Heltec Wireless Tracker Send/Recv ESP32 Installation Guide and Setup

This document will walk you through how to get setup with ESP32 and the various tools you will need in order to make things work properly.

Dependency Installation

You will want to download the latest version of Python: https://www.python.org/downloads/

You should then download our CodeMetal Hackathon Example git repo: https://github.com/CodeMetalAI/hackathon-esp32-examples?tab=readme-ov-file

- This repo is your one stop shop for building and running ESP32 / Arduino apps that utilize Code Metal tools and applications.
- README file in this repository contains up to date instructions on required Arduino libraries, MicroPython/CircuitPython firmware, etc.
- Instructions in this document are specific to Send/Recv examples.

You will want to download the Thonny IDE for Micropython/CircuitPython programming (NOTE: Thonny is much better than VSCode for MicroPython/CircuitPython): https://thonny.org/

You will also want to download the Arduino IDE: https://support.arduino.cc/hc/en-us/articles/360019833020-Download-and-install-Arduino-IDE

You will also need to download specific MicroPython firmware binaries: https://micropython.org/download/ESP32_GENERIC_S3/

- Select the [.bin] file from the Firmware Releases Section

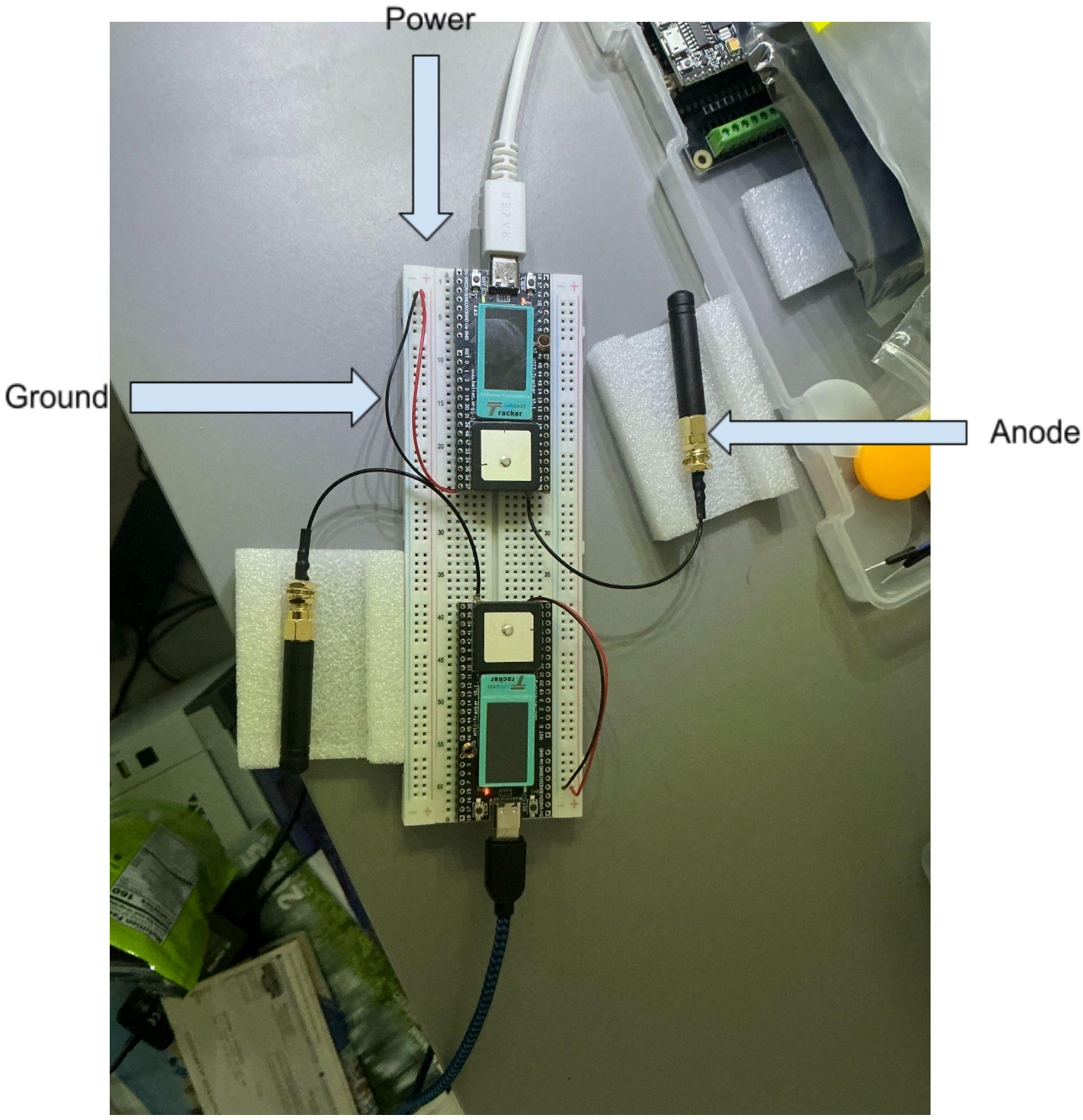
Some python libraries will need to be downloaded / cloned from GitHub so we can upload them to our devices:

- esptool.py for board flashing:
  - Open terminal and type: ***pip3 install esptool***
- MicroPySX126 for board programming: https://github.com/ehong-tl/micropySX126X
- [OPTIONAL] If you want to program the OLED Display, you'll need the ST7735 libraries: https://github.com/boochow/MicroPython-ST7735

Breadboard Setup

I had never done this kind of electrical engineering hardware setup / programming before, so guides talking about "anodes" (the black cylindrical thing) "power" (the red wire) and "grounding"

(the black wire) were foreign to me. In this instance, I am a 5 year old child, and in need of picture guides. Those were somewhat lacking online. Here's what the breadboard setup for the heltech-wireless-tracker sender/receiver looks like:



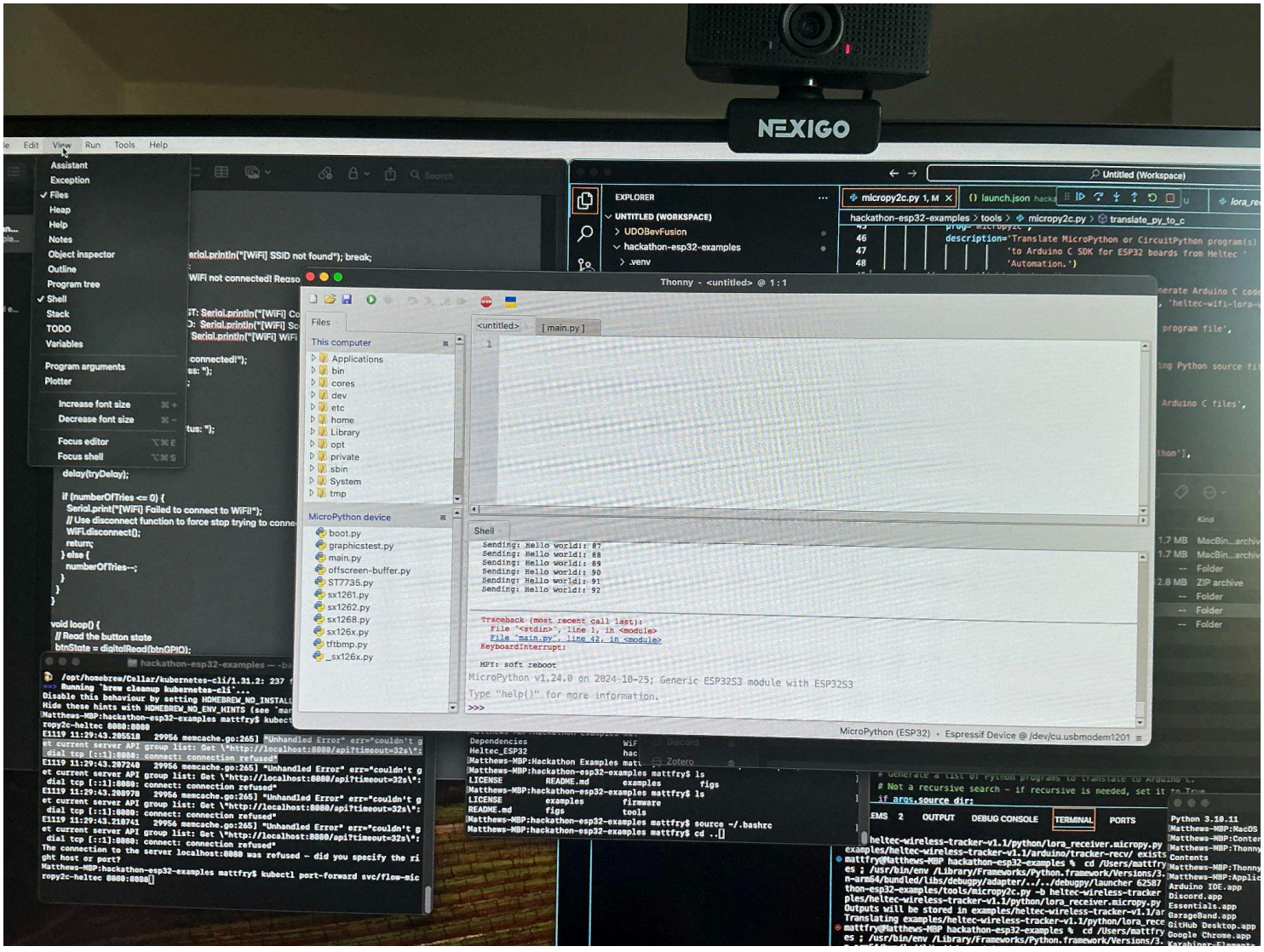
Thonny Setup / Firmware Flashing / Program Upload / Testing

Thonny Setup

First you're going to want to open Thonny and get your workspace setup. After opening Thonny, go to the top of the screen to the menu bar, and click "View":

- On the View checklist, checkmark "Files" and "Shell".

Your IDE should look something like this:



Additionally, you will want to allow your computer to run multiple windows of Thonny.

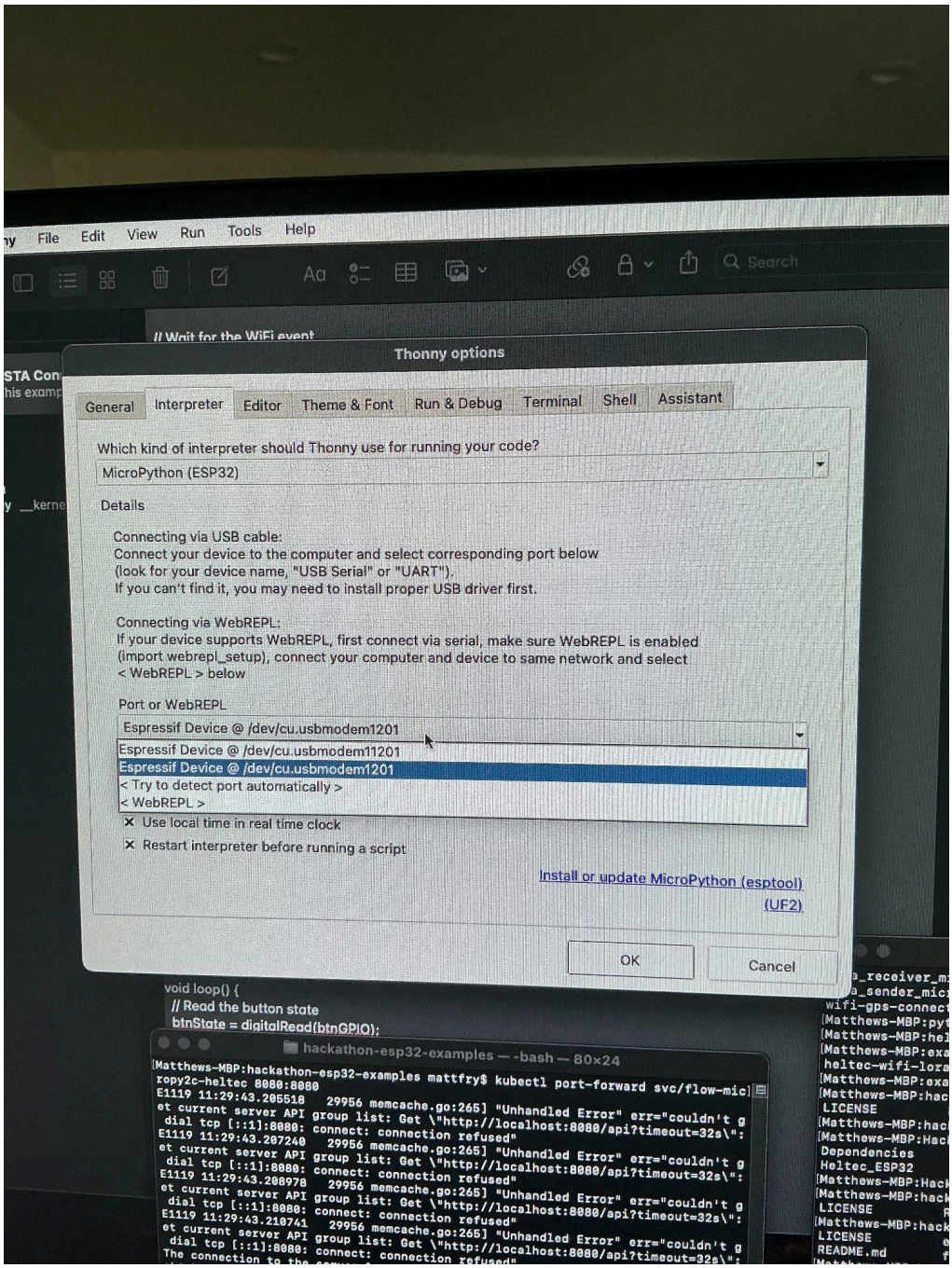
- In the menu bar, go to *Tools => Options*
- Under the “General” tab, uncheck the “Allow only single Thonny instance”
- Open a terminal and enter the following command to launch a second Thonny instance:
    - ***open -n -a thonny.app***

You should now have two Thonny windows open. Make sure to make Files and Shells viewable in this new instance by following the steps above.

Firmware Flashing

Next, you need to flash MicroPython onto the board. We can learn what our devices port numbers look like by perusing Thonny.

- In the menu bar, go to *Tools => Options*
- Select MicroPython (ESP32) from the Thonny Interpreter list
- In the Port or WebREPL section, the drop down menu should now contain two listed boards if everything has been wired correctly.
- This is what you might see:

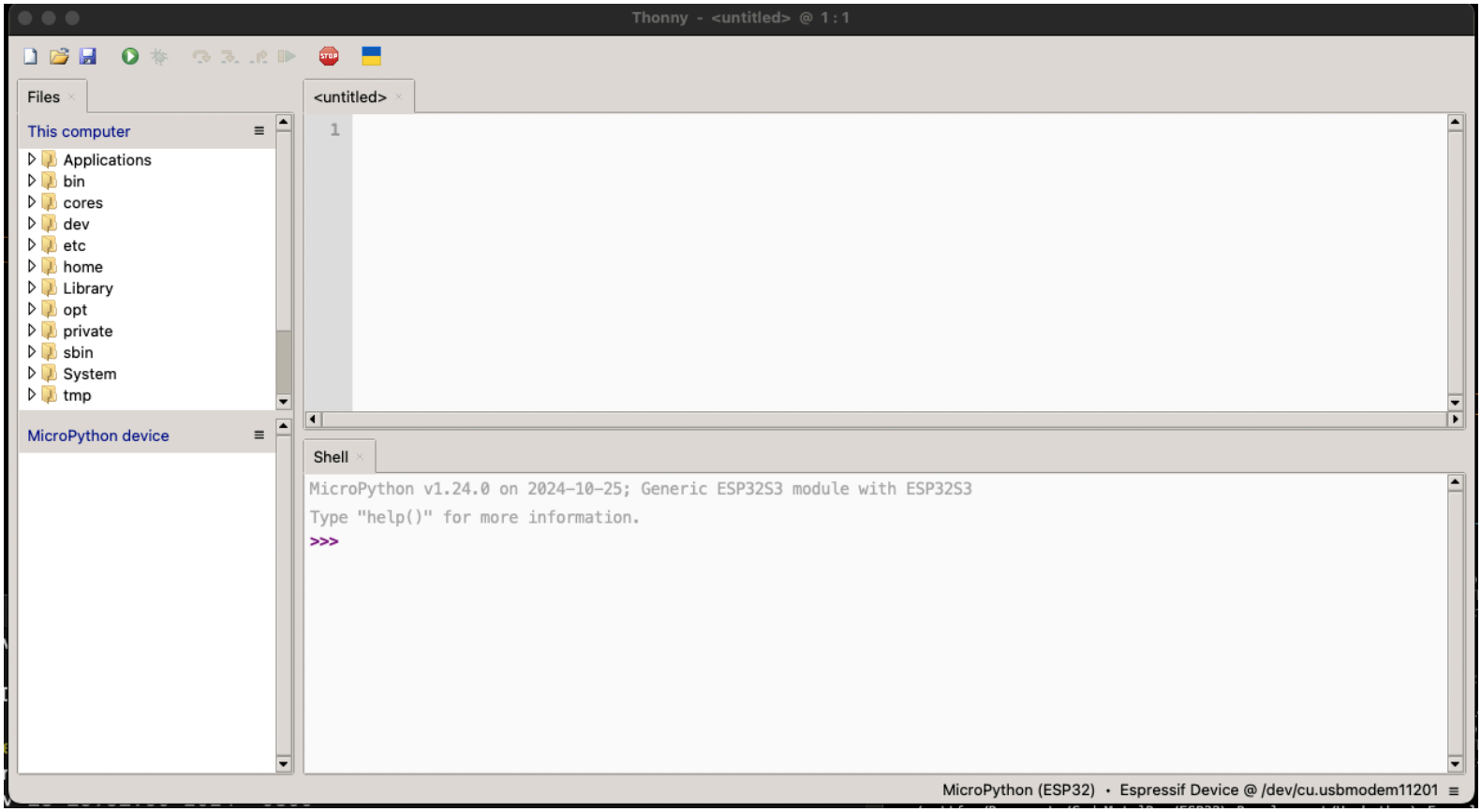


The “dev/cu.usbmodem1201” parts are what we’re after.

Next, open a terminal and run the following commands wherever you stored Firmware Binaries from the step above, replacing the <device-port> with the collected port names above:

- ./esptool/esptool.py --chip esp32s3 --port <device-port> erase_flash
- ./esptool/esptool.py --chip esp32s3 --port <device-port> write_flash -z 0 ESP32_GENERIC_S3-20241025-v1.24.0.bin

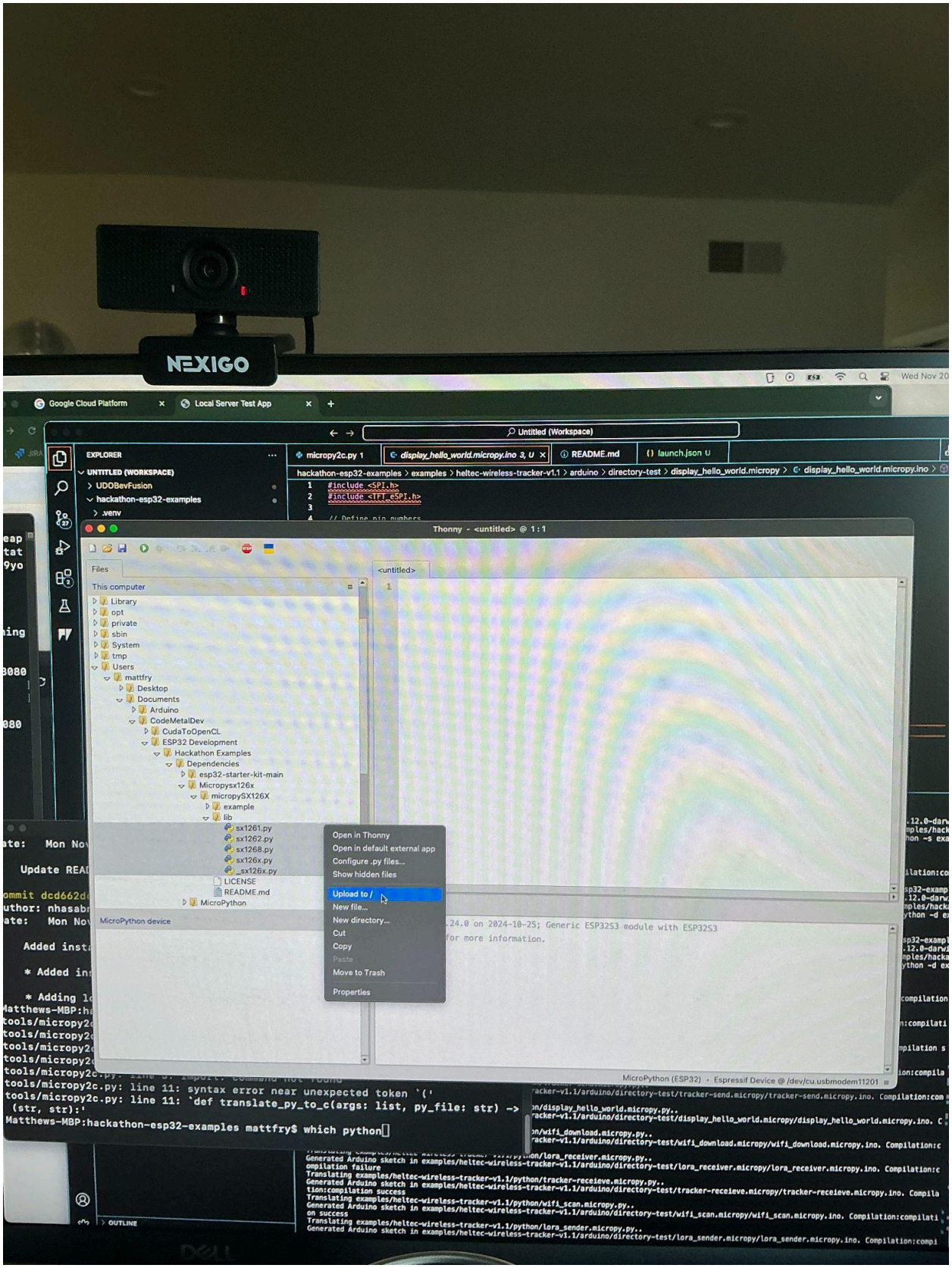
If the firmware was flashed correctly, you should now see the shell displaying an active python shell for you to use:



Program Upload

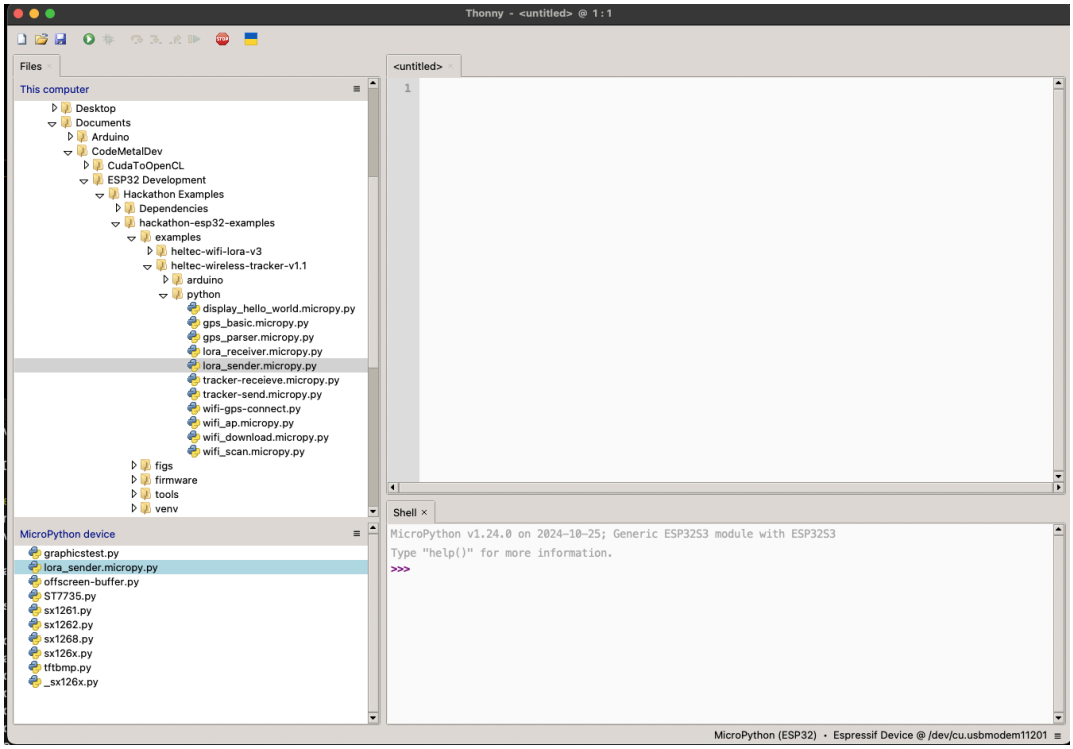
Now that the firmware has been successfully flashed, you’ll need to upload your micropython code to the device.

Navigate to the folders on your computer where you’ve stored the MicroPython libraries and code you wish to run. I want to tinker around with OLED displays and am using the send/receive code, so I need the ST7735 and Micropysx126X libraries. I am going to upload these python libraries to the device like so:

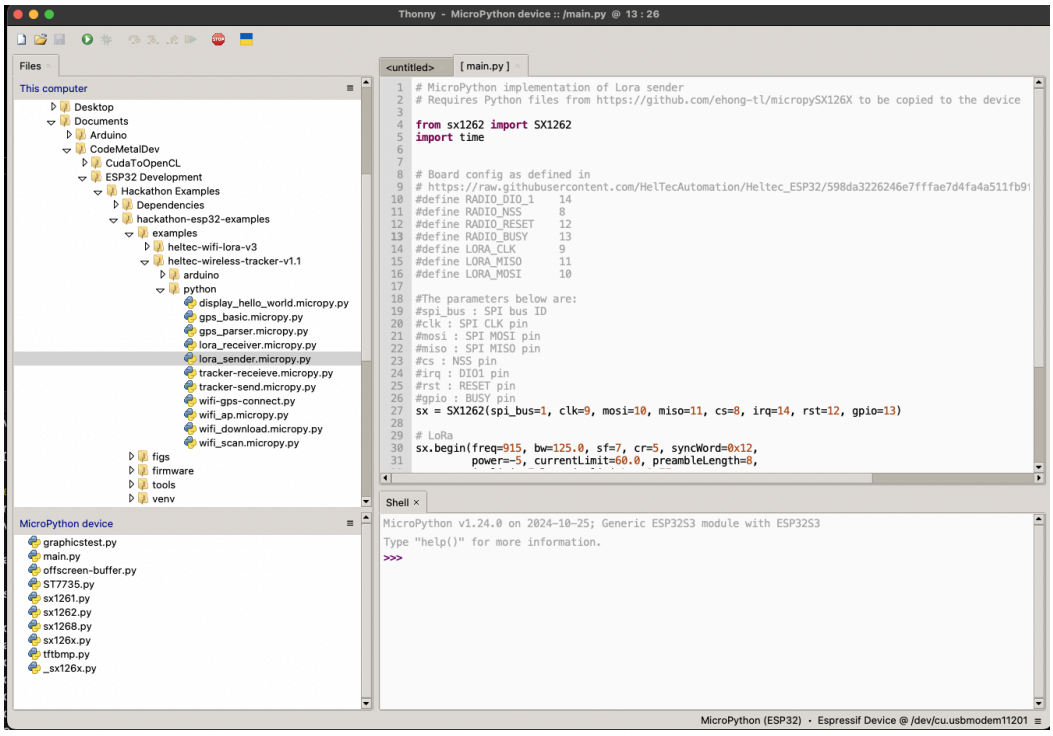

NEXIGO
Google Cloud Platform
Local Server Test App
Untitled (Workspace)
EXPLORER
UNTITLED (WORKSPACE)
UDOBevFusion
hackathon-esp32-examples
.venv
micropy2c.py 1
display_hello_world.micropy.ino 3, U
README.md
launch.json U
hackathon-esp32-examples > examples > heltec-wireless-tracker-v1.1 > arduino > directory-test > display_hello_world.micropy > display_hello_world.micropy.ino
#include <SPI.h>
#include <TFT_eSPI.h>
Thonny - <untitled> @ 1 : 1
Files
This computer
Library
opt
private
sbin
System
tmp
Users
mattfry
Desktop
Documents
Arduino
CodeMetalDev
CudaToOpenCL
ESP32 Development
Hackathon Examples
Dependencies
esp32-starter-kit-main
Micropysx126x
micropySX126X
example
lib
sx1261.py
sx1262.py
sx1268.py
sx126x.py
_sx126x.py
LICENSE
README.md
MicroPython
MicroPython device
<untitled>
Open in Thonny
Open in default external app
Configure .py files...
Show hidden files
Upload to /
New file...
New directory...
Cut
Copy
Paste
Move to Trash
Properties
for more information.
MicroPython (ESP32) · Espressif Device @ /dev/cu.usbmodem11201
tools/micropy2c.py: line 11: syntax error near unexpected token `('
tools/micropy2c.py: line 11: `def translate_py_to_c(args: list, py_file: str) ->
(str, str):'
Matthews-MBP:hackathon-esp32-examples mattfry$ which python
OUTLINE

Next I will upload the program I want to actually run. In order to run the program on the device, we must change the name of the program to “main.py”.

Before:

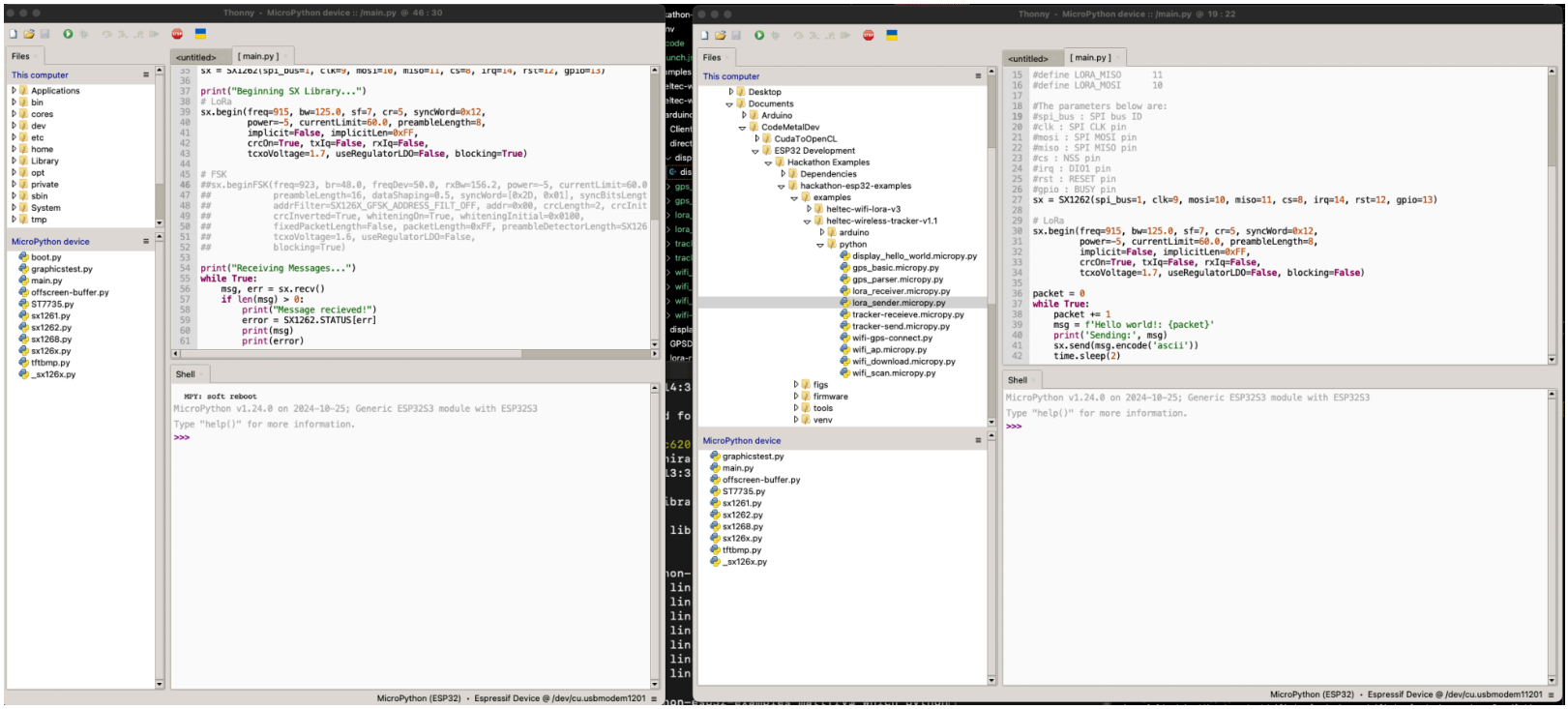


After:



Testing

Once you've done this process for both the sender and receiver code, you should have two Thonny IDE instances open, one with the sender python code, the other with the receiver python code:



Now you need only type "import main" on both shell terminals and the devices will begin operating the code you've uploaded: