Goals #1
due at team meeting, Week 2 (Tuesday 4/8, Wednesday 4/9/25)

The focus for the first full week is getting the hardware up and running, and everyone comfortable with coding on the Pi. We will not have/use any sensors, but confirm that the basic robot works and illustrate that following a straight line is hard without sensors.

Please use Gradescope for both the **prep and final** *submission,* **due before each meeting**. *Submit only one document and one set of code files per team. One person submits and links to all members!*

Goals:

1. Make sure you have a matched pair of motors.
2. Mechanically and electrically put the robot together.
3. Log into/prepare the Pi: Set up and test SSH, VNC, Samba (file access) *from both laptops.*
4. Do some simple coding to drive a square.

Prep Meeting: Before your prep meeting, please review the sample code (see Section 4.1). Then think about how you might implement the `Motor` object code (see Section 4.3) and the main code (for Sections 4.4-4.6).

Prep Submission: Please *briefly* specify and submit (i) the API for the Motor object, (ii) what its functions/methods will do, and (iii) how the main code will instantiate/call two Motor objects. Be ready to discuss this with your TA at the prep meeting.

Demo Meeting: Next Tuesday/Wednesday, we ask to see a demo of:

(a) SSH, VNC, and Samba working for both teammates.

(b) The robot driving in a square.

(c) We may improvise and ask you to change right turns to left turns or similar, just to see how tuned the solution is to a specific case.

Goals Submission: For this and all weeks, we ask you to submit a brief report and code via Gradescope. This week's report should contain:

(i) any issues you encountered during the build or Pi setup.

(ii) the code specification of what you implemented, possibly adjusted after discussion with the TAs: an API and a simple flow chart or pseudo code of the final program.

(iii) the levels and time values you selected, for both a 1m straight line and a 90 degree turn.

(iv) how far from the start your bot ended, when the battery was grossly off-center.

Please don't go overboard, but document the data clearly/concisely.

Regarding code: This week, most teams only have one file (`motor.py`). We will ask you to directly load that file (or all files, if you have multiple) into Gradescope. In future weeks, we'll use GitHub to easily manage and submit multiple files.

1 Motor Gear Ratio Check

Please follow the "Motor Gear Ratio Test Instructions" posted on Canvas. If you discover that your motors are not matched, please let us (myself or any TA) know and we'll swap either one to get you a symmetric setup.

2 Assembly

Please follow the "Bot Assembly Instructions" posted on Canvas. A few things to note/check:

(a) Make sure the body of the motors are snug with the base-plate and don't wobble. If necessary, you can insert a piece of paper as a shim to remove any play.

(b) Make sure the wheels don't rub against the body, especially once the bot is on the ground and carrying the battery.

(c) If a wheel slides off too easily, apply a small piece of tape over the end of the motor output shaft (hence in a "U" shape, not around the shaft). This should tighten the fit.

(d) Please strain-relief the motor power cable (from battery to motor controller board), i.e. zip-tie it to the body. You'll be unplugging this from the battery a lot and flexing/pulling can this break the small wires at the crimps.

(e) Try to keep the center of mass (while carrying the battery) closer to the wheels and away from the ball caster. This improves traction and reduces friction on the caster.

3 Logging in/Preparing the Pi

The Pi should be ready to go, but you need to connect both your laptops to the Pi.

So, with reference to the 'PiConnection.pdf' handout and for each laptop on the team, please:

(a) Try SSH-ing into the Pi to make sure the connection is good (see Section 5.1). Make sure your laptop is on `robotnet` WiFi. And remember the username on the Pi is `robot` with the password as given on the board.

(b) Set up a remote desktop connection using VNC (Section 2).

(c) Try mounting the Pi as a disk to your laptop (Section 3).

(d) Set up ssh keys - they will make life easier over the next 9 weeks (Section 5.2)

We will set up git next week, when we are dealing with multiple files.

For this week, on the Pi, please create a `~/goals1` folder. That is, a `goals1` folder under the user's home (main) folder. You can do this via the graphical interface or from the command line with:

```
mkdir ~/goals1
```

If you already made a `~/project` folder, please rename this to `~/goals1`. We will create a clean `~/project` folder as part of the git setup next week.

Please place all code (the given `motordemo.py` and your `motor.py` files) in the `~/goals1` folder.

4 Driving

In the following, try to keep the battery secured in the middle between both motors. That is, the center of mass above the midpoint between the wheels. We'll see at the end what happens if the battery shifts.

4.1 Sample Code

Please download the sample code `motordemo.py` from Canvas and place into the `~/goals1` folder on the Pi. This shows how you can access the motors. You can run this via

```
cd ~/goals1                    (do this once to "move into the goals1 folder")
python3 motordemo.py
```

This should energize one motor. Thus, the robot should move a little.

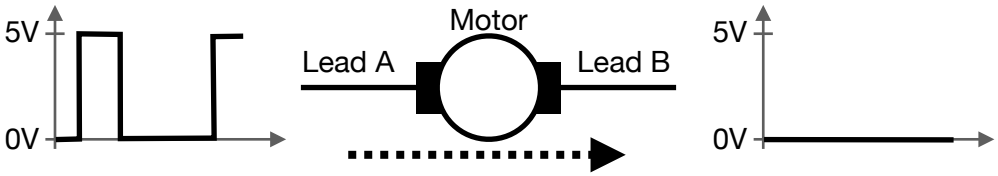
4.2 Note How to Control each Motor

We talked in class about how applying a voltage energizes a motor and creates either motion (if there are no obstacles) or torque (if the motor is blocked). Which is why it is so hard to get an exact movement - the motor may be slower/faster based on load.

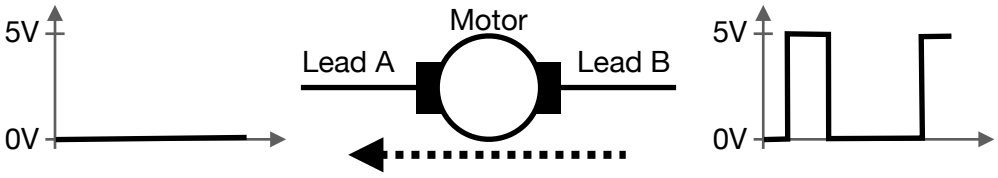
We also mentioned the Pi adjusts voltage by setting a PWM duty cycle: 0%...100% creates 0V...5V.

For direction, remember that the motor has two leads, usually called Lead A and Lead B. Each lead can be anywhere between 0V and 5V. So the direction the motor spins depends on which lead is pulled high, which remains low, and hence in which direction the current will flow. That is, across the motor, the voltage is applied in one direction or the other.

So we have two cases. To move in one direction, set the PWM for lead A to some non-zero duty-cycle. That means lead A will have some (positive) average voltage between 0V and 5V. But hold the other lead B at 0V (setting to 0% duty-cycle). The direction is thereby defined.



To reverse the direction, hold lead A at zero and set lead B PWM to a non-zero duty-cycle.



Always make sure only one pin has a nonzero duty-cycle, and the other pin is set to zero duty-cycle.

4.3 Write Code to Control a Motor

Create a new file `motor.py`. Using inspiration from the sample code (`motordemo.py`), create a `Motor` class (to be instantiated twice for the two motors). Write the class, such that it has

(a) an initialization `__init__()`. What arguments will you pass?

(b) a `stop()` method to halt the motor.

(c) a `setlevel(level)` method which sets the level in the range $-1.0 \cdots +1.0$, i.e. either positive or negative, relative to the max motor power. Given a constant friction, this effectively sets the speed.

These functions will have to set up and use the I/O pins appropriately.

In the main portion of `motor.py`

(a) instantiate the I/O object (as did `motordemo.py`),

(b) instantiate two motor objects, for the left and right motors,

(c) program the below behaviors,

(d) do not forget to stop the motors and the I/O object before the code exits.

Please discuss with the TAs to flesh out exactly how this will work!

4.4 Drive Forward (Straight) for One Meter

Using the motor objects, program the robot to drive forward in a straight line, for one meter. That means you need to determine

(a) what level (positive or negative depending on direction) to command to each motor. Note they could be slightly different if the motors aren't identical or something else is asymmetric.

(b) how long to keep the motors energized before stopping to get to exactly one meter.

4.5 Turn Right for 90 Degrees

Change the previous code to turn in place to the right for 90 degrees. Again, you'll need to know the motor levels as well as the timing.

4.6 Drive a Square

Finally combine the above, so the robot drives a perfect 1m square. That is, repeat four times.

4.7 How Sensitive is this?

For a final/quick test, move the battery as far off-center as you can (left/right or front/back). Then re-run the code. How much does this change the robot's path? In particular, how close to the starting position does it return?

Submission: Please report your final level/PWM and time values, for both straight line and turning. Also report the start-end error with an off-center battery, and submit the final `motor.py` code.