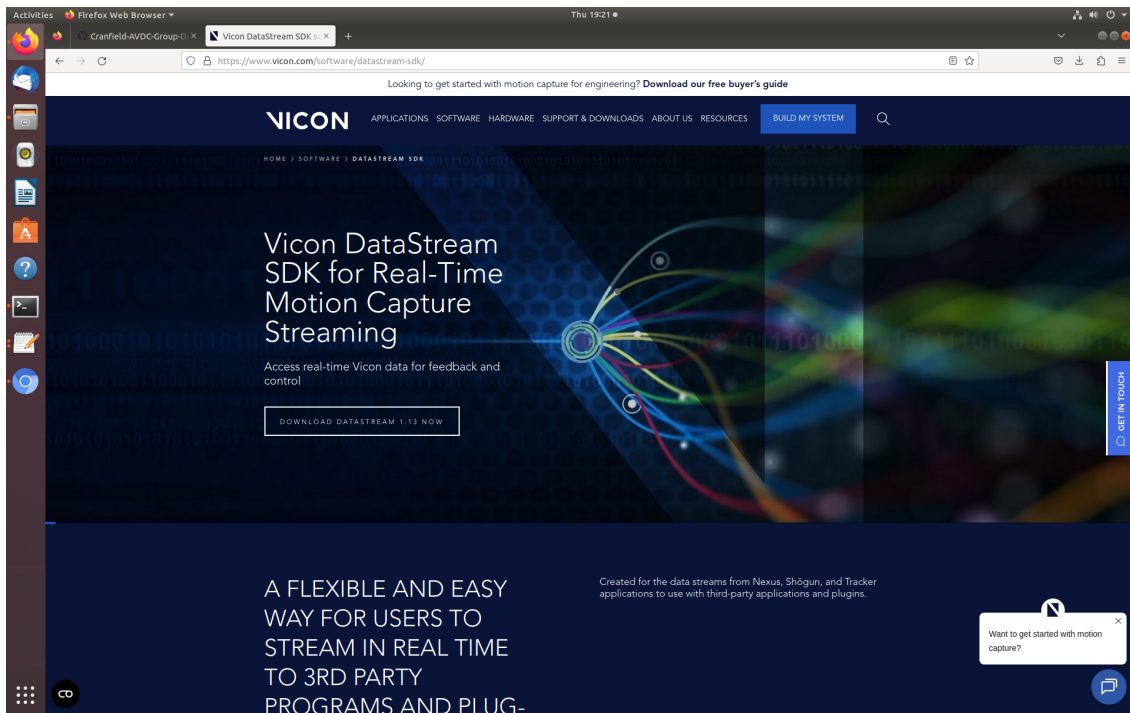


Figure 33: VICON SDK Download Website

Scroll down from the page to what is shown in figure 34:

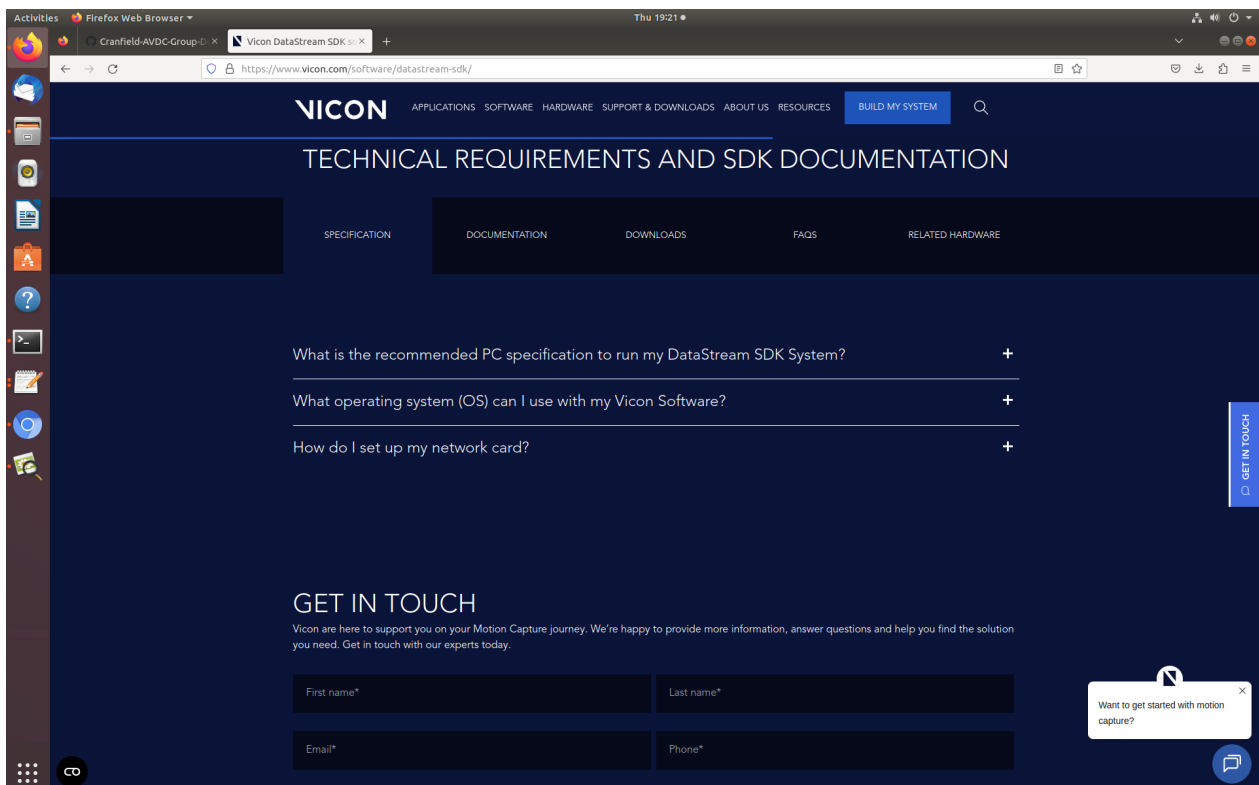


Figure 34: VICON SDK Download Page

Click the downloads tab, install the *DataStream SDK 1.10.0*, write down *an email*, *No* for existing customer, *Tracker* for the Software to Download, click *DOWNLOAD*, as seen in figure 35:

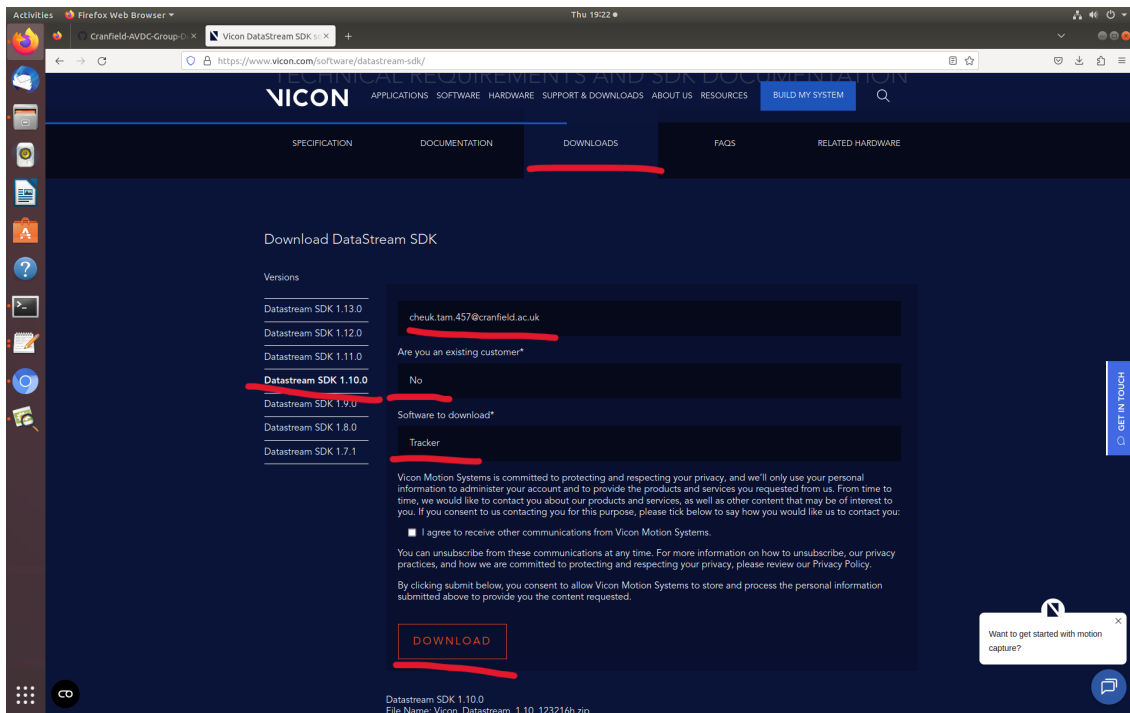


Figure 35: VICON SDK Download Information

Click *Download* as seen in figure 36

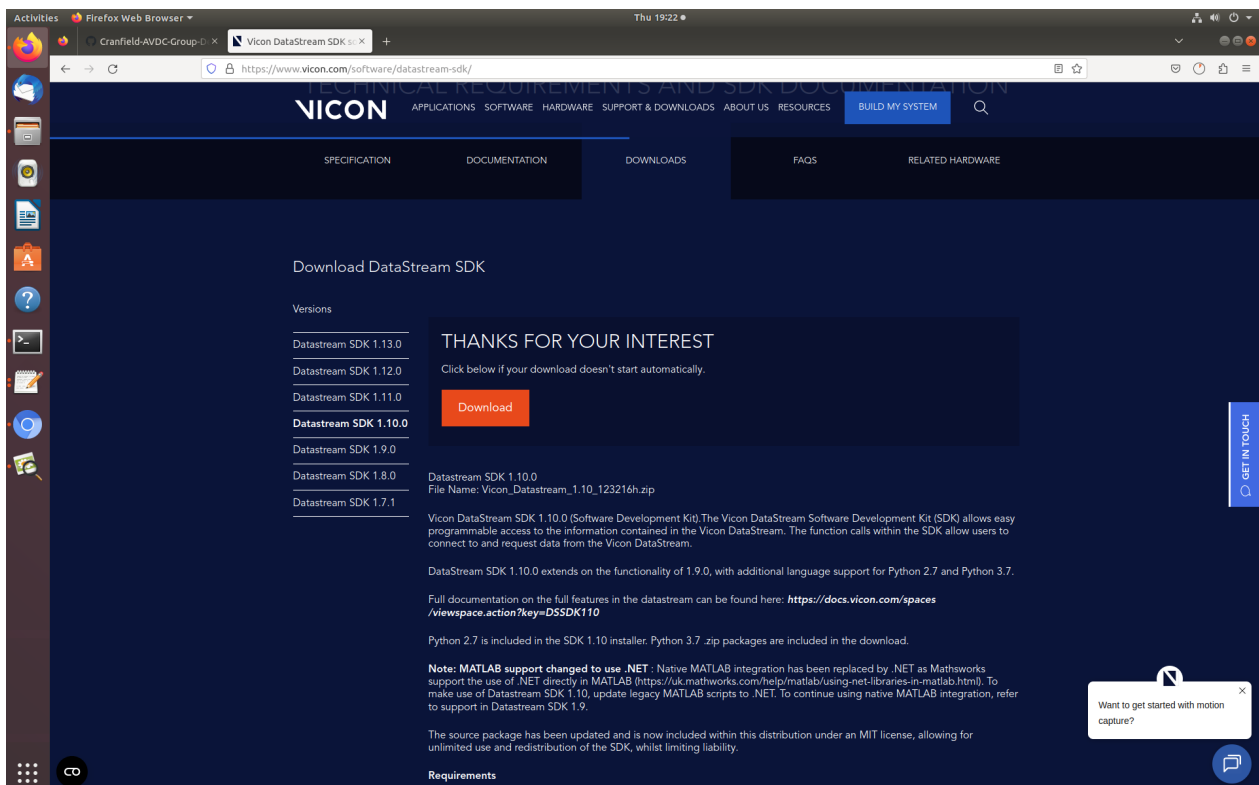


Figure 36: VICON SDK Download

Navigate to the your *Downloads* Folder, click into *Vicon_Datastream_1.10_123216h.zip*, click into *20200407_123216h*, into *Linux64*, into *Release*, as seen in figure 37

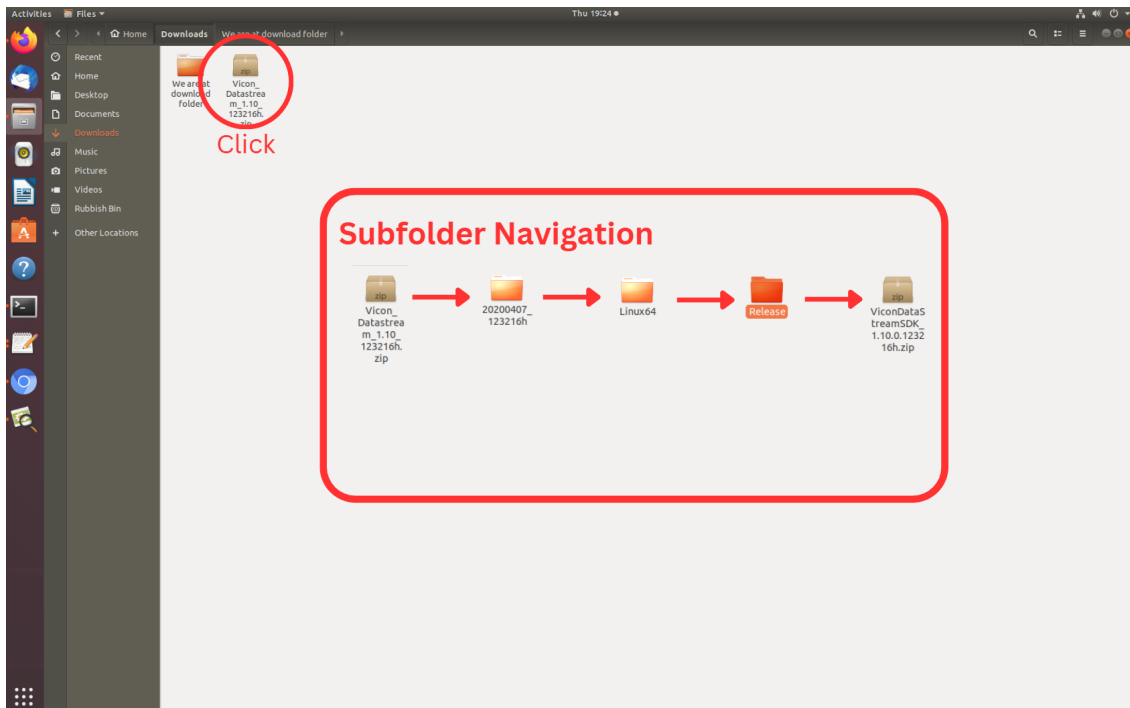


Figure 37: Navigate to Linux64/Release inside Vicon SDK

Right click on *ViconDataStreamSDK_1.10.0.123216h.zip*, extract the file, you should have a file like figure 38

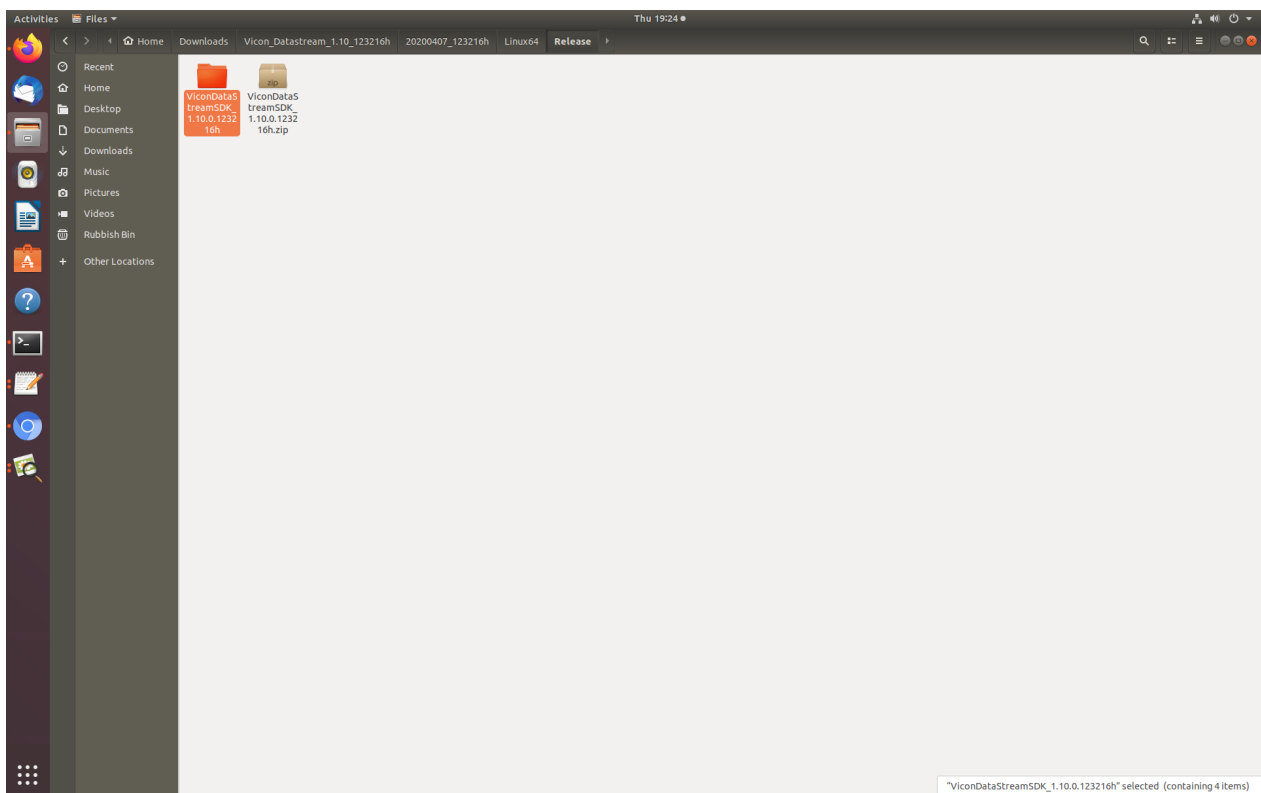


Figure 38: Extracted file of Linux64 Vicon SDK

Drag the extracted file of the Vicon SDK back to the download dictionary as shown in figure 39

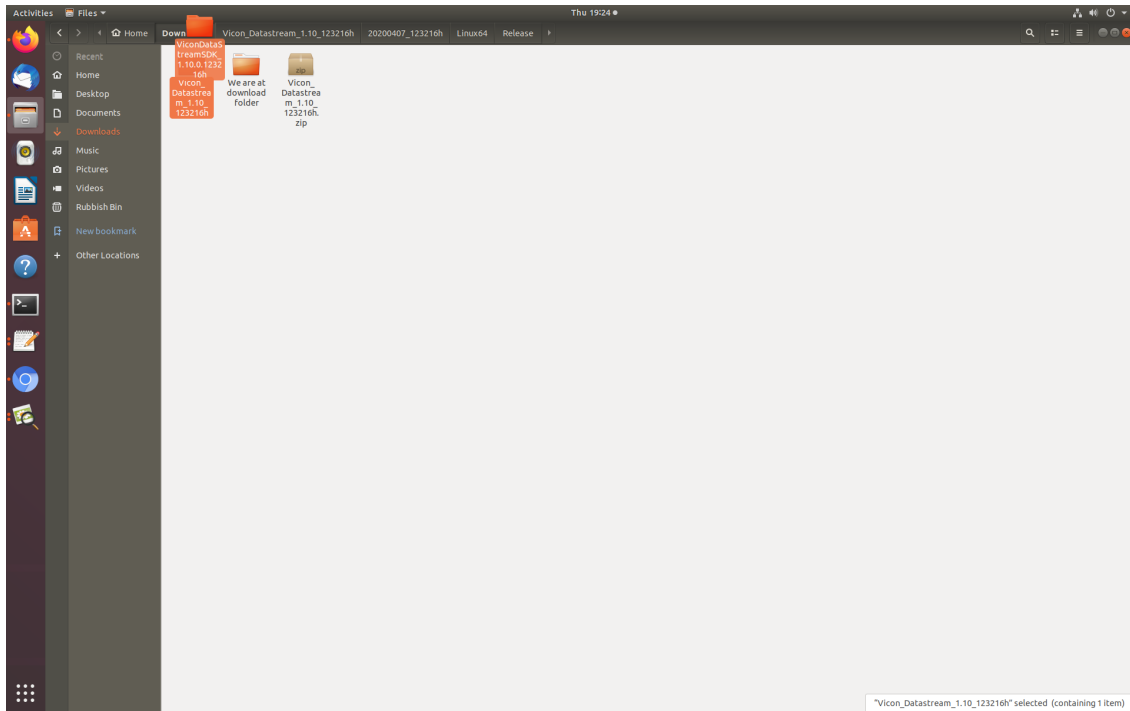


Figure 39: Moving Vicon SDK into download folders

Remove the original zip file, they are no longer needed, you should have a singular file as shown in figure 40.

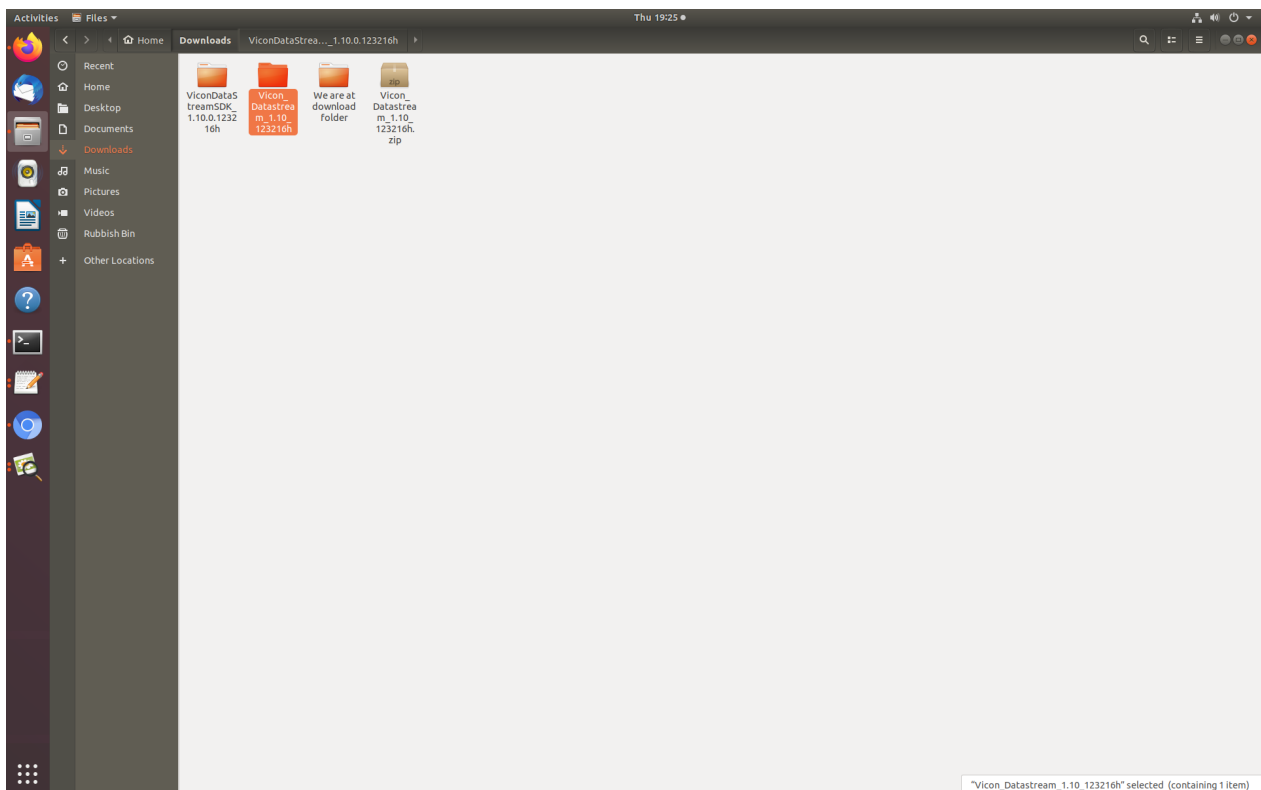


Figure 40: Linux64 Vicon SDK extracted in Downloads Folder

Click into the SDK folder (*ViconDataStreamSDK_1.10.0.123216h*), you should see 2 zipped files inside it like figure 41.

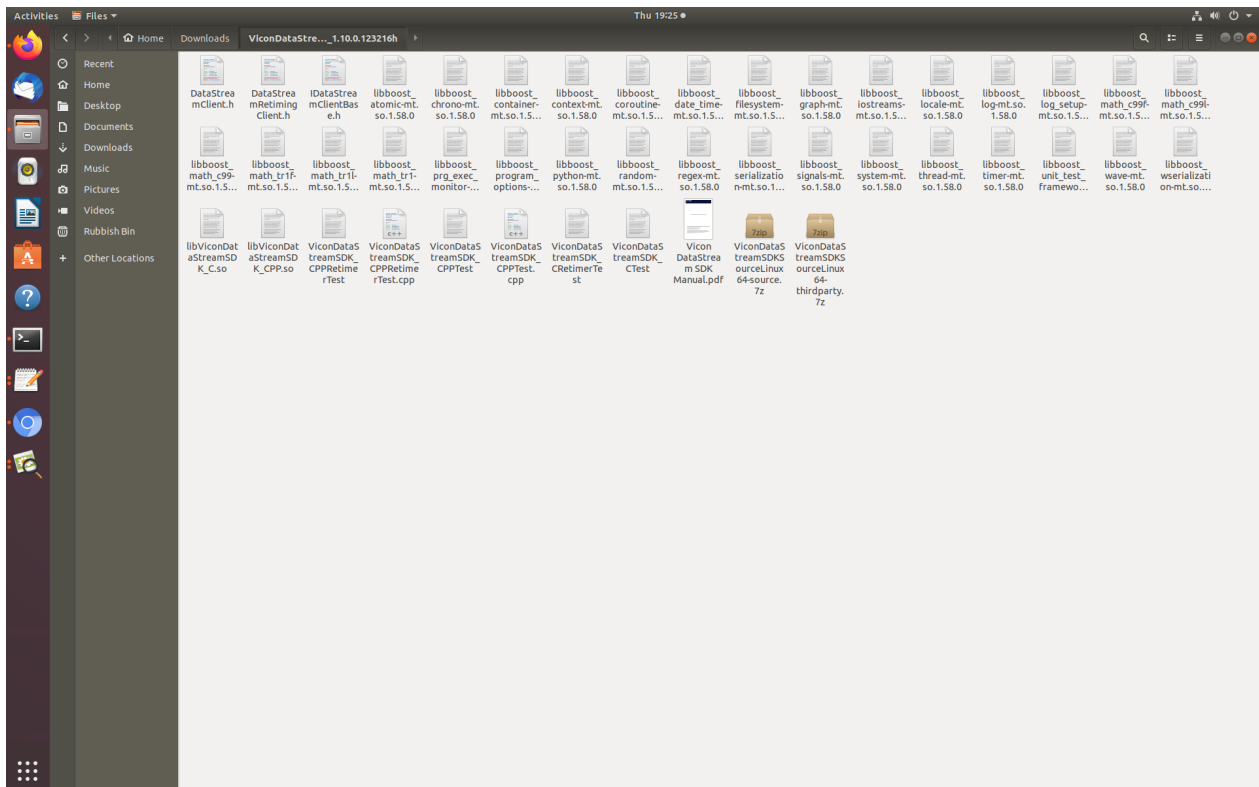


Figure 41: Vicon SDK Folder

Extract the 2 zipped files as shown in figure 42.

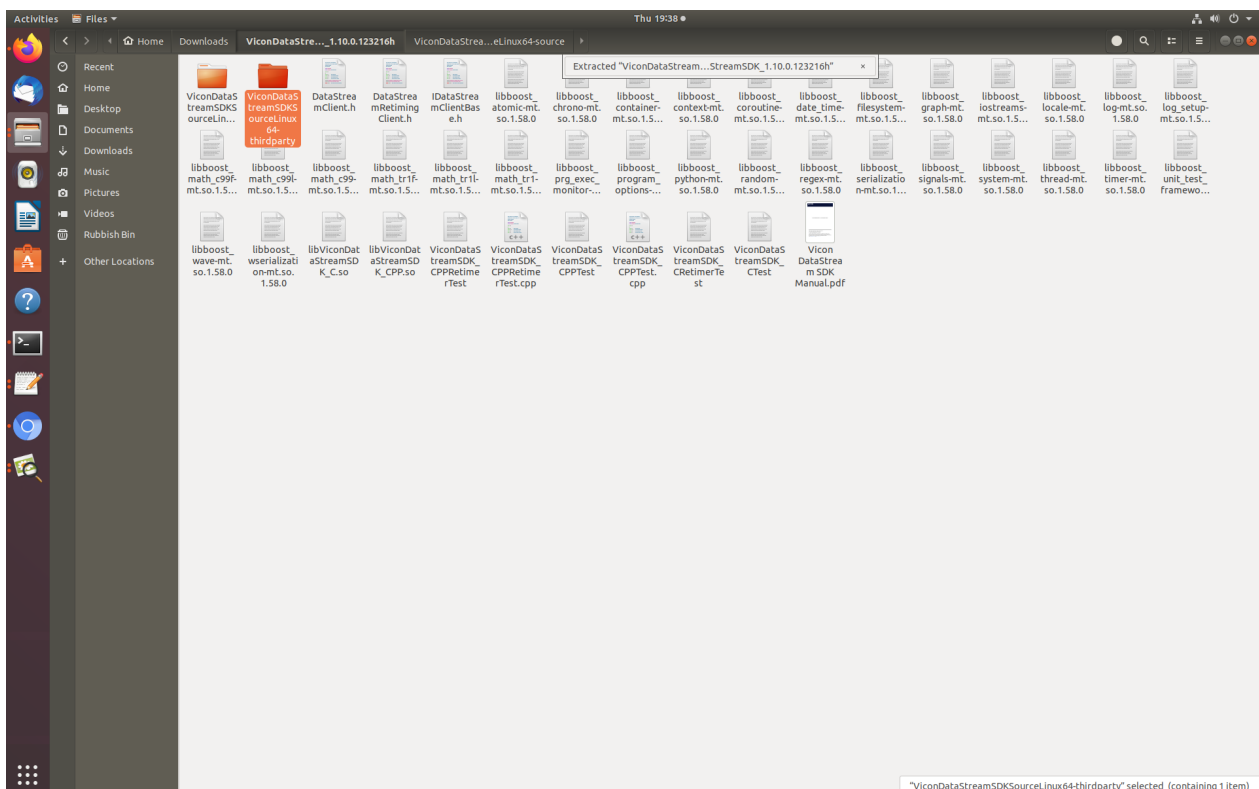


Figure 42: Extracted 2 Subfolders of Vicon SDK Folder

4.4 Downloading pyvicon

After SDK is installed, you will need install a custom python script by MathGaron [1] that bridges the VICON SDK to MAVPROXY. This is installed via <https://github.com/MathGaron/pyvicon>, as seen in figure 43:

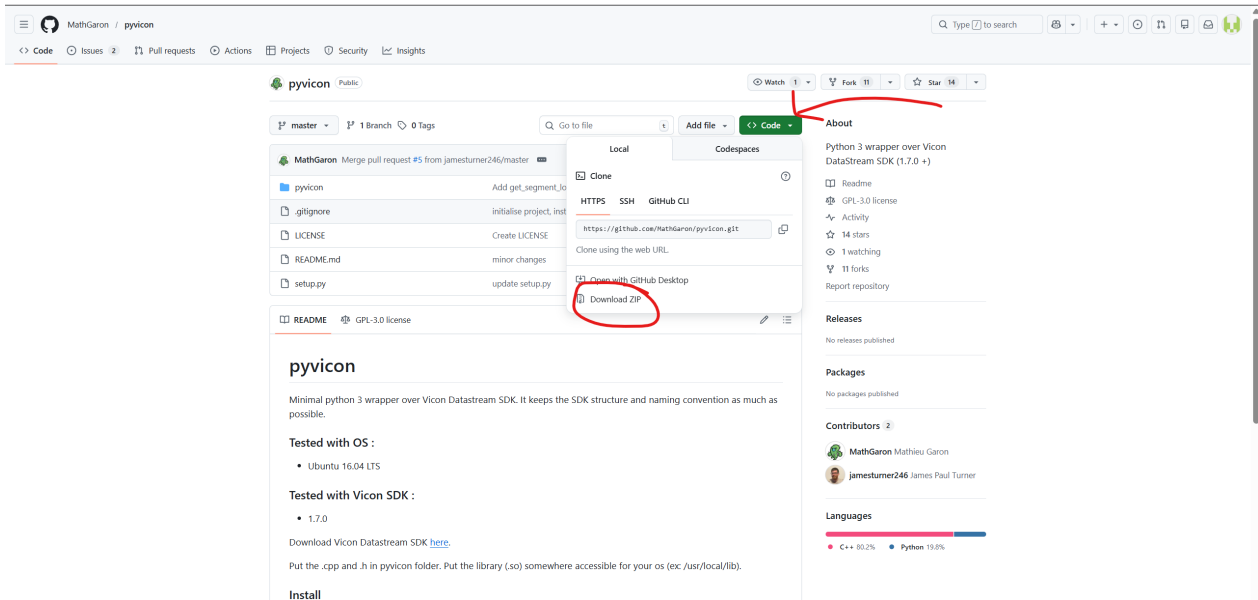


Figure 43: Install the Github file. (Click *Code*, and *Download Zip*)

Extract the zipped pyvicon folder into your download folders. Your download folder should now have the Vicon SDK folder & pyvicon folder, as seen in figure 44:

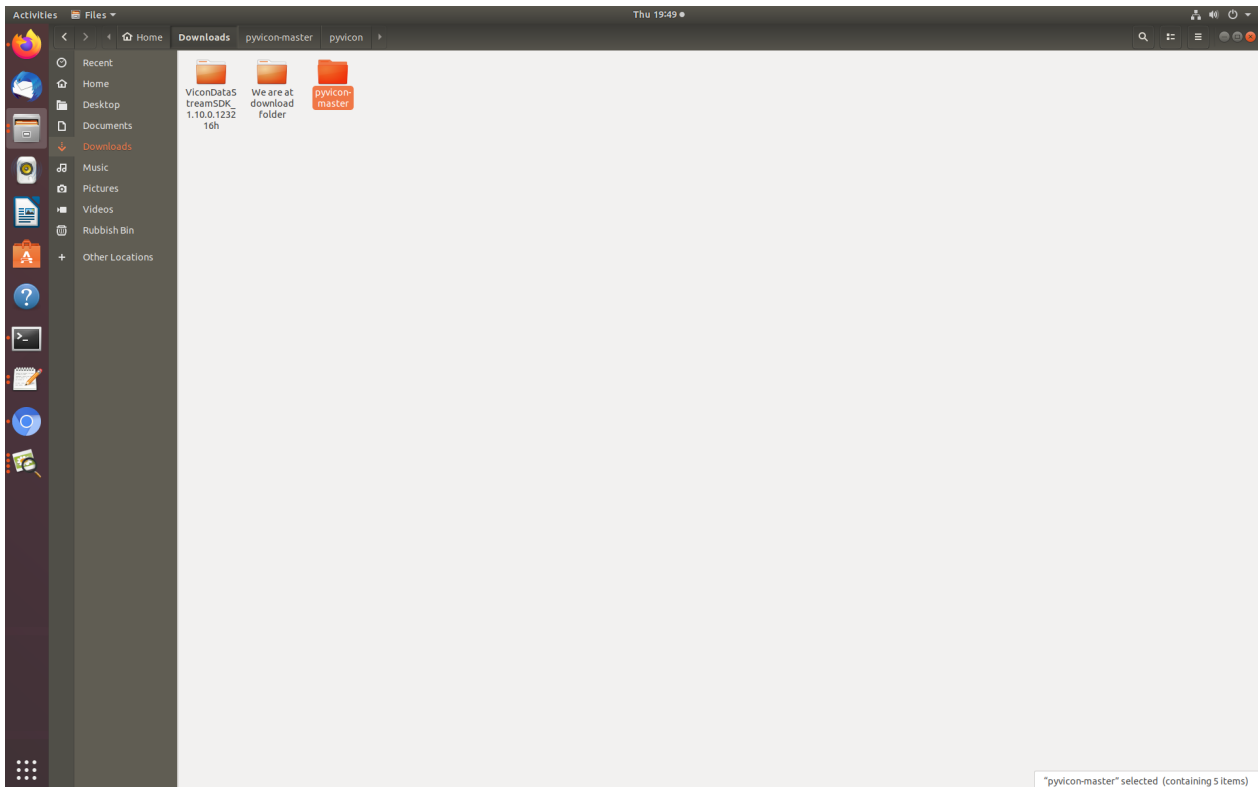


Figure 44: Vicon SDK folder and pyvicon folder in Downloads Folder

4.5 Merging VICON SDK into pyvicon

Next, you need to merge the .so files of the VICON SDK into your venv's lib directory. Run these following code in your terminal, as seen in figure 45:

```
(VICON) inspection@inspection-ThinkStation-P320:~$ find ~/Downloads/ViconDataStreamSDK_1.10.0.123216h -name "*.so" -exec mv {} /home/inspection/miniconda3/envs/VICON/lib/ \;
```

```
(VICON) inspection@inspection-ThinkStation-P320:~$ find ~/Downloads/ViconDataStreamSDK_1.10.0.123216h -name "*.cpp" -exec mv {} ~/Downloads/pyvicon-master/pyvicon/ \;
```

```
(VICON) inspection@inspection-ThinkStation-P320:~$ find ~/Downloads/ViconDataStreamSDK_1.10.0.123216h -name "*.h" -exec mv {} ~/Downloads/pyvicon-master/pyvicon/ \;
```

Figure 45: Three scripts to merge SDK into venv's lib folder and setting up pyvicon.

The first of the three commands above moves all the .so files from the Vicon SDK folder into your venv's lib folder, which is usually located in `miniconda3/envs/VICON/lib`:

```
find ~/Downloads/ViconDataStreamSDK_1.10.0.123216h -name "*.so" -exec mv {}  
/home/inspection/miniconda3/envs/VICON/lib/ \;
```

Your `miniconda3/envs/VICON/lib` folder should now have the .so files of Vicon SDK, it should look like figure 46:

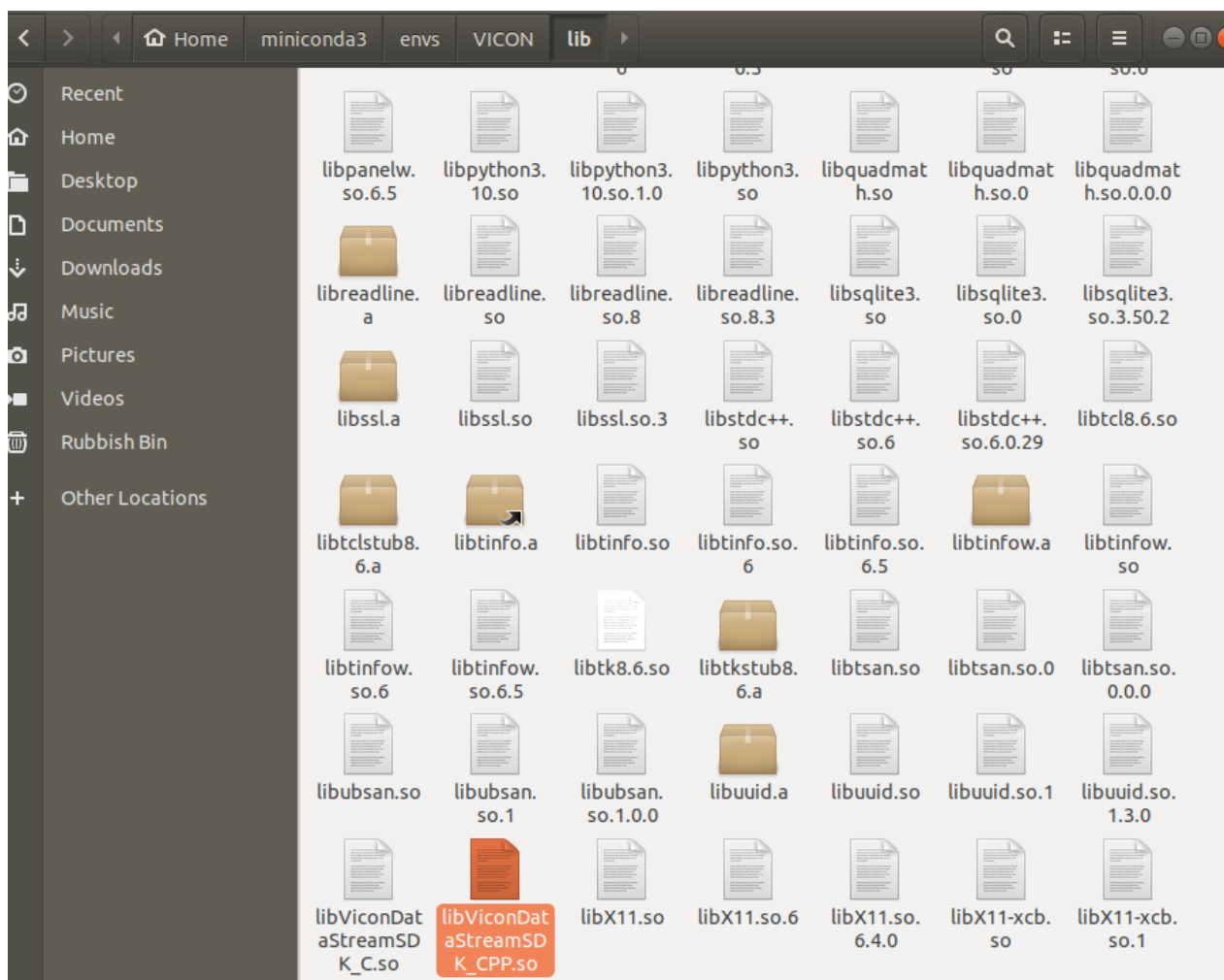


Figure 46: .so files of Vicon SDK merged into venv's lib folder

The 2nd and 3rd commands move the .cpp and .h files of ViconDataStreamSDK into pyvicon-master/pyvicon:

```
find ~/Downloads/ViconDataStreamSDK_1.10.0.123216h -name "*.cpp" -exec mv {} ~  
/Downloads/pyvicon-master/pyvicon/ \;
```

```
find ~/Downloads/ViconDataStreamSDK_1.10.0.123216h -name "*.h" -exec mv {} ~  
/Downloads/pyvicon-master/pyvicon/ \;
```

The pyvicon folder should now have the .cpp and .h files of Vicon SDK like in figure 47:

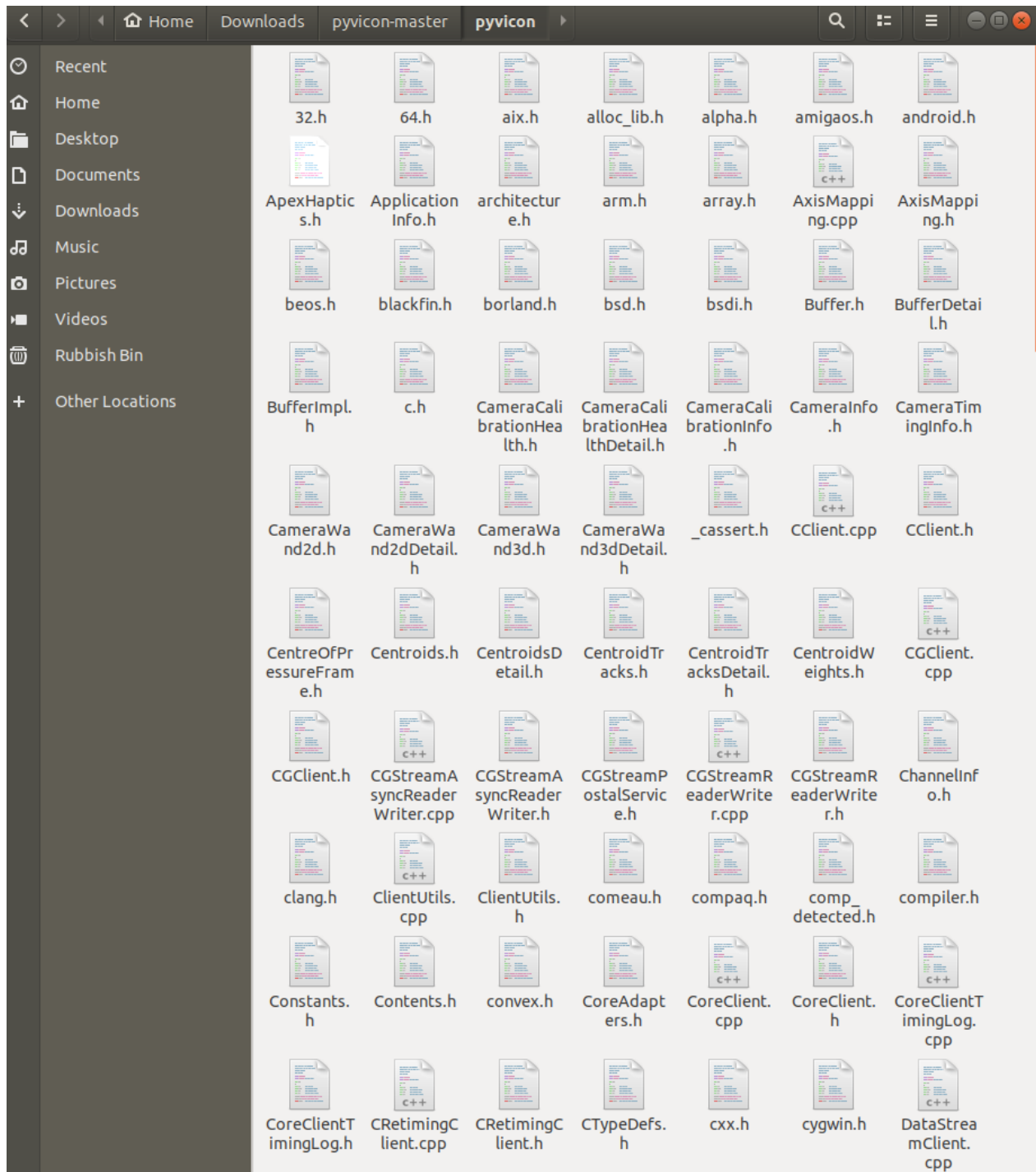


Figure 47: .cpp and .h files of Vicon SDK merged into pyvicon folder

4.6 Installing Dependencies Packages

A few dependencies are needed in order for pyvicon & Mavproxy to work. Below are the dependencies and the corresponding versions that I used for a working setup: (Do note wxpython takes approximately 20 minutes to install, so be patient when you see the installation stall at wxpython, it is actually installing)

```
sudo apt-get install libgtk-3-dev
sudo apt remove modemmanager -y
python3 -m pip install numpy==2.2.6
python3 -m pip install pathlib2==2.3.7.post1
python3 -m pip install matplotlib==3.10.9
python3 -m pip install opencv-python==4.13.0.92
python3 -m pip install Cython==3.2.4
python3 -m pip install future==1.0.0
python3 -m pip install wxpython==4.2.5
```

Similarly, you can visit my Github page

<https://matthewt0809.github.io/Vicon-Positioning-System-To-Pixhawk6C-Guide/> for the requirements.txt file that has all the dependencies listed in it, you can download this .txt file and install all dependencies by running the following code:

```
python3 -m pip install -r requirements.txt
```

4.7 Setting up pyvicon

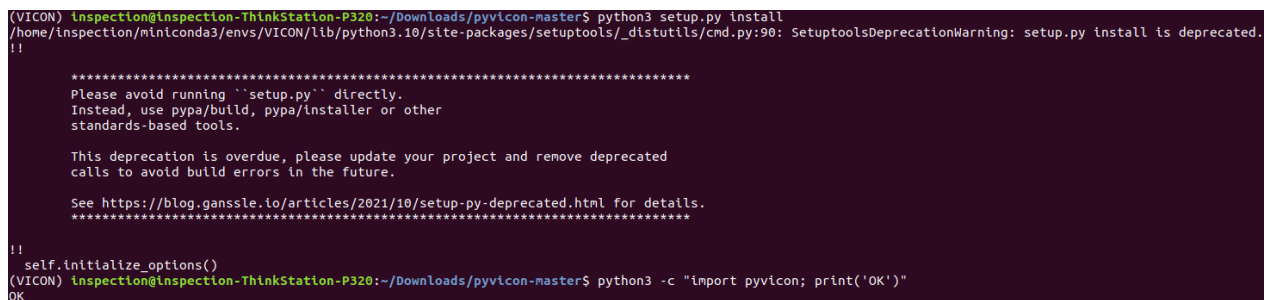
To setup pyvicon wrapper, run the following code inside the *Downloads/pyvicon-master* folder, as seen in figure 48:

```
cd ~/Downloads/pyvicon-master
```

```
python3 setup.py install
```

Test if the setup worked:

```
python3 -c "import pyvicon; print('OK')"
```



```
(VICON) inspection@Inspection-ThinkStation-P320:~/Downloads/pyvicon-master$ python3 setup.py install
/home/inspection/miniconda3/envs/VICON/lib/python3.10/site-packages/setuptools/_distutils/cmd.py:90: SetuptoolsDeprecationWarning: setup.py install is deprecated.
!!
*****
Please avoid running ``setup.py`` directly.
Instead, use pypa/build, pypa/installer or other
standards-based tools.

This deprecation is overdue, please update your project and remove deprecated
calls to avoid build errors in the future.

See https://blog.ganssle.io/articles/2021/10/setup-py-deprecated.html for details.
*****
!!
self.initialize_options()
(VICON) inspection@Inspection-ThinkStation-P320:~/Downloads/pyvicon-master$ python3 -c "import pyvicon; print('OK')"
```

Figure 48: Overall setup of pyvicon

If you see *OK* printed in the terminal, you have successfully setup pyvicon. If there is an error, it is mostly arising from a mismatch in boost libraries. To keep the guide concise, please refer to appendix A for common debugging solutions that has worked for me.

4.8 Installing Mavproxy & Mavlink

Next, the installation of Mavlink and Mavproxy. Run the following code as shown in figure 49:

```
python3 -m pip install pymavlink==2.4.49
python3 -m pip install mavproxy==1.8.74
```

```
Inspection@Inspection-ThinkStation-P320:~$ python3 -m pip install --upgrade --user pymavlink
Collecting pymavlink
  Using cached https://files.pythonhosted.org/packages/21/a2/0a4ce323178f60f869e42a2ed3844bead7b685807674bef966a39661606e/pymavlink-2.4.49.tar.gz
Collecting fastcrc (from pymavlink)
  Downloading https://files.pythonhosted.org/packages/b8/3a/60293f7eae505634bcec5d47fc2416fa88a64c5dbaa403ea475198f2a3b3/fastcrc-0.2.1-cp36-abl3-manylinux_2_5_x86_64.manylinux1_x86_64.whl (1.0MB)
  100% |#####| 1.0MB 1.2MB/s
Collecting lxml (from pymavlink)
  Using cached https://files.pythonhosted.org/packages/ff/78/1d8a946398eb47680d196118facc05c4404ba185365deef22f0fd6d8f6d/lxml-5.4.0-cp36-cp36m-manylinux_2_5_x86_64.manylinux1_x86_64.whl
Building wheels for collected packages: pymavlink
  Running setup.py bdist_wheel for pymavlink ... done
  Stored in directory: /home/Inspection/.cache/pip/wheels/9a/a1/27/dd34a2b33176d1d78e9dede896e92c04c03f52b522e346b
Successfully built pymavlink
Installing collected packages: fastcrc, lxml, pymavlink
Successfully installed fastcrc-0.2.1 lxml-5.4.0 pymavlink-2.4.49
Inspection@Inspection-ThinkStation-P320:~$
Inspection@Inspection-ThinkStation-P320:~$ python3 -m pip install --upgrade --user mavproxy
Collecting mavproxy
  Downloading https://files.pythonhosted.org/packages/5d/ae/a4903e8231cc2563af8f215e4ce89374ac75ff5996a372494d74ea53ae3/mavproxy-1.8.74-py3-none-any.whl (28.8MB)
  100% |#####| 28.8MB 55kB/s
Collecting pyserial==3.0 (from mavproxy)
  Downloading https://files.pythonhosted.org/packages/07/bc/f8a445451b253b285629263eb51c2d8e9bcea4fc97826266d186f96f558/pyserial-3.5-py2.py3-none-any.whl (90kB)
  100% |#####| 92kB 4.8MB/s
Collecting pynmeagps (from mavproxy)
  Downloading https://files.pythonhosted.org/packages/83/14/bcfa52eb0334eb9d216fd8df40919f0934cfed87ead9da9d74efdf76316/pynmeagps-1.0.13-py3-none-any.whl
Collecting pymavlink==2.4.14 (from mavproxy)
Collecting numpy (from mavproxy)
  Using cached https://files.pythonhosted.org/packages/45/b2/6c7545bb7a38754d63048c7696804a0d947328125d81bf12beaa692c3ae3/numpy-1.19.5-cp36-cp36m-manylinux1_x86_64.whl
Collecting fastcrc (from pymavlink==2.4.14->mavproxy)
  Using cached https://files.pythonhosted.org/packages/b8/3a/60293f7eae505634bcec5d47fc2416fa88a64c5dbaa403ea475198f2a3b3/fastcrc-0.2.1-cp36-abl3-manylinux_2_5_x86_64.manylinux1_x86_64.whl
Collecting lxml (from pymavlink==2.4.14->mavproxy)
  Using cached https://files.pythonhosted.org/packages/ff/78/1d8a946398eb47680d196118facc05c4404ba185365deef22f0fd6d8f6d/lxml-5.4.0-cp36-cp36m-manylinux_2_5_x86_64.manylinux1_x86_64.whl
Installing collected packages: pyserial, pynmeagps, fastcrc, lxml, pymavlink, numpy, mavproxy
Successfully installed fastcrc-0.2.1 lxml-5.4.0 mavproxy-1.8.74 numpy-1.19.5 pymavlink-2.4.49 pynmeagps-1.0.13 pyserial-3.5
```

Figure 49: Installing Mavproxy and Mavlink

To test if your mavproxy installation is correct, run the following:

```
mavproxy.py --master :14550 --console --map
```

You should see the console and map window pops out as shown in figure 50. It will mention that link1 is down from the console, which is correct as you have not connected to your drone yet.

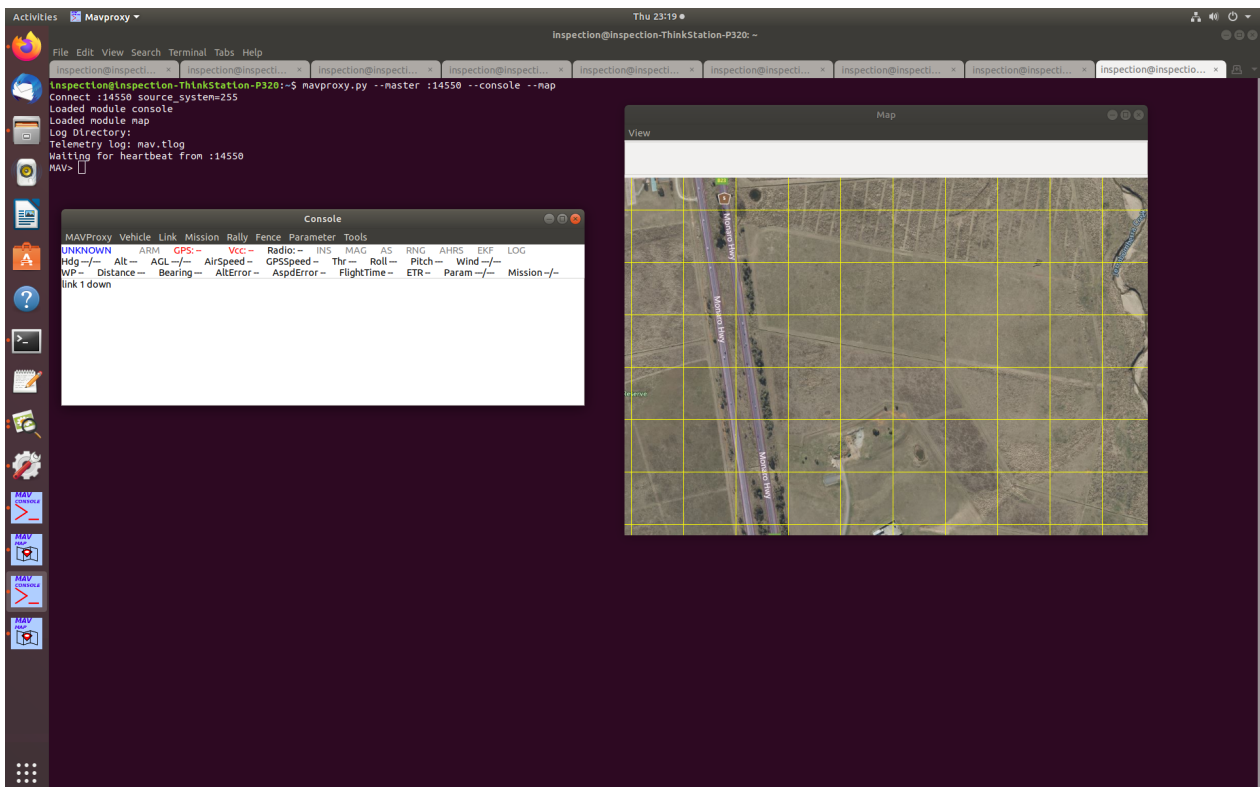


Figure 50: Mavproxy console and map

4.9 Network setup

Once all dependencies are installed, musculoskeletally move towards the Host PC with your GCS, locate the Vicon Ethernet Cable as shown in figure 51. Connect the ethernet cable to your GCS.

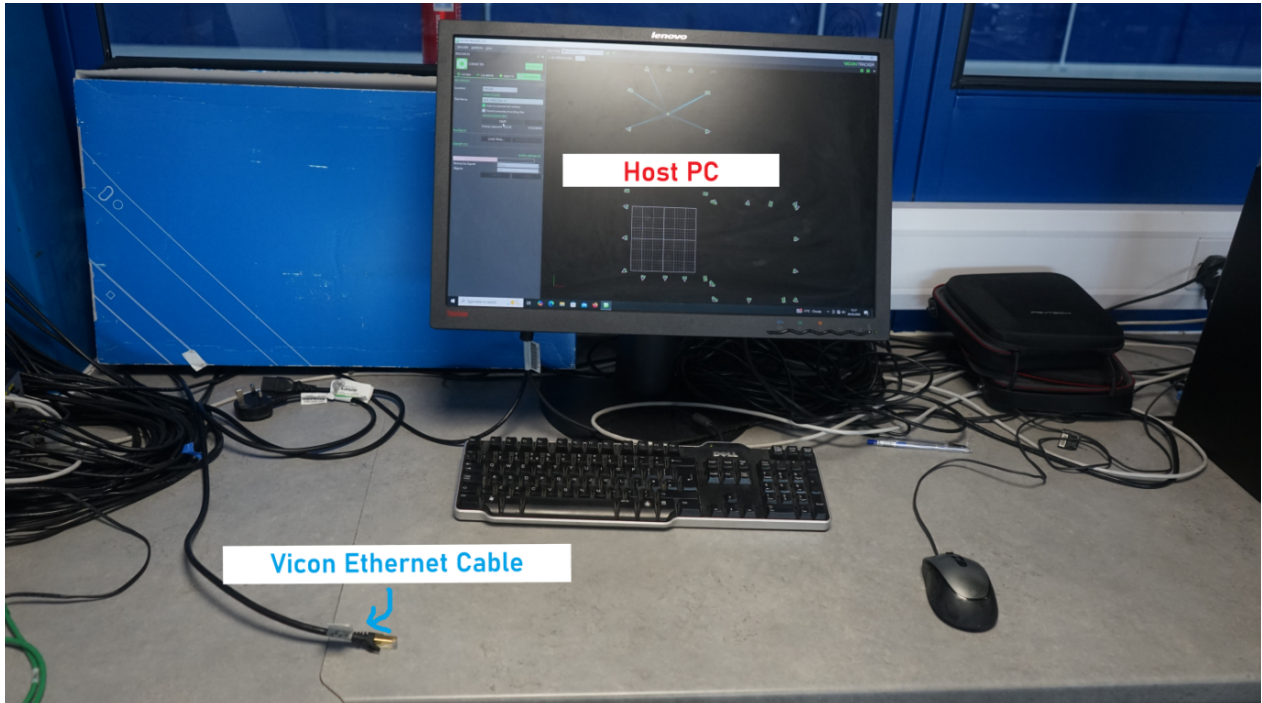


Figure 51: Photo of Vicon Ethernet Cable

Once connected to the Ethernet cable, you will need to name your source of your Vicon positional data's ip address as *vicon*, this can be done by typing this in Terminal, as shown in figure 52:⁶

```
sudo nano /etc/hosts
```

⁶If you do some crawling around, you will note that this ethernet cable actually comes from within the flight arena, which would be useful if you want your GCS within the flight arena to test localisation later on.

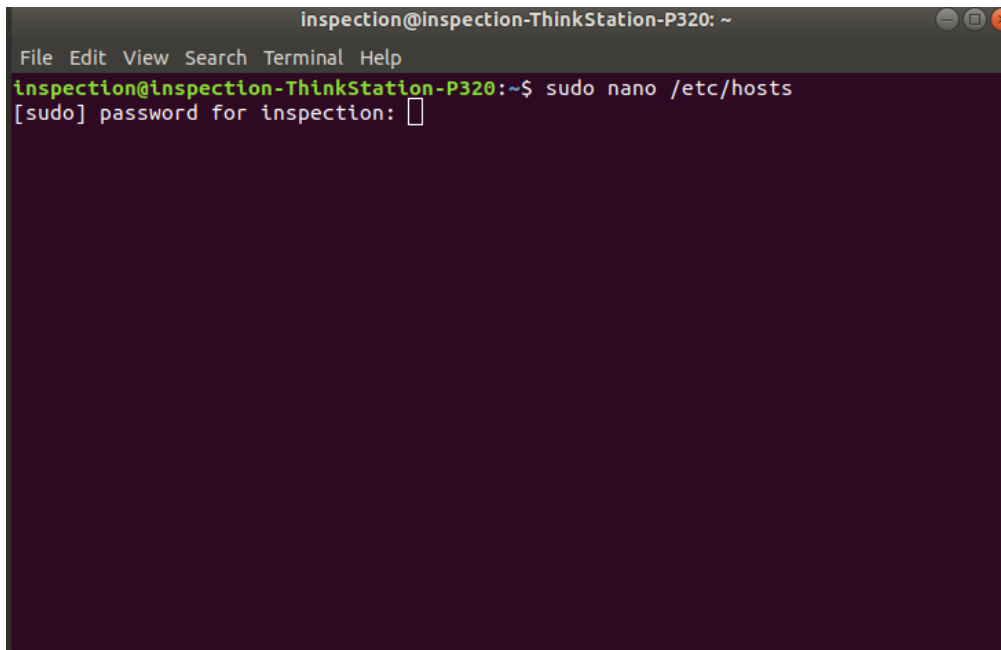


Figure 52: Code for network name editing

A window will show up like figure 53:

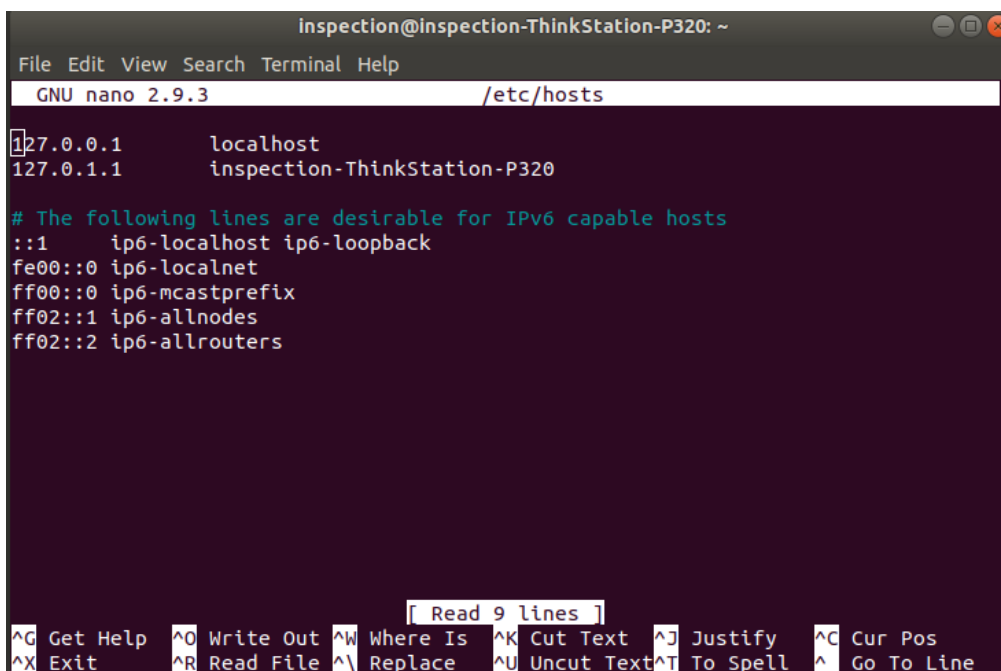
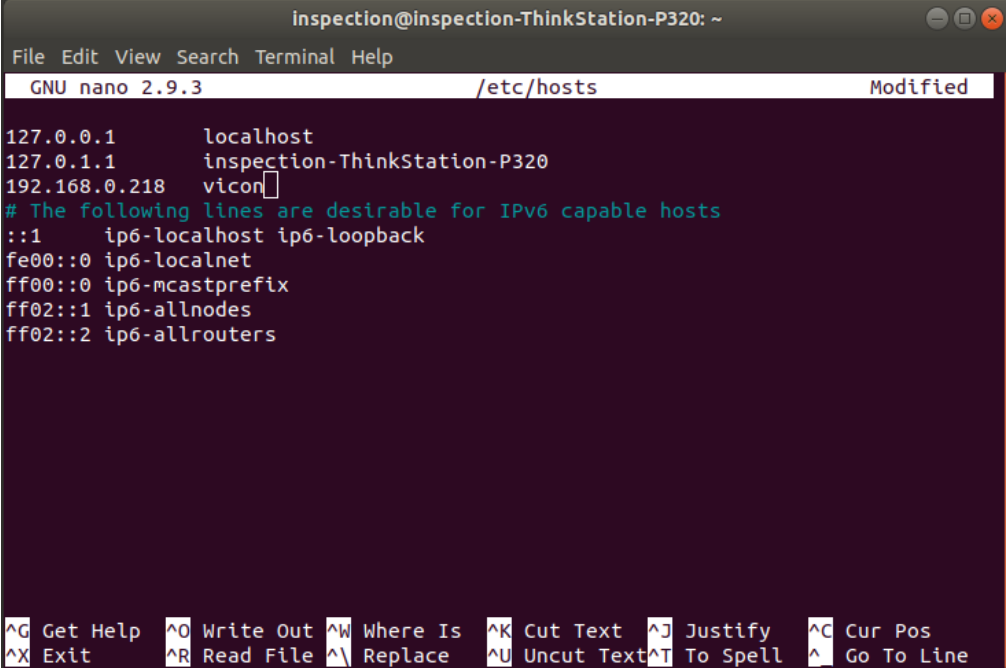


Figure 53: Hosts file for network name editing

Add *192.168.0.218 vicon*⁷ as shown in figure 54. Then press Ctrl+O, enter to save the changes, Ctrl+X to exit the window.

⁷This IP address is empirical, if you use another router with a different network card, you will need to adjust the IP address accordingly.



```
inspection@inspection-ThinkStation-P320: ~
File Edit View Search Terminal Help
GNU nano 2.9.3 /etc/hosts Modified

127.0.0.1    localhost
127.0.1.1    inspection-ThinkStation-P320
192.168.0.218 vicon
# The following lines are desirable for IPv6 capable hosts
::1         ip6-localhost ip6-loopback
fe00::0     ip6-localnet
ff00::0     ip6-mcastprefix
ff02::1     ip6-allnodes
ff02::2     ip6-allrouters

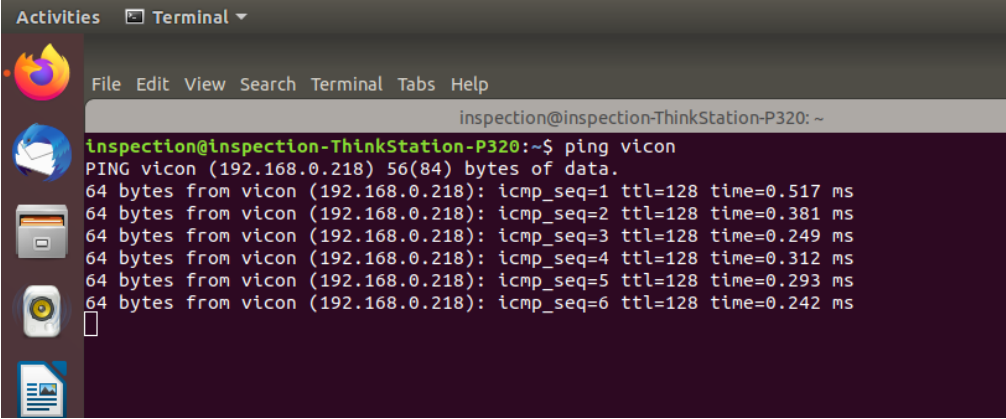
^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos
^X Exit    ^R Read File ^_ Replace ^U Uncut Text ^T To Spell ^_ Go To Line
```

Figure 54: Adding vicon IP address to hosts file

Test your connection to vicon via ethernet cable by doing:

`ping vicon`

If *64 bytes from vicon (192.168.0.218): icmp_seq=1 ttl=128 time=0.517ms* messages begin appearing like shown in figure 55, you have a correct connection between Vicon Tracker and your computer!



```
Activities Terminal
File Edit View Search Terminal Tabs Help
inspection@inspection-ThinkStation-P320: ~
inspection@inspection-ThinkStation-P320:~$ ping vicon
PING vicon (192.168.0.218) 56(84) bytes of data:
64 bytes from vicon (192.168.0.218): icmp_seq=1 ttl=128 time=0.517 ms
64 bytes from vicon (192.168.0.218): icmp_seq=2 ttl=128 time=0.381 ms
64 bytes from vicon (192.168.0.218): icmp_seq=3 ttl=128 time=0.249 ms
64 bytes from vicon (192.168.0.218): icmp_seq=4 ttl=128 time=0.312 ms
64 bytes from vicon (192.168.0.218): icmp_seq=5 ttl=128 time=0.293 ms
64 bytes from vicon (192.168.0.218): icmp_seq=6 ttl=128 time=0.242 ms
```

Figure 55: Successful ping to vicon

5 Drone Setup

In this section, things tends to be more bespoke and my instructions will start to become more arbitrary and generalised as the configuration differs for each drone. To appericate the process however, this section provides a working example of a functional communication link with the highlighted components used.

In this guide, an ESP8266 Wifi Module ⁸, a Pixhawk6C flight controller⁹¹⁰, a Radio Controller/Reciever configuration, IR Markers, and a custom drone is used. A photo of the configuration is shown in figure 56 below:



Figure 56: Hardware Required.

The pinout connection between the wifi module and the flight controller is as follows:

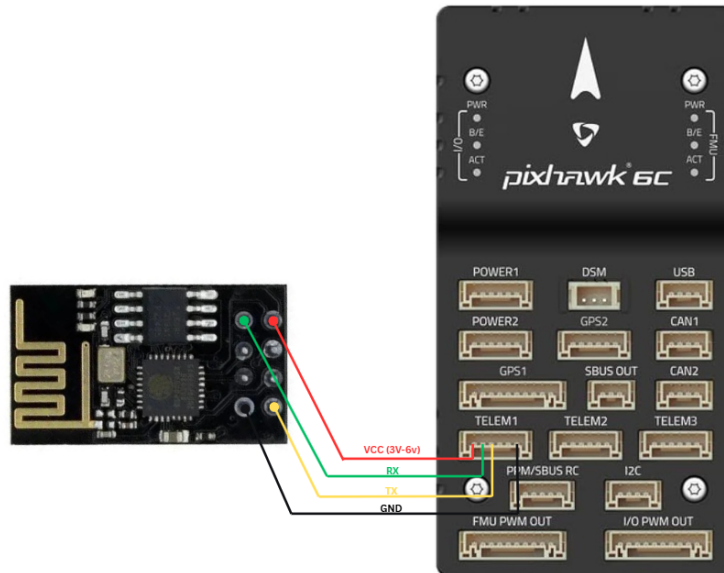


Figure 57: Pinout from Pixhawk 6c to ESP8266.

⁸ESP8266 is reported to have connection issues in swarms, use raspberry pi if doing swarm flights.

⁹You are more than welcome (encouraged even) to explore using host wifi features from raspberry pi, which I have been told to be more stable than the ESP8266.

¹⁰Do note that not all Pixhawk works the same when setting this up, I have had no success with older versions of Radiolink Pixhawk with this configuration, also I have not tried this with PX4.

5.1 Setting up the ESP8266

To set up the ESP8266, power up the Pixhawk (Remove the propellers).^{11 12} and connect your GCS to the wifi being hosted by the ESP8266. The wifi connection should be just a conventional wifi network as shown in figure 58. Connect to it and enter the password like a normal wifi network.

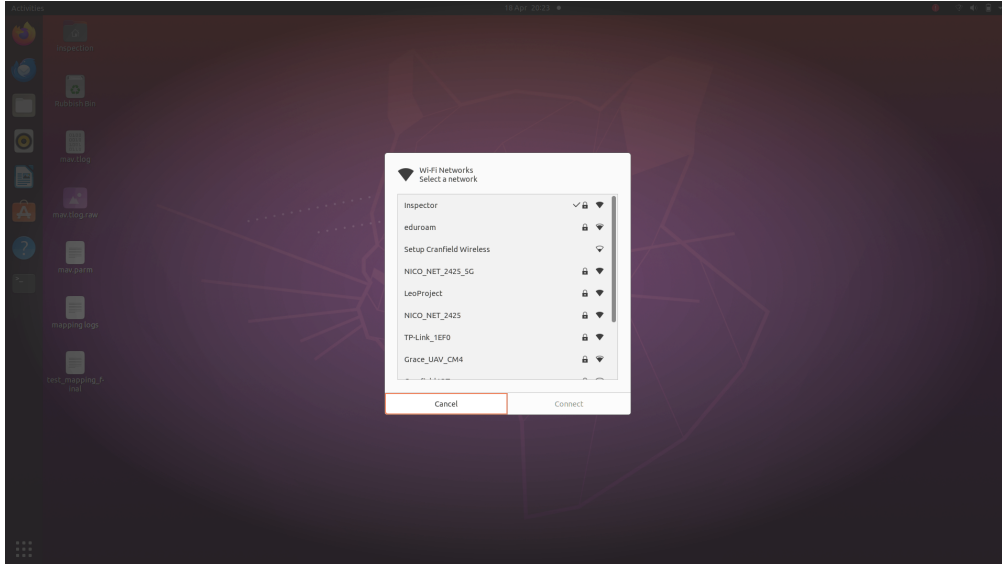


Figure 58: Wifi Network Hosted by ESP8266.

Once you are connected to the WiFi, you need to find out the IP address of the WiFi network, this is known by going into the wifi settings in 59:

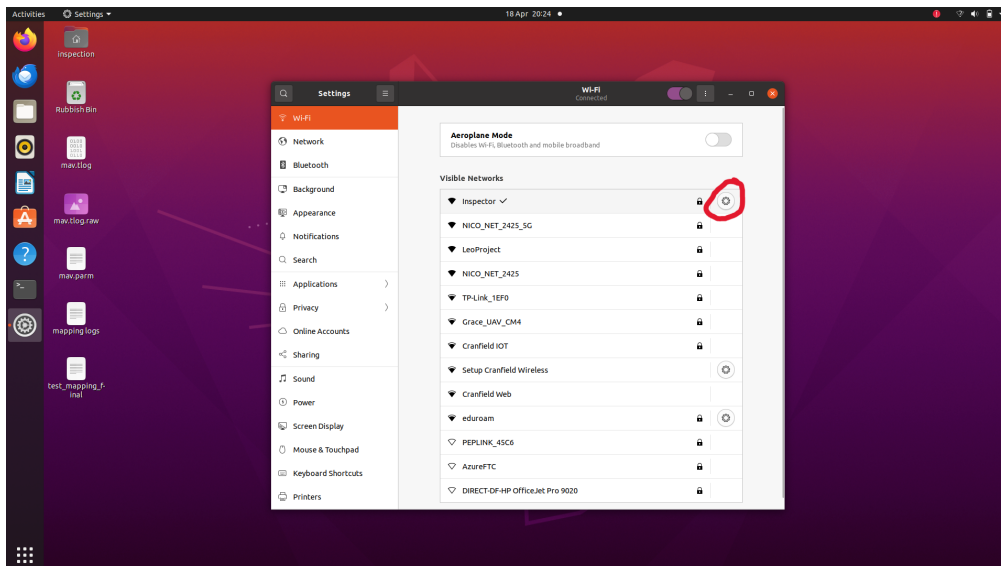


Figure 59: WiFi Settings.

Note down the WiFi's Default Route as seen in figure 60:

¹¹If the ESP8266 is being used for the 1st time, please see [2] for setting up wifi name and password.

¹²If you are using the drone after the 2025/2026 cohort, it is named as Inspector/Surveyour with password being inspector/surveyor correspondingly

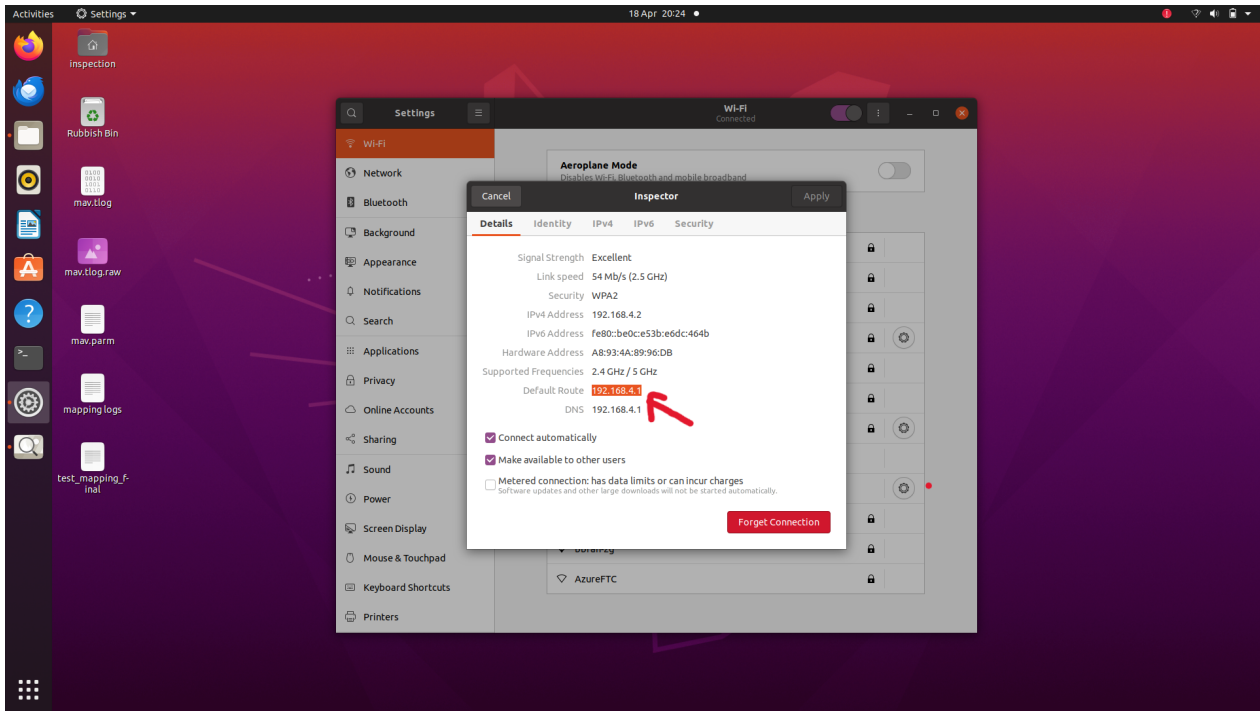


Figure 60: Default Route of the Wifi Network.

Enter the WiFi's default route into a browser, you should see a page as shown in figure 61 below:

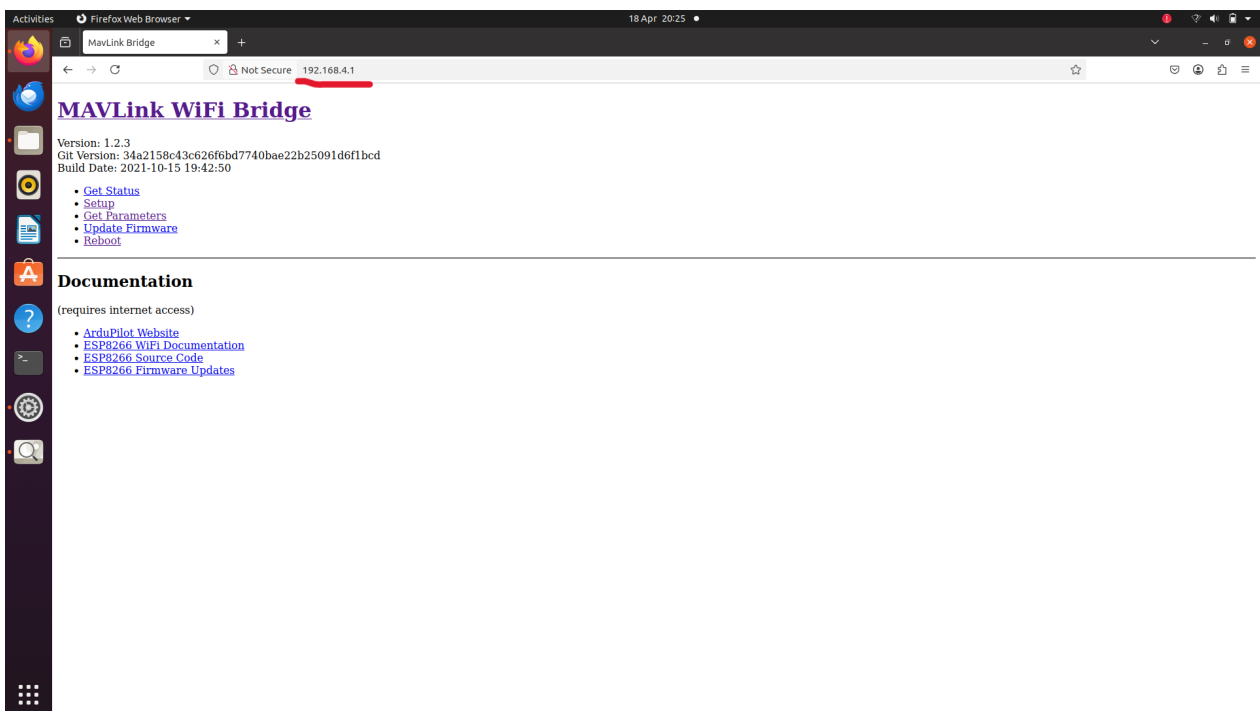


Figure 61: ESP8266 Configuration Page.

Press *Setup*, change your wifi SSID and password as you wish. Noting your host port's digit for later, as shown in figure 62:

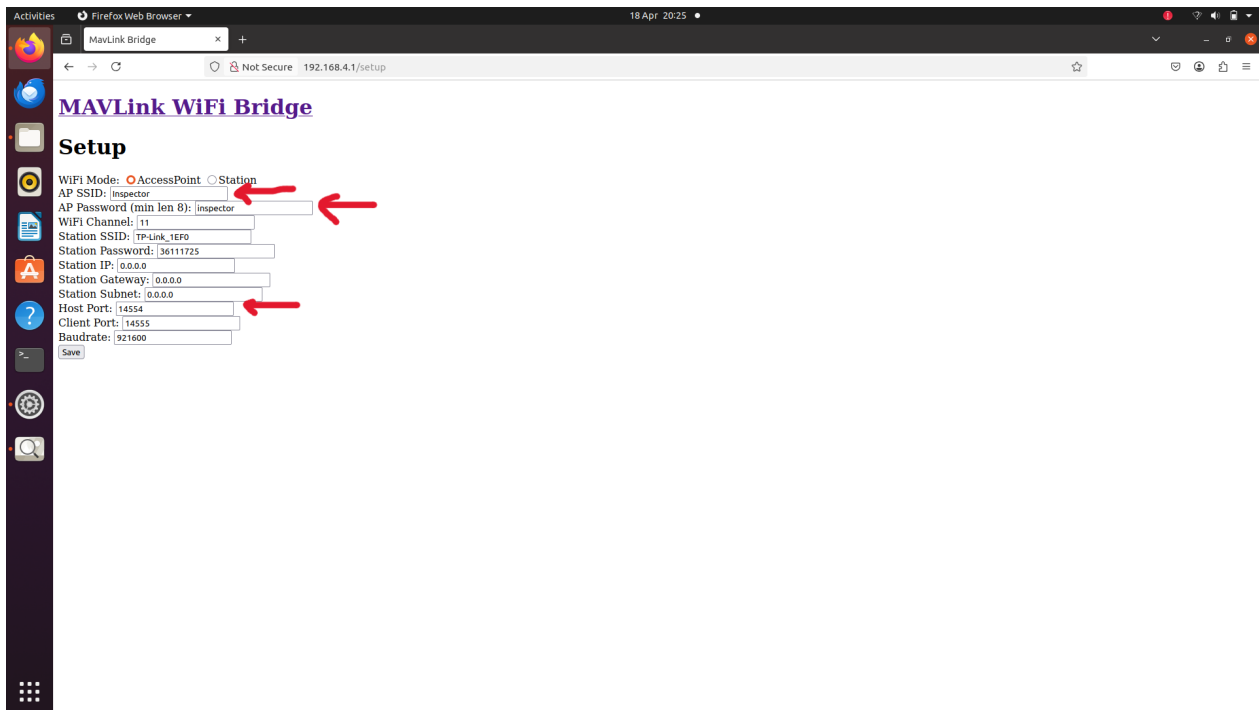


Figure 62: SSID and Password Setup

Return to the home page, and press *Reboot*, as shown in 63:

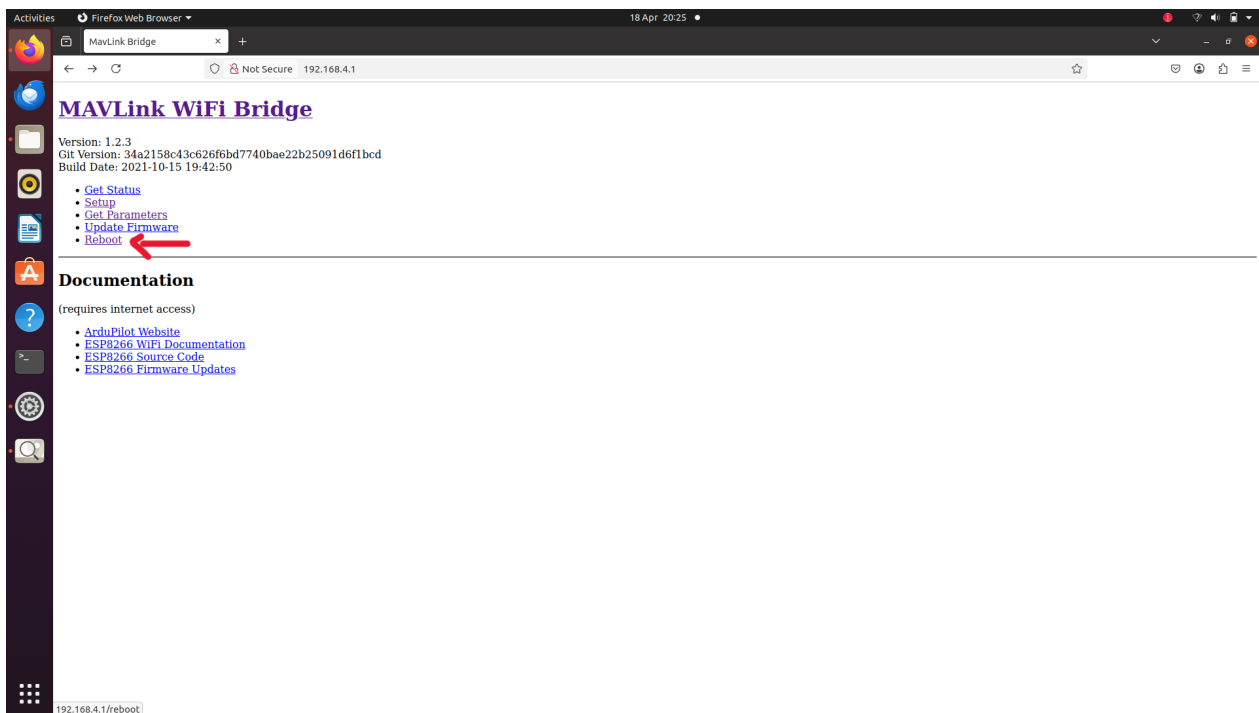


Figure 63: Saving and rebooting the ESP8266.

The ESP8266 is now configured. You now need to configure the Pixhawk so that it is compatible with the wifi module, this is done by using MissionPlanner, which is covered in the next subsection.

5.2 MissionPlanner Configuration

Connect the usb-c port of the Pixhawk6C to a computer as shown in figure 64 below:

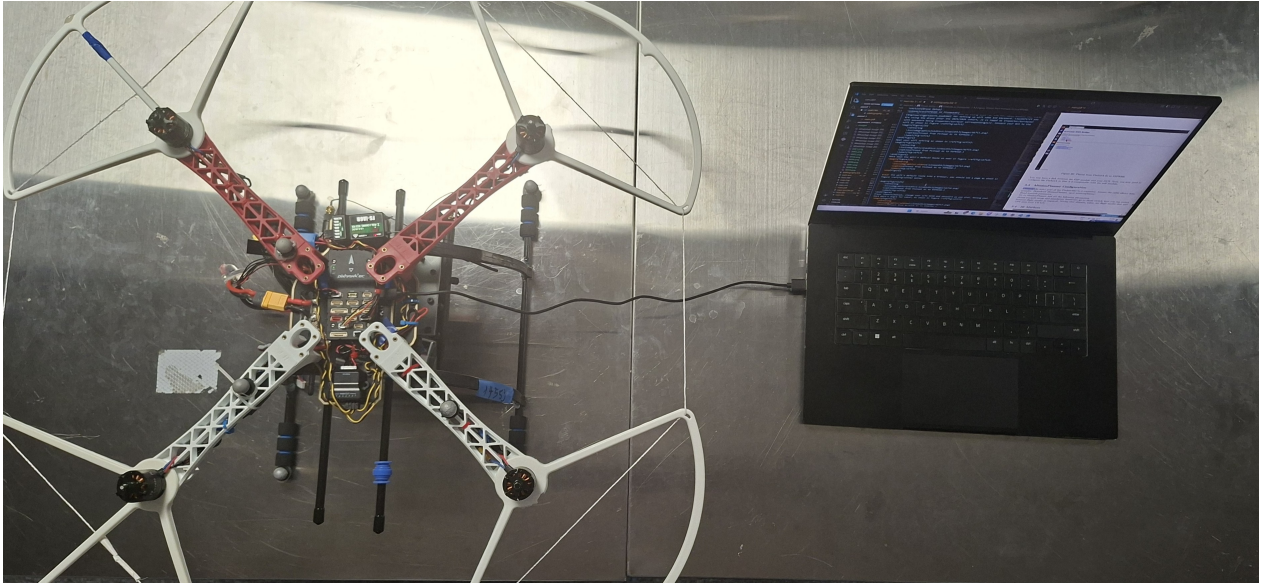


Figure 64: Connecting Pixhawk6C to Computer.

Download MissionPlanner in [3] and open it. On the top right corner of the welcome screen, select *AUTO* and press *Connect* as seen in figure 65:

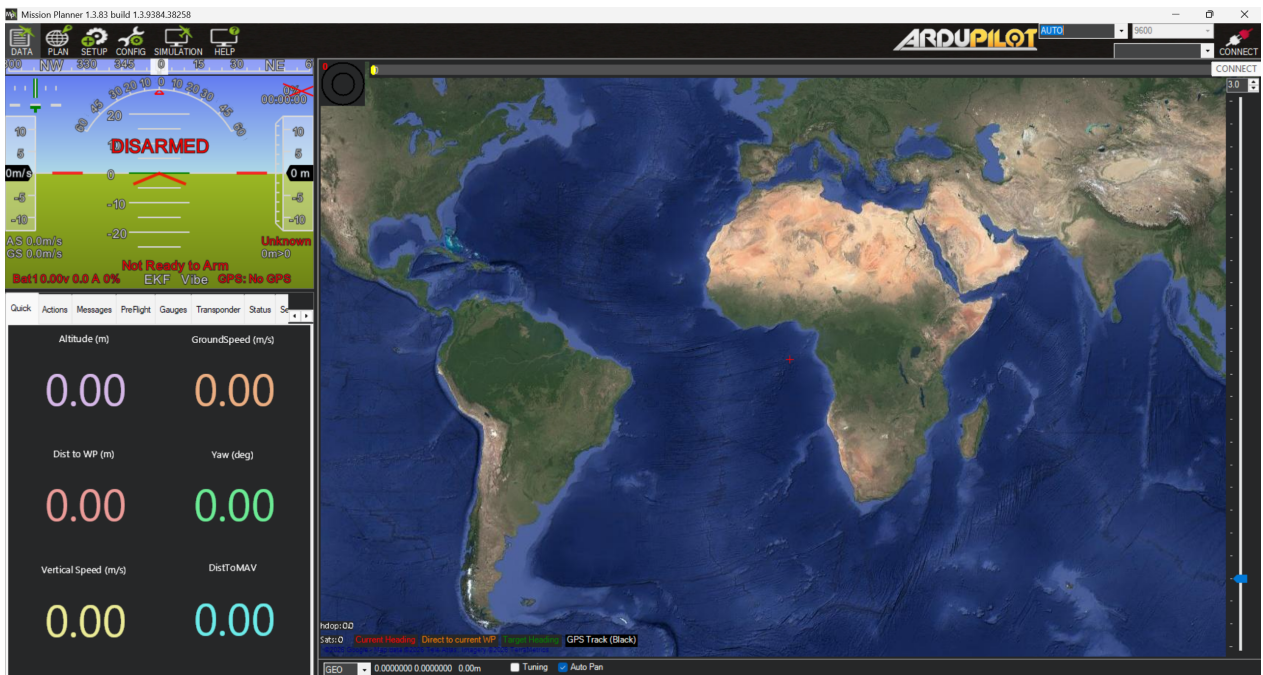


Figure 65: Connecting to the Drone with MissionPlanner.

Once entered, align your focus to the top left portion of the app. Press *Config*, then press *Full Parameter List* shown on the left side panel, a warning will show up, press *OK*, as seen below in figure 66:

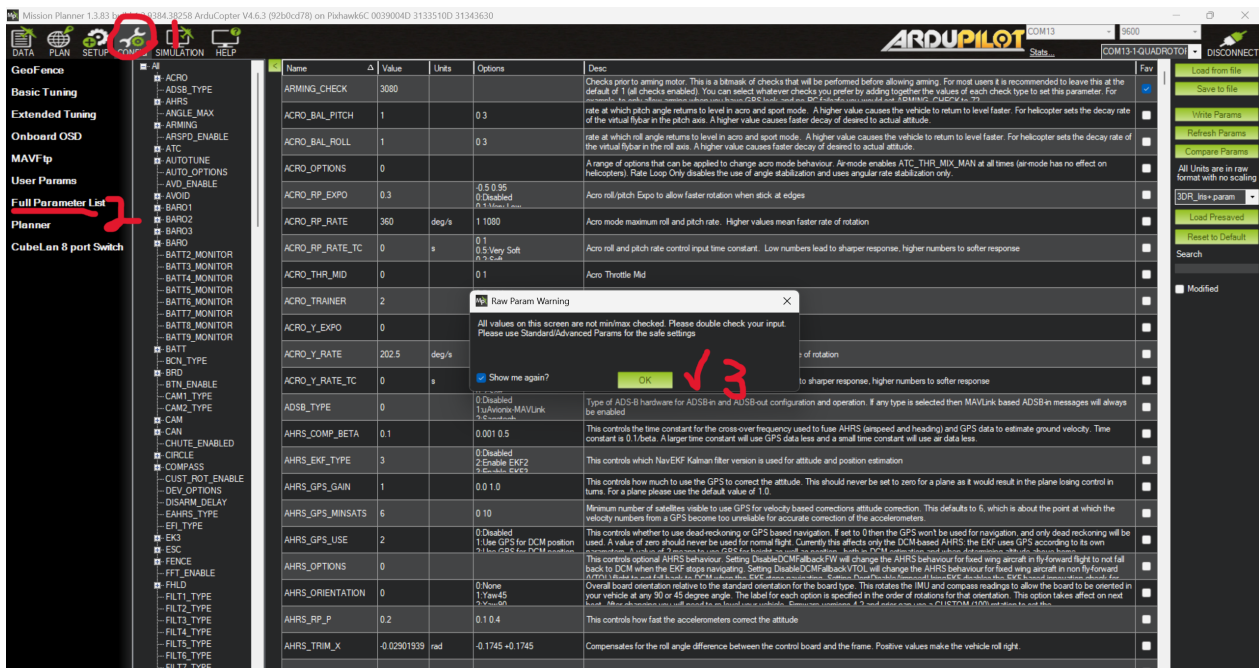


Figure 66: Navigating to full Parameter List.

Search for *AHRS EKF TYPE 3* in the searchbox on the right panel, then change the value of it to *3*, press *Write Params* on the right panel, as seen below in figure 67:

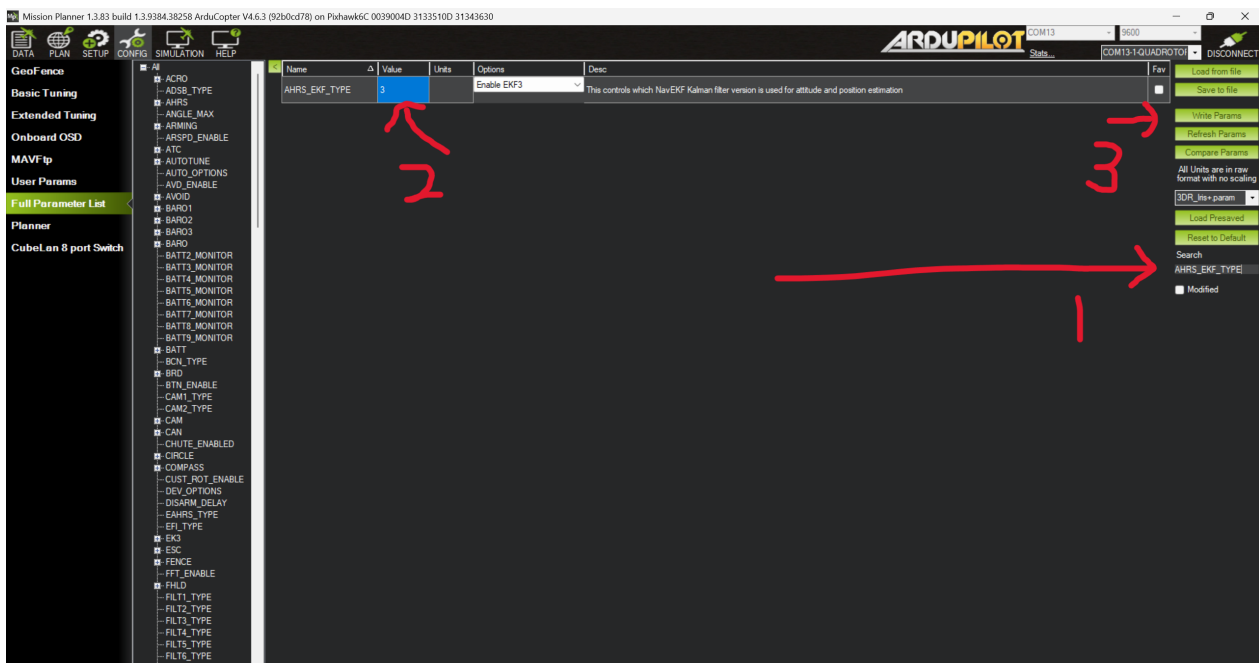


Figure 67: Changing the AHRS EKF TYPE Parameter.

A window will pop up asking for confrimation, click *Yes* as shown in figure 68



Figure 68: Confirming the Parameter Change.

Once confirmed, the value is now saved to the Pixhawk6c flight controller, a confirmation message will be shown like figure 69.

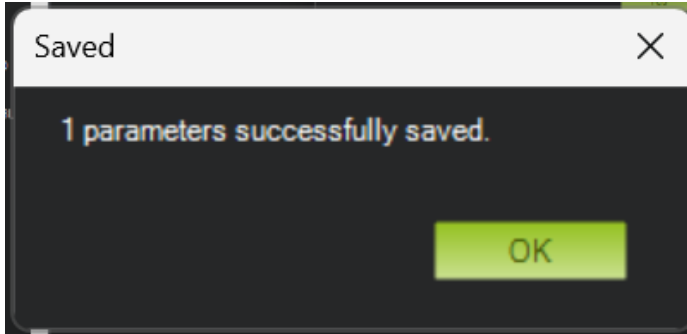


Figure 69: Confirmation of Parameter Change.

Using the same process, please also rewrite all the variables shown in table 70 on the right. These parameters are all related to the Extended Kalman Filter (EKF) and GPS configuration, which is important for the drone to be able to use the Vicon data as a source of position estimation.

Once all the parameters are saved, head towards *SETUP, Flight Modes*, you should navigate to a screen as shown in figure 71.

Parameter	Value
AHRS_EKF_TYPE	3
SERIAL1_BAUD	921
SERIAL1_OPTIONS	2
EK3_SRC1_POSXY	3
EK3_SRC1_POSZ	3
EK3_SRC1_VELXY	3
EK3_SRC1_VELZ	3
EK3_SRC1_YAW	2
EK3_SRC2_...	0
EK3_SRC3_...	0
EK3_MAG_CAL	5
EK3_POSNE_M_NSE	0.1
GPS1_TYPE	14
GPS1_DELAY_MS	50
COMPASS_USE	0
COMPASS_USE2	0
COMPASS_USE3	0

Figure 70: Configured ArduPilot Parameters

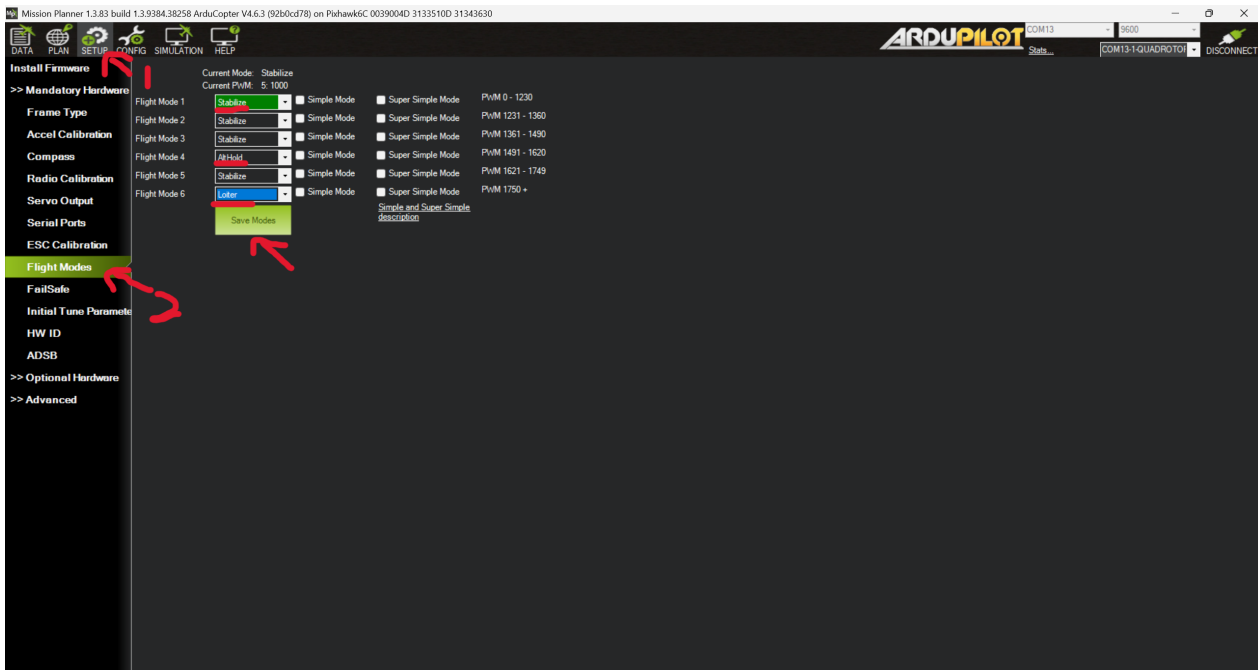


Figure 71: Setting up Flight Modes.

Change the Flight Modes 1, 4, 6 into *Stablize*, *Alt Hold*, and *Loiter*; Press *Save Modes* to save your edits afterwards. Here I am using radio channel 5 on my transmitter, which corresponds to a 3 way switch as my safety switch between Vicon indepedent flight mode and Vicon dependant flight mode.

5.3 Mavproxy Telemetry

To test if the telemetry communication link is working, return to the GCS and type in the mavproxy command (Remeber to select the ESP8266 Wifi Network). If successful, telemetry should start showing up like figure 72.

```
mavproxy.py --master :14554[Replace with your port number here] --console --map
```

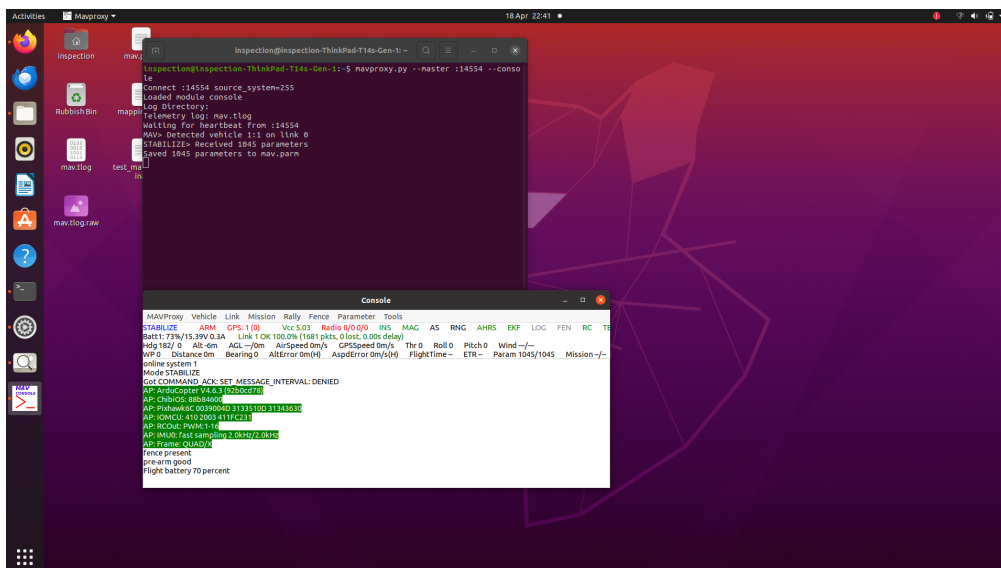


Figure 72: Telemetry Data Showing up in Mavproxy Console.

Recalling the Host Port number from before in figure 61, make sure to replace 14554 with the number of your port.

5.4 IR Markers

Now your GCS connetions are all done, we are moving towards the final steps before actually flying the drone with VICON! Mount the IR markers with hot glue on location of your drones that is easily seen by the cameras. Have at least 4+ per drone and place them in an assymptical arragement, an example is shown in figure 73 below:



Figure 73: Example of Drone IR Markers layout.

If you are flying more than 1 drone, your assortment of IR markers must be geometrically different, otherwise the Vicon System may mistake the wrong object to track. (This has happened to me before and caused a crash.)

5.5 Final Intergration

Finally, let's test the overall system. Plug in your GCS to the Vicon ethernet port as seen in figure 51, connect the GCS laptop to the ESP8266's wifi, and place your drone inside the flight arena.

Setup the Vicon system by following section 3.1 above. This includes turning on the cameras, creating a mask, calibrating cameras, setting the origin, and define your now ready to go drone as a new object in section 3.7.

One last thing, we must let pyvicon know the name of the object so it knows which object to track. To do this, go into the to GCS's venv, and navigate to `/miniconda3/envs/VICON/lib/python3.1/site-packages/MAVProxy` and open the file named `mavproxy_vicon.py`, as shown in figure 74:

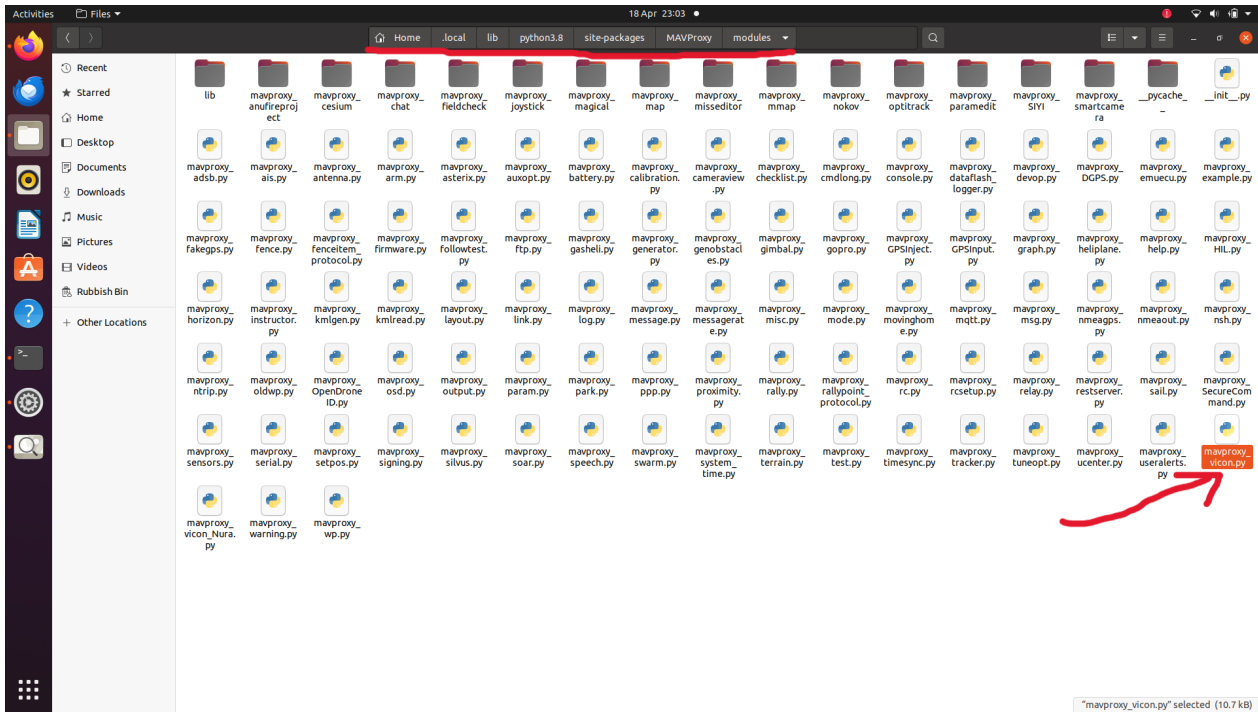


Figure 74: Location of `mavproxy_vicon.py`.

Align your focus on line 29 to 36, and change `(origin_lat)(origin_lon)(origin_alt)` to 0 (Otherwise your drone will fly towards Australia, where the origin is defaulted at the glorious Canberra Model Aircraft Club Flying Field), `(object_name)` as your object name. (Here, `MATT_INSPECTOR` is the name of my drone as defined within the Vicon Host PC). Press *Save* on top right to save, as shown below in figure 75.

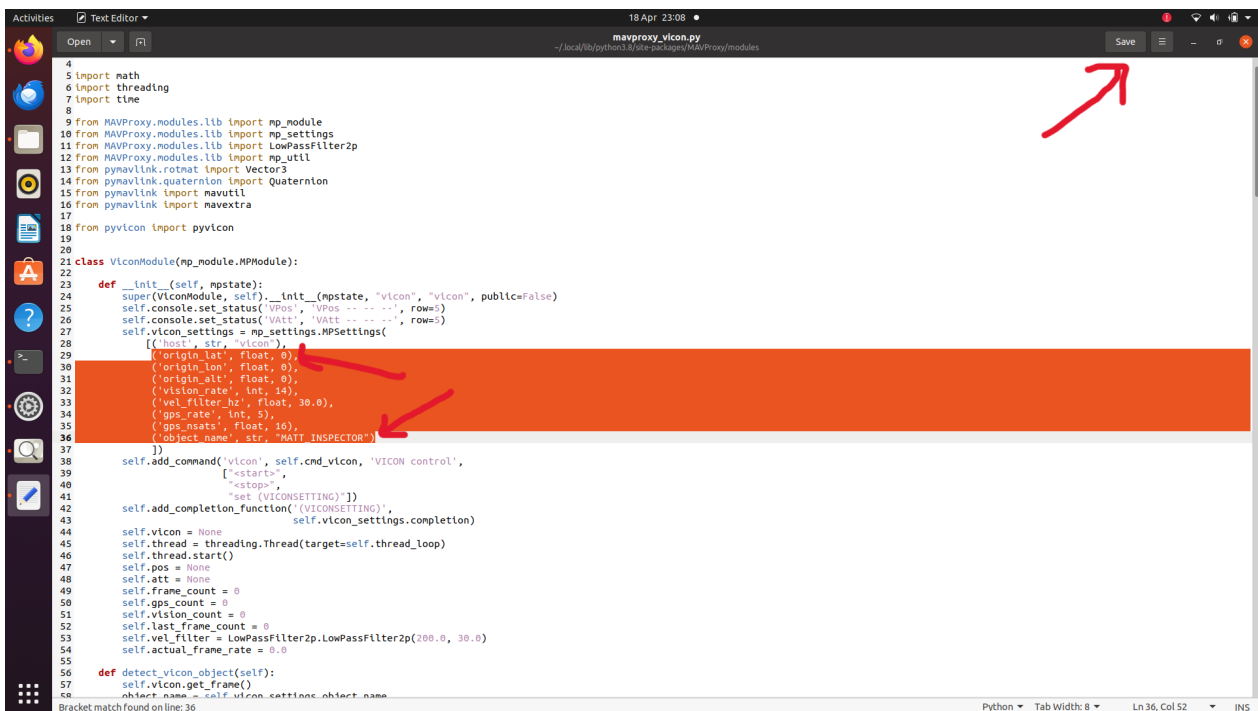


Figure 75: Changing the origin and object name inside `mavproxy_vicon.py`.

Once completed, activate MAVproxy console, once again, this is done by opening terminal and typing:

```
mavproxy.py --master :14554 --console --map
```

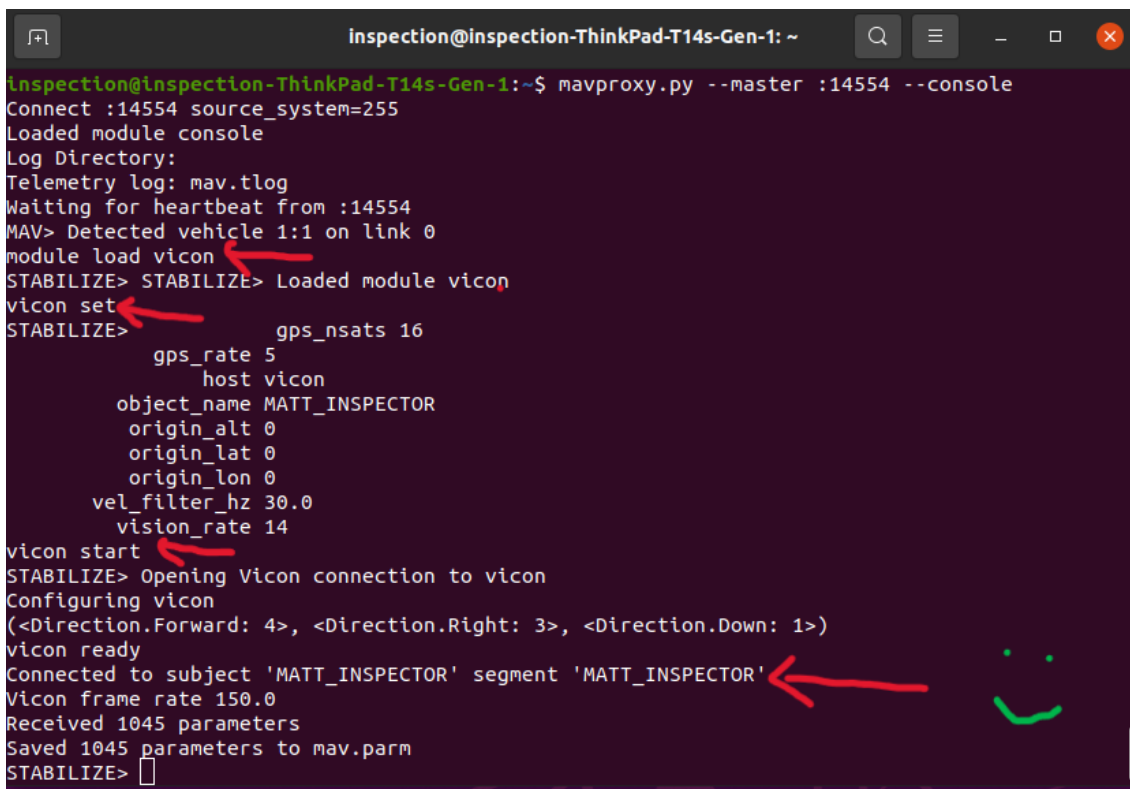
Within mavproxy, type the following to load the Vicon Module:

```
module load vicon
```

```
vicon set
```

```
vicon start
```

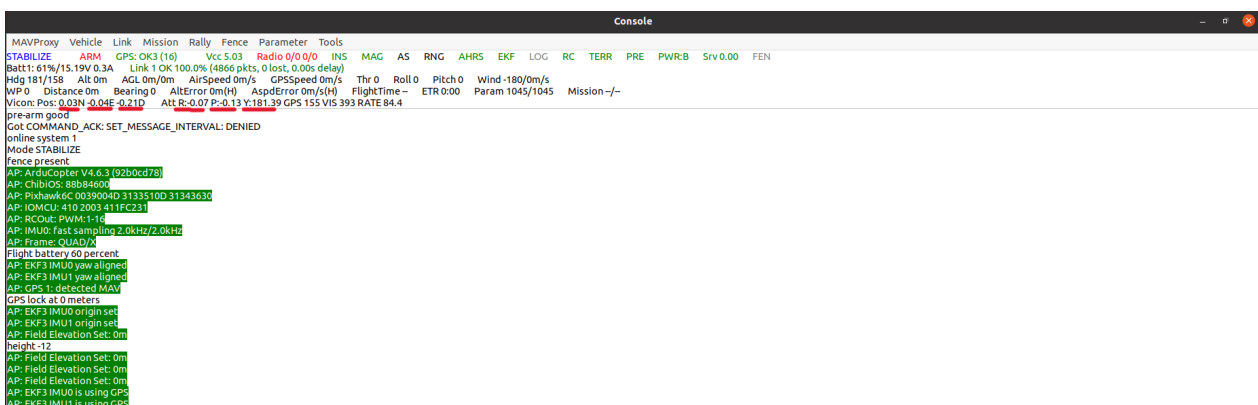
You should have the same results as figure 76. A good way to verify if the object is being tracked correctly within the scirp is when entering *vicon start*, it will only return the statement *Connected to subject "MATT_INSPECTOR" segment "MATT_INSPECTOR"* when it is working correctly.



```
inspection@inspection-ThinkPad-T14s-Gen-1: ~$ mavproxy.py --master :14554 --console
Connect :14554 source_system=255
Loaded module console
Log Directory:
Telemetry log: mav.tlog
Waiting for heartbeat from :14554
MAV> Detected vehicle 1:1 on link 0
module load vicon
STABILIZE> STABILIZE> Loaded module vicon
vicon set
STABILIZE>          gps_nsats 16
                gps_rate 5
                host vicon
                object_name MATT_INSPECTOR
                origin_alt 0
                origin_lat 0
                origin_lon 0
                vel_filter_hz 30.0
                vision_rate 14
vicon start
STABILIZE> Opening Vicon connection to vicon
Configuring vicon
(<Direction.Forward: 4>, <Direction.Right: 3>, <Direction.Down: 1>)
vicon ready
Connected to subject 'MATT_INSPECTOR' segment 'MATT_INSPECTOR'
Vicon frame rate 150.0
Received 1045 parameters
Saved 1045 parameters to mav.parm
STABILIZE> 
```

Figure 76: Expected outputs of activating mavproxy.py

If everything is working well, within the console, you should see the the NED (North East Down) and Euler angles (Roll Pitch Yaw) of your drone - Yippee!



```
MAVProxy Vehicle Link Mission Rally Fence Parameter Tools
STABILIZE ARM GPS: OK3 (16) Vcc 5.03 Radio 0/0/0/0 INS MAG AS RNG AHRS EKF LOG RC TERR PRE PWR/B Srv 0.00 FEN
Batt1: 61%/15.19V 0.3A Link 1 OK 100.0% (4866 pkts, 0 lost, 0.00s delay)
Hdg 181/158 Alt 0m AGL 0m/0m AirSpeed 0m/s GPSSpeed 0m/s Thr 0 Roll 0 Pitch 0 Wind -180/0m/s
WP 0 Distance 0m Bearing 0 AltError 0m(h) AspError 0m/s(h) FlightTime - ETR 0:00 Param 1045/1045 Mission -/-
Vicon Pos: 0.03N -0.04E -0.21D Alt R: 0.07 P: -0.13 Y: 181.39 GPS 155 VIS 393 RATE 84.4
pre-arm good
Got COMMAND_ACK: SET_MESSAGE_INTERVAL: DENIED
online system 1
Mode STABILIZE
Fence present
AP: ArduCopter V4.6.3 (92b0cd78)
AP: ChibiOS: 88b84600
AP: PixhawkK&C 0030004D 3133510D 31343630
AP: iOMCU: 410 2003 411FC231
AP: RCOut: PWM: 1-16
AP: IMU0: fast sampling 2.0kHz/2.0kHz
AP: Frame: QUAD04
Flight battery 60 percent
AP: EKF3 IMU0 yaw aligned
AP: EKF3 IMU1 yaw aligned
AP: GPS is detected MAV
GPS lock at 0 meters
AP: EKF3 IMU0 origin set
AP: EKF3 IMU1 origin set
AP: Field Elevation Set: 0m
height -12
AP: Field Elevation Set: 0m
AP: Field Elevation Set: 0m
AP: Field Elevation Set: 0m
AP: EKF3 IMU0 is using GPS
AP: EKF3 IMU1 is using GPS
```

Figure 77: Correct console information.

When console mentions height is at -12 while your drone is on the ground, this is referring a 12cm offset from the bottom to the center of the drone. The position of the drone is defined as the geometric center of the assortment of IR markers you placed.

5.6 Finishing

From this section, we have managed to merge our understanding with the operation Vicon System, the GCS system you developed, and the drones that we prepared. We now have a working architecture that feeds Vicon postional data to the drone via the GCS. This is splendid and we are now ready for test flights! Congratulations and well done on makign it this far! The next section will be about your first flight, and how to use the Vicon data to fly the drone.

6 Concluding Remark

I hope this guide has been helpful. No matter where or what you plan to do with this, I wish the best in your projects. My hope is that maybe one day someone with enough time will create a much more streamline system that can make this whole system plug and play, but until then, you are stuck with me and my words!

References

- [1] Mathieu Garon. pyvicon: Python interface for vicon datastream sdk.
<https://github.com/MathGaron/pyvicon>, 2024. Accessed: 2026-03-26.
- [2] Adnan Aqeel. Esp8266 pinout, datasheet, features and applications.
<https://www.theengineeringprojects.com/2018/08/esp8266-pinout-datasheet-features-applications.html>, August 2018. Accessed: 30 March 2026.
- [3] ArduPilot Documentation Team. Mission planner installation guide.
<https://ardupilot.org/planner/docs/mission-planner-installation.html>, n.d.
Accessed: 2026-04-18.