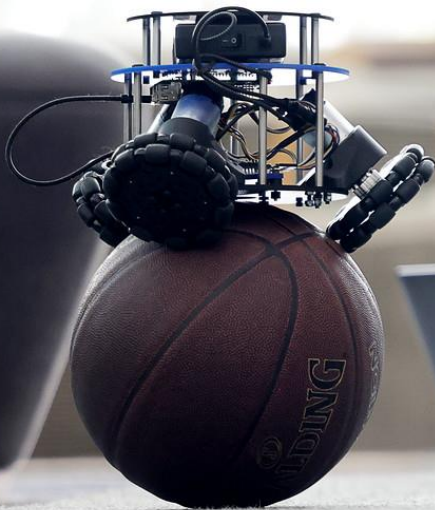


Robotics 311 : How to build robots and make them move

Prof. Elliott Rouse

GSI Yves Nazon MS

Fall 2022

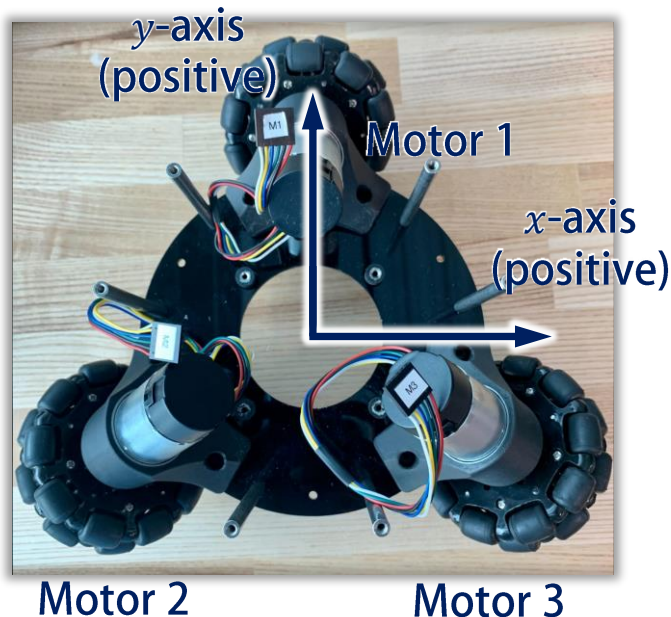


ROB 311 – Lab 8

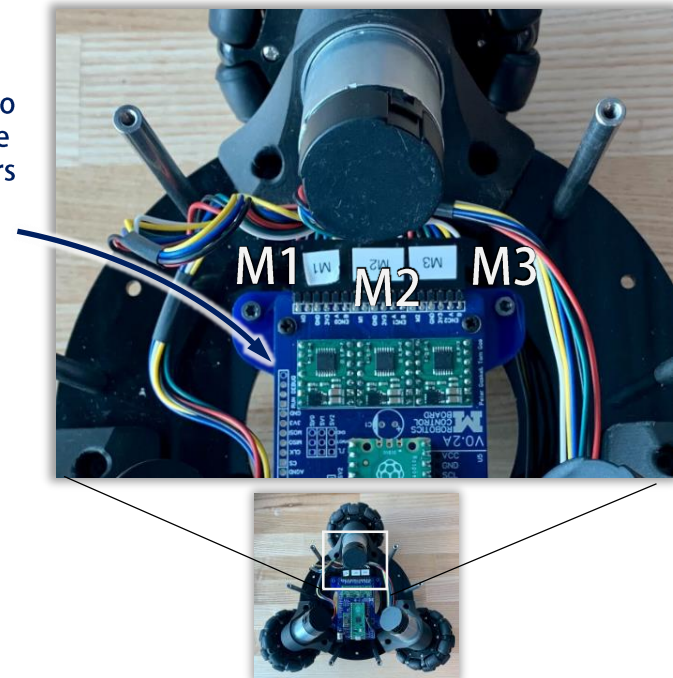
- Today we will
 - Wire your ball-bot for power and motors
 - Learn to flash your Pico
 - Learn to read and save data from your RPi
 - Learn how to import and plot your data in MATLAB
 - Interact with the ball-bot and inspect the acquired data

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

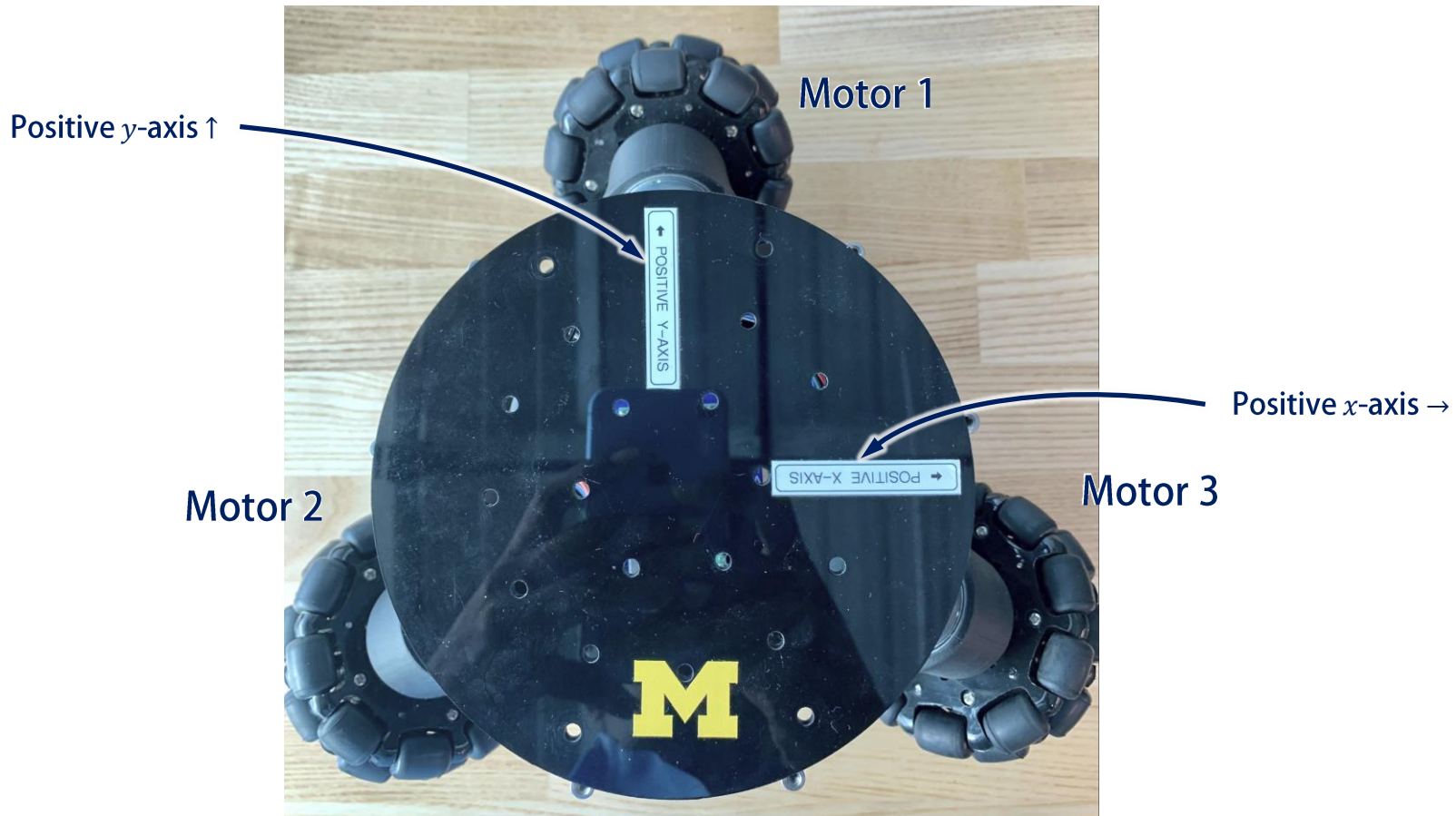


Orient the Pico board to have the connectors near Motor 1



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

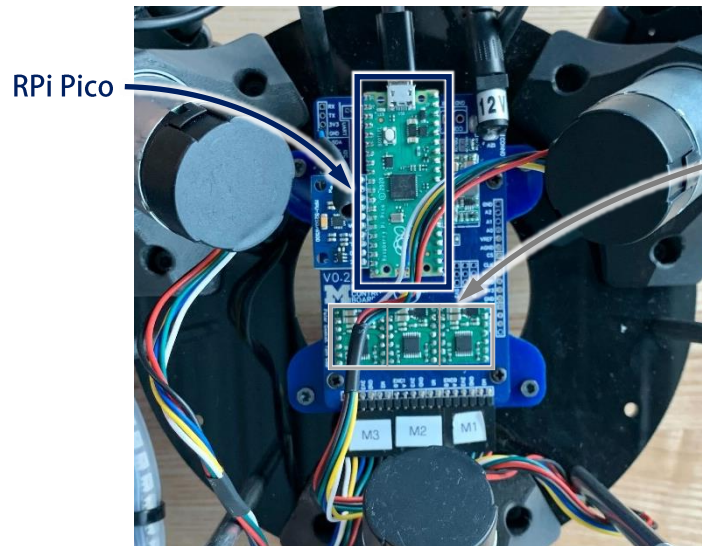


Raspberry Pi Pico

- 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](#))



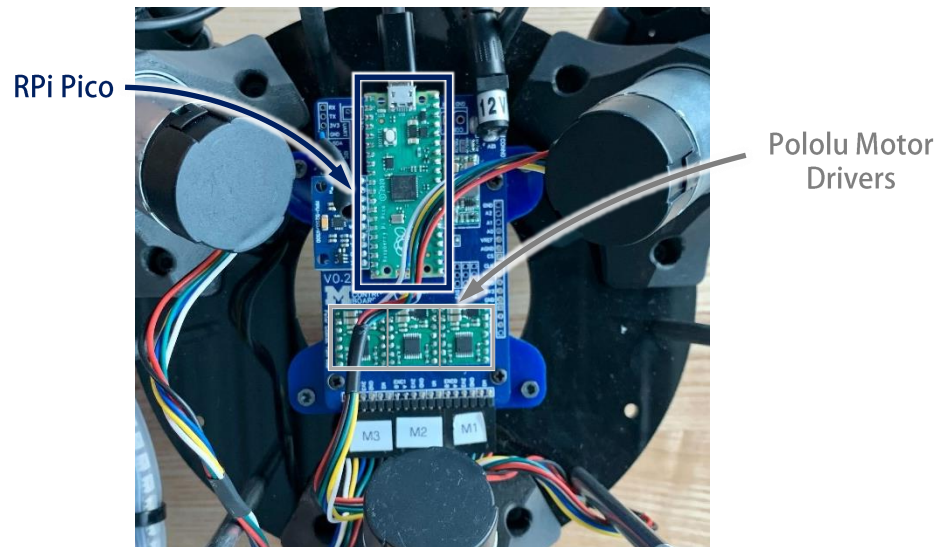
Pololu Motor
Drivers
(1 per motor)



Pololu DRV8874
([link](#))

Raspberry Pi Pico

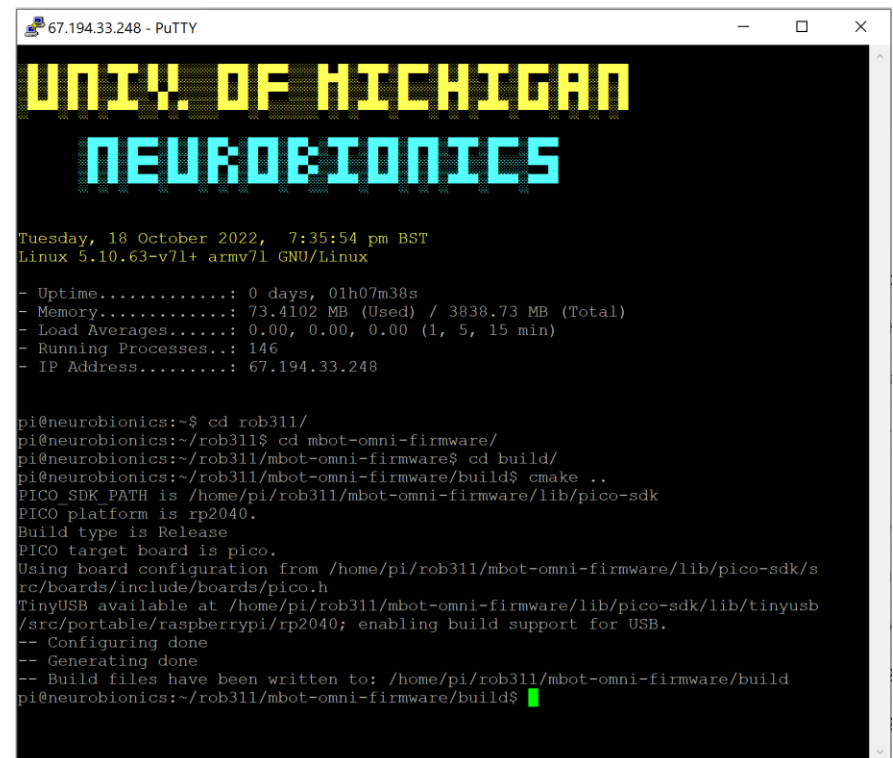
- We need to add instructions to the Pico, a process known as ‘flashing’
- The file with the instructions is known as ‘firmware’
- Firmware is generally one set of instructions (instead of an OS)
 - Once flashed, we won’t need to re-flash unless we change the firmware (unlikely)
- To flash our Pico,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



The screenshot shows a PuTTY terminal window titled "67.194.33.248 - PuTTY". The terminal displays the following content:

```
UNIV. OF MICHIGAN
NEUROBIONICS

Tuesday, 18 October 2022, 7:35:54 pm BST
Linux 5.10.63-v7l+ armv7l GNU/Linux

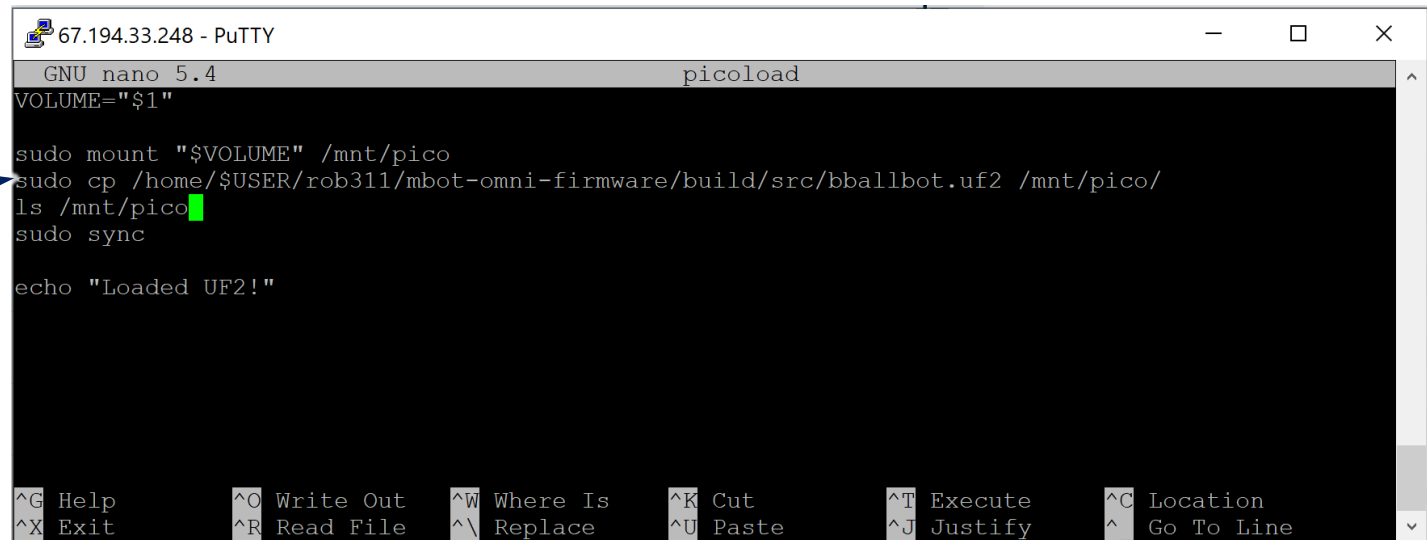
- Uptime.....: 0 days, 01h07m38s
- Memory.....: 73.4102 MB (Used) / 3838.73 MB (Total)
- Load Averages.....: 0.00, 0.00, 0.00 (1, 5, 15 min)
- Running Processes..: 146
- IP Address.....: 67.194.33.248

pi@neurobionics:~$ cd rob311/
pi@neurobionics:~/rob311$ cd mbot-omni-firmware/
pi@neurobionics:~/rob311/mbot-omni-firmware$ cd build/
pi@neurobionics:~/rob311/mbot-omni-firmware/build$ cmake ..
PICO_SDK_PATH is /home/pi/rob311/mbot-omni-firmware/lib/pico-sdk
PICO platform is rp2040.
Build type is Release
PICO target board is pico.
Using board configuration from /home/pi/rob311/mbot-omni-firmware/lib/pico-sdk/s
rc/boards/include/boards/pico.h
TinyUSB available at /home/pi/rob311/mbot-omni-firmware/lib/pico-sdk/lib/tinyusb
/src/portable/raspberrypi/rp2040; enabling build support for USB.
-- Configuring done
-- Generating done
-- Build files have been written to: /home/pi/rob311/mbot-omni-firmware/build
pi@neurobionics:~/rob311/mbot-omni-firmware/build$
```

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



```
67.194.33.248 - PuTTY
GNU nano 5.4 picoload
VOLUME="$1"

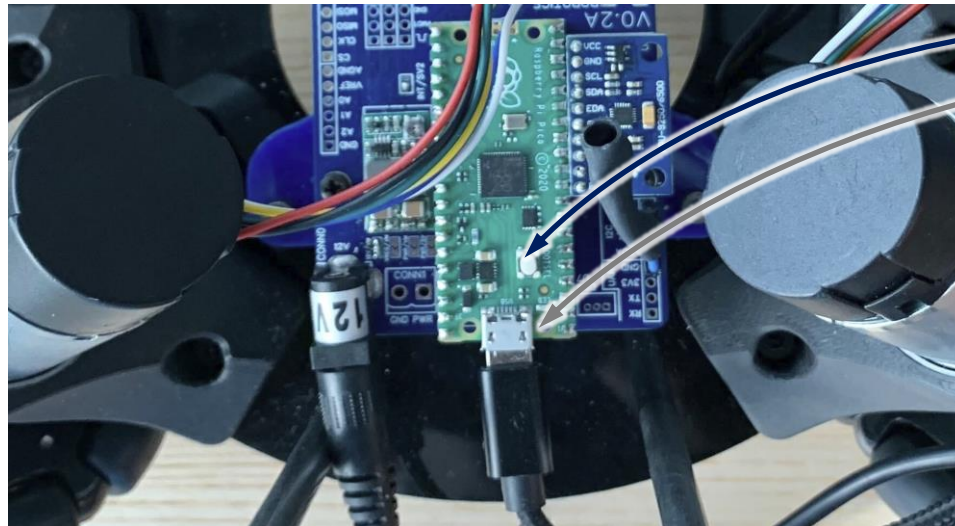
sudo mount "$VOLUME" /mnt/pico
sudo cp /home/$USER/rob311/mbot-omni-firmware/build/src/bballbot.uf2 /mnt/pico/
ls /mnt/pico
sudo sync

echo "Loaded UF2!"

^G Help      ^O Write Out  ^W Where Is   ^K Cut        ^T Execute    ^C Location
^X Exit      ^R Read File  ^\ Replace    ^U Paste      ^J Justify    ^_ Go To Line
```


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



Press this

While inserting
this

Then release
button

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

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

```
pi@neurobionics:~/rob311$ ./picoload /dev/sda1
bballbot.uf2  INDEX.HTM  INFO_UF2.TXT
Loaded UF2!
pi@neurobionics:~/rob311$
```

- 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`

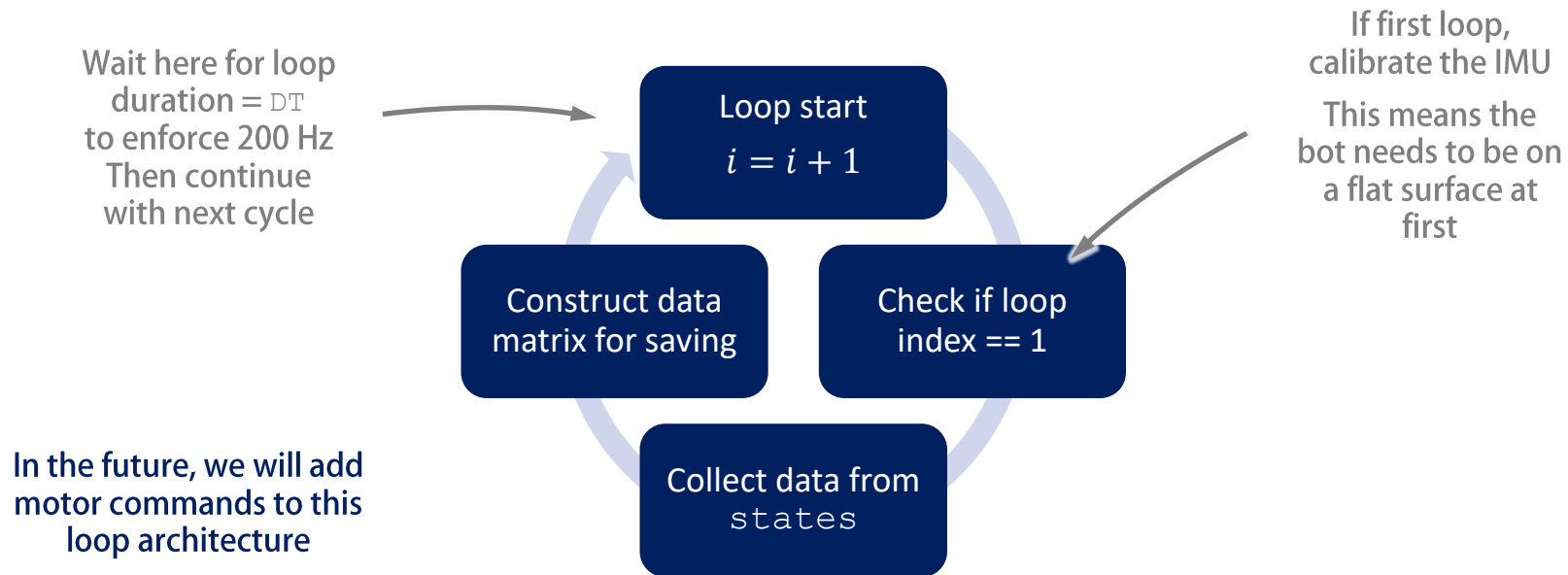
Needs to match /dev

```
67.194.33.248 - PuTTY
pi@neurobionics:/dev$ ls
autofs          hwrng          mmcblk0p2      ram6            stdin          tty24          tty42          tty60          vcs4           video10
block           initctl        mqueue         ram7            stdout         tty25          tty43          tty61          vcs5           video11
bsg             input         net            ram8            tty            tty26          tty44          tty62          vcs6           video12
btrfs-control  kmsg          null           ram9            tty0           tty27          tty45          tty63          vcsa           video13
bus             log           port           random          tty1           tty28          tty46          tty7           vcsa1          video14
cachefiles     loop0         ppp            raw             tty10          tty29          tty47          tty8           vcsa2          video15
cec0            loop1         ptmx           rfcill          tty11          tty3           tty48          tty9           vcsa3          video16
cec1            loop2         pts            rpivid-h264mem tty12          tty30          tty49          ttyAMA0        vcsa4          video18
char            loop3         ram0           rpivid-hevcmmem tty13          tty31          tty5           ttyprintk      vcsa5          watchdog
console        loop4         ram1           rpivid-intcmem  tty14          tty32          tty50          uhid           vcsa6          watchdog0
cuse           loop5         ram10          rpivid-vp9mem   tty15          tty33          tty51          uinput        vcsu-cma       zero
disk           loop6         ram11          sdb             tty16          tty34          tty52          urandom        vcsu
dma_heap       loop7         ram12          sdb1            tty17          tty35          tty53          v4l            vcsu1
dri            loop-control  ram13          serial1         tty18          tty36          tty54          vchiq          vcsu2
fd             mapper        ram14          sg0             tty19          tty37          tty55          vcio           vcsu3
full           media0        ram15          shm             tty2           tty38          tty56          vc-mem         vcsu4
fuse           media1        ram2           snd             tty20          tty39          tty57          vcs            vcsu5
gpiochip0      mem           ram3           spidev0.0       tty21          tty4           tty58          vcs1           vcsu6
gpiochip1      mmcblk0       ram4           spidev0.1       tty22          tty40          tty59          vcs2           vga_arbiter
gpiomem        mmcblk0p1     ram5           stderr          tty23          tty41          tty6           vcs3           vchi
pi@neurobionics:/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)



Reading Data From Your RPi

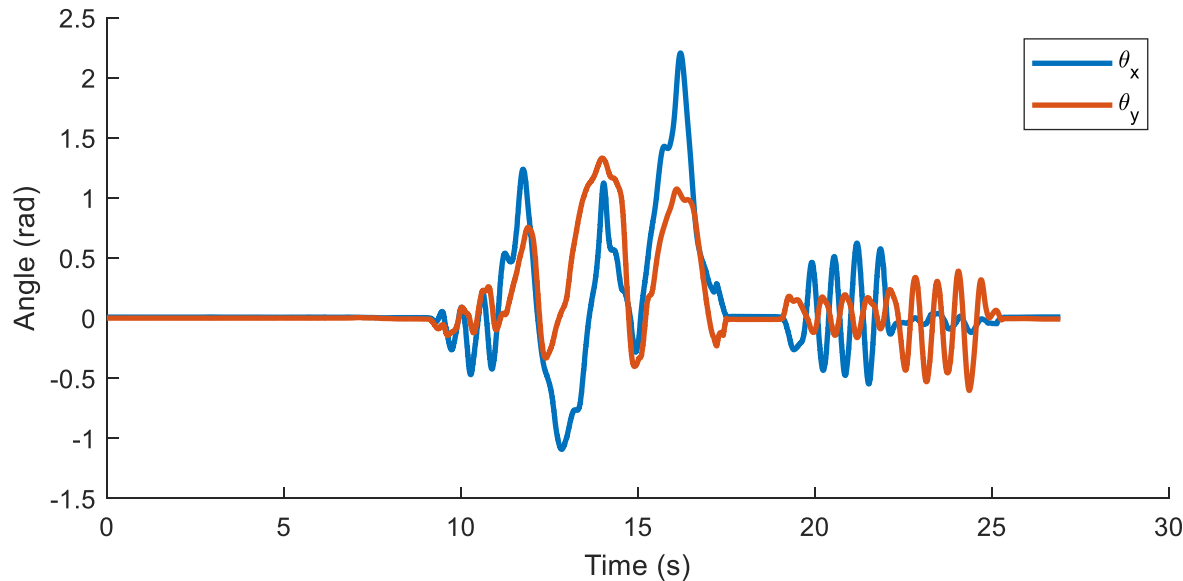
- To collect IMU data, first download files from Canvas\Labs\Lab 8
- You will need
 - `Datalogger.py`
 - `ROB311_sensor_read_demo.py`
 - `ROB311_ball_bot_data_analysis.m`
- Add the Python files to the `ballbot-omni-app` folder on your RPi
- Run the python script using `python ROB311_sensor_read_demo.py`
- 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`

This will
show the trial
number you
entered



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



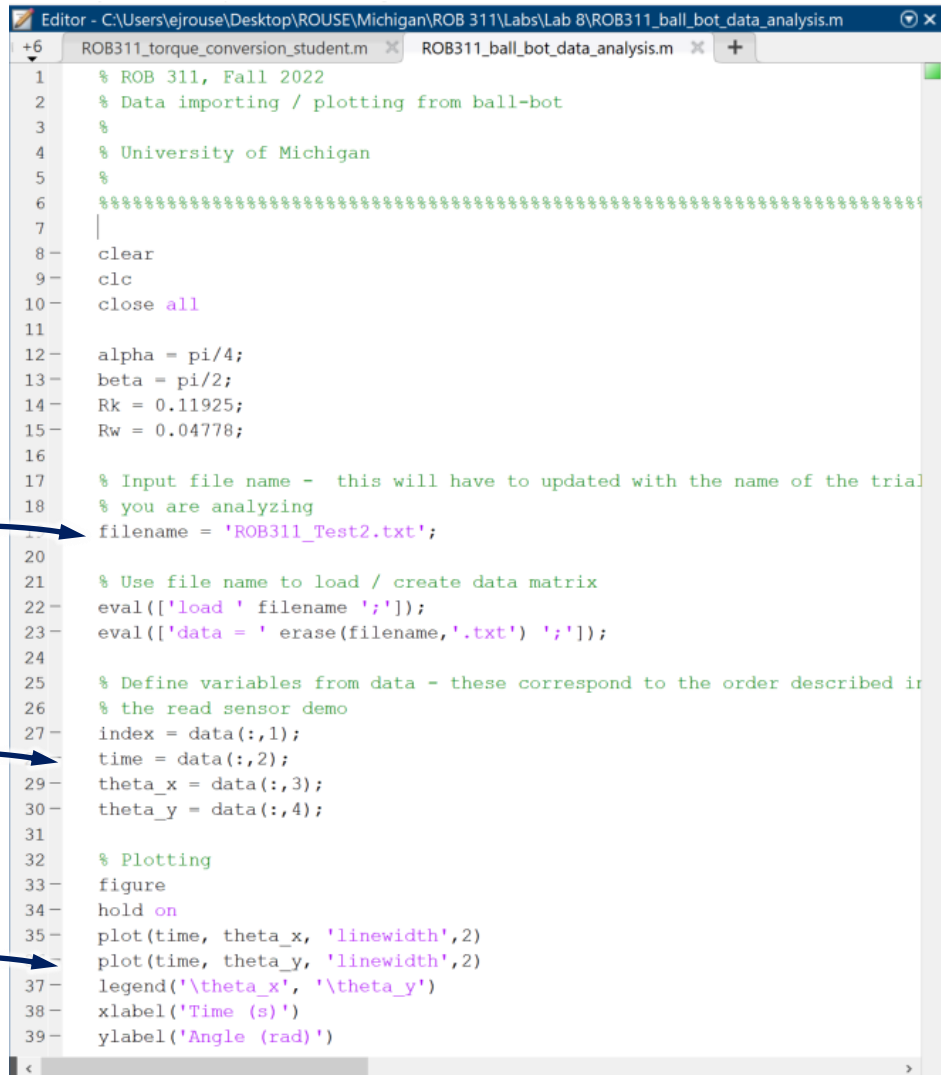
Reading Data From Your RPi

- 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



```
1 % ROB 311, Fall 2022
2 % Data importing / plotting from ball-bot
3 %
4 % University of Michigan
5 %
6 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
7 |
8 clear
9 clc
10 close all
11
12 alpha = pi/4;
13 beta = pi/2;
14 Rk = 0.11925;
15 Rw = 0.04778;
16
17 % Input file name - this will have to be updated with the name of the trial
18 % you are analyzing
19 filename = 'ROB311_Test2.txt';
20
21 % Use file name to load / create data matrix
22 eval(['load ' filename ';' ]);
23 eval(['data = ' erase(filename, '.txt') ';' ]);
24
25 % Define variables from data - these correspond to the order described in
26 % the read sensor demo
27 index = data(:,1);
28 time = data(:,2);
29 theta_x = data(:,3);
30 theta_y = data(:,4);
31
32 % Plotting
33 figure
34 hold on
35 plot(time, theta_x, 'linewidth',2)
36 plot(time, theta_y, 'linewidth',2)
37 legend('\theta_x', '\theta_y')
38 xlabel('Time (s)')
39 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 (ψ)
- 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



```
1 import numpy as np
2
3 mo_cmds_dtype = np.dtype([
4     ("kill", np.double),
5     ("motor_1_duty", np.double),
6     ("motor_2_duty", np.double),
7     ("motor_3_duty", np.double)
8 ])
9
10 mo_states_dtype = np.dtype([
11     ("timestep", np.double),
12     ("theta_roll", np.double),
13     ("theta_pitch", np.double),
14     ("theta_yaw", np.double),
15     ("dpsi_1", np.double),
16     ("dpsi_2", np.double),
17     ("dpsi_3", np.double),
18     ("psi_1", np.double),
19     ("psi_2", np.double),
20     ("psi_3", np.double)
21 ])
22
23 mo_pid_params_dtype = np.dtype([
24     ("theta_kp", np.double),
25     ("theta_ki", np.double),
26     ("theta_kd", np.double)
27 ])
```

Reading Data From Your RPi

```
C:\Users\ejrouse\Desktop\ROUSE\Michigan\ROB 311\Labs\Lab 8\ROB311_sensor_read_demo.py - Sublime Text (...
File Edit Selection Find View Goto Tools Project Preferences Help

ROB311_sensor_read_demo.py x message_defs.py MIT_Knee_Complex_State_Machine.py + ▾

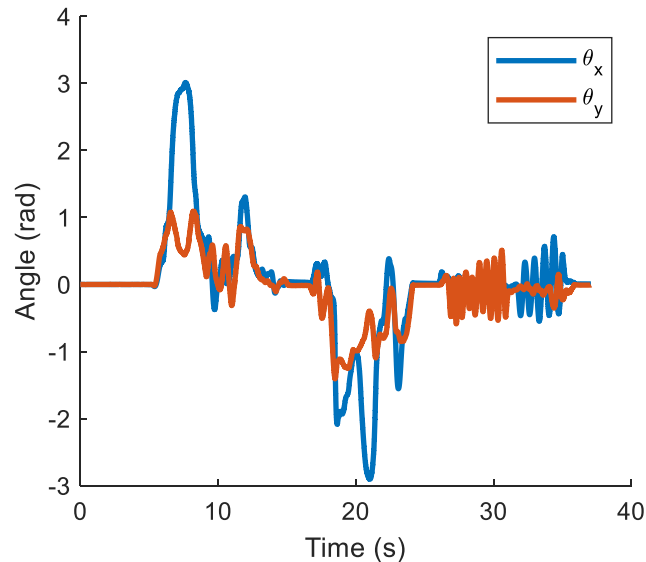
199
200 # Init serial
201 serial_read_thread = Thread(target = SerialProtocol.read_loop, args=(ser_dev,), daemon=True)
202 serial_read_thread.start()
203
204 # Local structs
205 commands = np.zeros(1, dtype=mo_cmds_dtype)[0]
206 states = np.zeros(1, dtype=mo_states_dtype)[0]
207
208 commands['kill'] = 0.0
209
210 dpsl = np.zeros((3, 1))
211
212 # Time for comms to sync
213 time.sleep(1.0)
214
215 ser_dev.send_topic_data(101, commands)
216
217 print('Beginning program!')
218 i = 0
219 cal_flag = 0
220
221 for t in SoftRealtimeLoop(dt=DT, report=True):
222     if i == 0 and cal_flag == 0:
223         print('Calibrating...')
224         cal_flag = 1;
225     try:
226         states = ser_dev.get_cur_topic_data(121)[0]
227     except KeyError as e:
228         continue
229     if i == 0:
230         print('Finished calibration\nStarting loop...')
231         t_start = time.time()
232         i = i + 1
233         t_now = time.time() - t_start
234
235         dpsl[0] = states['dpsi_1']
236         dpsl[1] = states['dpsi_2']
237         dpsl[2] = states['dpsi_3']
238
239         ser_dev.send_topic_data(101, commands)
240
241         # Define variables for saving / analysis here - below you can create variables from the avail
242         theta_x = (states['theta_roll'])
243         theta_y = (states['theta_pitch'])
244
245         # Construct the data matrix for saving - you can add more variables by replicating the form
246         data = [i] + [t_now] + [theta_x] + [theta_y]
247         dl.appendData(data)
248
249         print("Saving data...")
250         dl.writeOut()
251         print("Resetting Motor Commands.")
252         commands['kill'] = 1.0
253         commands['motor_1_duty'] = 0.0
254         commands['motor_2_duty'] = 0.0
255         commands['motor_3_duty'] = 0.0
256         ser_dev.send_topic_data(101, commands)
257
```

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

