2021-2022 IoT Project Guide

Anish Shenoy

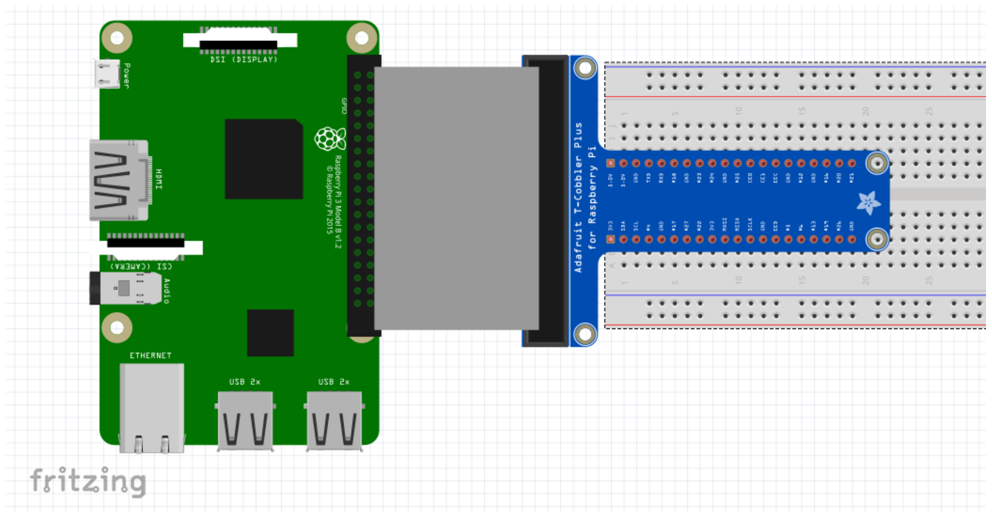
Hardware

Materials used

1. Raspberry Pi 4 Starter Kit 4GB, any raspberry pi 4 will do.
2. Assembled Pi T-Cobbler Plus - GPIO Breakout [Pi A+, B+, Pi 2/3/4, Zero] : ID 2028
3. Hook-up Wire Spool Set - 22AWG Solid Core - 6 x 25 ft : ID 1311 : $15.95 : Adafruit Industries, Unique & fun DIY electronics and kits
4. Multi-size wire stripper & cutter [5023] : ID 147
5. 12Pcs Small Solderable Breadboard 170 Tie-Points Mini Breadboard Kit Mini PCB Prototype Solderless Breadboards Bread Boards Circuits Beadboard(Lkelyonewy): Amazon.com
6. 4x ADXL343 - Triple-Axis Accelerometer (+-2g/4g/8g/16g) w/ I2C/SPI [STEMMA QT / Qwiic]
7. 2x Pull-up resistors (preferably 1K8)

Raspberry Pi Setup

1. Follow this guide to set up the Pi. The last few steps assume access to an external monitor. If you don't have an external monitor, this link includes alternative methods such as using a direct ethernet connection and SSH.
2. Connect the Adafruit T-cobbler to the Pi and the breadboard



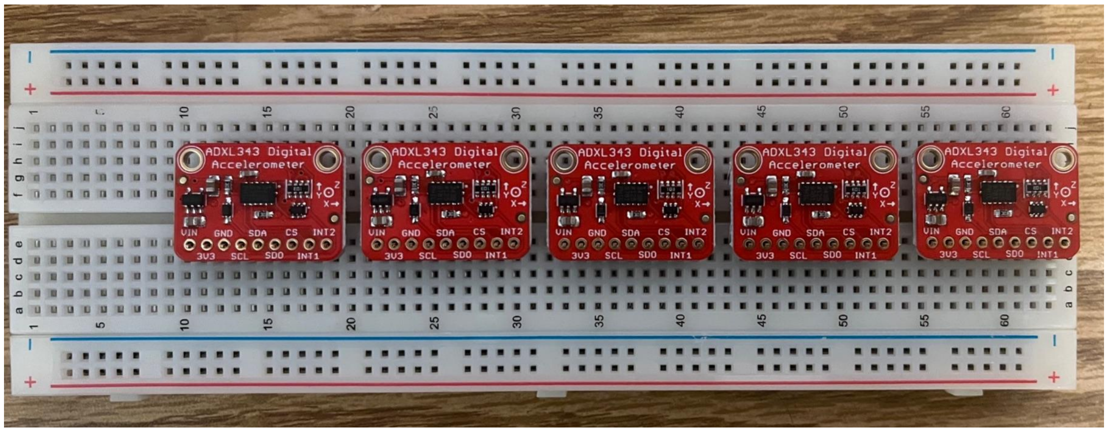
   This will make wiring easier because the pins will be clearly labeled.
3. I connected my Raspberry Pi to Caltech Secure since connecting to Caltech Visitor was giving me problems. If you prefer Caltech Visitor, look at the WiFi section of Kyle's guide. To connect to Caltech Secure, add these lines to your /etc/wpa_supplicant/wpa_supplicant.conf (edit as appropriate):

```
network = {
        ssid="Caltech Secure"
        proto=RSN
        key_mgmt=WPA-EAP
        auth_alg=OPEN
        eap=PEAP
        identity="<your caltech alias i.e ashenoy>"
        password="<your password>"
        phase1="peaplabel=0"
        phase2="auth=MSCHAPV2"
        priority=1
}
```

4. You can either work from your raspberry pi using a monitor, or SSH into it. I preferred to SSH into it. To do that, you'll need to get the IP address of your Pi using the `$ifconfig` command. I also set up remote VS code using this guide which seamlessly handles all the file syncing and SSH'ing.

Sensor setup

1. Solder the sensors using this guide for all of the sensors.



2. Install CircuitPython (a version of micropython that is able to interface with hardware) and Adafruit libraries using this guide.
3. Enable an additional I2C bus:
   a. Run the command `$ sudo nano /boot/config.txt`
   b. Modify the following existing lines:
      i. `dtparam=i2c_arm=on`
      ii. `i2c_arm_baudrate=100000`
   c. Add the following line:
      i. `dtoverlay=i2c-gpio,bus=4,i2c_gpio_delay_us=2,i2c_gpio_sda=23,i2c_gpio_scl=24`
   d. Reboot

   This will add an additional I2C bus (#4) with pin 23 as SDA and 24 as SCL. This is required since we have 4 sensors and the Pi only has one I2C bus (#1). You can verify that the new I2C bus works by running the command `$ sudo i2cdetect -y 4`
4. Add 1K8 pullup resistors to the new I2C ports.
5. Connect the sensors:
   a. For every I2C bus, connect two sensors, one of which has the SDO wired to HIGH to use an alternate address. This is needed because each I2C bus can only handle sensors with different addresses.
   b. This guide provides more details on how to wire the sensors using the I2C protocol.
   c. You can refer to the data sheet here for more information on the technical details of the sensor.
6. Get the sensors attached to the door (or anything else). You will have to be creative about wiring. Make sure that cables are secured so that they don't come out during movement and don't present a tripping hazard.
7. Verify that the sensors are sending data to the Pi by running the `$ python accel_test.py`

Local Processing and IoTPy

1. Clone the IoTPy repo to the Raspberry Pi.
2. The code that handles local processing for the door detection project is in *door_multiprocessing.py*. This code makes use of the IoTPy library to process the accelerometer data and identify when to send data to the cloud for ML processing. You can use it as an example for your own local processing code.
3. The *cloud_func* function in python is what sends the appropriate window of sensor data to the cloud. For now, you can leave this function blank (or simply, print the window)
4. Refer to the notebooks *Stream.ipynb* and *ExampleOperators.ipynb* for good explanations of how the code works behind the scenes.

Training a model (locally)

1. The *cloud_func* method returns the desired window as a numpy array. Data should be saved using *np.save* for training.
2. The specific model is dependent on the kind of problem you are solving. However, you'll probably need some sort of time-series model. Here's some of the research I was able to find
3. Training data and code from this project is in the */examples/door-detection-anish/ML* folder of the repo.

ML algorithm research

Book: https://moa.cms.waikato.ac.nz/book/ or
https://direct.mit.edu/books/book/4475/Machine-Learning-for-Data-Streamswith-Practical

1. Online learning:
    a. https://radicalbit.io/online-machine-learning-into-the-deep/
2. Semi-supervised:
    a. https://github.com/yassouali/awesome-semi-supervised-learning
    b. http://www.acad.bg/ebook/ml/MITPress-%20SemiSupervised%20Learning.pdf
    c.
3. Combining Both:
    a. https://proceedings.mlr.press/v80/wagner18a/wagner18a.pdf
4. Clustering:
    a. https://en.wikipedia.org/wiki/Data_stream_clustering
    b. https://arxiv.org/pdf/2007.10781.pdf (Survey of different data stream clustering)

Libraries:

1. https://towardsdatascience.com/river-the-best-python-library-for-online-machine-learning-56bf6f71a403 (River, online learning)
    a. https://github.com/online-ml/river
2. https://www.sktime.org/en/stable/ (Sktime, time-series classification)

Azure Setup

1. Sign up for an Azure Student account at azure.microsoft.com using your caltech.edu email address.
2. Set up a machine learning workspace by following this guide
   The machine learning workspace allows you to keep track of different models and manage deployments.
3. You can also explore Azure's IoT Hub which includes features like sending live telemetry. Here are some guides for that:
   a. Send telemetry using SDK (only for setting up iot hub, device registration)
   b. Using the python SDK on the Pi: https://github.com/Azure/azure-iot-sdk-python
   c. Upload files from devices to Azure IoT Hub with Python | Microsoft Docs Upload files like .npy to azure.

Communicating with Azure

1. Once you have a model trained locally, you need to deploy it to the cloud using Azure. Deploying the model means hosting the model in the cloud and having a web endpoint that you can send data to and receive predictions from.
2. Follow this guide to deploy your model using Azure.
3. The accompanying code is in the */examples/door-detection-anish/ML* folder of the repo.
4. If you would like to use Twilio to incorporate SMS into you workflow (i.e. to send final predictions), follow this guide.