

Robot Workflow

ROB 102: Introduction to AI & Programming

Lab Session 2

2021/09/17

Today...

1. Some common issues for Project 0
2. Code structure for Project 1
3. Intro to the MBot-Omni
4. MBot-Omni workflow
5. Time to work on Project 1 (Part 1)

Find your teammate and grab your robot!

Admin

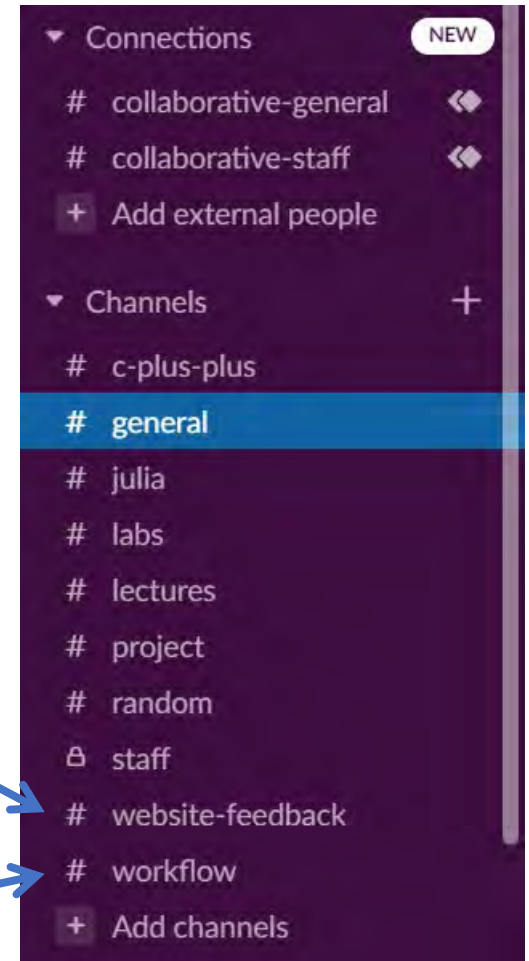
Project 0 due **Monday, October 4th @ 11:59 PM**

Project 1 due **Monday, October 4th @ 11:59 PM**

No quiz next week.

Post here if you find typos or
other issues with the website

Post here if you have any issues
with compiling and running code,
Git, Docker, etc.



Some common issues

- Compile and run your code in Docker! (`./docker_run.sh`)

Docker terminal →

```
root@0aa2bbca2929:/# cd code/src/hello_world/  
root@0aa2bbca2929:/code/src/hello_world# ls  
hello_world.cpp  
root@0aa2bbca2929:/code/src/hello_world#
```

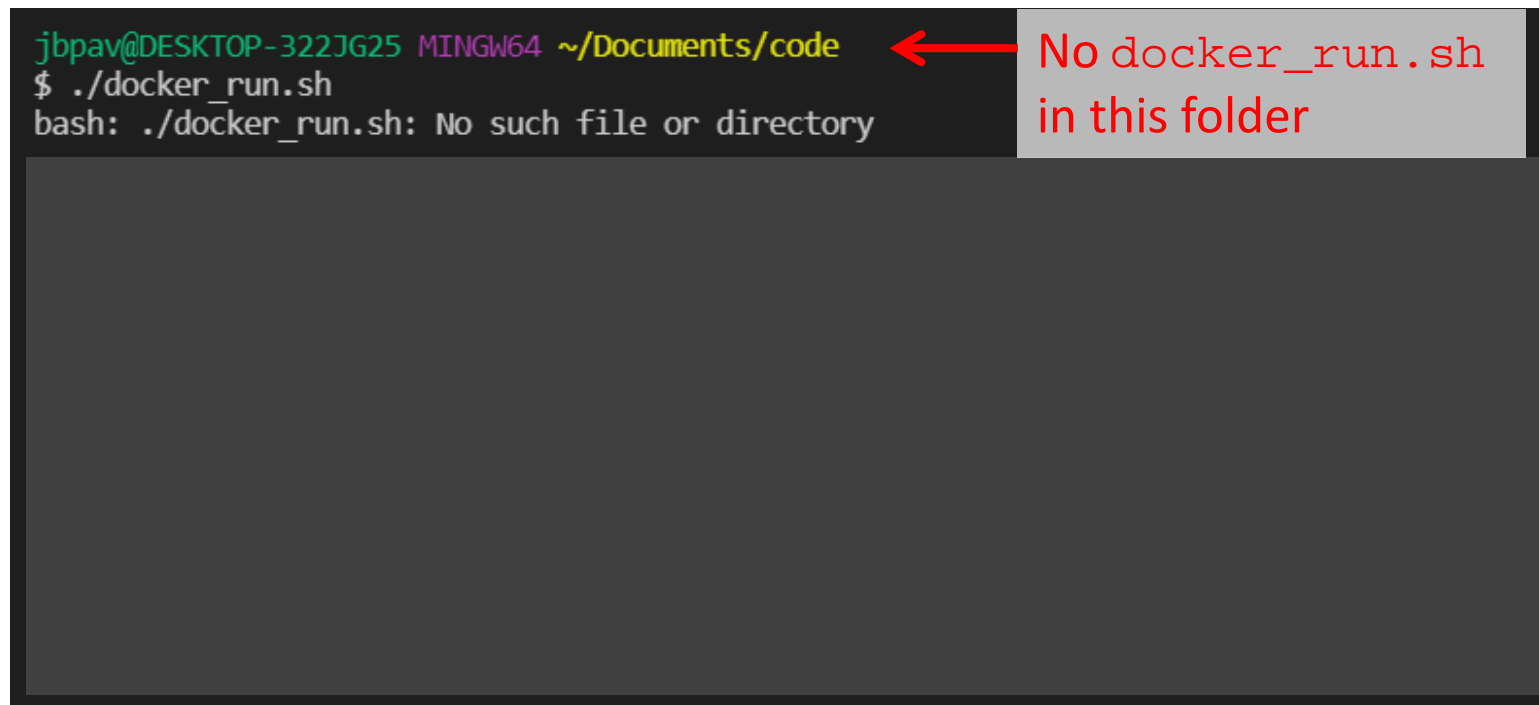
```
jbpav@DESKTOP-322JG25 MINGW64 ~/Documents/rob102-code/p0-intro-cpp-janapavlasek  
(main)  
$
```



Not Docker terminal!

Some common issues

- Compile and run your code in Docker!
- Make sure you're in the right folder when you're running commands.



```
jbpav@DESKTOP-322JG25 MINGW64 ~/Documents/code  
$ ./docker_run.sh  
bash: ./docker_run.sh: No such file or directory
```

← No docker_run.sh in this folder

The image shows a terminal window with a dark background. The prompt is 'jbpav@DESKTOP-322JG25 MINGW64 ~/Documents/code'. The user enters '\$./docker_run.sh' and the output is 'bash: ./docker_run.sh: No such file or directory'. A red arrow points from the text 'No docker_run.sh in this folder' to the path '~/Documents/code' in the prompt.

Some common issues

- Compile and run your code in Docker!
- Make sure you're in the right folder when you're running commands.

```
jbpav@DESKTOP-322JG25 MINGW64 ~/Documents/code  
$ ./docker_run.sh  
bash: ./docker_run.sh: No such file or directory  
  
jbpav@DESKTOP-322JG25 MINGW64 ~/Documents/code  
$ cd p0-intro-cpp-janapavlsek/src/  
  
jbpav@DESKTOP-322JG25 MINGW64 ~/Documents/code/p0-intro-cpp-janapavlsek/src (main)  
$ ./docker_run.sh  
bash: ./docker_run.sh: No such file or directory
```

← No docker_run.sh in this folder

← Not in here either...

Some common issues

- Compile and run your code in Docker!
- Make sure you're in the right folder when you're running commands.

```
jbpav@DESKTOP-322JG25 MINGW64 ~/Documents/code  
$ ./docker_run.sh  
bash: ./docker_run.sh: No such file or directory  
  
jbpav@DESKTOP-322JG25 MINGW64 ~/Documents/code  
$ cd p0-intro-cpp-janapavlasek/src/  
  
jbpav@DESKTOP-322JG25 MINGW64 ~/Documents/code/p0-intro-cpp-janapavlasek/src (main)  
$ ./docker_run.sh  
bash: ./docker_run.sh: No such file or directory  
  
jbpav@DESKTOP-322JG25 MINGW64 ~/Documents/code/p0-intro-cpp-janapavlasek/src (main)  
$ cd ..  
  
jbpav@DESKTOP-322JG25 MINGW64 ~/Documents/code/p0-intro-cpp-janapavlasek (main)  
$ ./docker_run.sh  
root@1b839c02e08e:/#
```

← No docker_run.sh
in this folder

← Not in here
either...

The script is in
the root of the
repository!



Some common issues

- Compile and run your code in Docker!
- Make sure you're in the right folder when you're running commands.

Docker terminal ↗

```
root@1b839c02e08e:/# g++ hello_world.cpp -o hello_world
g++: error: hello_world.cpp: No such file or directory
g++: fatal error: no input files
compilation terminated.
```

No hello_world.cpp!

Some common issues

- Compile and run your code in Docker!
- Make sure you're in the right folder when you're running commands.

Docker terminal →

```
root@1b839c02e08e:/# g++ hello_world.cpp -o hello_world
g++: error: hello_world.cpp: No such file or directory
g++: fatal error: no input files
compilation terminated.
root@1b839c02e08e:/# cd code/src/hello_world/
root@1b839c02e08e:/code/src/hello_world# ls
hello_world.cpp
```

No hello_world.cpp!

ls and cd are
your friends!

Some common issues

- Compile and run your code in Docker!
- Make sure you're in the right folder when you're running commands.

Docker terminal →

```
root@1b839c02e08e:/# g++ hello_world.cpp -o hello_world
g++: error: hello_world.cpp: No such file or directory
g++: fatal error: no input files
compilation terminated.
root@1b839c02e08e:/# cd code/src/hello_world/
root@1b839c02e08e:/code/src/hello_world# ls
hello_world.cpp
root@1b839c02e08e:/code/src/hello_world# g++ hello_world.cpp -o hello_world
root@1b839c02e08e:/code/src/hello_world# ls
hello_world  hello_world.cpp
root@1b839c02e08e:/code/src/hello_world# ./hello_world
Hello World! My name is Jana
root@1b839c02e08e:/code/src/hello_world#
```

No hello_world.cpp !

ls and cd are
your friends!

Compile before
running again

The working directory

Some common issues

- Compile and run your code in Docker!
- Make sure you're in the right folder when you're running commands.
- Go through the [Lab 1 slides](#) and the [Project 0 instructions](#) again if something isn't working.

Project 1: Code Structure

main 1 branch 0 tags Go to file Add file Code Use this template

janapavlashek Add license file

build	Add template for wall follower
include	Add template for wall follower
lib	Add template for wall follower
omnibot_msgs	Add template for wall follower
src	Add reference to tak number
.gitignore	Add template for wall follower
CMakeLists.txt	Add template for wall follower
LICENSE.txt	Add license file
README.md	Add template for wall follower

Source code (Modify here)

main wall-follower / src /

janapavlashek Add reference to tak number

common

drive_safe.cpp

drive_square.cpp

wall_follower.cpp

Source code for parts 1-3

5 days ago

Project 1: Code

main 1 branch 0 tags		
janapavlassek Add license file		
build	Add template for wall follow	
include	Add template for wall follow	
lib	Add template for wall follow	
omnibot_msgs	Add template for wall follow	
src	Add reference to tak numbe	
.gitignore	Add template for wall follow	
CMakeLists.txt	Add template for wall follow	
LICENSE.txt	Add license file	
README.md	Add template for wall follow	

```
1  #include <iostream>
2  #include <cmath>
3
4  #include <wall_follower/common/utils.h>
5  #include <wall_follower/common/drive.h>
6
7
8  int main(int argc, const char *argv[])
9  {
10     /**
11      * TODO: (P1.1.1) Write code to make the robot drive in a square. Then,
12      * modify your code so that the robot drives in a square 3 times.
13      *
14      * HINT: A function to send velocity commands to the robot is provided. To
15      * use it, use the following code:
16      *
17      *     drive(vx, vy, wz);
18      *
19      * Replace vx, vy, and wz with the velocity in the x direction (vx), y
20      * direction (vy), and the angular velocity (wz). You can also use this code:
21      *
22      *     sleepFor(secs);
23      *
24      * to sleep for "secs" seconds (replace with desired number of seconds).
25      */
26
27     // Stop the robot.
28     drive(0, 0, 0);
29     return 0;
30 }
```

← Headers let us include code in other files

← Write your code here

Project 1: Code

main 1 branch 0 tags		
janapavlashek Add license file		
build	Add template for wall follower	
include	Add template for wall follower	
lib	Add template for wall follower	
omnibot_msgs	Add template for wall follower	5 days ago
src	Add reference to tak number	4 days ago
.gitignore	Add template for wall follower	5 days ago
CMakeLists.txt	Add template for wall follower	5 days ago
LICENSE.txt	Add license file	4 days ago
README.md	Add template for wall follower	5 days ago

include/wall_follower/common/drive.h

```
1  #ifndef WALL_FOLLOWER_COMMON_DRIVE_H
2  #define WALL_FOLLOWER_COMMON_DRIVE_H
3
4  #include <lcm/lcm-cpp.hpp>
5
6  #define MULTICAST_URL "udpm://239.255.76.67:7667?ttl=2"
7  #define MBOT_MOTOR_COMMAND_CHANNEL "MBOT_MOTOR_COMMAND"
8
9  static lcm::LCM lcmInstance(MULTICAST_URL);
10
11 void drive(const float vx, const float vy, const float wz);
12
13 #endif // WALL_FOLLOWER_COMMON_DRIVE_H
```

The function is defined
in a *header file*.

Project 1: Code



main 1 branch 0 tags Go to file Add file

janapavlassek Add license file

- build Add template for wall follower
- include Add template for wall follower
- lib Add template for wall follower
- omnibot_msgs Add template for wall follower
- src Add reference to tak number
- .gitignore Add template for wall follower
- CMakeLists.txt Add template for wall follower
- LICENSE.txt Add license file
- README.md Add template for wall follower

include/wall_follower/common/drive.h

```
1 #ifndef WALL_FOLLOWER_COMMON_DRIVE_H
2 #define WALL_FOLLOWER_COMMON_DRIVE_H
3
4 #include <lcm/lcm-cpp.hpp>
5
6 #define MULTICAST_URL "udp://239.255.76.67:7667?ttl=2"
7 #define MBOT_MOTOR_COMMAND_CHANNEL "MBOT_MOTOR_COMMAND"
8
9 static lcm::LCM lcmInstance(MULTICAST_URL);
10
11 void drive(const float vx, const float vy, const float wz);
12
13 #endif // WALL_FOLLOWER_COMMON_DRIVE_H
```

src/common/drive.cpp

```
1 #include <omnibot_msgs/omni_motor_command_t.hpp>
2
3 #include <wall_follower/common/utils.h>
4 #include <wall_follower/common/drive.h>
5
6 void drive(const float vx, const float vy, const float wz)
7 {
8     omnibot_msgs::omni_motor_command_t cmd;
9     cmd.utime = getTimeMicro();
10     cmd.vx = vx;
11     cmd.vy = vy;
12     cmd.wz = wz;
13
14     lcmInstance.publish(MBOT_MOTOR_COMMAND_CHANNEL, &cmd);
15 }
```

The function is implemented in a *source file*.

Project 1: Code

main 1 branch 0 tags		
janapavlassek Add license file		
build	Add template for wall follow	
include	Add template for wall follow	
lib	Add template for wall follow	
omnibot_msgs	Add template for wall follow	
src	Add reference to tak numbe	
.gitignore	Add template for wall follow	
CMakeLists.txt	Add template for wall follow	
LICENSE.txt	Add license file	
README.md	Add template for wall follow	

src/drive_square.cpp

```
1  #include <iostream>
2  #include <cmath>
3
4  #include <wall_follower/common/utils.h>
5  #include <wall_follower/common/drive.h>
6
7
8  int main(int argc, const char *argv[])
9  {
10     /**
11      * TODO: (P1.1.1) Write code to make the robot drive in a square. Then,
12      * modify your code so that the robot drives in a square 3 times.
13      *
14      * HINT: A function to send velocity commands to the robot is provided. To
15      * use it, use the following code:
16      *
17      *     drive(vx, vy, wz);
18      *
19      * Replace vx, vy, and wz with the velocity in the x direction (vx), y
20      * direction (vy), and the angular velocity (wz). You can also use this code:
21      *
22      *     sleepFor(secs);
23      *
24      * to sleep for "secs" seconds (replace with desired number of seconds).
25      */
26
27     // Stop the robot.
28     drive(0, 0, 0);
29     return 0;
30 }
```

By including the drive.h header, we can use the drive() function!

You will only modify SOURCE code for Project 1.

Compiling Project 1

Since we use multiple files and some external libraries in the code, we will use a tool called **CMake** (instead of raw g++ commands).

CMake finds all the code and dependencies and generates *Makefiles* to compile them. We can compile many executables at once. To compile, do:

Replace with your repository name
↓
`cd [wall-follower-dir]/build/` ← Need to be in the `build` folder
`cmake ..` ← Look for the CMake file one directory up
(in `[wall-follower-dir]/`)
`make` ← Compile the code

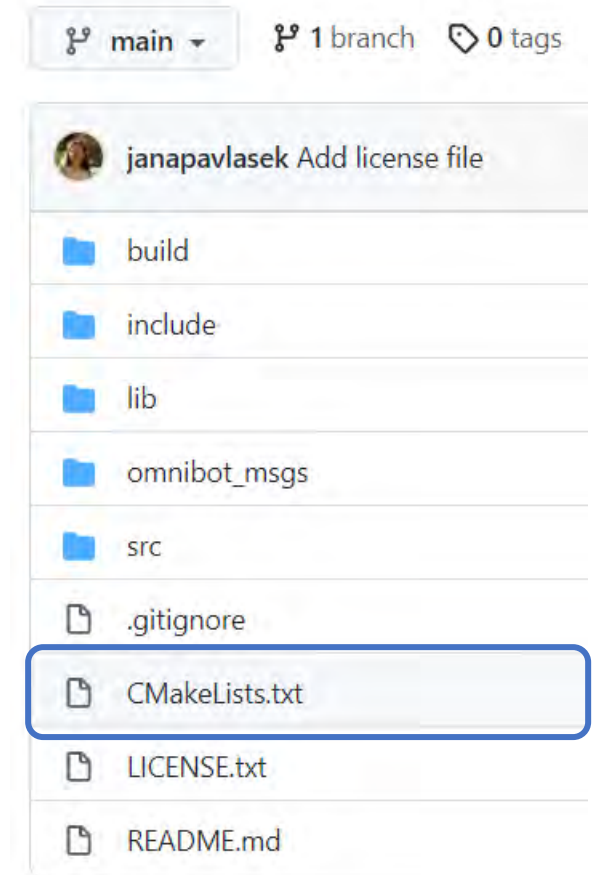
Compiling Project 1

To compile, do:

```
cd [wall-follower-dir]/build/  
cmake .. ← Reads CMakeLists.txt.  
make      Only needs to be called once.  
           Make every time you change your source code.
```

To execute, in the build folder, do:

```
./drive_square
```



The MBot-Omni



The MBot-Omni

An **Omnidrive** robot platform.

The MBot-Mini's (EECS 467, ROB 550) younger cousin.



MBot-Omni: Computing

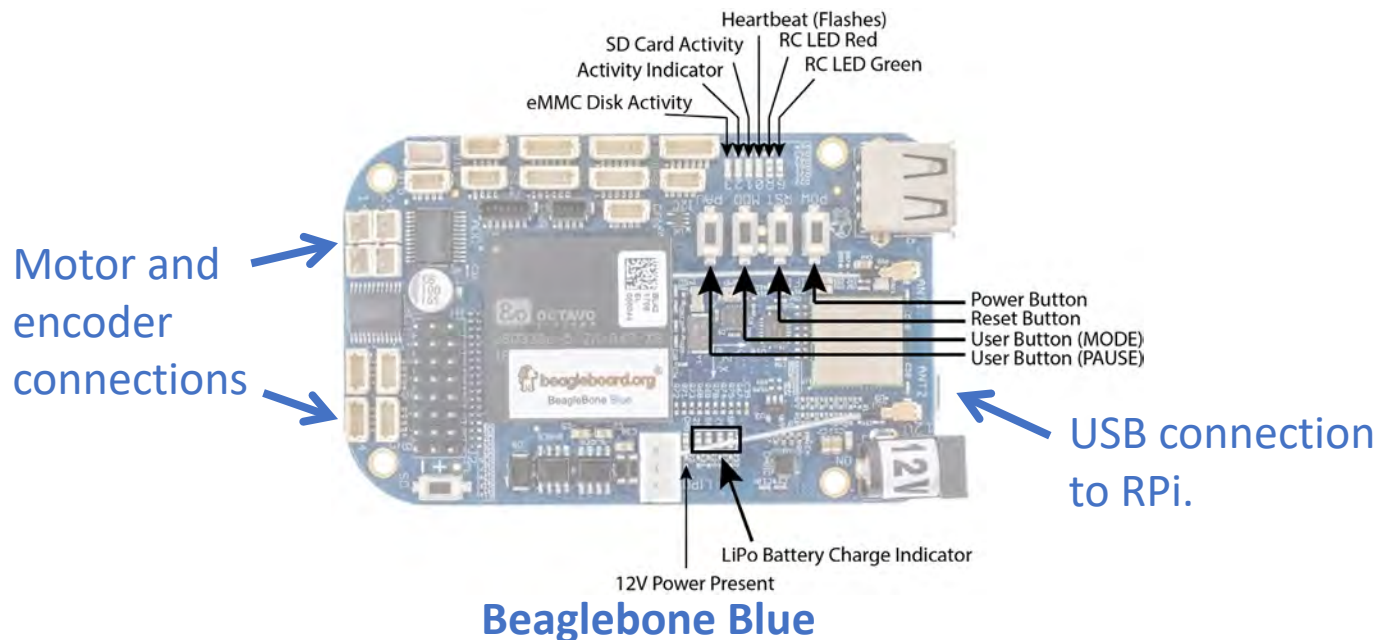
The MBot-Omni has two embedded computers: a **Raspberry Pi** and a **Beaglebone Blue**.

Sometimes we'll call them the RPi and the BBB for short.



MBot-Omni: Computing

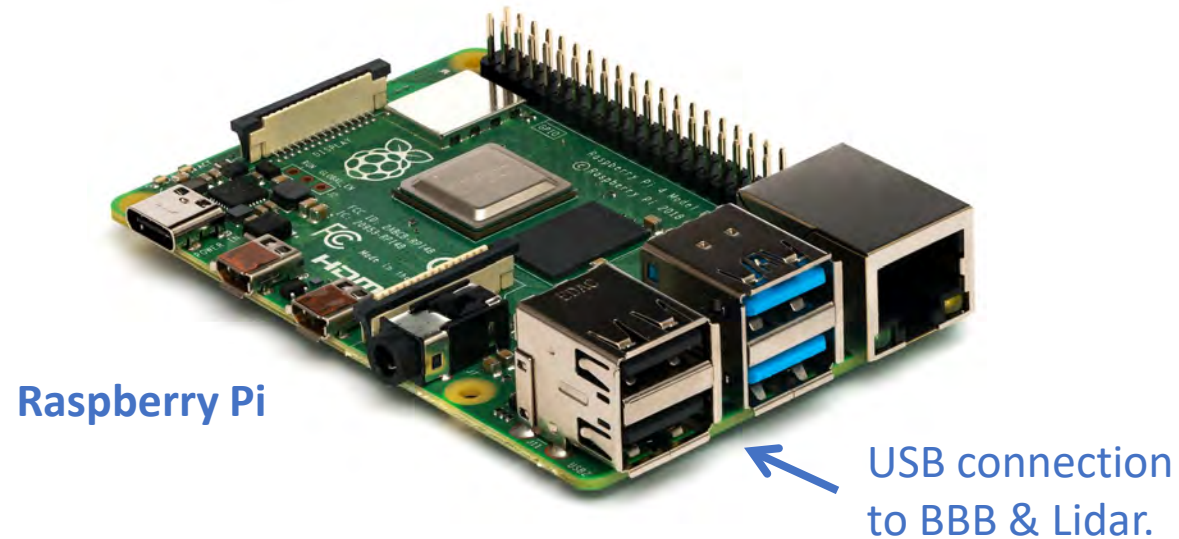
The **motors** and **encoders** are connected to the Beaglebone. It turns commands into signals to send to the wheels.



MBot-Omni: Computing

The **Lidar** is connected to the Raspberry Pi. It sends commands to the Beaglebone to move the motors.

All code in ROB 102 will run on the RPi.

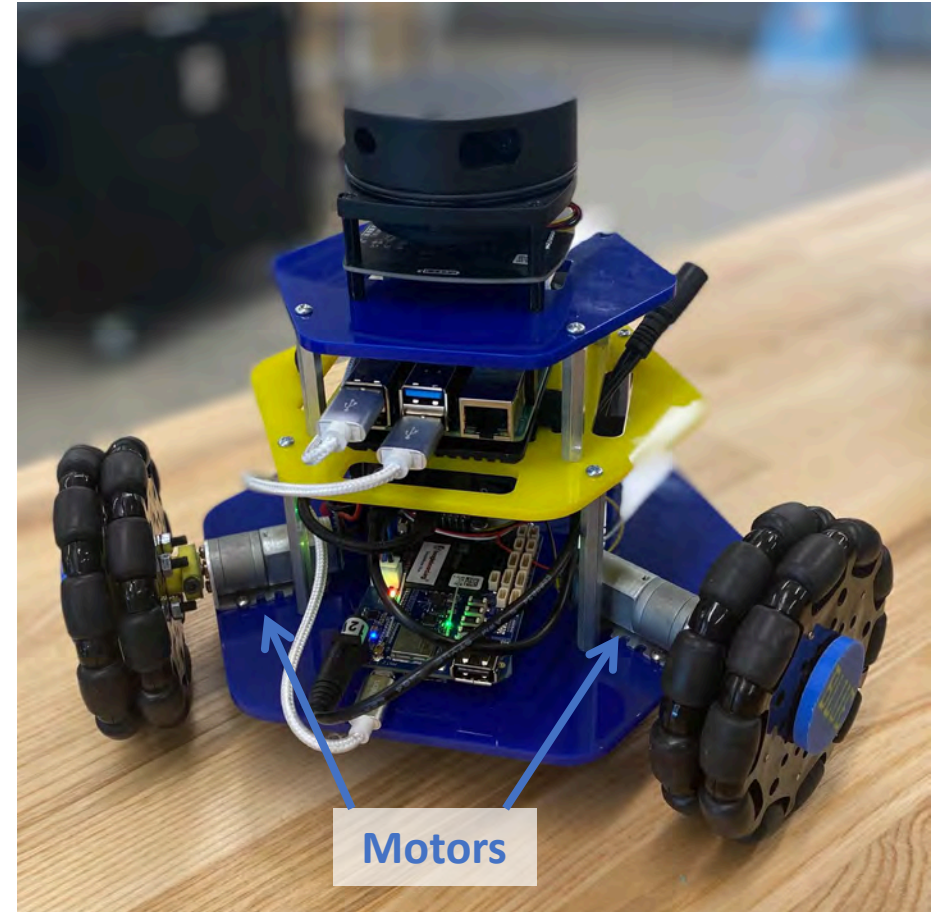


MBot-Omni: Motors & Control

The MBot-Omni has 3 motors that are connected to the Beaglebone.

The Beaglebone controls the motors with two parameters:

1. **Duty:** How fast to spin the motors (a value from 0 to 1).
2. **Direction:** Forward or backward.



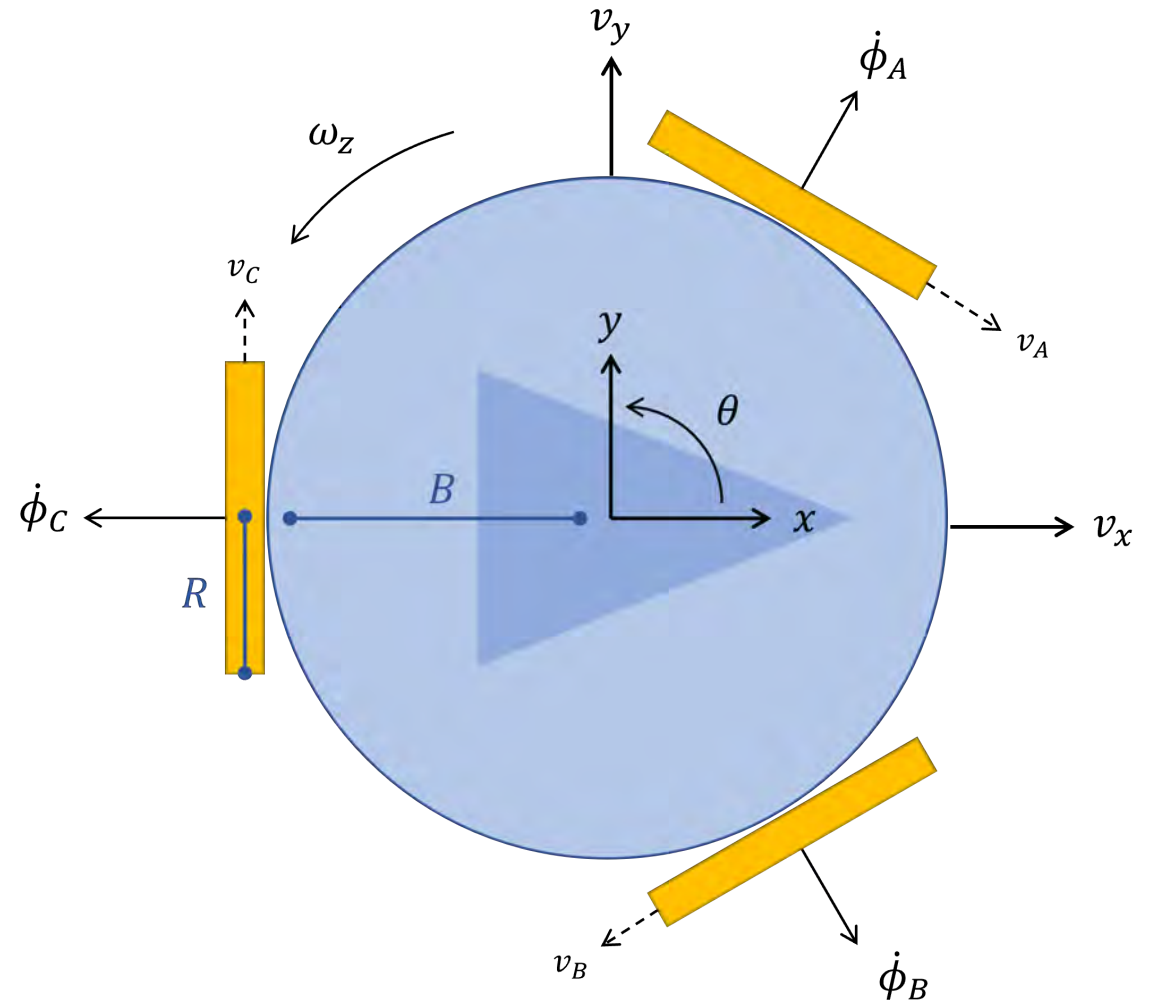
MBot-Omni: Kiwidrive Control

Since all we can control is how fast the wheels spin, it's intuitive to drive the robot by giving it velocity commands:

$$\begin{matrix} & [v_x, & v_y, & \omega_z] \\ & \uparrow & \uparrow & \uparrow \\ \text{Velocity in the x- and y-axis} & & & \text{Angular velocity} \end{matrix}$$

The Beaglebone runs code that converts velocity commands from the RPi to motor commands.

Why don't we use position commands to control the robot? (x, y, θ)



MBot-Omni: Coordinate System

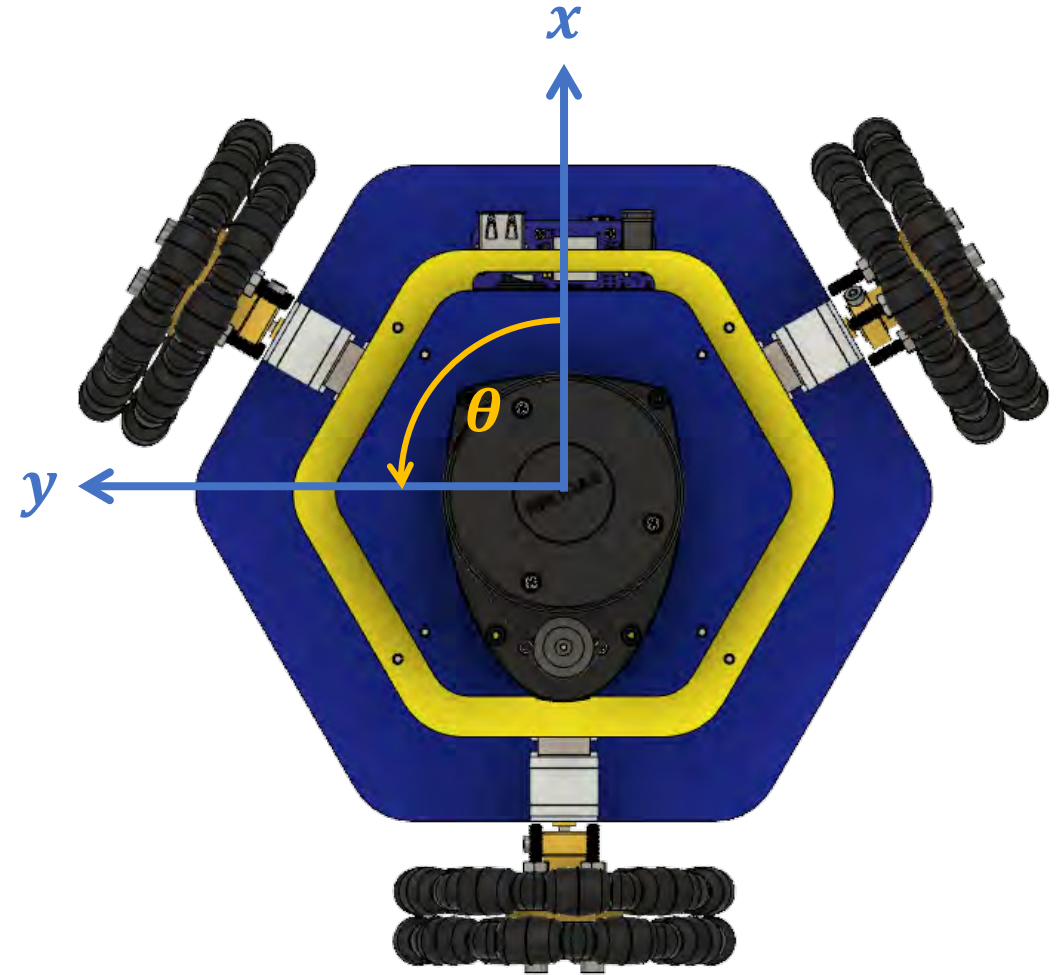
In robotics, it's very important to use a consistent coordinate system.

We choose the following convention for the MBot-Omni:

+x: forward

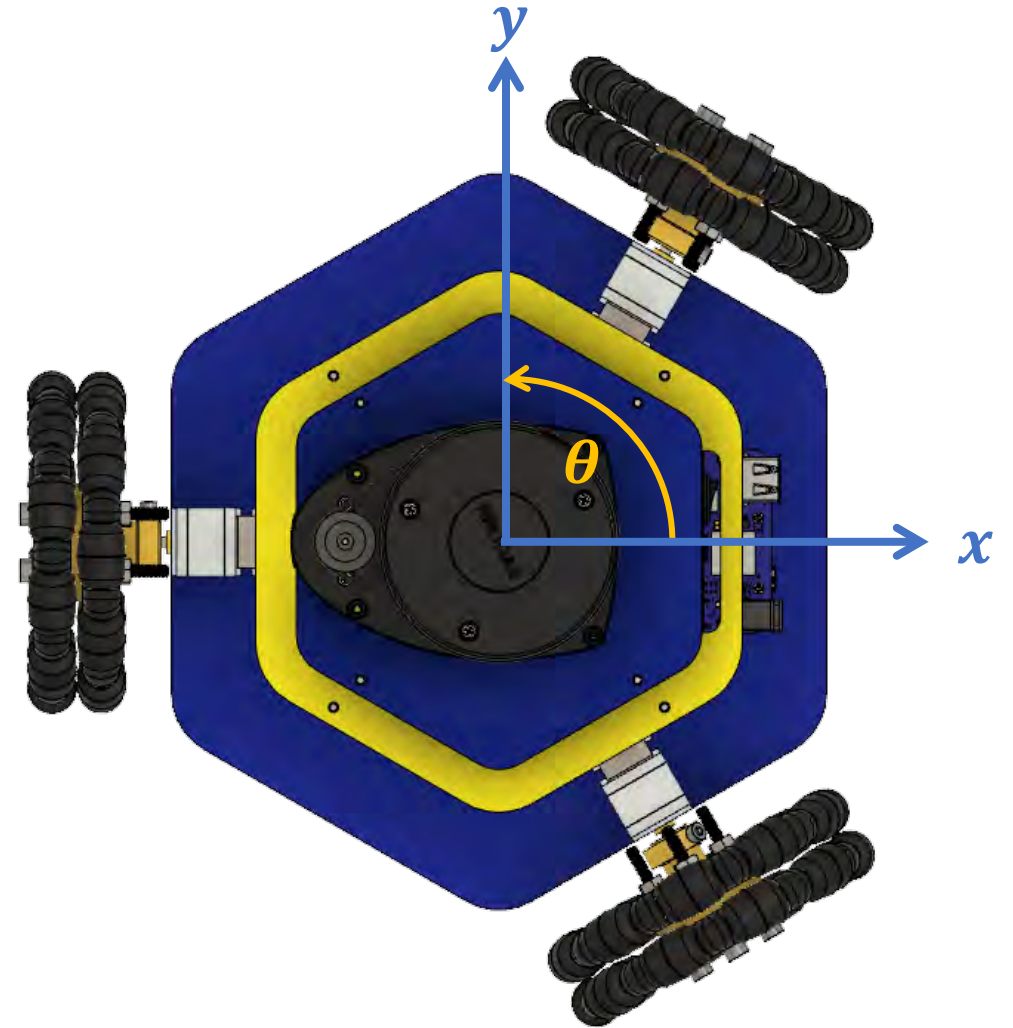
+y: left

+ θ : counterclockwise



MBot-Omni: Coordinate System

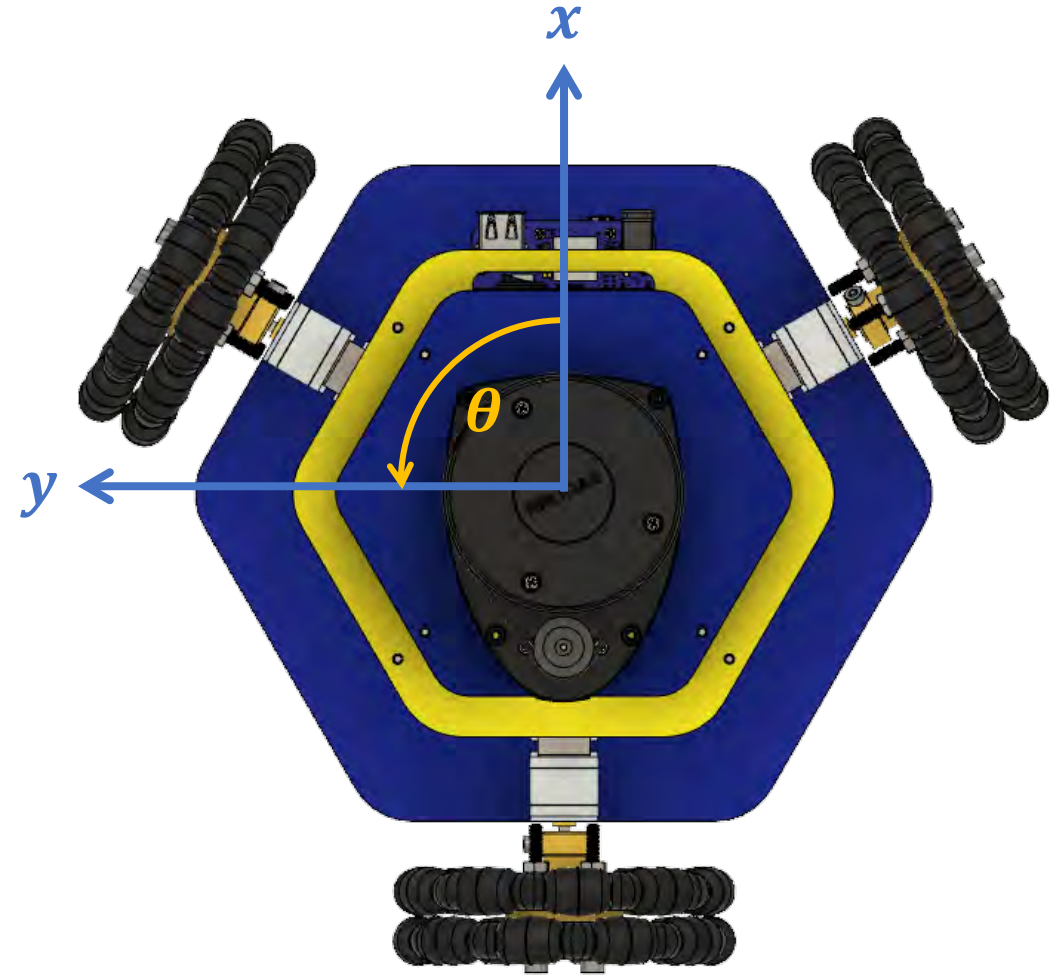
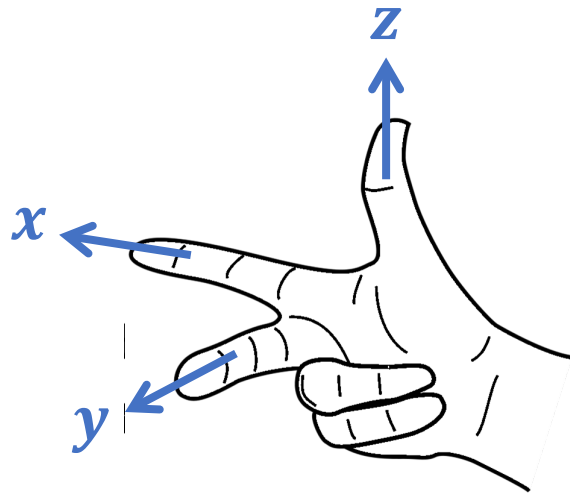
Rotating 90° makes it look like the familiar cartesian coordinate system.



MBot-Omni: Coordinate System

This is a **right-handed** coordinate system.

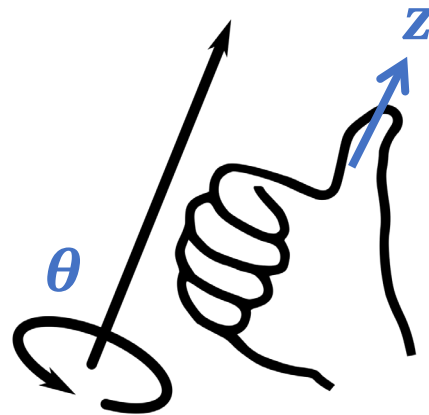
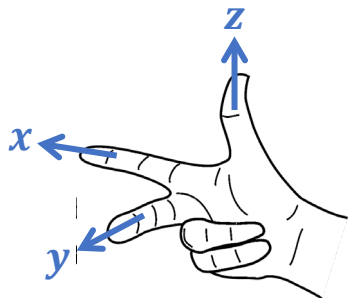
The fingers of your right hand point in the positive direction of each axis.



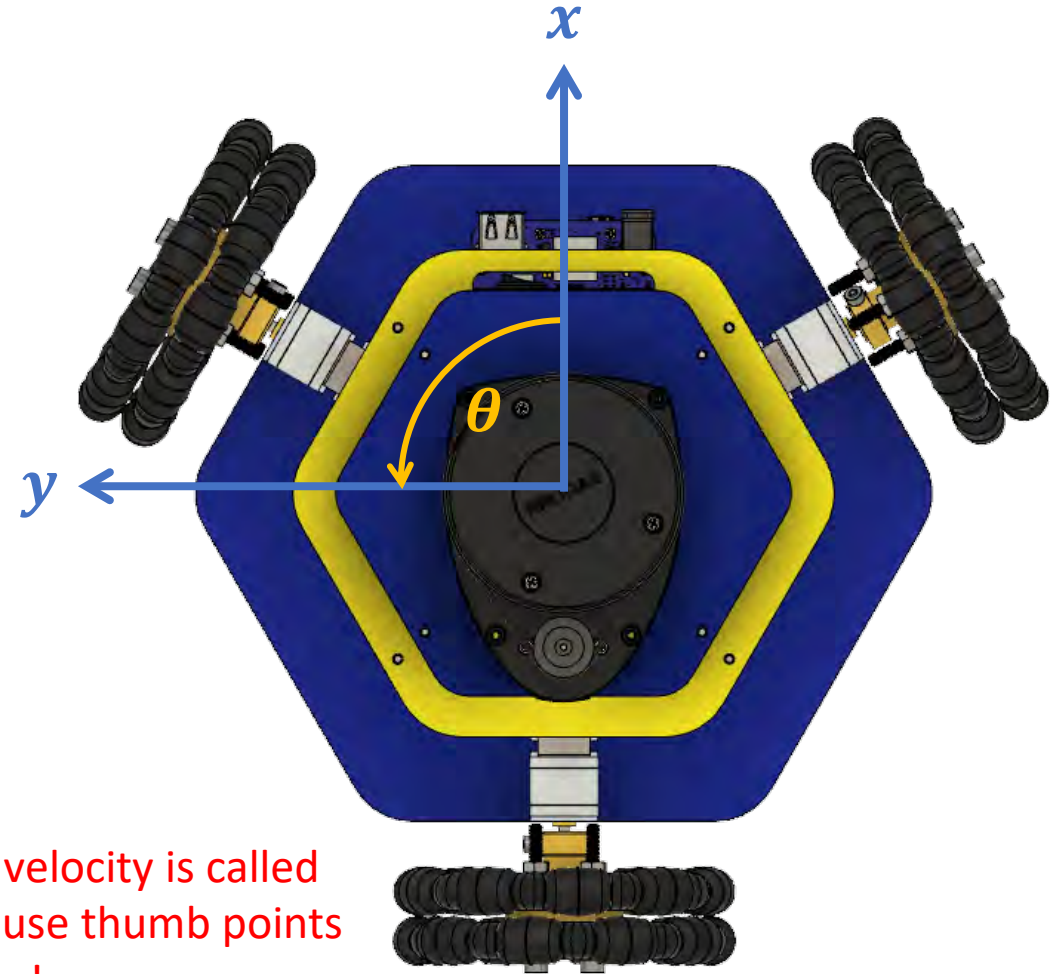
MBot-Omni: Coordinate System

This is a **right-handed** coordinate system.

When you point your thumb in the positive direction of an axis, your fingers wrap in the positive angular direction.



Angular velocity is called ω_z because thumb points towards z!

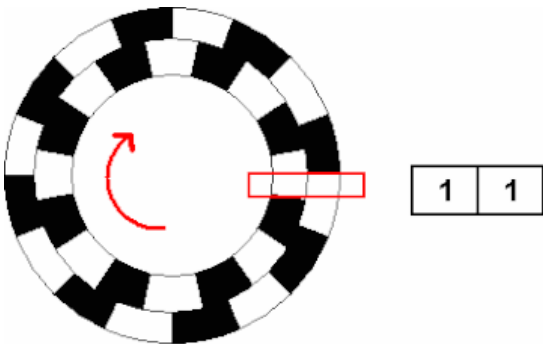


MBot-Omni: Encoders & Odometry

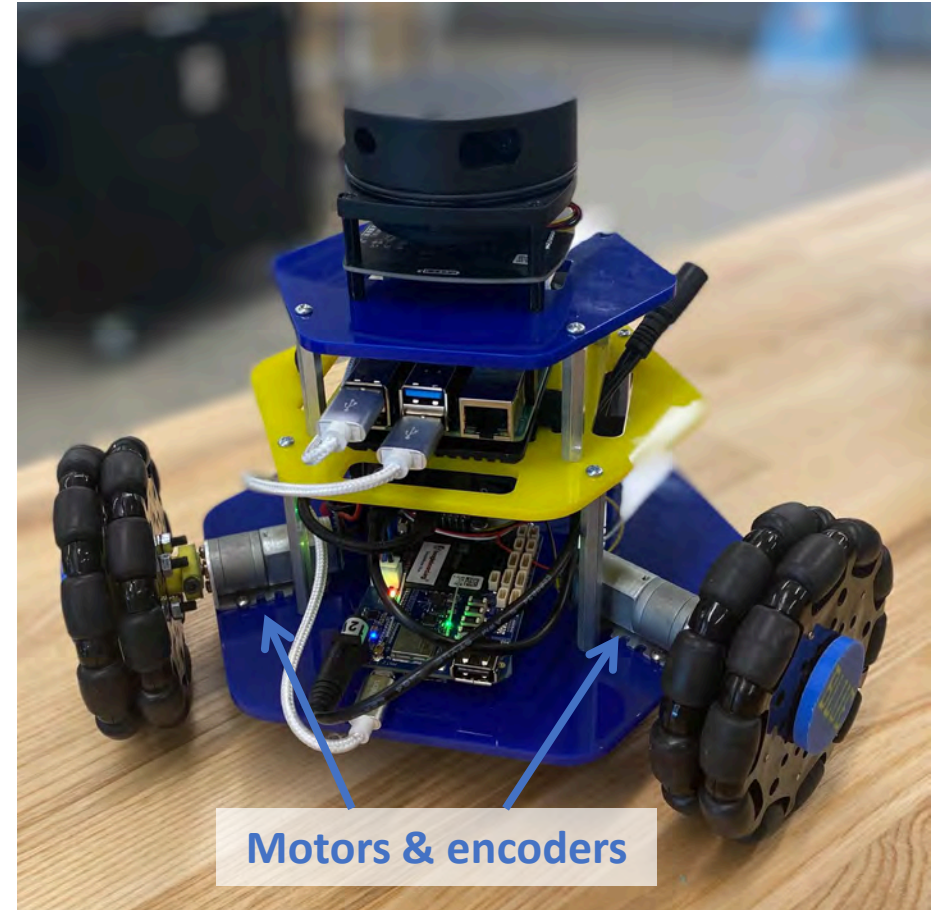
The MBot-Omni has an **encoder** on each motor which measures how much the wheel turns.

Odometry uses encoder values to measure how much the robot has moved.

The Beaglebone has code to read the encoders and perform odometry.



More on this in ROB 103
and EECS 467!

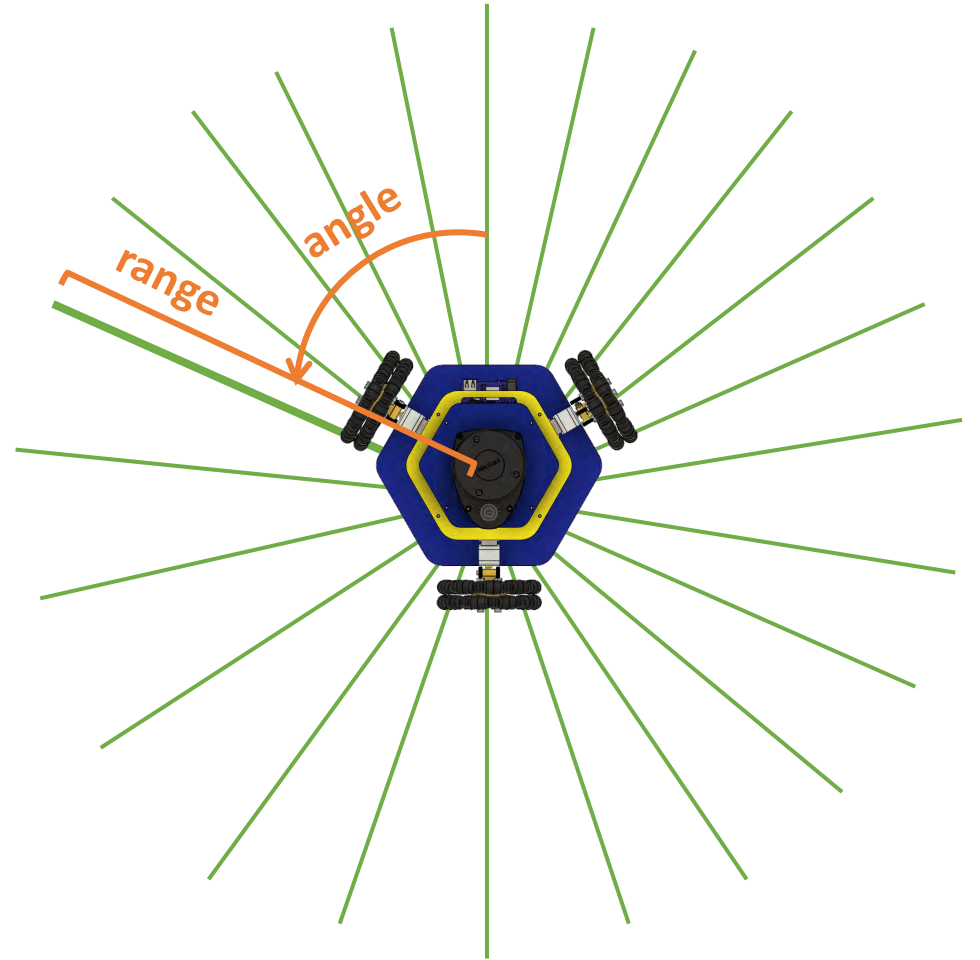


MBot-Omni: Lidar

The **Lidar** sends out a series of **rays**.

Each **scan** is a list of rays with the following data:

- range (length in meters)
- angle (in radians)
- intensity
- time of scan



MBot-Omni: Lidar & Mapping

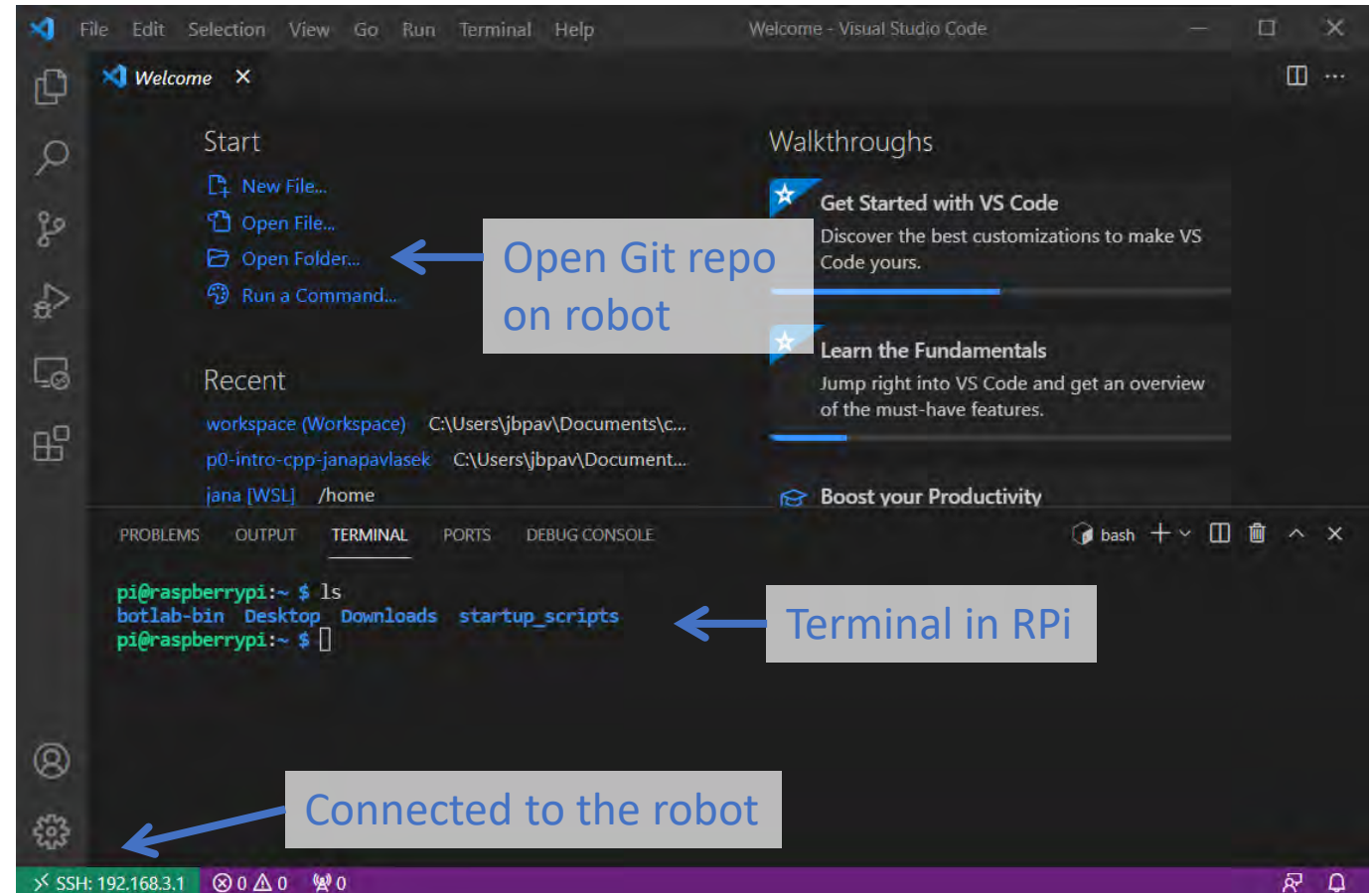
The Raspberry Pi reads the Lidar data. It also has code to perform mapping & localization.



More on this in EECS 467
and ROB 550!

MBot-Omni Workflow

1. Connect to the Raspberry Pi Wifi access point.
2. Open a remote session to the Raspberry Pi in VSCode.
3. Clone your Git repo on the robot.
4. Write, compile, and run code on the robot from VSCode.



<https://robotics102.github.io/tutorials/robot.html>

MBot-Omni Usage

Turning the robot **ON**:

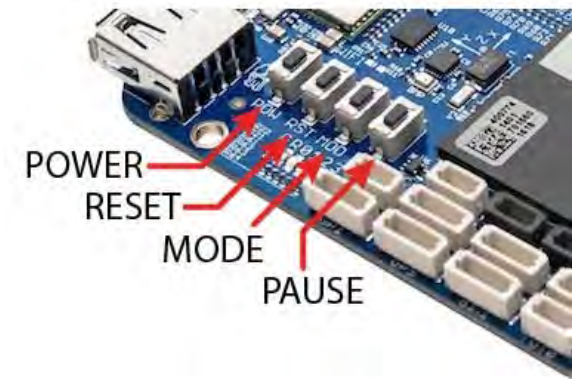
Flick the switch on the power bank to ON (—)

Turning the robot **OFF**:

Flick the switch on the power bank to OFF (o)

Press the POW button on the Beaglebone

Robot is only OFF if all the LEDs are off.



MBot-Omni Usage

Wifi:

Connect to **OMNI-RPI-XXXX** (substitute your robot number)

Wifi password: **iloverobots**

Connecting:

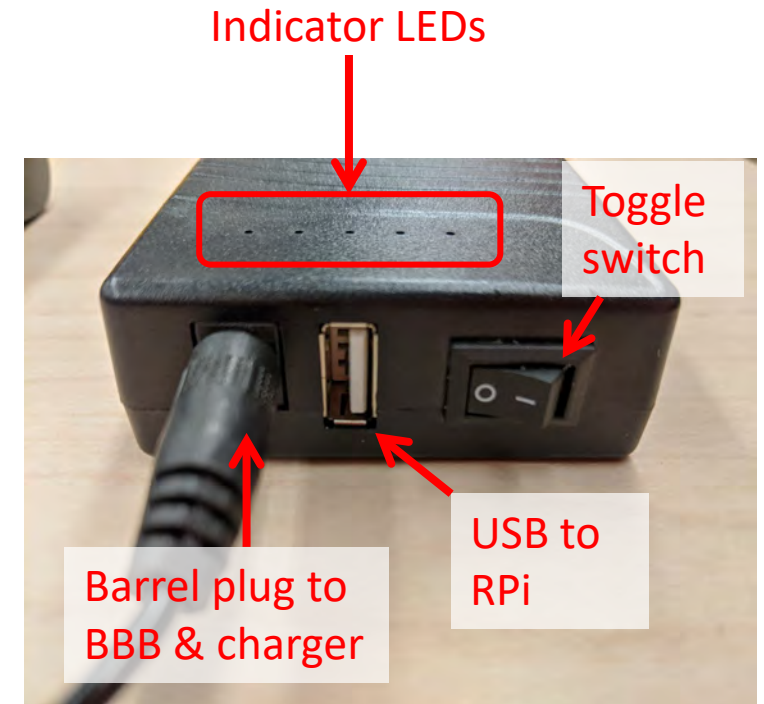
Raspberry Pi IP address: **192.168.3.1**

Raspberry Pi password: **i<3robots!**

MBot-Omni Batteries

Batteries have 5 green LEDs on top. Recharge when 3 LEDs are on.

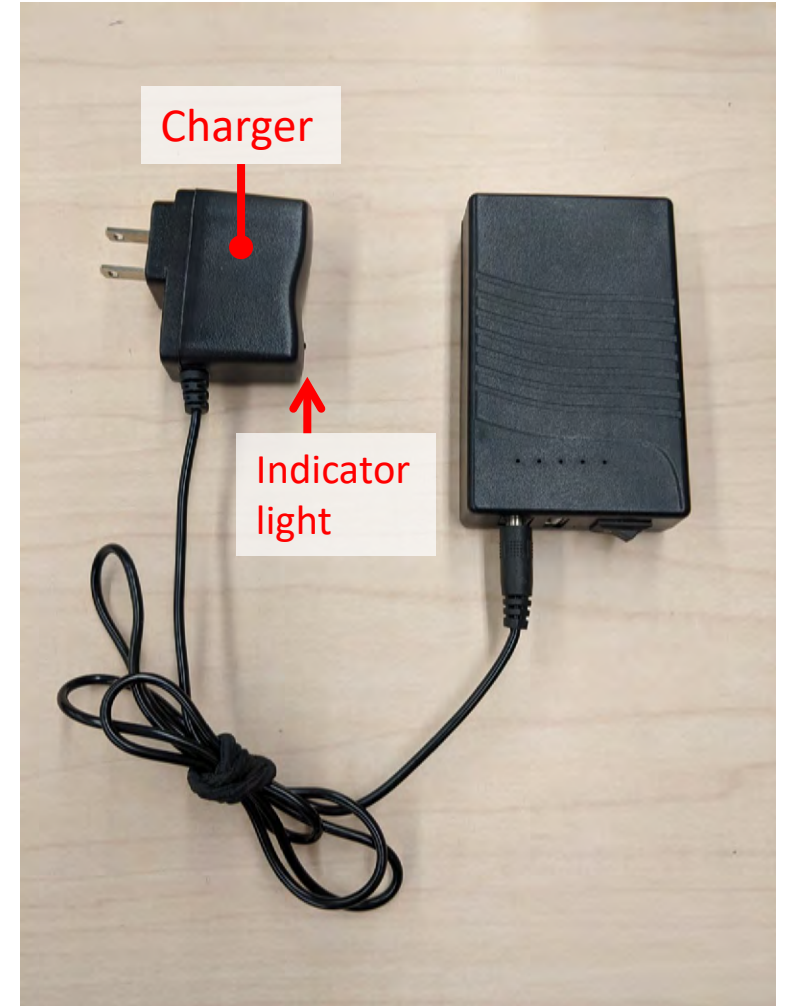
When replacing the battery, make sure the *barrel plug* (to BBB) and *USB* (to RPi) are plugged into the battery.



MBot-Omni Batteries

Plug the battery in and switch to **ON (—)** to charge. The indicator on the charger is **RED** when charging.

Unplug the battery and switch to **OFF (◦)** to store. The indicator is **GREEN** when fully charged.



MBot-Omni Rules & Etiquette

You may get your team's robot from the cabinet in the Robotics lab (FRB 2010) at any time. **Return robots to the cabinet in FRB 2010 when you are done.**

The robots must stay in FRB at all times. You may work in FRB 2000.

Only use the robot assigned to your group.



MBot-Omni Rules & Etiquette

Charged batteries are in the cabinet in FRB 2010 in the “CHARGED BATTERIES” box.

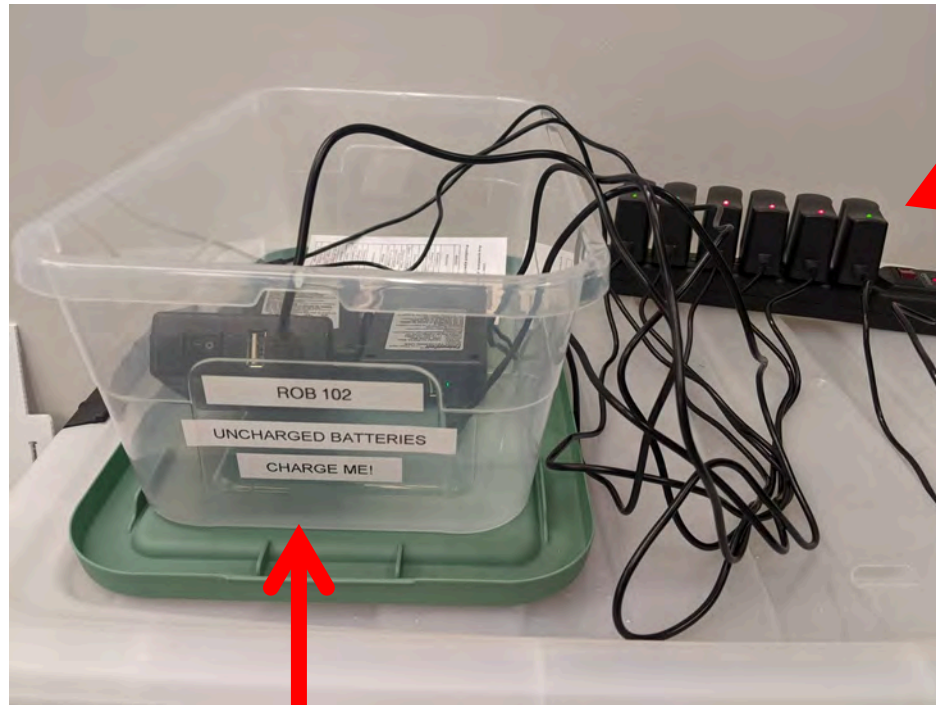
CHARGE OLD BATTERIES when you replace them. They take a long time to charge. Put uncharged batteries in the “UNCHARGED BATTERIES” box.

If you see a battery that is finished charging, put it in the “CHARGED BATTERIES” box.

Charged
batteries



MBot-Omni Rules & Etiquette



Uncharged
batteries

Chargers



Charged
batteries



MBot-Omni Rules & Etiquette

- Treat your peers and the staff with respect.
 - [Michigan Robotics Values](#) apply.
- Keep the lab space clean.
 - Return robots, equipment, and batteries to FRB 2010 when done
 - Leave FRB 2000 like you found it (or better!)
 - Make sure tables, chairs, and maze are in a state usable by others
- Be respectful of other students using the space outside of lab times
- Return any tools you use to where you found them.

Help! My robot is broken!

If your robot is not working as expected:

1. Read the instructions again. Make sure you're following the steps exactly.
2. Restart the robot. Try again.
3. Post on Slack on channel **#mbot-omni** (if no instructors are around).
4. Ask instructors for help.

If your robot is broken, we can repair or replace it. Please ask before switching robots!

TODO: Today

1. Follow the instructions to [connect to the robot](#) on the website
2. [Accept the assignment](#) for Project 1 on GitHub
3. Follow the instructions to [code on the robot](#) on the website
4. Do [Part 1 \(Drive Square\)](#) in Project 1
5. Bring your robot back to FRB 2010. Charge any dead batteries.

Robot Tutorial: <https://robotics102.github.io/tutorials/robot.html>

Project 1: <https://robotics102.github.io/projects/a1.html>