Line-Following Robot Setup

A guide for students in setting up the Line-Following Robot hardware and software

I. Installing the software

The robot consists of a chassis with a left and a right motor, three IR sensors and an Arduino board. Your robot has already been wired for you. Please **DO NOT** alter the wiring unless you find that your motors are not running properly when performing the steps under section "II. Testing the Robot".

DO NOT CONNECT THE ROBOT TO YOUR COMPUTER until you are instructed to do so.

A. Install the latest version of Arduino IDE (1.8.2 as of Mar 23, 2017)

https://www.arduino.cc/en/Main/Software

It is NOT necessary to contribute to download the software. Click on "JUST DOWNLOAD" once you choose the appropriate OS version.

B. Install the necessary packages to MATLAB

This section walks through the installation of Arduino Hardware Support Packages to MATLAB.

1. Make sure you have MATLAB version R2016a, R2017b, or newer.

2. Run MATLAB and go to the "HOME" tab.

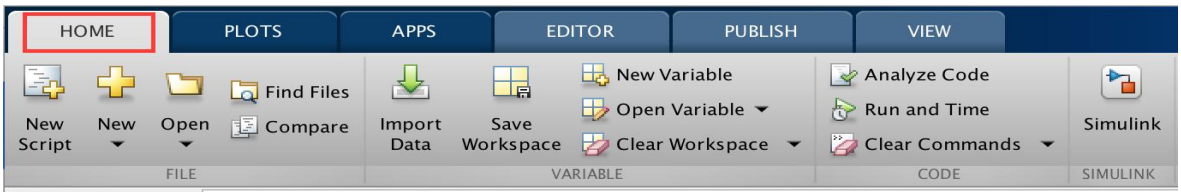


Figure 1. Home Tab

3. Click on "Add-Ons" icon.

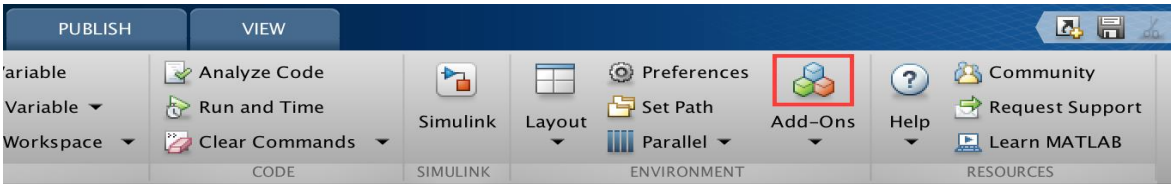


Figure 2. Add-Ons

4. Select “Hardware Support Packages” from the left panel in the pop-up window.

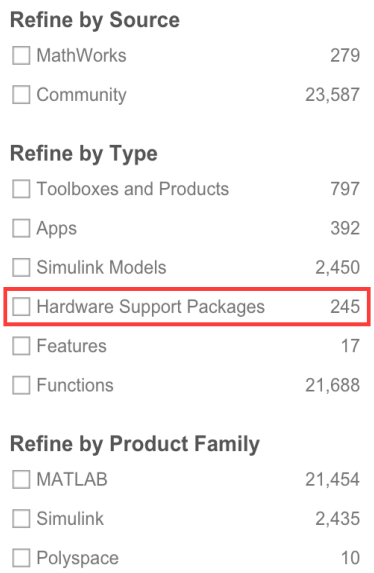
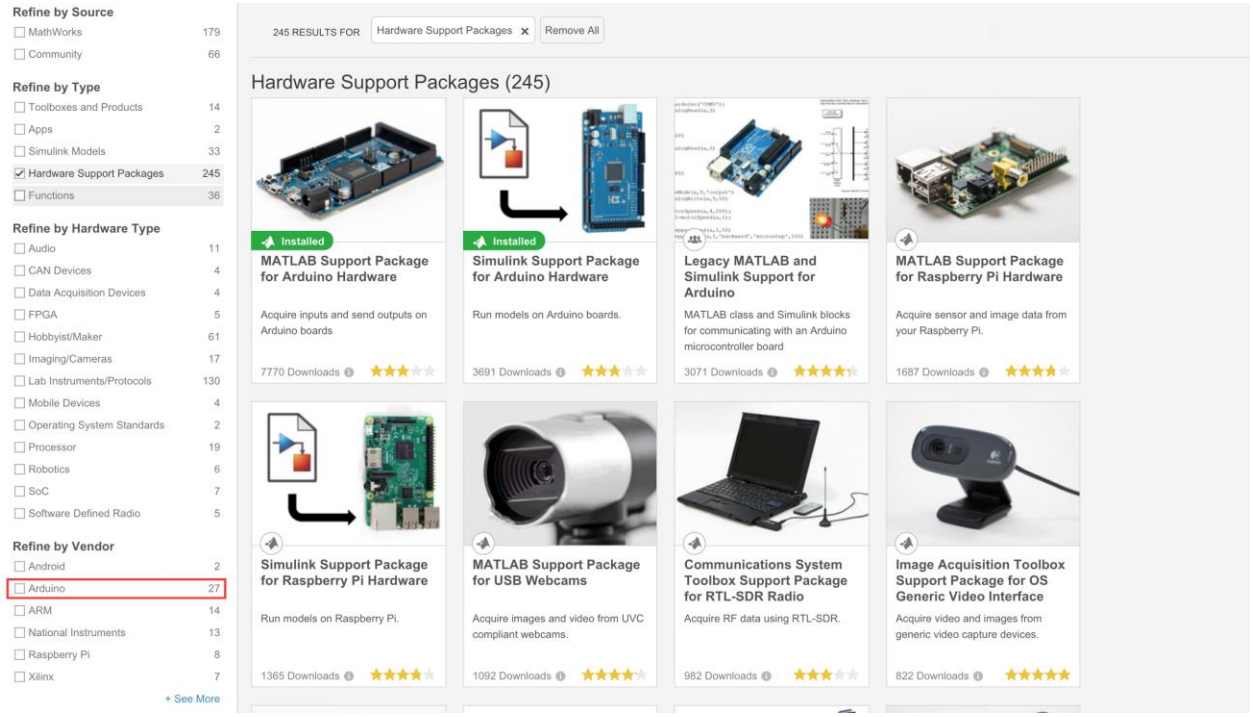


Figure 3. Hardware Support Packages



5. The results will look something like this:

Figure 4. Arduino packages

6. Select “Arduino” from the “Refine by Vendor” panel.

7. Scroll through the results and locate the following two packages (you can also search for them using the bar in the top right corner).

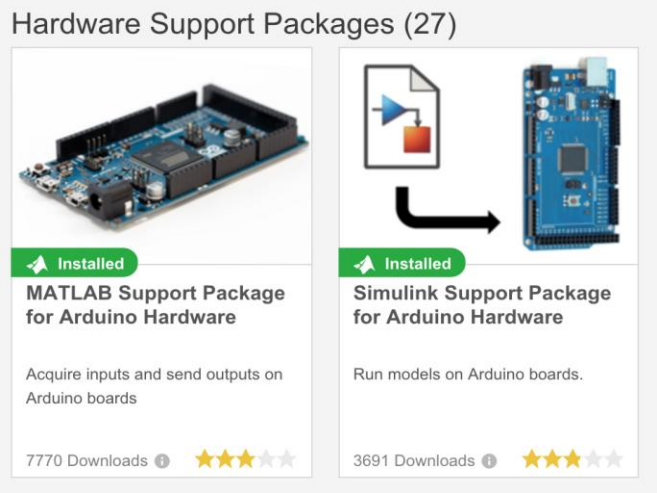


Figure 5. Hardware Packages needed

8. Install each one by clicking on it and following the instructions.

C. Instructions for Connecting the Arduino

1. Download all files from the following shared Google folder:

https://drive.google.com/drive/folders/0BxgDAEu-h2ZBRVFXR3V0ODhUNGc?usp=sharing

2. After the files are downloaded, copy the .m files to your working MATLAB Directory.

3. Connect the USB cable to the Arduino and your computer.

4. Open the Device Manager on your computer.

In Device Manager, you can identify what is connected to your COM ports. There are many ways to do this depending on what version of Windows you are running. If you don't know how to do this, Google it or ask a friend.

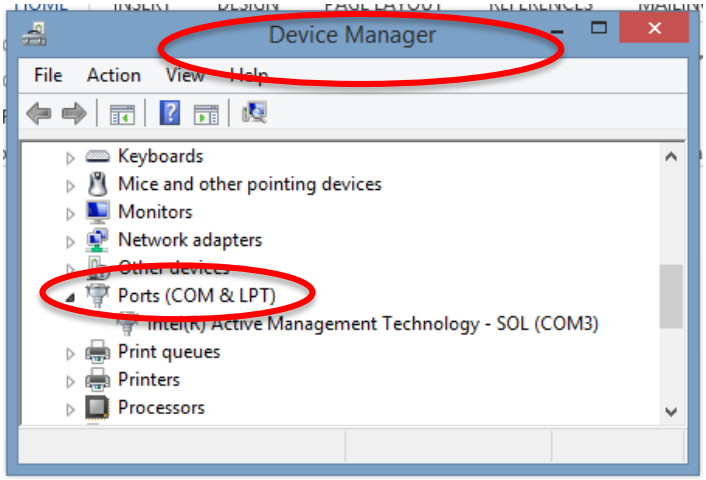


Figure 6. Device Manager

5. In "Device Manager," find the port through which Arduino is communicating.

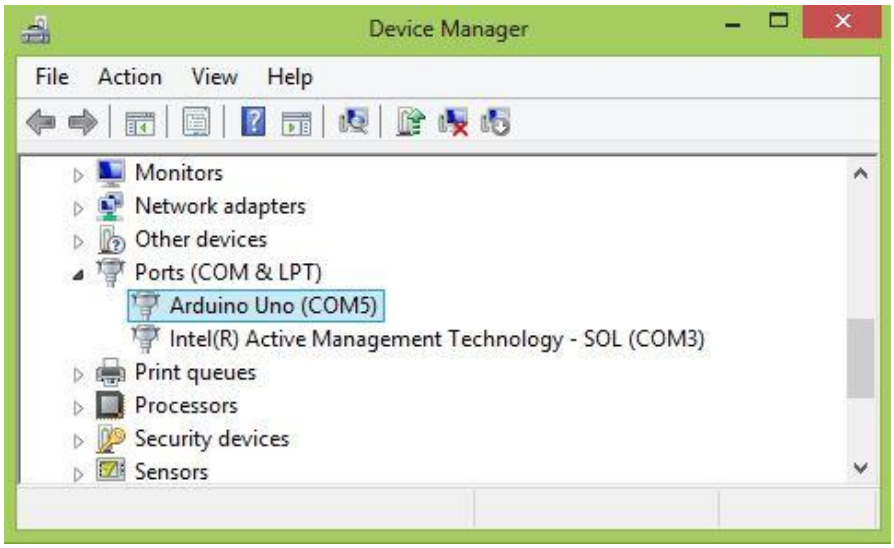


Figure 7. Determining COM-port used

Note which COM-port is being used by the Arduino.
In the figure above, we see that COM5 is being used for Arduino.

6. Navigate to the folder with the files you downloaded.

7. Open Robot_Connect.m in the script editor and change the port to the appropriate name.

8. Run the Robot_Connect script.
9. If there is an error consult Troubleshooting section (below), or test the robot motor and sensor functionality.

Troubleshooting

Make sure you have the latest Arduino installed (1.8.2 as of Mar 23, 2017).
https://www.arduino.cc/en/Main/Software

After installation follow the instructions in the prompts and try connection again from MATLAB.

If there is an error regarding legacy Java framework, search for the framework for your file system and install it. It is used for programming the Arduino boards with the server code.

II. Testing the Robot

This section assumes you have completed the instructions in the previous section.

Note that the robot is powered through the USB cable and draws its power from the laptop it is connected to. It would be a good idea plug your computer in while running the robot.

Robot_Connect is used to establish the connection between your computer and the Robot. Sensor_Test is used for testing the sensors and Motor_Test is used for testing the motors. Stopm is used to stop the motors if they keep running after one of the other programs terminates. See the Appendices for a description of these programs.

As you test your robot you will be recording results in the file Robot_Testing_Results.docx (which you should have already downloaded).

1. **Establish the connection with the Arduino** – Determine which COM port your Arduino is connected to (use the Device Manager). Open Robot_Connect.m in the editor. Edit line 8 to reflect the COM port your Arduino is using. Run Robot_Connect.m. This program calls arduino_new.m. You do not need to run Arduino separately.

   After running Robot_Connect you should see a variable "**a**" in your workspace. As long as the variable "**a**" exists in the workspace you should not need to run Robot_Connect again. If you use the 'clear' command or need to restart MATLAB for any reason, you will need to run Robot_Connect again.

2. **Motors –** The goal of this section is to **a.** verify that your motors are running in the correct direction, **b.** understand how various motor settings cause the robot to move, and

c. determine the "maximum speed" settings required for the robot to track straight. As you test the motors be careful not to let the robot run off the table, or run into other objects. **Also note that the USB cable connecting the robot to the computer can interfere with the robot motion and has limited length. You may have to support the cable to reduce its effect.**

a. Check to see if the motors run in the correct direction. Hold the robot in the air. Run Motor_Test.m. Enter 1 as the speed for both the left and right motors. If the motors do not run, see Appendix D and check your robot's wiring. Verify that BOTH motors run in a direction that would cause the robot to move forward. If a motor causes the wheel to rotate in the wrong direction you will need to change the wiring for that motor. As an example, Figure 8 shows what to do if the right motor is turning opposite to what is desired.

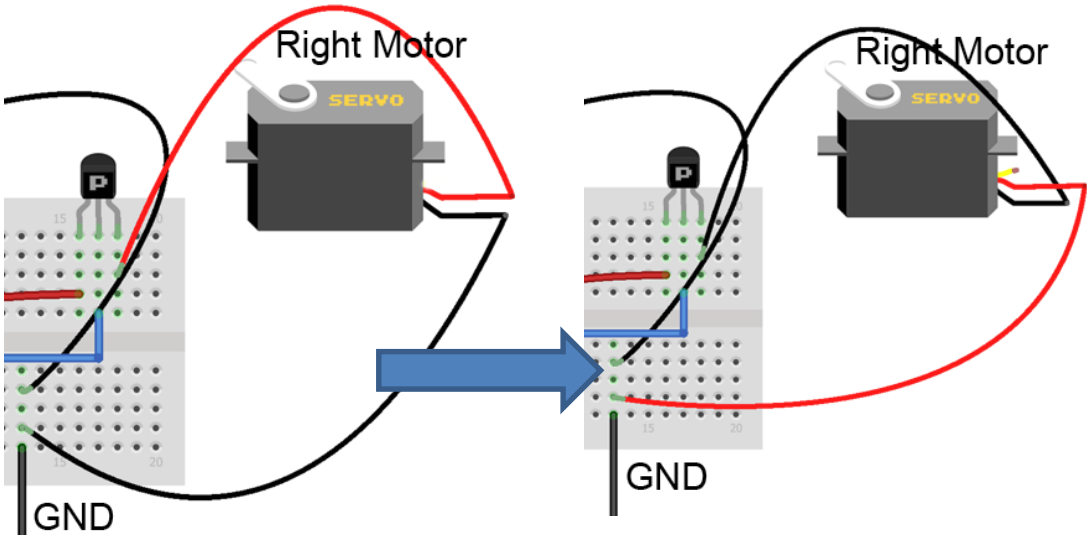


Figure 8. Switch Motor wires as shown if motor is not turning in the forward direction

b. Place the robot on a CLEAN area on the floor or on the table. Use the motor speeds/levels below in your motor testing program. Record what the robot does, e.g., moves straight, turns right, turns left, stop, in Table 1 of the Robot_Testing_Results file
   i. Left motor = 1, Right motor = 0
   ii. Right motor = 1, Left motor = 0
   iii. Both motors = 1

c. In the previous step, with both motors levels at 1, did your robot travel in a straight line? (Do not allow the USB cable to influence the motion of the robot). Your goal in this step is to balance the motors, by adjusting the motor power levels (the values of lmHigh and rmHigh) until the robot tracks in a straight line. The power level for one of the motors will remain at 1 (the maximum), but you may need to adjust the power level for the other to a lower value, to balance motors. Run Motor_Test for multiple values of lmHigh (the high power level for

the left motor) and rmHigh (the high power level for the right motor) looking for a combination that causes the robot to track straight. **Record the values of rmHigh and lmHigh and whether the robot travels straight, turns right, or turns left for each pair of settings you test in Table 2 in the Robot_Testing_Results file**. Don't let your robot fall off the table.

Make note of the values of lmHigh and rmHigh that make your robot track straight. These are the values you will use in your program when you test your line-following algorithm next class.

3. **Sensors** – The goal of this section is to **a.** determine typical sensor readings when the robot is in various positions over a black line and **b.** to determine a threshold value that can be used to indicate that a sensor is **directly above** the black line. You will take sensor readings multiple times, recording the results in **Table 3 in the Robot_Testing_Results file**.
    a. Run Sensor_test.m
    b. Physically move your robot according to the 15 different conditions listed in the file Robot_Testing_Results, one condition at a time. Record in Table 3 the sensor reading for each of the listed conditions. Notice that you are repeating each reading 3 times so that you can determine an average value for determining your threshold. You should move the robot away from the line and reposition it for each reading.
    c. Based on the sensor readings you recorded, identify a threshold value that indicates that a sensor is over the black line. You are looking for a value which you would say means the sensor is over the black line. You will compare your sensor readings to this value in your line-following robot algorithm to determine whether or not the sensor is over the black line. You will use the same threshold for all sensors, so the value you use should work for all sensors.

III. Finishing up

1. Before disconnecting your robot from your computer
    a. Stop all programs (Make sure you have the >> prompt in the command window).
    b. Clear your MATLAB workspace (type clear all)
    c. Uninstall the Arduino from the COM port using the Device Manager (but do not uninstall the drivers.)
2. Return the robot and the connecting cable to the front of the class.
3. Make sure the names of all group members present are recorded in the Robot_Testing_Results file, save it as a PDF, and upload one copy of the PDF per group to Canvas.

IV. Appendix A

Robot_Stub and MATLAB/Robot communication commands

The following MATLAB code initializes several constants that **1)** will make your robot programs easier to understand and **2)** make it easier change the program if a change in the hardware wiring is required. You will use this program as the basis for your line-following robot code. Add the logic needed to cause the robot to follow the line below line 46 in the USER CODE SECTION.

```
1   % EngE 1215 Spring 2017
2   % This program is used to help a Line-Following Robot follow the line.
3   % The first several sections set up the constants used in the communcation
4   % between this program and the arduino.
5   % Each student will add his/her own code to the User Code section with the
6   % logic to follow the line.
7
8   % This program assumes that a connection has been established to the
9   % arduino using the program Robot_Connect.
10  % The variable 'a' must be established in the command window.
11
12  %% Constants for reading sensors - do not change statements in this block
13  % unless there is a change to the physical robot
14  SENSOR_PIN = 'D13';     % sensor control pin (all sensors)
15  LEFT_SENSOR_PIN = 'A0';  % pin to read left sensor values from
16  CENTER_SENSOR_PIN = 'A2';% pin to read center sensor values from
17  RIGHT_SENSOR_PIN = 'A4'; % pint to read right sensor values from
18
19  %% Changes and Additions
20  % pins have digital and analog designations, ex 13 => 'D13', 2 => 'A2'
21  % pinMode setup
22  a.pinMode(SENSOR_PIN,'DigitalOutput');
23  a.pinMode(LEFT_SENSOR_PIN,'AnalogInput');
24  a.pinMode(CENTER_SENSOR_PIN,'AnalogInput');
25  a.pinMode(RIGHT_SENSOR_PIN,'AnalogInput');
26
27  %% Constants for motor control - Change lmhigh and rmhigh based on testing
28  %your robot.  Do not change other statements in this block
29  % unless there is a change to the physical robot
30  LEFT_MOTOR = 'D5';      % set pin 5 as the pin to control the left motor
31  RIGHT_MOTOR = 'D6';     % set pin 6 as the pin to control the right motor
32  OFF = 0;            % motor speed of 0, 0 is off
33  lmHigh = 1;        % left motor speed, range of acceptable values 0-1
34  rmHigh = 1;        % right motor speed, range of acceptable values 0-1
35
```

Figure 9. Portion of the Robot_stub.m code

V. Appendix B

Motor Test

Motor_Test.m is a program that will allow you to test the robot's response to different motor speeds and to identify a combination of speed settings for the left and right motors that will allow the robot to track straight.

a. Commands for controlling the motors

```
a.analogWrite(LEFT_MOTOR,lmHigh) % turn left motor on  (speed/level = lmHigh)
a.analogWrite(LEFT_MOTOR,OFF) % turn left motor off (speed/level = 0)
a.analogWrite(RIGHT_MOTOR,rmHigh) % turn right motor on (speed/level = rmHigh)
a.analogWrite(RIGHT_MOTOR,OFF) % turn right motor off (speed/level = 0)
```

b. This program features:
   i. A while loop that will execute as long as the user indicates that they wish to continue
   ii. An input statement to request the user to enter the desired motor speed levels (rmHigh and lmHigh)
   iii. Commands to turn the motors on (at the indicated levels) or off
   iv. Commands to pause the program and let the motors run until the user hits a key

VI. Appendix C

Sensor Test

Sensor_Test.m - a program that will allow you to take sensor readings for a user-entered number of robot positions

a. Command for controlling the motors

```
a.digitalWrite(SENSOR_PIN,1);    % turns sensors on
left = a.analogRead(LEFT_SENSOR_PIN); % reads the left sensor storing the value in variable left
center = a.analogRead(CENTER_SENSOR_PIN);
right = a.analogRead(RIGHT_SENSOR_PIN);
a.digitalWrite(SENSOR_PIN,0);    % turns sensors off
sensors = [left center right]; % stores three sensor readings in a variable
disp(sensors) % displays three sensor readings on a single line with no label
```

b. This program features:
   i. A while loop that will execute a user-specified number of times.
   ii. A command to turn the sensors on
   iii. Statements to read the sensors
   iv. A command to turn the sensors off
   v. Statements to display the sensor values
   vi. A “pause” which causes the program to wait for the user to hit any key, allowing as much time as needed to record the sensor values and to move your robot to set up for the next reading.

VII. Appendix D:

Checking the robot's wiring

Each motor is controlled through a motor driver circuit which is built using a 4401 Transistor. Each motor has two wires (PWR and GND). The motor will run either forward or backward depending on which wire is attached to *collector* of the transistor. Since the motors are exactly the same, but are mounted in opposite directions, to make the robot move forward, one motor will have the red wire connected to the *collector* and the other will use the black wire. The *emitter* of the transistor is connected to (+5V) from the Arduino. The *base* of each transistor is connected to a signal pin on the Arduino. The base of the transistor for the left motor is connected to pin 5, for the right motor to pin 6.

There are six wires to the sensors. The wire marked (+) on the sensor supplies power to the sensors (+5V), the wire marked (G) is the ground wire. The wire marked (S) wire provides the instruction from the Arduino to power the sensors and the wire marked (L), (C), and (R) wires carry the signal from the left, center, and right sensors, respectively, to the Arduino.

The breadboard on the robot is used to allow multiple connections (through the underlying terminals) to +5V and GND on the Arduino.

A capacitor is connected to the +5V and ground on the breadboard. The negative side of the capacitor is connected to ground and the positive side is connected to +5v as shown in Figure 2 by the purple wire.

Use the diagrams below (figures 10 and 11) to confirm that your robot is wired correctly.

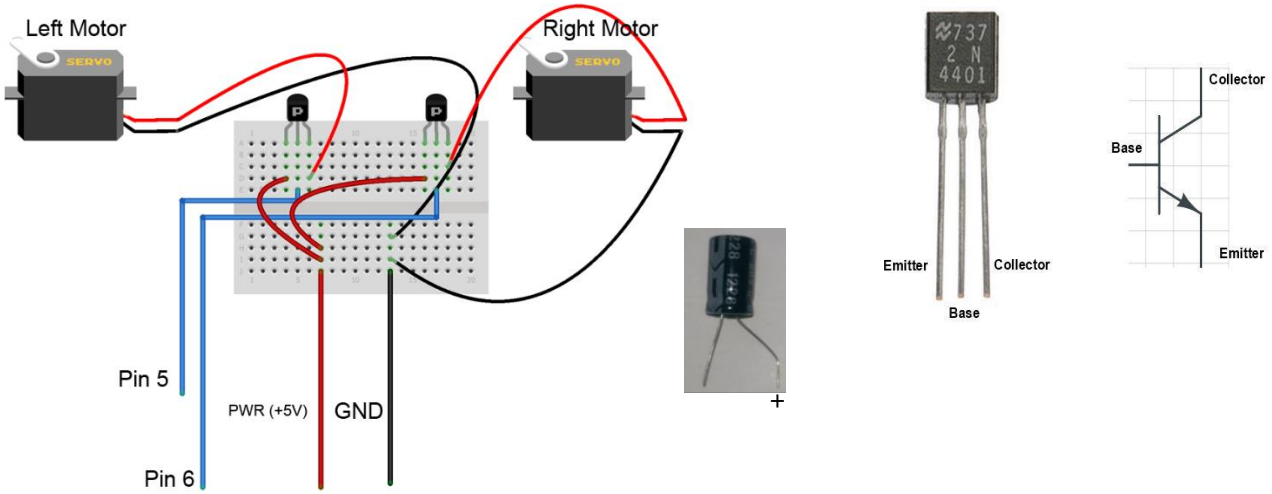


Figure 10. Line - following robot wiring, and detail of Capacitor and 4401 Transistor

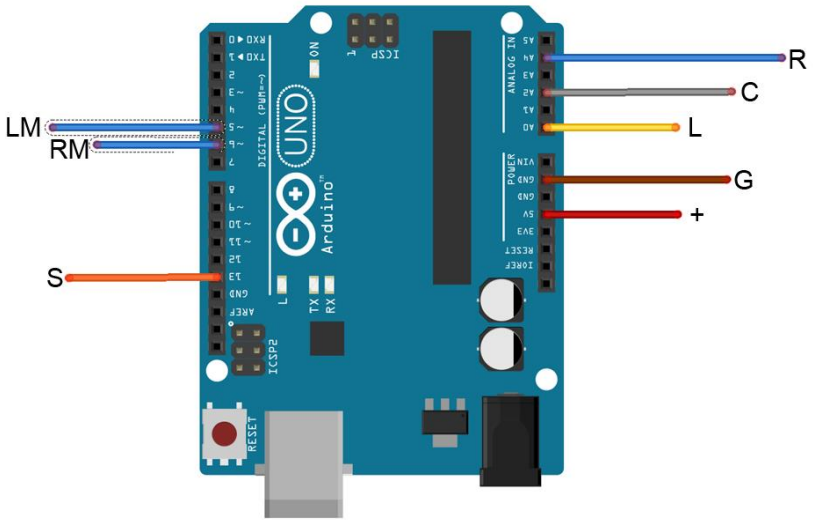


Figure 11. Arduino Pinout

** Please note that the wires are not color specific but marked by letters on the sensors

VIII. Appendix E

Engineering Units Explanation

Two functions used in the MATLAB script involve units: analogRead(pin), analogWrite(pin)

Analog Read

In the line following robots, this function returns the voltage reading from the line sensors. When the robot is powered by usb (which delivers 5V) these voltages will range from 0V to 5V. The returned value is a floating point value from 0 to 5.

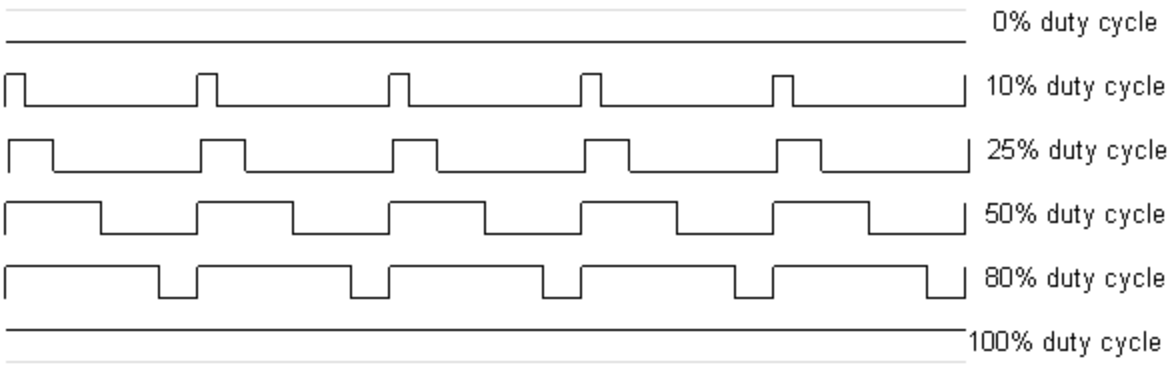
Example Transcript:
>> Sensor_Test
How many sensor readings will you take? 3
Position the robot and press enter for sensor readings
Left Center Right
  0.5914 0.5914 0.5963

Position the robot and press enter for sensor readings
Left Center Right
  0.4057 0.4545 4.0029

Position the robot and press enter for sensor readings
Left Center Right
  0.3959 3.9785 3.8221

Analog Write

This function accepts the value from 0 to 1, which corresponds to a duty cycle of the square wave that will be simulated on the output pin.



The figure above is taken from https://www.arduino.cc/en/Tutorial/SecretsOfArduinoPWM, which contains a much more in depth explanation of the underlying hardware that makes up the analogWrite functionality.

Since the robot cannot start and stop instantaneously due to both mechanical and electrical effects, this high frequency jitter is not noticeable, but rather the average effects of the entire wave.

Higher duty cycle sends on average more current to the motors causing the motors to spin faster.

The exact values for the appropriate duty cycle are up to experimentation, but it is unlikely that the robot will be able to overcome the internal friction at duty cycles smaller than 0.5 and therefore will not move.

There is absolutely no harm at using it full speed (i.e. 100% duty cycle or simply 1) and in most cases it is preferred.

Example Transcript:

```
>> Motor_Test
Enter desired left motor speed level (0.00-1.00): 0.95
Enter desired right motor speed level (0.00-1.00): 0.95
Hit any key to stop motors
Run again?  1 for Yes, 0 for No: 0
>>
```