Robotics 311 : How to build robots and make them move

Prof. Elliott Rouse

GSI Yves Nazon MS

Fall 2022

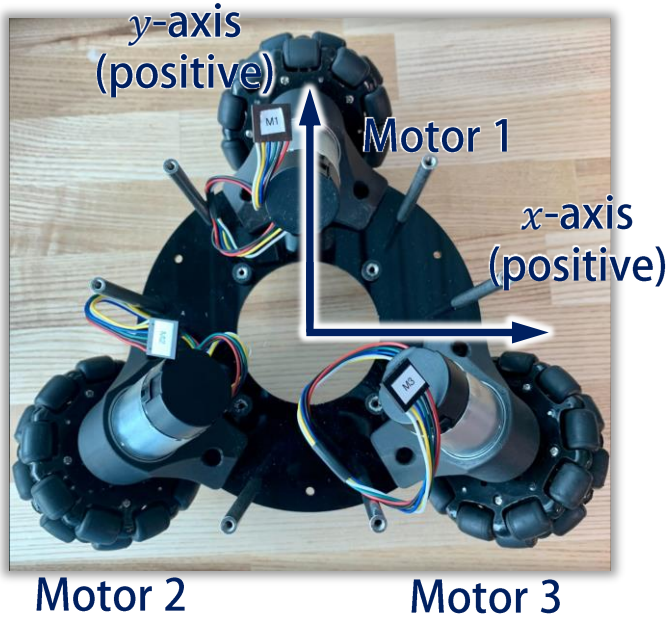
Photo credit: Brenda Ahearn / U-M CoE

ROB 311 – Lab 8

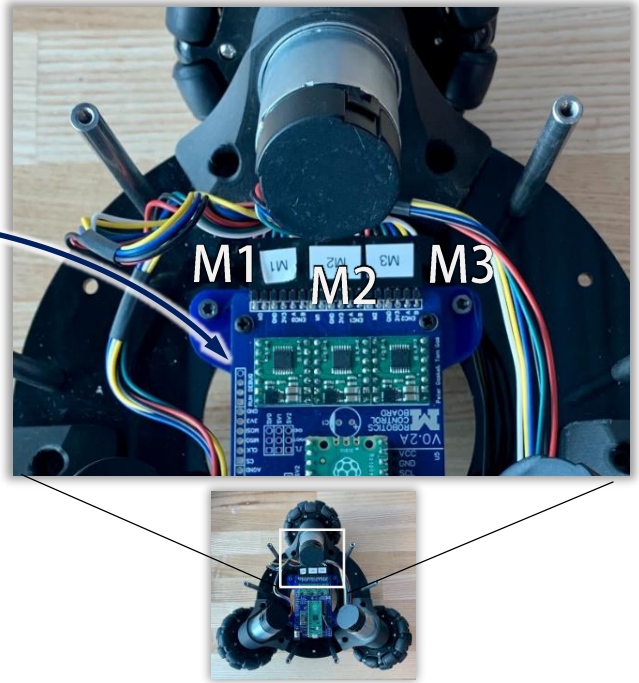
- Today we will
  - Wire your ball-bot for power and motors
  - Learn to flash your Pico
  - Learn to read and save data from your RPi
  - Learn how to import and plot your data in MATLAB
  - Interact with the ball-bot and inspect the acquired data

Wiring Your Ball-Bot

- To wire your ball-bots, there are three items that need to be connected
  - Motors plug into your Pico board (white wires on the side nearest to motor 3 when viewed from above)
  - USB-A to USB-C connects your battery to your Rpi
  - Battery also connects barrel jack to Pico board
  - USB-A to micro-USB connects the RPi to the Pico

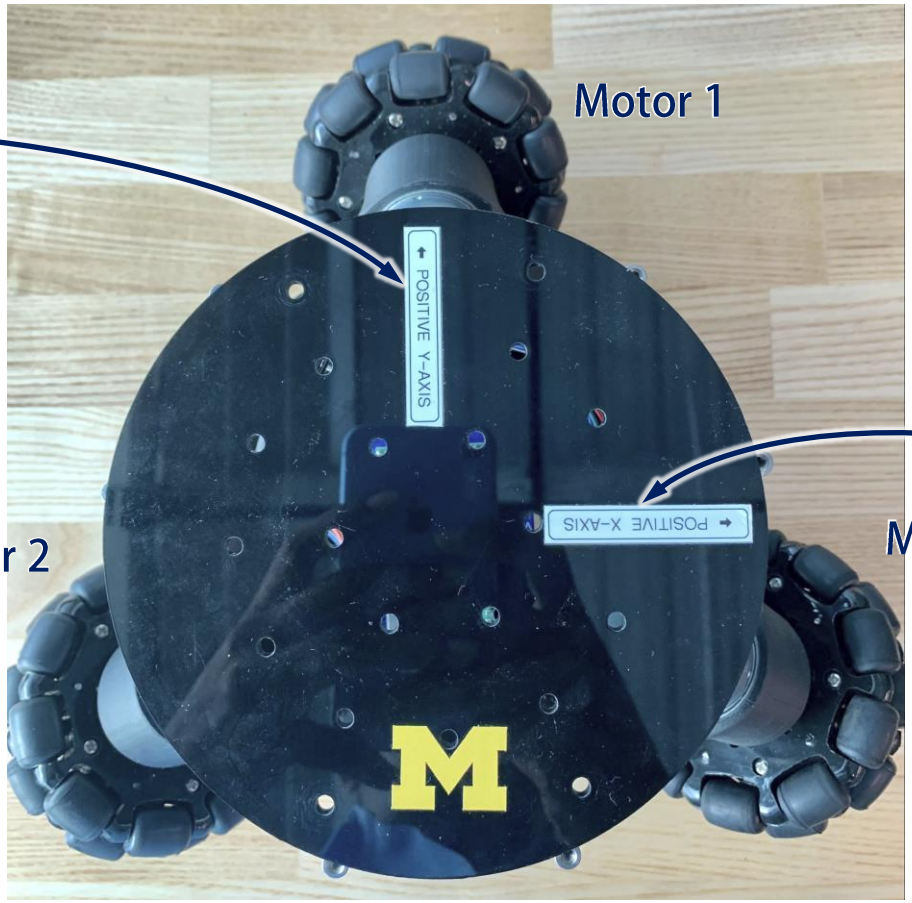


Orient the Pico board to have the connectors near Motor 1



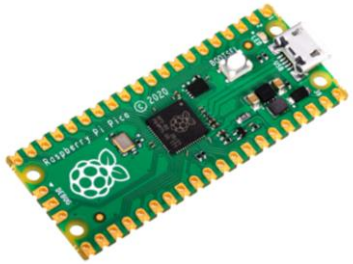
Wiring Your Ball-Bot

- To facilitate understanding / control, label your axes with the label maker
- Print / add labels similar to below to show your axis configuration
- Motor 1 is along the y-axis

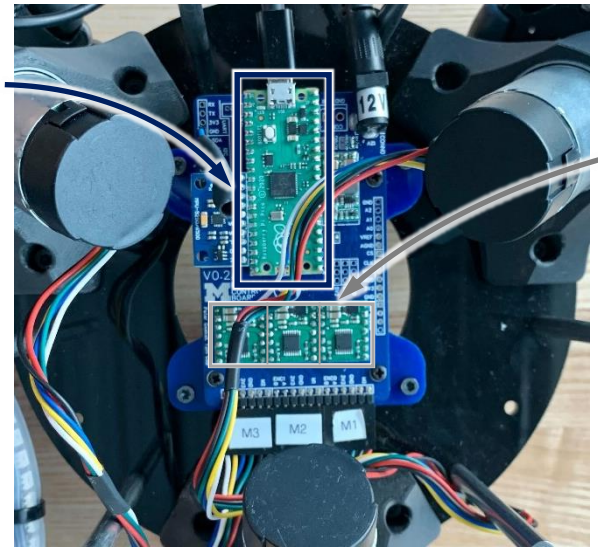


Raspberry Pi Pico

- Robotics projects often include the integration of sensing, control, and actuation
- We will use the U-M Robotics Pico board to help with sensing, actuation, and communication
- It uses an RPi Pico—we will use this setup for its convenience and motor drivers, but we could run the system without it
- Your Pico will act as a pass-through—it does not do any computation and instead, collects data, shares with the RPi, and runs the motor drivers



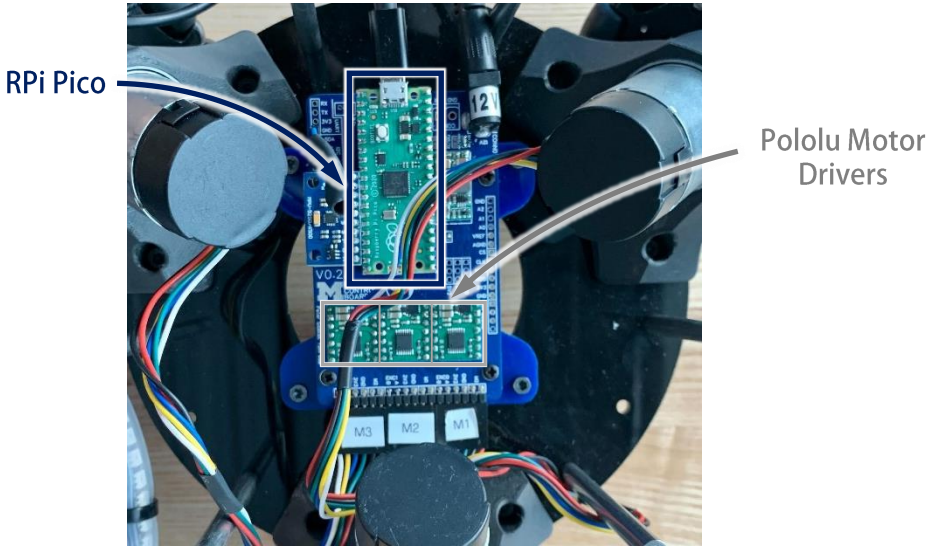
Raspberry Pi Pico
(link)



Pololu DRV8874
(link)

Raspberry Pi Pico

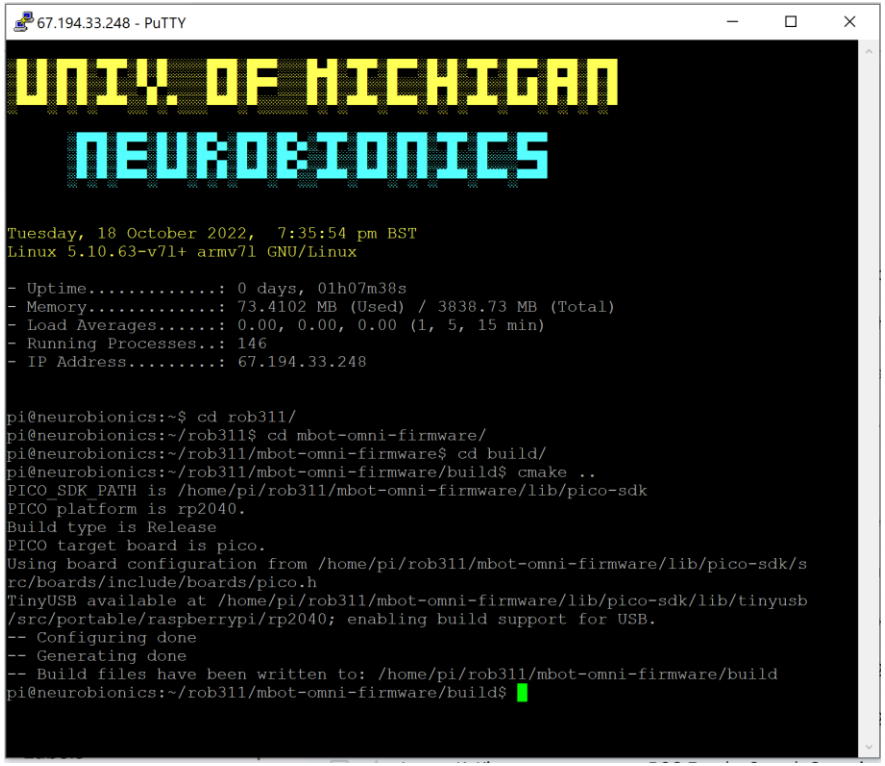
- We need to add instructions to the Pico, a process known as ‘flashing’
- The file with the instructions is known as ‘firmware’
- Firmware is generally one set of instructions (instead of an OS)
  - Once flashed, we won’t need to re-flash unless we change the firmware (unlikely)
- To flash our Pico, we first need connect to our Raspberry Pis



Flashing the Pico

- Connect to your RPi using its IP address
- Navigate to `~/rob311/mbot-omni-firmware/build`
- If the build directory does not exist, you can make it with `mkdir build`
- Then, run the following commands to compile the firmware
- After compiling the binaries, we will download onto the Pico

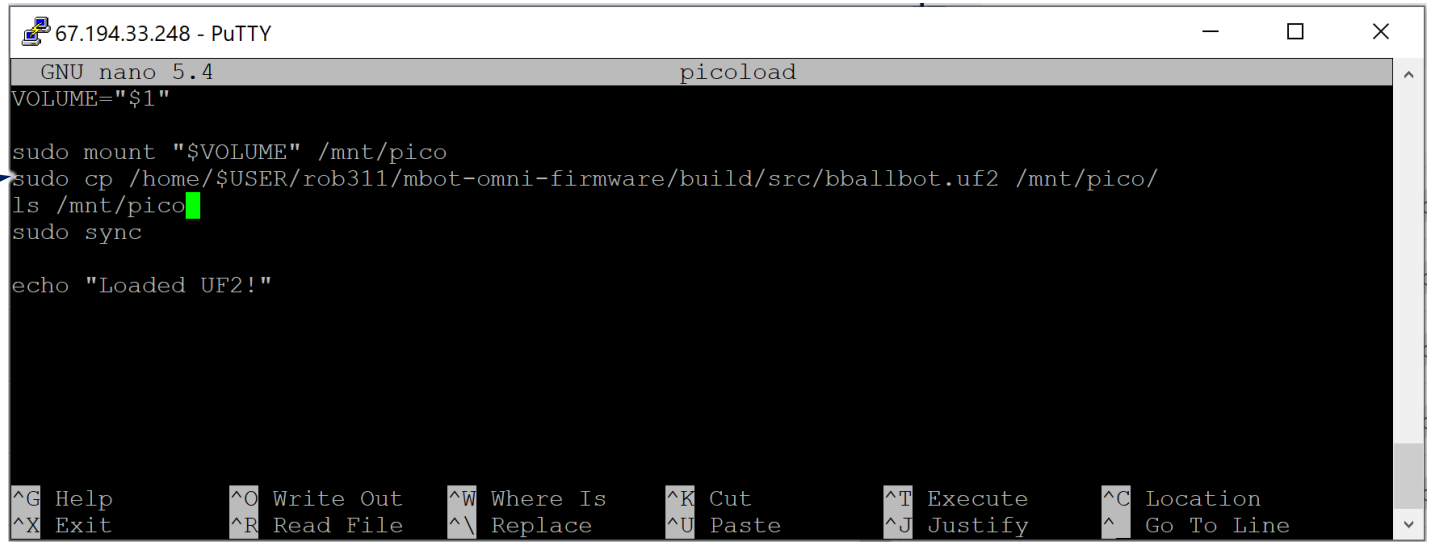
```
cd rob311
cd mbot-omni-firmware
cd build
cmake ..
make
```



Flashing the Pico

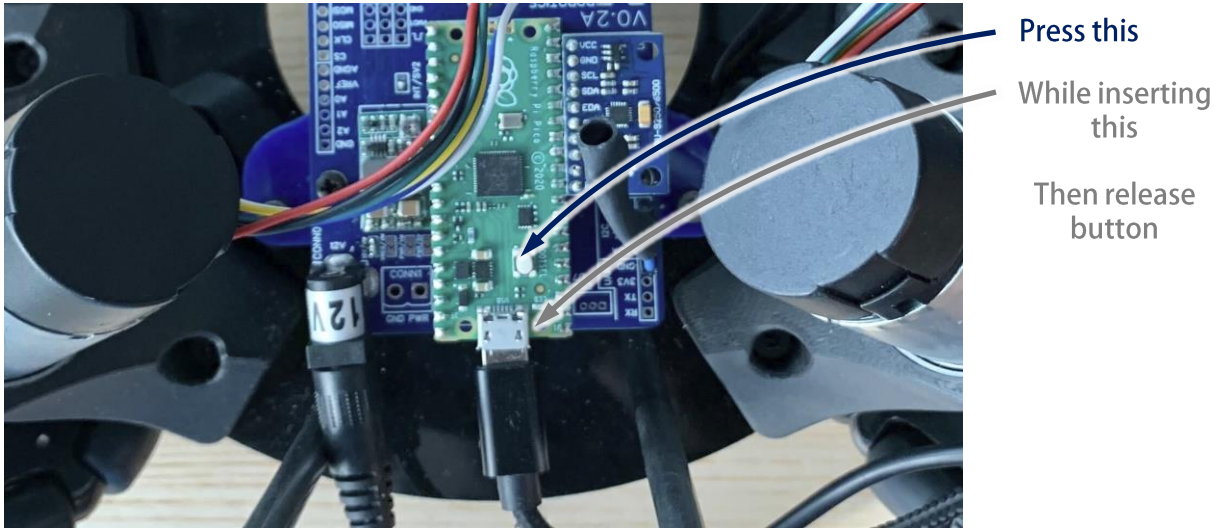
- First, we want to confirm the locations of the binaries (in case you changed the directory structure of your fork)
- Navigate to the `rob311` outer directory
- Inspect the picoload bash script by typing `nano picoload`
- Make sure the directory listed is correct (using WinSCP or VSCode) by verifying that the .uf2 file is in the specified directory

Verify this line points to the correct file location



Flashing the Pico

- Before we flash the Pico, we need to tell it that we will be sending it new instructions
- This is done by pressing the white button while the micro USB connector is inserted
- This tells the Pico to bootload the instructions



- Now you can run the picoload script by typing the following in the terminal:

```
./picoload /dev/sda1
```

- When completed, it should echo '`Loaded UF2!`' without errors

Troubleshooting

- If `cmake` or `make` is not recognized, we may need to add some packages
- In the terminal, execute the following commands:

```
sudo apt-get update
sudo apt-get install cmake
sudo apt-get install gcc-arm-none-eabi
```

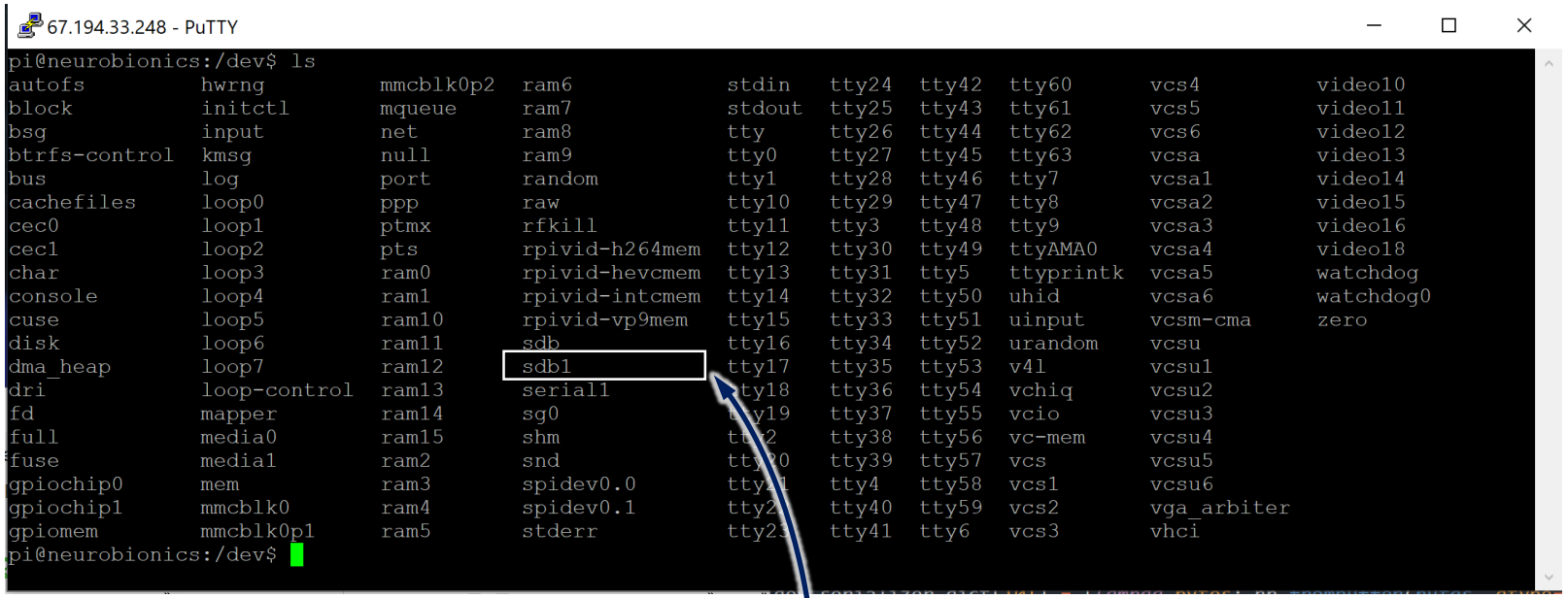
- If picoload fails, you may need to create a directory

```
cd /mnt
sudo mkdir pico
```

Troubleshooting

- If it doesn't work and you've been unplugging / re-plugging in your Pico, your device name may have changed
- Your device can be found by looking in `/dev/` folder (`cd /dev`), then `ls`
- To check, look for `sda1`, `sdb1`, `sdc1`, etc.
- This needs to be the correct device name in `./picoload /dev/sda1`

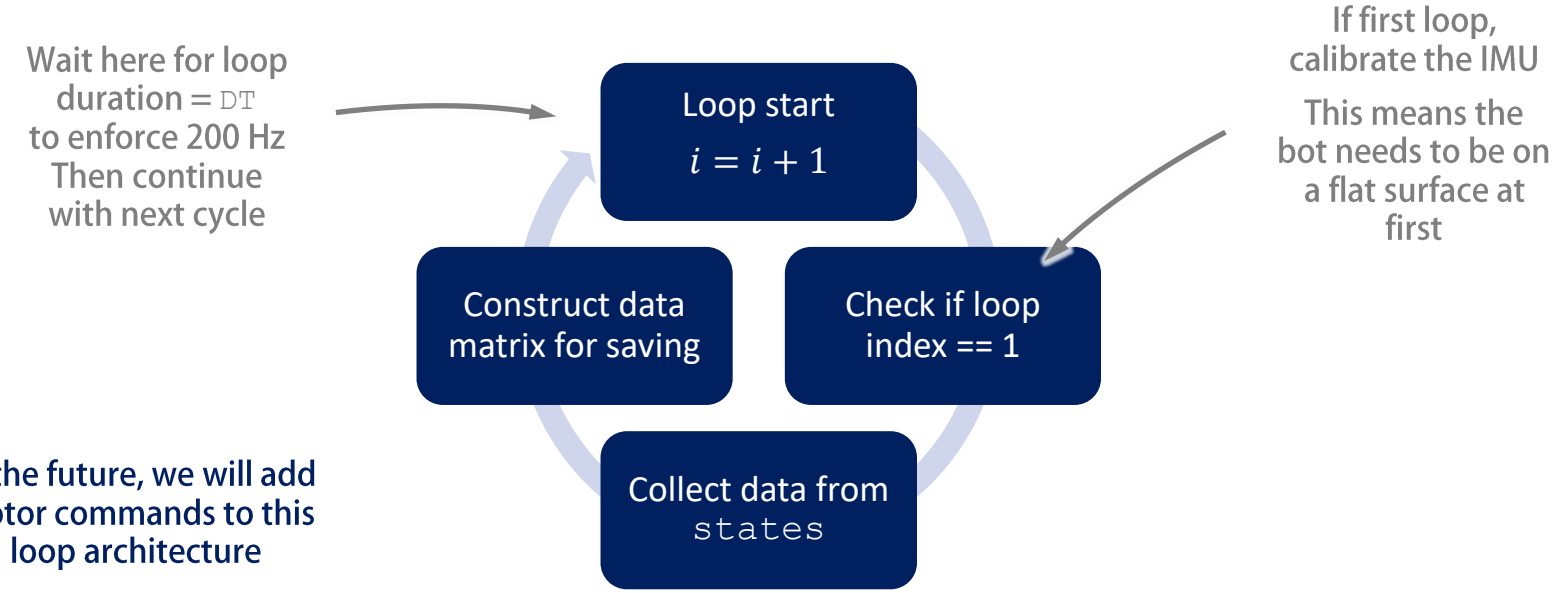
Needs to match /dev



Re-named `sdb1` by unplugging and re-plugging in

Reading Data From Your RPi

- Now your system is ready to run
- Today, we will read and plot data, but not turn on the motors
- We will read from the **IMU** and **motor encoders**
  - The IMU will tell you rotation around the x and y axes
    - Units: Radians
  - The motor encoder will tell you the rotation of each motors
    - Units: Radians (I think)



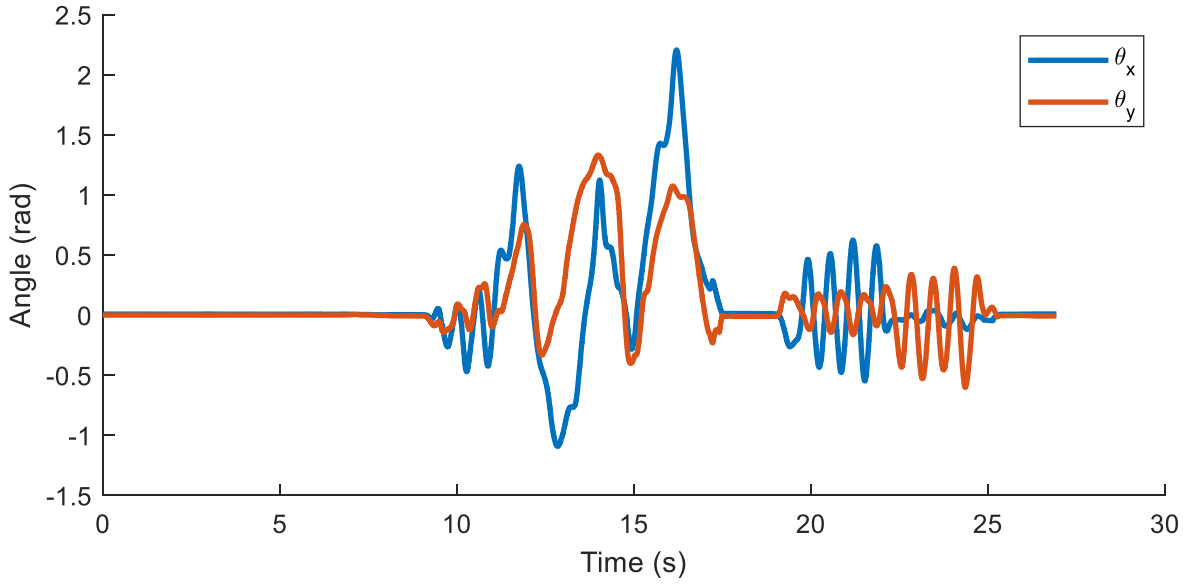
In the future, we will add motor commands to this loop architecture

Reading Data From Your RPi

- To collect IMU data, first download files from Canvas\Labs\Lab 8
- You will need
  - `Datalogger.py`
  - `ROB311_sensor_read_demo.py`
  - `ROB311_ball_bot_data_analysis.m`
- Add the Python files to the `ballbot-omni-app` folder on your RPi
- Run the python script using `python ROB311_sensor_read_demo.py`
- It will prompt you for a trial number for the data to be saved
- Calibration must be done while the ball-bot is on a flat surface
- Following calibration, rotate the ball bot around the x and y axes
- Use `ctrl+c` to stop the program
- It will save a file named `ROB311_TestX.txt`

Reading Data From Your RPi

- When you're finished, moved the saved .txt file from your RPi to your working MATLAB directory.
- Download the m-file from Canvas and move your data to this folder.
- Edit the filename to be sure it matches the name of your .txt file
- Run the m-file and it will plot your x and y-axis rotations



Reading Data From Your RPi

- Place your data from the RPi in the same folder as this MATLAB file
- This enables quick and easy viewing of your data
- We will show a real-time plotter next lab

Edit the filename to match

Create data vectors (add any new variables you are saving (in the right order)

Plot for inspection

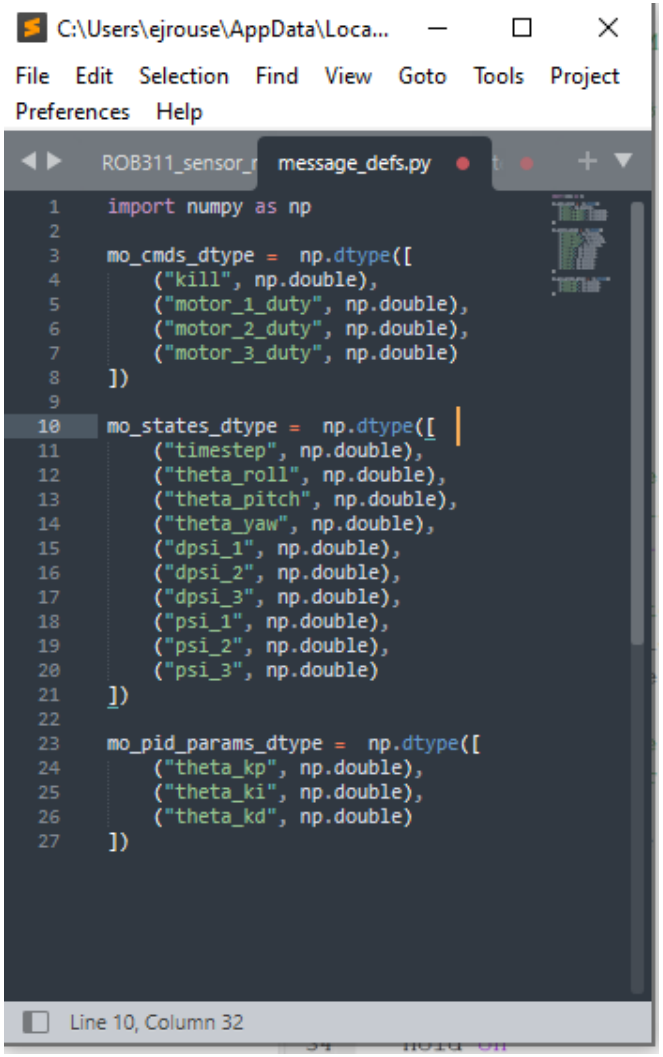
Editor - C:\Users\ejrouse\Desktop\ROUSE\Michigan\ROB 311\Labs\Lab 8\ROB311_ball_bot_data_analysis.m

ROB311_torque_conversion_student.m | ROB311_ball_bot_data_analysis.m

```
% ROB 311, Fall 2022
% Data importing / plotting from ball-bot
%
% University of Michigan
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear
clc
close all

alpha = pi/4;
beta = pi/2;
Rk = 0.11925;
Rw = 0.04778;

% Input file name -  this will have to updated with the name of the trial
% you are analyzing
filename = 'ROB311_Test2.txt';

% Use file name to load / create data matrix
eval(['load ' filename ';']);
eval(['data = ' erase(filename,'.txt') ';']);

% Define variables from data - these correspond to the order described in
% the read sensor demo
index = data(:,1);
time = data(:,2);
theta_x = data(:,3);
theta_y = data(:,4);

% Plotting
figure
hold on
plot(time, theta_x, 'linewidth',2)
plot(time, theta_y, 'linewidth',2)
legend('\theta_x', '\theta_y')
xlabel('Time (s)')
ylabel('Angle (rad)')
```

Reading Data From Your RPi

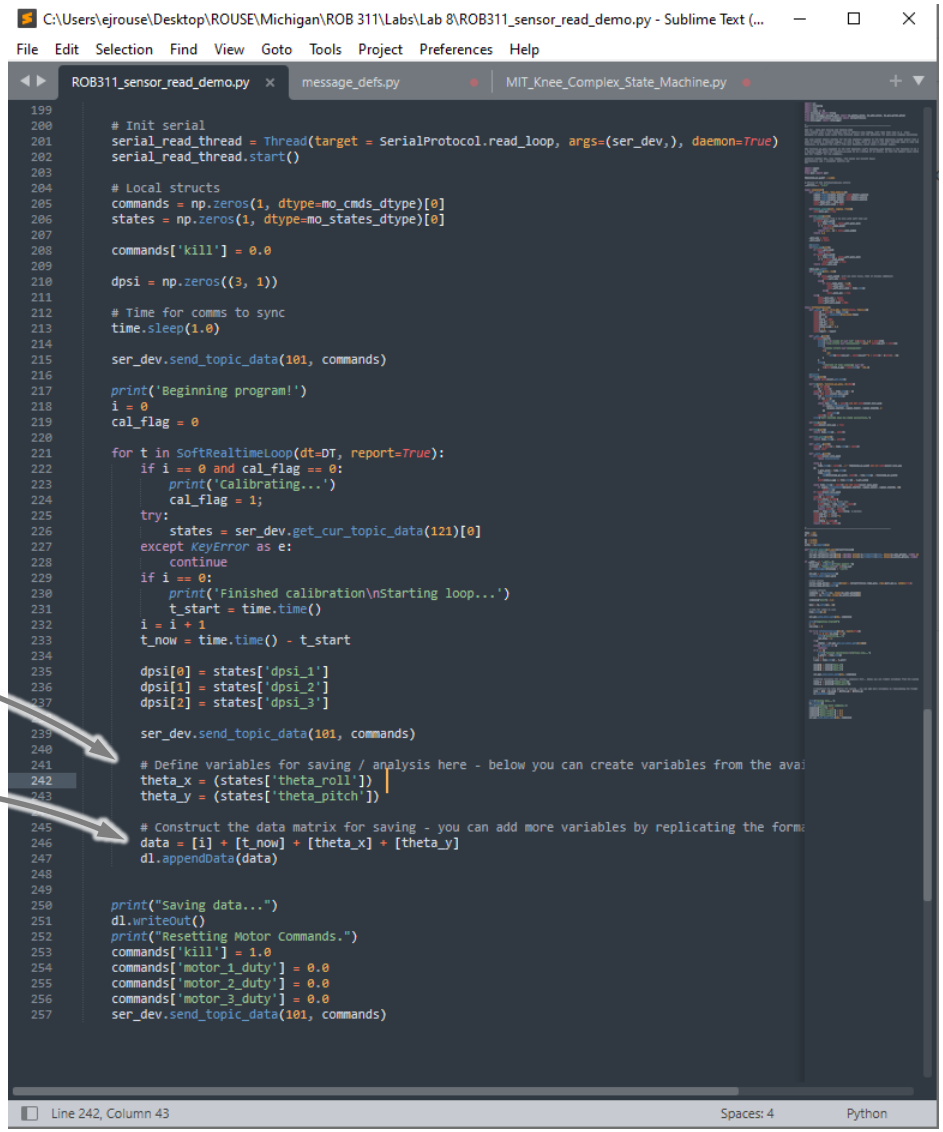
- You have access to many variables collected and sent by the Pico
- These variables are defined in `message_defs.py`
- You can access these variables using the same command we used to get x and y axis rotations
- Edit your python script to also save wheel rotations (`psi`)
- Add them to the data matrix to be saved
- Match the formatting / syntax of how the data are received and stored in the data matrix



Reading Data From Your RPi

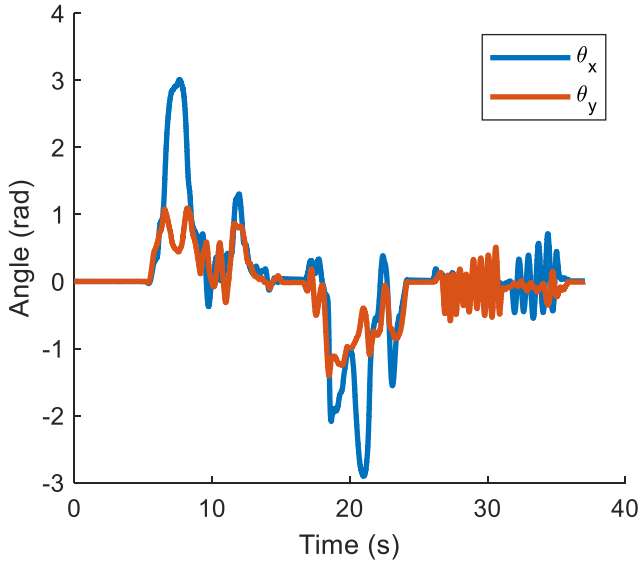
Add more variables here

To save variables, add them to this data matrix and remember the order for importing to MATLAB (keep the brackets)



Reading Data From Your RPi

- Collect data while manually driving your ball-bot
- Push down on the top and try to roll it with the basketball
- Save your data and plot the result in MATLAB
- Does it look correct? Can you tell if / when the wheels slipped?



Push down and horizontally (x-y plane) and you should be able to backdrive the ball-bot and make it roll