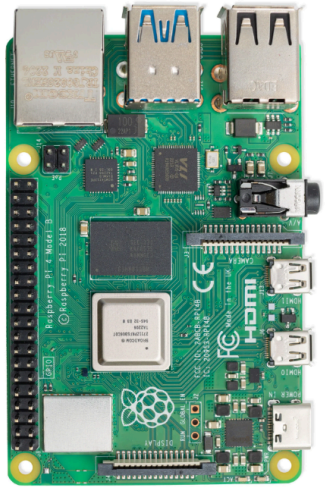
Jacurracy Scope Build Instructions

Initial Release 9/20/2023

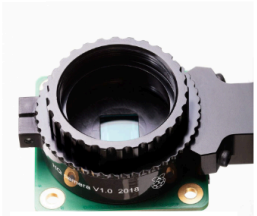
Hello! Thank you for downloading the build guide and software for the Jaccuracy Scope! I hope you are familiar or seeking to become familiar with the art of long range ballistic target shooting, I think you will have a fun time with this build! Boiled down to a simple sense, we will be utilizing the following subsystems to build our baseline scope:



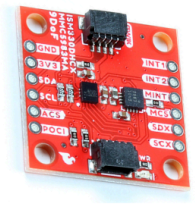
Raspberry Pi 4 (or Zero 2W)

Arducam 8-50mm C-mount Lens (OR your choice of lens)

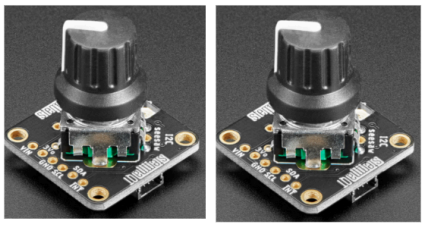
Pi Camera Ribbon Cable



Pi Camera HQ (IMX477)



SparkFun 9DoF IMU board

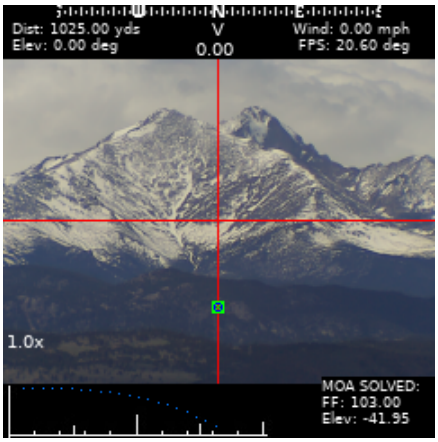


Adafruit I2C Stemma QT Rotary Encoder x 2

TFT 1.3" LCD

The Parts list for these items and their alternatives can be found in the document package! The fun part of this build is that these sub components can be changed with others if you choose, you will just need to modify the main coding files to accommodate changes.

The 9DoF sensor and the Stemma QT Rotary encoders were chosen for their ease of integration, they both utilize the Qwiic connection ports for a daisy chain on the I2C bus of the raspberry pi, meaning they have little to no soldering required to build your unit! The only soldering you will need to do for these components is on the back of the encoder units to choose the encoder I2C address which is very simple to do with a blob of solder. We will dive into the specifics in the later section for the encoder itself.



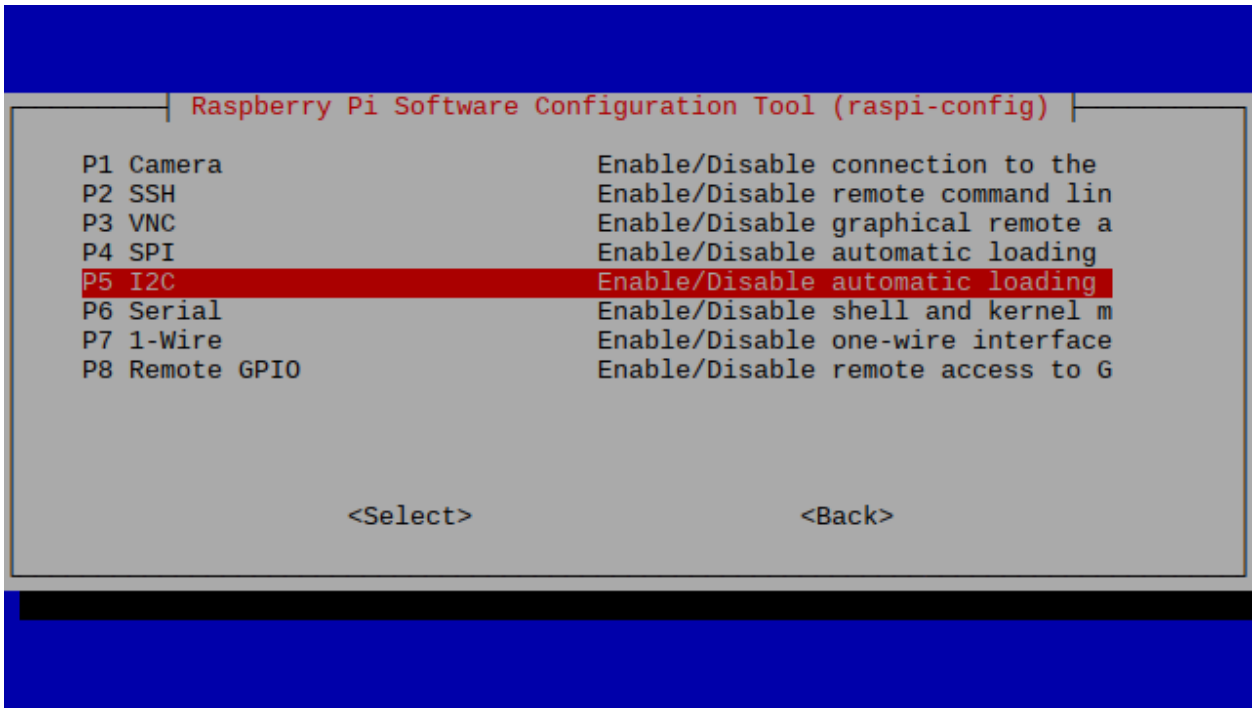
The primary ballistics software was originally built by myself with converted matlab coding after learning to do it the proper way, my results seemed to generate very close results but in order to match perfect with most online ballistic calculators I opted to include a modified compiled C library for that ballistics that runs faster (YAY!) and was written originally by David Yates. You can search for his original source code. My algorithm is still deployed during the Lobster mode for fun, both can be seen in the Ballistics Threader python files.

Starting with the Raspberry Pi

The Raspberry Pi single board computer will need to be set up in the following steps from scratch. There are many Raspberry Pi tutorials out there that do this but I will try to summarize them here. It is very important that you set up the imaging to include your home Wifi SSID and Passcode so you can remotely edit and view the files for installation. It makes it easy, trust me. Alternatively you can plug a monitor, mouse and keyboard into the Pi to work directly on the desktop interface:

1. Download the the Raspberry Pi imager software https://www.raspberrypi.com/software/
2. Follow the following tutorial to get your Pi up and running and connected to your Wifi network. https://www.raspberrypi.com/tutorials/how-to-set-up-raspberry-pi/
3. We want to install the 64bit Rasbian Bullseye debian (NOT BOOKWORM).

When your Pi is up and running, let's enable some GPIO interfaces to allow us to connect to the devices we intend to drive with the header pins. Use the following command to enter the configuration: "*sudo raspi-config*". Select the interface options and enable the Camera, VNC (optional), SPI, I2C and Serial options. You will need to also Expand the Filesystem from this menu to utilize more card space. Also choose the option to expand the filesystem so we have full access to the SD card memory!! When finished, press "Yes" to reboot the Pi.



You will need to know the IP address of your pi if you want to tap in to it externally, like with VNC or SSH through Putty to make your life easier. I recommend finding out your pi's local IP address for the install process, although you can always do the classic keyboard/mouse/hdmi setup temporarily so long as you have access to the internet to pull the software repository.

Automated Installation of the needed Software Packages

I did my best to write up a bash installation script included in the Github repo. Bear in mind this is my first ever Bash script so I've really got my fingers crossed that the speed is adequate… Install on a fresh pi by doing the following:

1. Keep your dependencies up to date using the following two commands:
   sudo apt update
   sudo apt upgrade

2. *cd ..* - *just makes sure we are at the home directory… you should already be there anyways….*
3. *sudo mkdir /home/pi/share* - Creates a Share Folder on the filesystem for easy sharing

4. cd share - move into the new Share folder we have created.

5. *sudo git clone https://github.com/ouch3994/JaccuracyScope* - Downloads (clones) the repository from Github that includes all the files you will need

6. cd JaccuracyScope - move into the downloaded folder

7. sudo chmod +x jaccuracyinstall - Elevates permissions of the installer file

8. sudo ./jaccuracyinstall Starts the installation process… godspeed.
9. Hopefully this works without errors! If it hangs up on the installation of any of the packages you can always see the manual install section below for installing the needed items on the side.

10. sudo reboot - Please reboot the Pi using this command at this time…

11. The portion of this that takes the longest is the installation of OpenCV (needed to record internal videoclips and future image processing capability…). You will want a fast internet connection for this so I recommend doing this separate from the bash installer script. Install this using the following command:

    pip install opencv-python-headless

    Try not to interrupt this installation as it can be very time consuming.

Manual Installation of the needed Software Packages

In case the bash installer script doesn't work… you'll need the following items installed to your pi. Some of these come included in the Pi image but its best to run through these one at a time and check that they are installed in some capacity.I ended up using a lot of software packages here for referencing our devices and internal libraries of code for Python 3 to use. PIP is a package installer for python that should be installed already on the Pi when you install the firmware to the SD card. We will be using PIP for alot of these installs. Below is a breakdown of the packages you need and a description of what each of them will do.

12. Keep your dependencies up to date using the following two commands:
    sudo apt update
    sudo apt upgrade

13. *sudo mkdir /home/pi/share* - Creates a Share Folder on the filesystem for easy sharing

14. *sudo apt-get install samba samba-common-bin* - installs SAMBA file share server
    a. When this is completed, lets configure the SAMBA file share for easy use
       sudo nano /etc/samba/smb.conf
    b. When in the text Editor, add the following code:

```
[ScopeShare]
path = /home/pi/share
writeable=Yes
create mask=0777
directory mask=0777
public=Yes
force user = pi
force group = pi
```

    c. Use the following commands to final configure the share folder
       sudo smbpasswd -a pi
       sudo systemctl restart smbd

    At this point, you should be able to access the file system of share folder on your computer by entering the Pi's IP address into the file explorer bar. If you are working locally on the Pi and need to find your IP, use the *ipconfig* command.

15. The Pi Camera HQ will need the following library of python code to connect.
    sudo apt install -y python3-picamera2

16. The ST7789 240x240 Display Unit will require the following package:
    sudo pip install st7789

17. Our code uses the Pillow image creation tool for Python to generate each of the frames.
    sudo pip install pillow

18. The Sparkfun 9DoF IMU (ISM330DHCX, MMC5983MA) uses the following library
    sudo pip install adafruit-circuitpython-lsm6ds
19. An option in the code when driving the Pi from VNC or a display is to show the live feed on the screen, this is done using the imagetk library and can be used for troubleshooting the code in the future, all wirelessly when VNC'd into the Pi.
    sudo apt-get install python3-pil.imagetk

20. Open CV is needed for the video recording features on the Pi, we will be using the headless install package to get these features.
    pip install opencv-python-headless

21. Ctypes is a package to allow us to use precompile C coding for the ballistics calculator to run fast and efficiently. We need this!
    pip install ctypes

22. (Optional) the Thermal Receipt Printer:
    sudo pip3 install adafruit-circuitpython-thermal-printer

Once you get through all of these installations we will need to take a closer look at the SPI protocol (the next section). This will take a bit of trickery in the text files of your pi to achieve the best frame rate for the scope!

The Spidev and ST7789 Hack

My research on the SPI protocol has led to me an interesting hack that is needed to get the maximum frame rate for our scope system. It seems that the SPI packet sending buffer is limited to a size of 4096 bytes. We can speed this up by 'hacking' the base code for the Spidev protocol and the ST7789 library that we have already installed. What I have done is changed all the accounts of 4096 in these files to 65536 and created a new set of functions to allow for the ST7789 display to always be accepting a full 240x240 image each time, rather than the slow default way of first telling the display the image size prior to sending. This helps global frame rate increase dramatically. (I found this works great on the Pi 4b so far, the Pi 5 was struggling to keep up with workarounds to the GPIO library stuff…. Its going to be a while to fix).

The file __init__.py can be overwritten with out specially modified one using the bash installer script. (the cp command will do this). This mod on the st7789 library will always consider a 240x240 pixel output and the modified SPI buffer of 65536 size so it can slam the display fast with pixel data.

Verify your Remote folders work

If the SAMBA share portion of the installer worked properly, you should have open access to the file system folder (anything under the “share” folder you created). Simply go to a windows file explorer tab and enter the IP address into the destination bar. On my local network, the IP is 192.168.1.56, so i’ll enter it as such: \\192.168.1.56

Now you can browse the code over wifi and make edits on your computer, or you can download any saved video and image files that were created during use! I have done this too over a cell phone with the iOS files explorer app, and with android.

Test the Software

Now that we have our files moved over and all dependent packages installed, it's time we give the software a test run from the terminal. We will need to make sure all the hardware (Camera, IMU, Encoders, ST7789 display) are attached in the correct method that we specified earlier to avoid hitting errors.

From a fresh terminal window, lets navigate to the master code file by doing the following command: *cd /share/JaccuracyScope/Display.* When you are ready to test your software execute the following command: *sudo python3 zDisptest.py*

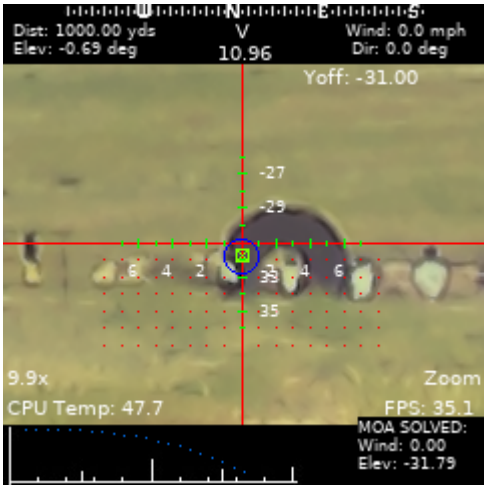
You should see a thermal image test pop up on screen and display the FPS achieved in the command window! Exciting that displays can be driven this fast with a single lane SPI line. To stop this test, use CNTRL + C on the terminal window to stop the frame rate test. You can also make sure that video libraries work by using the videoplay.py file.

Next, we will test out the full suite of the rifle scope considering everything is wired up and connected to the GPIO pins properly. Use the following command to run the software:
command: *sudo python3 Disptest46*
To stop this test at any point, use CNTRL + C on the terminal window a few times.

We should see the Terminal window connecting to the camera and beginning the iteration loops. The ST7789 should light up and start displaying the main program. To view the performance of the PI as the software is going, open a separate opened terminal window and enter *htop* to view a live feed of the CPU and memory usage. This is a good way to keep an eye on the program if you make future custom edits.

The Display has a few helpful items to monitor while testing the scope out, such as the FPS and the CPU temperature (Celcius) displayed in the corners of the image. Its a good idea to keep an eye on these if you do not have a good heat sink or fan installed on the Pi’s main processor. We

are using the Pi to a great potential here so heat management is important so we don’t burn our processor off the board.



The Temperature and FPS seen above

Verify that each of the input buttons and potentiometers are functioning properly, and that the you are able to navigate to the menu window.

To stop this test at any point, use CNTRL + C on the terminal window a few times.

Final Step, Crontab

If the test run of the software went well, we can now instruct the Pi to run the code automatically at boot up! We do this using the crontab method, essentially a text file to automatically start running whatever we tell it to during the boot up sequence of the Raspberry Pi. Let's tap into the Crontab file by running the following command in a fresh terminal:

sudo crontab -e

It may ask for the Editor you want to use, chose option 1…

We are going to want to navigate to the bottom of the text file to insert the following line:

@reboot cd /home/pi/share/Display && python3 disptest45_Laser2_smalldisp.py &

Press CTRL+ S to save the file, then use CTRL + X to exit the window. If all went well, we can now reboot the Pi using *sudo reboot* and expect the scope software to run automatically when we power up!

This completes the scope software installation sequence! In the next section you will see a breakdown of my code and how it utilizes the four cores of the Pi's processor to run individual tasking to maximize speeds.

SOFTWARE USAGE GUIDE

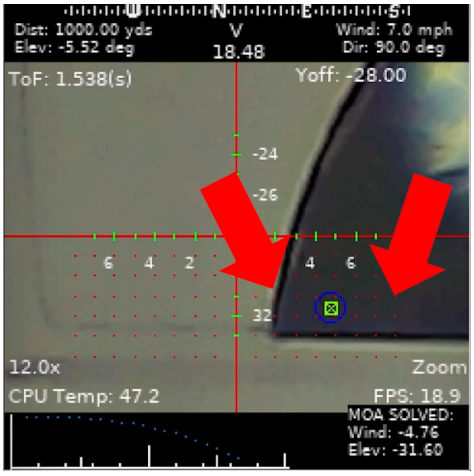
The Main Screen

The Scope body is free for you to build for a comfortable layout, where the software will apply different functions to the rotary encoder dials. I call my top mounted encoder the “Encoder 1”. When you are in the primary screen window with the crosshairs (Screen 0), the Encoder 1 can manually adjust the target distance found in the top left of the image. This can only be done while the scope is digitally zoomed out all the way to the displayed 1.0x Zoom. If you are unsure if you are zoomed out all the way, either rotate the Encoder 2 while ‘Zoom’ mode is displayed on the right side or push the Encoder 1 down twice to snap in and snap out to 1.0x Zoom. Adjust the Target Distance to the desired length by rotating Encoder 1.

You will notice as you adjust the target distance in and out that the green box location changes with distance. This is the Impact Indicator which shows the user where in visual space the fired bullet will be located at the prescribed distance. This location changes algorithmically to match the image feed from the camera and allow the user to just hover the impact box over what they want to hit. For a 1,000 yard target, dial in 1,000 yards for the set distance, check your windage settings in the menu to match your conditions, and hover the green impact box over the steel target to hit it. This is a great easy visual indicator for the elevation and windage adjustment that normal long distance shooters need to align manually by eye with the MOA hash marks of a normal rifle scope. Be sure that you have mechanically aligned your scope to a set zeroing and that your bullet statistics are entered correctly first though! The solution data is located at the bottom right of the image for if you need to communicate your shot solution when used as a spotting scope.

Watching this impact indicator move with distance is very educational for a new shooter to learn the shot path of a bullet over long distances. You will find as well the effect of an uphill or downhill target will be incorporated to the solution as well which is equally educational. This

ballistics computation takes real time information from the 9Dof IMU to compute these values, so calibration of your sensors is a must.



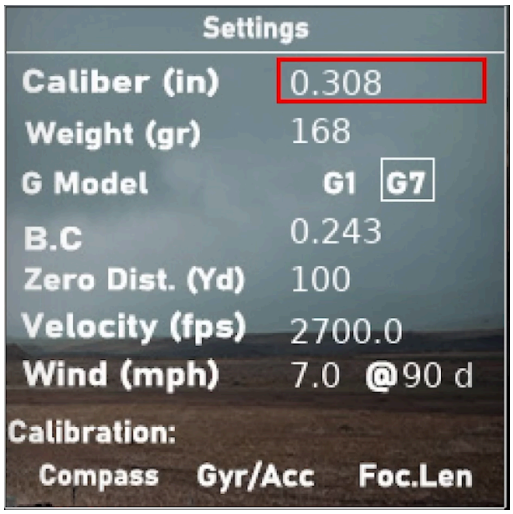
A 12.0x Digital Zoom

Aiming the spotting scope upward far enough will display a fun bonus mode meant for educational purposes only. This is the ‘Lobster Mode’ which gives a rough approximation for how high the projectile will reach before returning back to the initial height. This is a very very rough approximation to showcase the customizable features of the python scripting and I am not liable for your stupidity if you fire a firearm in the air at an irresponsible angle. Do not do this.

Lobster Mode Easter Egg

The Setting Menu

Long range shooters are very well familiar with the settings listed in the Settings menu's main screen. This is where your bullet of choice's flight characteristics are plugged in to ensure an accurate flight computation. User's should familiarize themselves with the following settings to properly match their rifle and projectile combination. Damn it, I forgot the scope height in the menu settings… see the last page of this doc to see what to do for that adjustment in the code.



The Menu can be navigated using the Encoder 1 dial to move the highlighted box. Encoder 2 will adjust the setting that the red indicator box is hovering over and correspond to the listed units next to the setting. All stats are saved to a configuration file upon leaving the Menu by pressing the designated Menu Button on scope and are reloaded automatically during boot up.

The Wind input parameter is marked with Miles Per Hour units and the directional heading in degrees, so for a wind blowing in the pure North direction insert 0 degrees. For a East direction insert 90 degrees, for a South wind enter 180 degrees and for a Western wind direction enter 270 degrees. These directions are not relative to the shooter rather globally relative as you would measure on a compass.

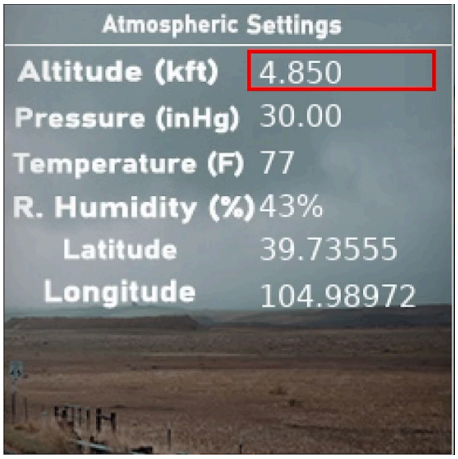
The bottom three menu features are for the sensor calibration features to tune in the static and dynamic values of the Accelerometer, Gyroscope and Compass separately from the 9DoF sensor. The Accelerometer and the Gyroscope are calibrated at the same time by prompting the user to leave the scope completely stationary and rested in a perfectly level orientation. This will save these calibration values internally for a more accurate elevation reading during shot calculation. The Compass is calibrated in a separate way and the steps to complete this are displayed on the scope screen. This should be performed after mounting the scope to a tripod baseplate (for spotter scope configuration) or after mounting to any other large metal surface so the hard metal magnetic effects of that surface are measured and taken into account for the calibration. This will require the user to point the scope in various orientations as displayed on the screen.

The Focal Length Calibration will be necessary to perfectly match the on screen crosshairs and angle indicator marks to a true measurement. The two encoders can be used in

this screen to increase the zoom level (Encoder 2), and to adjust the internally saved Focal Length (Encoder 1). If you put a 50mm lens on the front of your image sensor you can expect to find a value near 50 mm works well but may need some fine tuning due to manufacturing tolerances. For example, I use a 50mm lens coupled with a 1.5x teleconverter. I should expect to get (50 * 1.5) = 75 mm but after calibrating to a 1" grid at 100 yards, I found that I needed around 77.25 mm to perfectly match the real life 1" grid.

The Focal Length Calibration Screen

Page 2 of the menu screen can be accessed by scrolling the Encoder 1 past the Focal Length setting and homes the Atmospheric Conditions inputs. These will need to be input manually for an accurate ballistics calculation. In the future I will create a third page that includes the Atmospheric Conditions at the time of Zeroing the rifle, this is not included at the moment but will improve accuracy of the shot solution and the impact indicator. There are place holder inputs for the Latitude and Longitude for future updates to include the Coriolis effect. Another parameter neglected for now is the rifling parameter for accounting for Spin Drift. I will work to include these in the future.

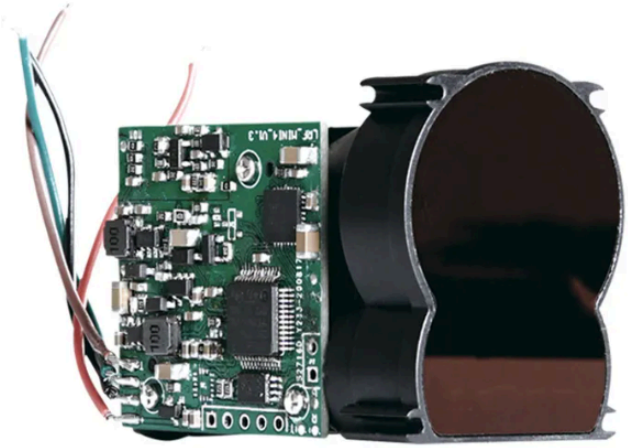


Page 2 of the Menu

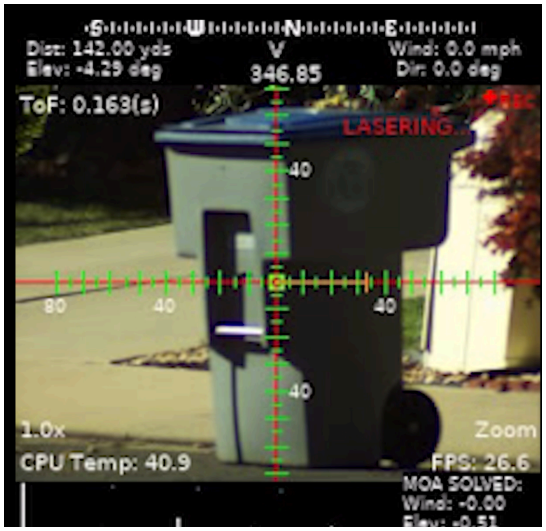
(Optional) The Laser Range Finder

If you find a good mechanical solution to mount the Laser Range Finder up with the scope, you can use Button 1 to command the scope to take a single distance reading. This will display “LASERING” in the top right corner of the image and will automatically apply the distance to the Target Distance parameter (rounded). You may need to run some visual tests to ensure you have a well aligned laser parallel to your scope’s lens and make mechanical adjustments if necessary. Ensure the laser is on and connected to the listed Serial Port connection pins that are listed in the section about the Laser Range Finder. If you have kept this device powered separately through it’s included CR2 Battery container, ensure that it is powered up first before taking a reading. The minimum distance that the MINI4-1000m module can measure is around 4.5 yards, anything closer than this will not register a reading. https://www.aliexpress.us/item/3256801536029580.html?spm=a2g0o.order_list.order_list_main.5.21ef1802a6qToL&gatewayAdapt=glo2usa

Wiring the Laser to the UART is a easy 3 wire connection, with this kit.



The Mini4-1000m Laser Range Finder Module



The Lasering Indicator in Top Right (red)

Python Code Structure

You may be curious to tinker with the python code itself and you will need to know the basic architecture of this coding. At a top level, there are four 'threads' being run where the split off 3 are called from the main script py file. Their responsibilities can be viewed as following:

Main Script	Camera Threader	Ballistics Thread	Sensor Threader
Uses the PIL Library to create a blank 240x240 frame. Reads the Sensor Threader IMU input Inputs IMU data to Ballistic Threader Reads output of Ballistic Threader for shot data Reads Camera Threader to paste 240x240 image to the frame Pastes the other telemetry and plots on the frame Draws the Crosshairs and impact indicator on frame Commands the ST7789 to display the generated Frame Listens for button pushed event callback scripts for Menu, Lasering, Recording/Screenshot	Opens a scaled image stream from the IMX477 Image Sensor. (Uses the picamera2 library to do this) Applies a sharpening setting and auto exposure setting command to the IMX477 sensor Dictates commanding of the image sensor for zoom and scaler crop commands Measures FPS performance of camera in background.	Takes inputs from RAM settings for shot data and uses the compiled C function to compute the ballistic ODE solvling. Creates the 2D plot of the shot trajectory Includes my Ballistic Model and GNU ballistic model. GNU on by default for better accuracy results. Mine will be improved in future for shots farther than 1,000 yards Exports data results to Main Script for display Commands the Printing Feature for shot table printing if installed. (Activated by button 1 in place of lasering). Commands/Reads the Laser Range finder if installed (Activated by button 1 in place of lasering).	Reads the IMU sensors and compiles a Moving Average filter for smoothing. Reads the two Rotarty Encoder Boards for inputs Allows for Main Scirpt to access all values for functional use. Commands the I2C bus for these sensor inputs over the daisy chained QWICC wire connections. Can be throttled to improve CPU usage and FPS.

Functions of the Script Threads

With these four cores working in unison to write and read information from memory, the scripts can be edited for optimization purposes and power management.

Future Updates Ideas:

There is so much potential for the Pi to handle more modes and operations, and I am always thinking of ways to expand this project. A few ideas are listed below that may find their way to future versions if I find the time, or if python-savvy contributors can show off their coding skills.

Future Possible Ideas	
Low Power Mode	A low power mode entered when the scope is pointed downwards
Battery Indicator	When a solid power solution is found, a battery indicator
GPS	GPS capabilities
Internal Humidity Sensor	Detect humidity with attached peripheral
Wireless Video Feed	Live Stream Feed to another device (like a smart phone)
Attachable Chronograph	Hall effect sensors to mount for shot data collect and BC computing on the go
Software Adjustable Zero	Adjust Zero reference point with digital crosshairs

Parts Guide:

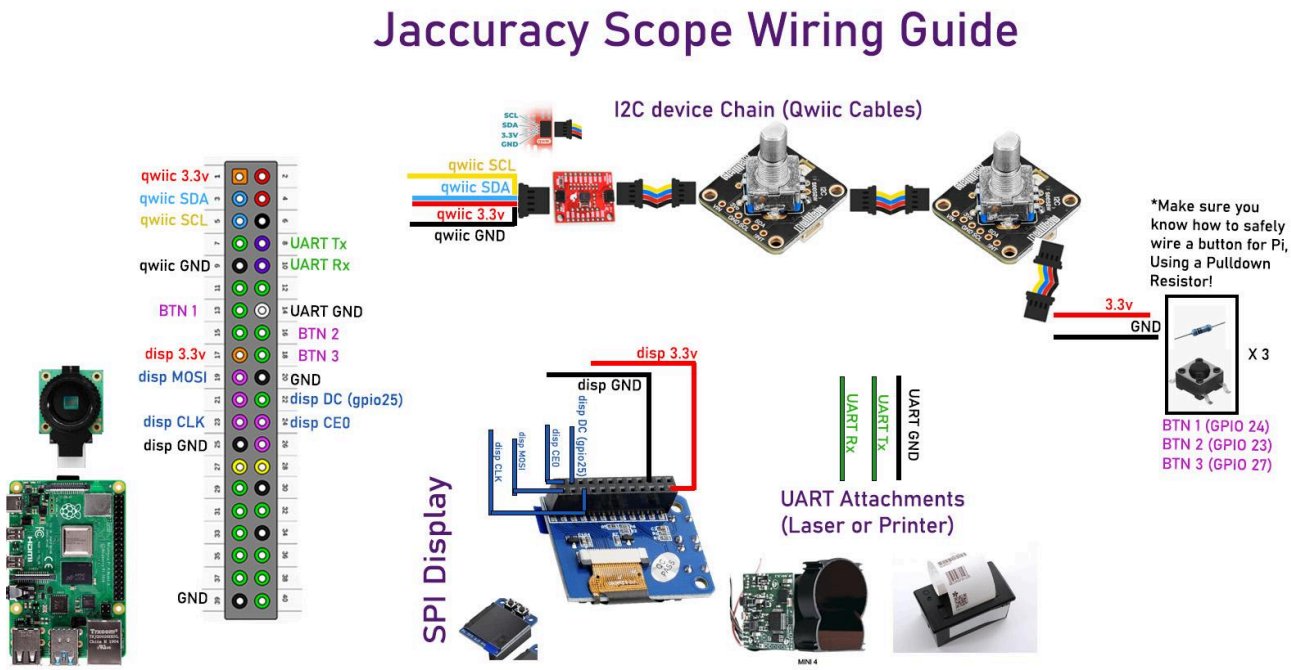
Needed Parts:

Pi 4	$60.00	https://a.co/d/aChKUkU
IMX477 Img Sensor	$59.00	https://www.amazon.com/Arducam-Raspberry-Camera-Sensitivity-Adapter/dp/B09YHN5DBY/ref=sr_1_4?crid=1JFV4AIF9SXK&keywords=pi%2Bhq%2Bcamera&qid=1694228553&s=electronics&sprefix=pi%2Bhq%2Bcamera%2Celectronics%2C117&sr=1-4&ufe=app_do%3Aamzn1.fos.006c50ae-5d4c-4777-9bc0-4513d670b6bc&th=1
50mm Lens	$50.00	https://a.co/d/dRSHxOI
9 dof sensor	$39.00	https://www.sparkfun.com/products/19895
240x240 TFT screen	$23.99	https://www.amazon.com/PiTFT-Display-240x240-Raspberry-ST7789/dp/B08F9XTKGK Alternative, but different wiring… https://www.adafruit.com/product/4313 or any240x240 TFT (SPI) alternative
Heat Sink with Fan	$15.00	https://www.amazon.com/Raspberry-Model-Aluminum-Cooling-Metal/dp/B07VQLBSNC/ref=sr_1_25?crid=2ZS57UEIEA3PP&keywords=pi%2B4%2Bfan%2Bheatsink&qid=1694225642&s=industrial&sprefix=pi%2B4%2Bfan%2Bheatsink%2Cindustrial%2C117&sr=1-25&th=1
Rotary Boards (x 2)	$10.00	https://www.adafruit.com/product/4991?gclid=CjwKCAjw-b-kBhB-EiwA4fvKrGGaGpFwRwUeAEPJiBHzoUaBYfY-vvQu-TTIGk6DpPNtqjLeOnZmTBoC3B8QAvD_BwE
Electronics Total	**$266.99**	This was cheaper two years ago… inflation

Optional Gadgets:
-Thermal Printer: https://www.adafruit.com/product/597
-Laser Range Finder: https://tinyurl.com/mrpsfrwt
(The Printer and Laser are Either-Or right now, since the hardware only has one available serial port for device attaching to the GPIO pins…)

EyePiece Helpers:
Rubber Viewer Eyepiece: https://a.co/d/fVEvzOL
Intermediate lens to help view screen: https://a.co/d/gmBvpHN (requires a main lens in front of it and a 3d print design to hold this in place).

Wiring Guide:



You will need to wire the device up in this way, or if you are Pi-savvy enough, edit the script to work with your changes (like references to the buttons). The buttons will require you to understand how to create a pulldown/pullup resistor configuration to be safe for the raspberry pi hardware to use. I wire my mine with a Pull Up resistor as the Python file configures GPIO 24, 23 and 27 to be. My software uses only 3 buttons, with other input coming straight from the Rotary Encoder knobs and push buttons on there for settings.

I will do a video explaining all this for clarification, if I find the time.

END

At the time of writing this, I have hope for the community of long range shooting enthusiasts to develop this further as my time to dedicate here has run short. I hope for the following features in the evolution of this project:

-Inclusion of Coriolis and Eotvos Effects with Coordinate input in the settings.
-A build using an HDMI small screen for compact builds, Raspberry Pi 5 capability, and support faster frame rate and features.
- A Low Power mode where the ballistic calculations and screen will turn to a slow frame rate to save power consumption wildly.
- A Separate Menu option for Atmospheric conditions during the time of Zeroing the rifle
- Inclusion of Scope Height Adjustment (right now its assumed to be 1.75 inches, but can be edited in the BallisticThreader4Prin….. files , edit the 'self.scope_height = 1.75' line for your rifle. For now.
-A physical build kit inlcuding a harness for easy building! (I don't even know how to start with that…).

Best way to get long distance viewing: Pair up the 50mm lens with the following 1.5x teleconverter!! Now you have a 75mm lens equivalent at the cost of low-light performance!

For now,
Thank you all for your interest in the project and please don't be intimidated by diving into a new challenge.

-Jack