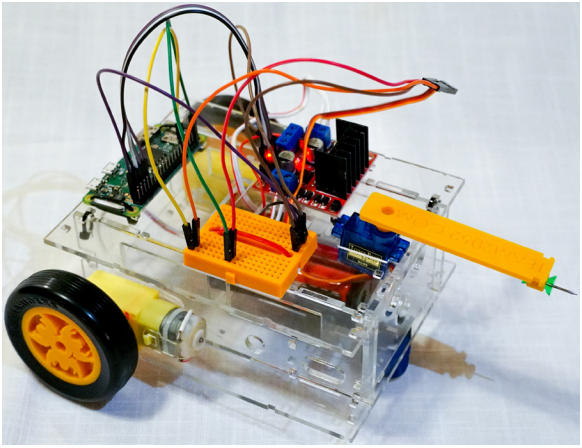
RASPBERRY PI ROBOTS



BY FRC TEAM 2052 KNIGHTKRAWLER

GOAL

This booklet will help you create a small robot using a Raspberry Pi. **With a balloon on the back and a pushpin on the front, this robot can participate in balloon-popping battles.** You will assemble, wire, and program a robot that can drive around the room. The Appendix (page 23) includes a parts list and links for purchasing all of the parts you need on Amazon. The same, or equivalent, parts can also be acquired both in retail stores or online.

NOTE: You can tweet questions to @Team2052 or send messages in THAT Slack (for THAT Conference students) to scottkdavis.

CONTENTS

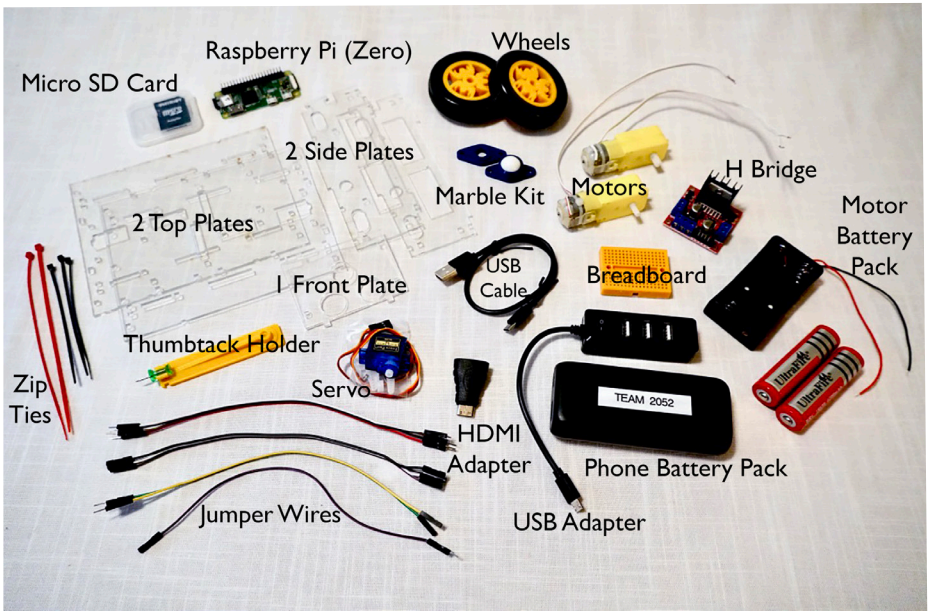
PART 1:

ASSEMBLING YOUR ROBOT

NOTE: For the simplicity of this guide, these instructions are written assuming you are using one of KnightKrawler's kits. If you are using your own parts, you may need to adapt the instructions for your robot.

CHAPTER 1: ASSEMBLY

There is a full list of the parts you will need for your robot on page 24 of this book. Below are the items:



We will begin by assembling the chassis, or frame, of your robot.

1. Take one of the largest acrylic chassis pieces to use as the underside of the robot. The two medium-sized pieces are the sides. With the circular holes on the sides facing down, place the sides into the slots in the bottom piece. (Figure 1)
2. Place the front plate with the 2 side-by-side holes on the top, into the bottom plate on the side with the rectangular hole. (Figure 2)
3. Place the other large piece on the top of the assembly. Make sure the rectangular holes in both large pieces are facing the same direction. (Figure 3)

Be careful when handling the chassis in the next few steps. You may need to take the top and front on and off throughout the assembly.

Now you will attach the motors, which will allow the wheels to move.

4. Feed the wires for one motor through the smaller round hole near the open end of the chassis. (Figure 4)
5. Feed a zip tie through the small top hole on the motor, then through the chassis wall, the chassis floor, and around the outside of the motor. By doing this you will secure the motor to the side and bottom in one loop. You may need to connect two zip ties together to make the complete loop, if using short zip ties. (Figure 4)
6. Repeat on the other side with the other motor. Use your snips to trim the ends of the zip ties.
7. **Marble Wheel**: Place the marble in the blue holder. Place the thin blue piece on top. Secure to chassis bottom with 2 zip ties, as shown. (Figures 5 and 6)

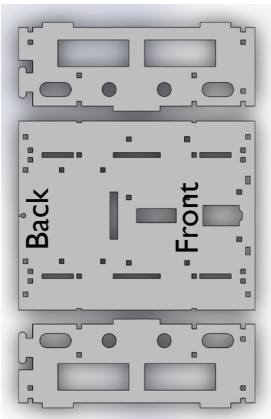


Figure 1

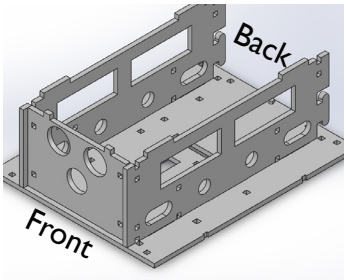


Figure 2

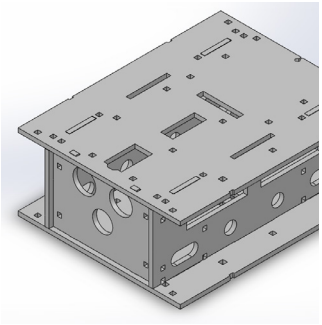


Figure 3

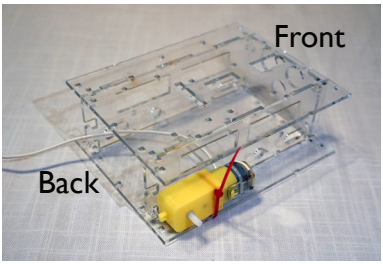


Figure 4

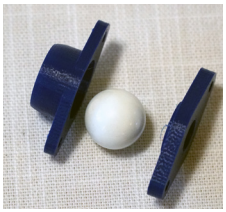


Figure 5

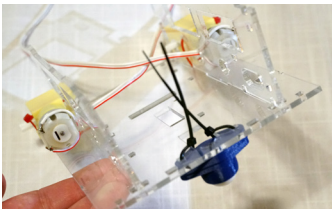


Figure 6

Grab the top panel of the chassis to add the robot's electronic components. See the parts diagram on page 4 if you are unfamiliar with the part names.

8. **Raspberry Pi:** Align the Raspberry Pi on the top as shown. Secure it to the top with 2 zip ties. (Figure 7)
9. **H-Bridge:** Align the H-Bridge on the top as shown. Using 2 zip ties, push one up through the bottom and the other down from the top in the same hole. Secure by tightening the zip ties both above and below the H-Bridge. Do this in 2 opposite holes. (Figure 8)
10. **Breadboard:** Do not remove the paper backing. Align on the chassis top as shown. Secure with a zip tie. (Figure 9)
11. **Servo:** put the wires through the top of the chassis and place the servo as shown. (Figure 10)
12. Secure the top of the chassis to the body. Make sure the servo and the marble wheel are aligned. Zip ties in opposite corners will secure everything nicely. (Figure 12)
13. Feed the wires for the servo and motors through the holes in the top of the chassis in the middle to prepare for wiring.

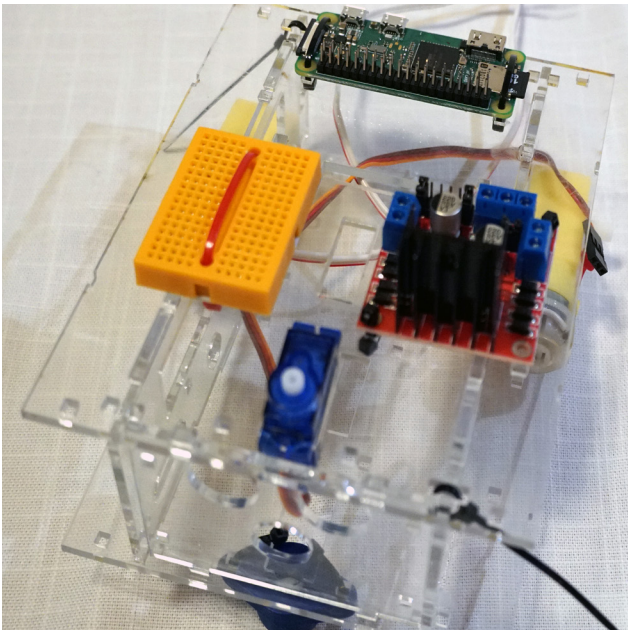


Figure 12

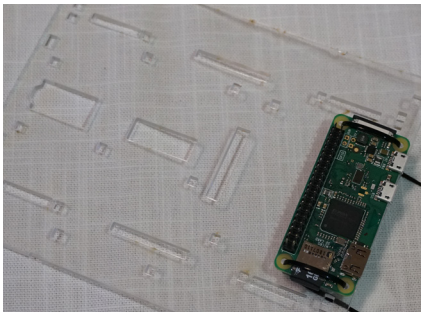


Figure 7

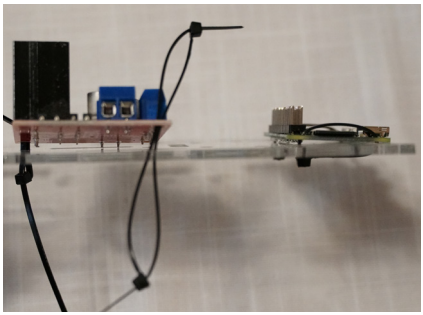


Figure 8

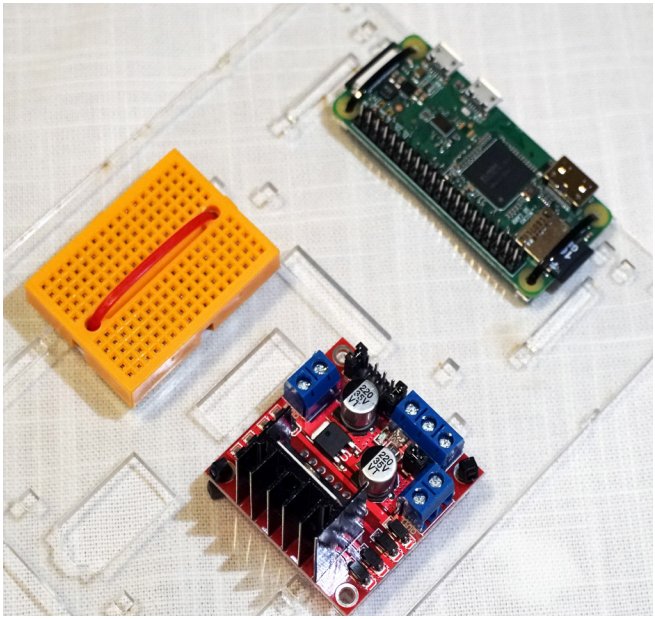


Figure 9

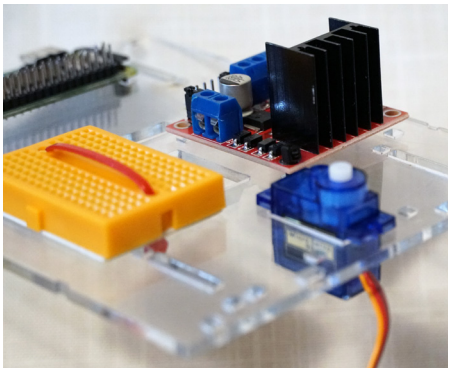


Figure 10

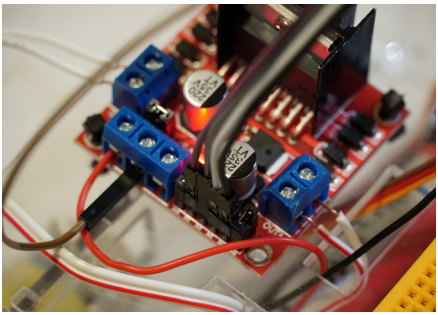
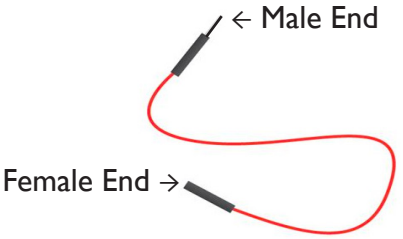


Figure 11

CHAPTER 2: WIRING

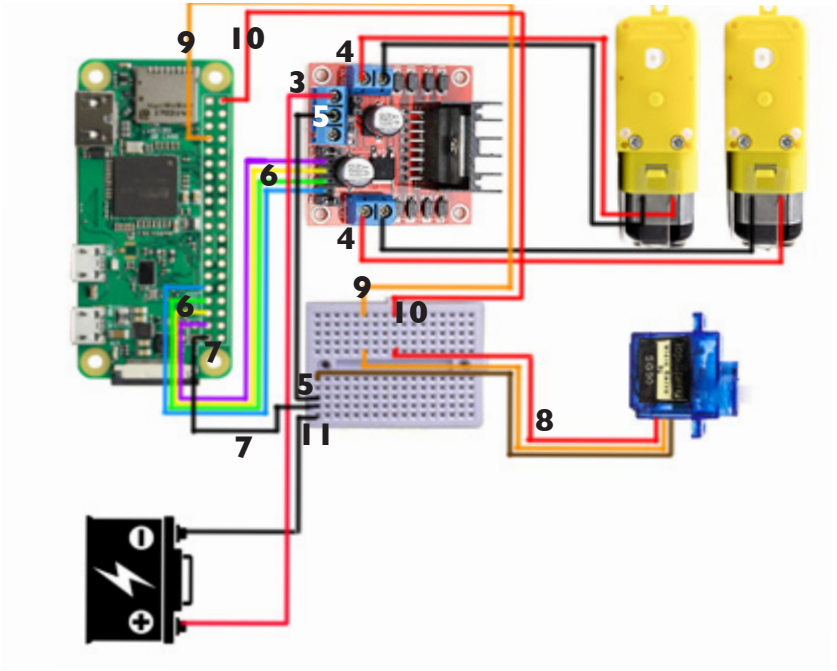
Now that your robot is put together, you will need to wire it to connect the components. There are three types of jumper wires you will be using:



- Male to male (two male ends)
- Male to female (one male end and one female end)
- Female to female (two female ends)

NOTE: The colors of the jumper wires you use doesn't matter.

This chapter will be walking you through the steps to wire your robot. Refer to the diagram below while following the instructions. On the next page (page 10) there is a diagram of the pins on the Raspberry Pi.

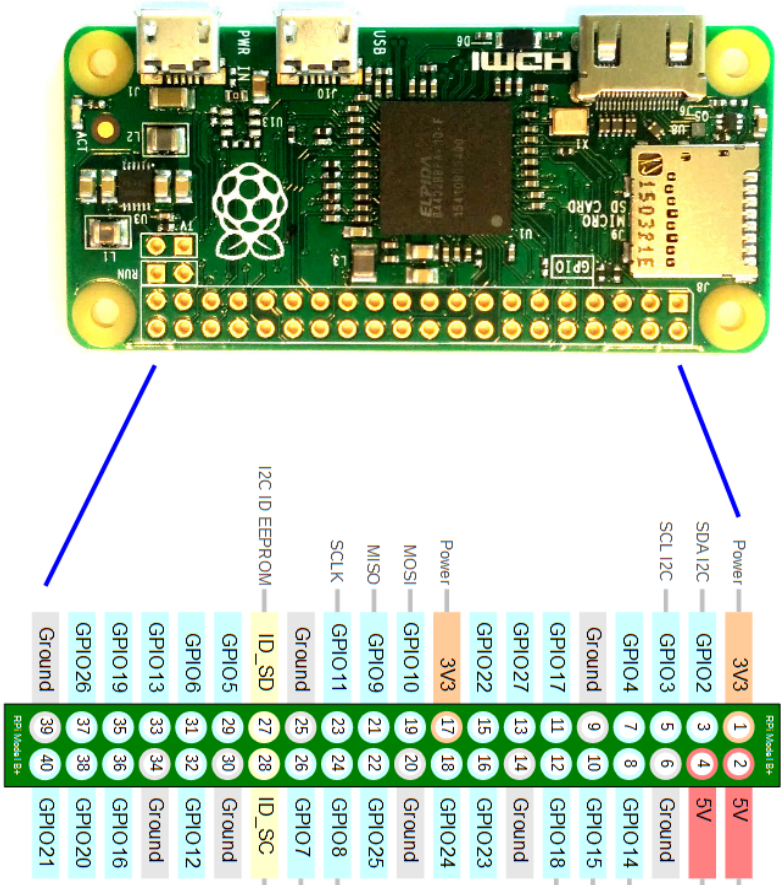


1. Battery Holder: Insert the batteries into the holder. Be sure the flat ends of the batteries are against the springs in the battery holder.
2. Place the battery holder beneath the top chassis panel and feed the battery holder wire through the top of the chassis.
3. Loosen the screws on the H-Bridge. Put the red wire from the battery pack into the H-Bridge as shown and re-tighten the screw. (See figure 11 on page 7 for a close-up of the H-Bridge)
4. Motors: Loosen the screws on the two pairs of blue housings. Take the wires from one of the motors and put one wire end into each housing. Re-tighten the screws.
5. Loosen the screw in the center blue housing shown. Insert one end of the male-to-male jumper wire into that housing and re-tighten the screw to secure the wire. Firmly push the other end of the jumper wire into the last row of the breadboard (see diagram). **NOTE:** Be sure to push all wires into the breadboard firmly to get a good connection.
6. Find 4 jumper wires with female/female ends. Push the ends of one side over the pins on the H-Bridge. Push the ends of the second side over Physical Pins 31, 33, 35, 37 on the Raspberry Pi as shown. (Refer to the pin diagram on page 10 for pin numbers)
7. Find a male/female jumper wire. Place the female end over Physical Pin 39 (a ground pin) on the Raspberry Pi. Place the male end into the last row of the breadboard.
8. Find 3 male/male jumper wires. Plug them into the wires from the servo. Refer to the image for the placement of the wires' other ends.
9. Find a male/female jumper wire. Plug the male end into the breadboard as shown in orange on the diagram. Plug the female end into Physical Pin 7 on the Raspberry Pi as shown.
10. Find another male/female jumper wire. Plug the male end into the breadboard as shown in red. Plug the female end into Physical Pin 2 (the pin for 5 volt power) on the Raspberry Pi as shown.
11. **Do not complete this step until you are ready to test your code.** Insert the black wire from the battery pack into the corner of the breadboard.

Add the finishing touches:

1. Place the yellow push-pin part on top of the servo. Press together firmly. (Refer to the image on this booklet's cover for pin placement)
2. Place one wheel on each motor's hub.
3. CONGRATULATIONS! You have built a robot!

PIN DIAGRAM


Power
SDA I2C
SCL I2C
Power
MOSI
MISO
SCLK
I2C ID EEPROM
I2C ID EEPROM
3V3
GPIO2
GPIO3
GPIO4
Ground
GPIO17
GPIO27
GPIO22
3V3
GPIO10
GPIO9
GPIO11
Ground
ID_SD
GPIO5
GPIO6
GPIO13
GPIO19
GPIO26
Ground
5V
5V
Ground
GPIO14
GPIO15
GPIO18
Ground
GPIO23
GPIO24
Ground
GPIO25
GPIO8
GPIO7
ID_SC
Ground
GPIO12
Ground
GPIO16
GPIO20
GPIO21
Power
UART0_TXD
UART0_RXD
PCM_CLK
CE0_N
CE1_N
I2C ID EEPROM

PART 2:

WRITING CODE

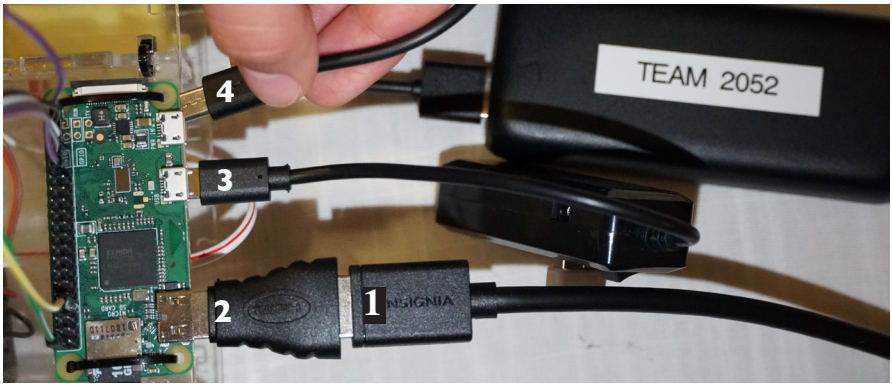
Instructions for Using a Chromebook (in-class method):

If you are using a Chromebook supplied to you at a robot camp, the setup has been completed for you to connect to the Raspberry Pi over the network. On the Chrombook, open the VNC app from the bottom of the screen. Find the number on your Raspberry Pi. When prompted to connect, enter the IP address of the Raspberry Pi. It should be "192.168.2.___," where the last number is the number on your Pi. Once connected you will see the Pi desktop.

Instructions for Using a Monitor (recommended at-home method):

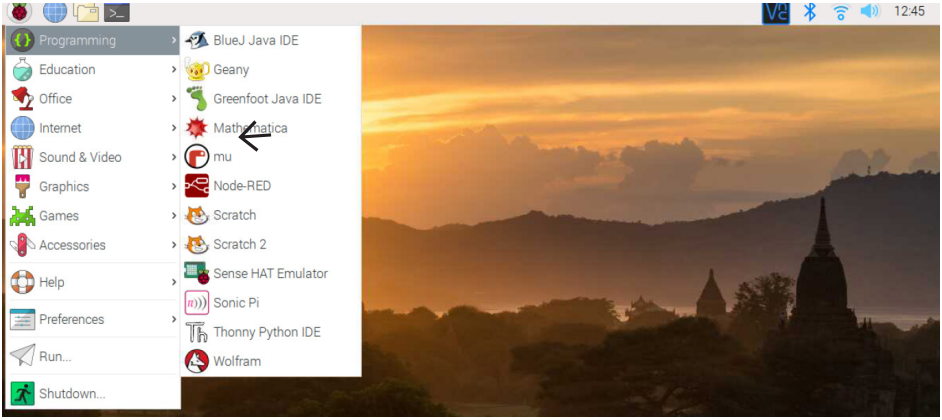
To program your robot you will need to connect your Raspberry Pi to a monitor.

1. Attach the HDMI adapter to the HDMI cable from the monitor.
2. Plug the HDMI cable into the Raspberry Pi.
3. Plug the USB hub into the Raspberry Pi. Plug your keyboard and mouse into the hub.
4. To power the Pi, plug the phone charger battery into the Pi. It may take a couple of minutes to start up. You must have your monitor plugged in before you power on the Pi, or the Pi will not show anything on the screen.



Writing Your First Program

In our class, we will use a program called "Mu." This comes preinstalled with Raspian Buster. If you are using a different version of Raspian, you may use "Python Idle" or "Thonny." You can open Mu by clicking the Raspberry menu, then "Programming."

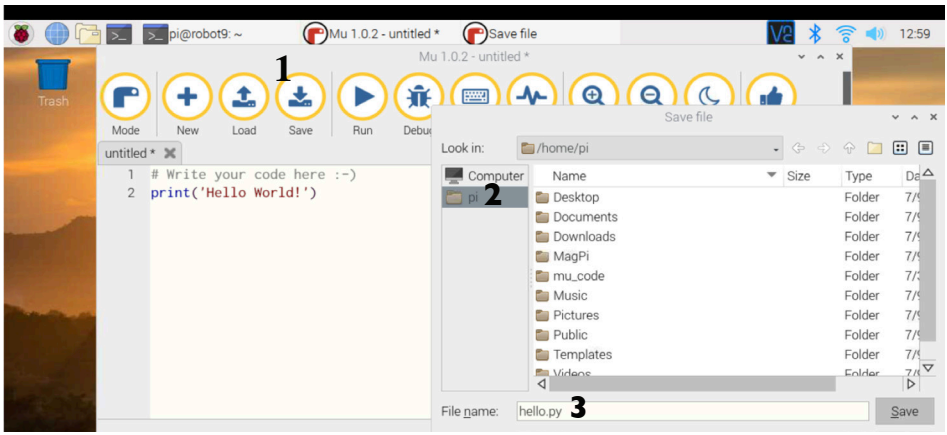


If Mu asks you to "Select Mode" when opening, select "Python3."

Let's write a sample program. In Mu, enter the following code:

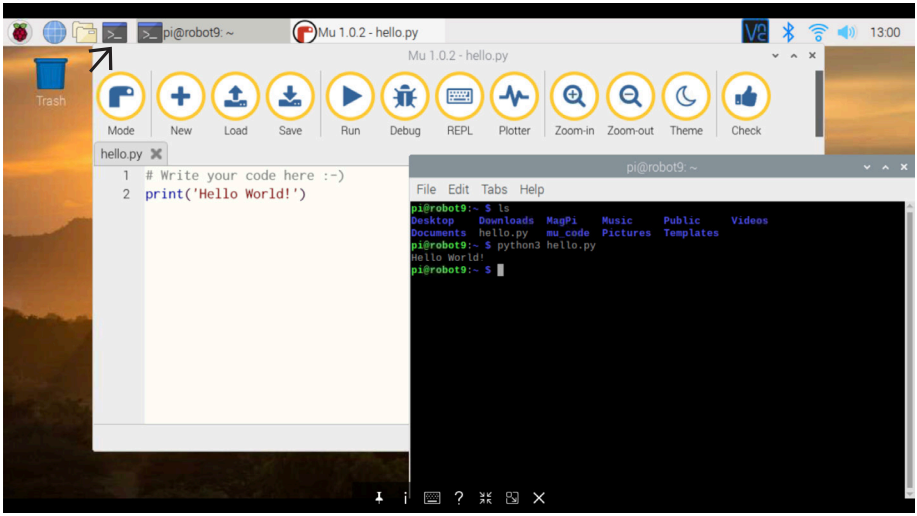
```
print('Hello World!')
```

Save the file(1). Mu will want to save the file in a Mu folder, but to make things easier for us, select the "Pi" folder (2).
Save this file as "hello.py"(3).



To run your test program, you could hit the "Run" button, however that won't work for our robot code. Instead, click on the terminal icon at the top of the screen (see photo on page 14). This will open a command-line terminal (a black screen). You can type the "ls" command (short for "list"), which will give you a listing of all the files and folders in the current path. You should be in the "Pi" folder when the screen opens and should see our "hello.py" program on the list. If the "hello.py" program is not on the list, try saving it again under that name and make sure you are saving it to the "Pi" folder.

To execute the program, type "python3 hello.py". This command tells python3 to execute our test program in the hello.py file. You should see "Hello World!" print on the screen.



One last fun test program, try this:

```
for x in range(25):
    print('I love robots!')
```

Save the file again, and execute it.

Now you're ready to program your robot!

TIPS:

- Text in the boxes below is code. Enter this text in the editor when programming.
- The "#" character is a comment. Comments help explain the code. You don't have to add comments to your code. Adding them won't change how your code runs, but can help you remember how your code works when you are testing it.
- You will be writing your code in a language called "Python." Spacing matters in Python. When your code is indented (tabbed), the left side of the code must line up. Otherwise it might not run correctly.
- If the motors don't run when tested, refer to page 22.

STARTING TO CODE:

All our code will be in a single file for programming our robot. However, we will be using some code written by other programmers. We start our file by importing the code from other Python developers using the import statement.

```
import time  #allows us to pause the code using the sleep command
import curses  #used to get keyboard presses
import datetime  #gives us the current date and time
import gpiozero #makes it easy to use pins on the Raspberry Pi
```

Next we need to create some variables to keep track of the values we need to run the robot. We will create 4 variables to name the pins we are using to control the motors. Then 4 variables that will control the forward and backward speeds for the motor. Finally, we create a servo variable, to control the balloon popper. You don't need to worry about what the numbers mean right now, just copy the code to match. Add this code below your imports:

```
#Make variables to keep track of what the pins are connected to
LEFT_FORWARD_PIN = 6
LEFT_BACK_PIN = 13
RIGHT_FORWARD_PIN = 19
RIGHT_BACK_PIN = 26

#Make variables to control the motors' forward and backward signals
leftForward = gpiozero.PWMOutputDevice(LEFT_FORWARD_PIN, True, 0, 1000)
leftBack = gpiozero.PWMOutputDevice(LEFT_BACK_PIN, True, 0, 1000)
rightForward = gpiozero.PWMOutputDevice(RIGHT_FORWARD_PIN, True, 0, 1000)
rightBack = gpiozero.PWMOutputDevice(RIGHT_BACK_PIN, True, 0, 1000)


#Make a variable to control our balloon popper servo
servo = gpiozero.Servo(4, 0, .0005, .0025)
```

We are almost ready to test our motors. The following code will be used to run our first test.

Main is a method. This is a block of code with a name. We will write more methods later, and talk more about how methods are used.

The "def" word is used to define a method. The "screen" between the parentheses is called a parameter, which is a variable that is passed to the method. When Curses calls our Main method via the Wrapper method, it will send us a reference to the screen window, which will show up in the "screen" parameter. If this sounds complicated, don't worry. There will be simpler examples later.

The motor has "forward" and "backward" wires. To run the motor, you need to send a speed to one wire on the motor and turn off the other wire. In the example below, we will tell the forward wire on the motor to run at 100% power by setting the value = 1, but the other side to not run at all by setting the value to 0.

```
#The Main method is a common way to start your code
def main(screen):
        #Tell the left motor to drive forward at 100% speed
        leftForward.value = 1
        leftBack.value = 0
        #Pause for 3 seconds
        time.sleep(3)
        #Tell the left motor to stop
        leftForward.value = 0
        leftBack.value = 0

#This must be the last line of code in the program
#The Wrapper method will start the main method for us
#It also reset the screen to normal mode when the program ends
curses.wrapper(main)
```

Now complete step 11 on page 9.

RUN YOUR CODE – Refer to your test program on page 13 for help. (save this file as "robot.py")

Did the left motor run forward? If not, see the troubleshooting section (page 21) on switching your motor wires. Switch your wires, and run the code again. Do this until the left motor on your robot drives forward when you run the code. Once the left motor is working correctly, switch your code above to use the "rightForward" and "rightBack" motors, and repeat the steps to make sure the right motor is wired correctly.

Time to "MOVE"

It can be a pain to write two lines of code to control the forward and backward sides of the motor, which is double if you want to control both the left and right motor. To avoid writing 4 lines of code every time we want to drive, let's make a new method.

The "Move" method will accept two parameters, a speed for the left wheel, and a speed for the right wheel. We use an "if" statement to determine if the motor should be forward or backward. Finally, this method will return the current time when the motors were changed. This will be important later.

IMPORTANT: add this method just above the "Main" method you created in in the last step

```
#Move method to make it easier to control the motors
def move(left, right):
    if left > 0: #we want to move the left wheel forward
        leftForward.value = left
        leftBack.value = 0
    else: #we want to move backwards or if value is 0, it will stop
        leftForward.value = 0
        leftBack.value = -left

    if right > 0: #we want to move the right wheel forward
        rightForward.value = right
        rightBack.value = 0
    else: #we want to move backwards or if value is 0, it will stop
        rightForward.value = 0
        rightBack.value = -right
    return time.time() #send back the current time
```

Now that you have your new move method, it is time to test your motors again. Change your main method to call the move method.

```
#The Main method is a common way to start your code
def main(screen):
        #Tell the left motor to drive forward at 100% speed
        move(1,0)
        #Pause for 3 seconds
        time.sleep(3)
        #Tell the left motor to stop, and run the right motor forward
        move(0,1)
        #Pause for 3 seconds
        time.sleep(3)
        #Tell the right motor to stop
        move(0,0)
```

RUN YOUR CODE

If the code ran correctly, you can play around with changing the main method to test that the correct motors run backward. Better yet, it may be helpful to build a method to test all the motor driving directions, then call the new test method from the Main method. Keep the main method as the last method in your code and create a new method above it. Change the main method to only have one line of code to call the test method.

```
def selfTest():
        move(1, 0)
        time.sleep(1)
        move(-1, 0)
        time.sleep(1)
        move(0, 1)
        time.sleep(1)
        move(0, -1)
        time.sleep(1)
        move(1, 1)
        time.sleep(1)
        move(-1, -1)
        time.sleep(1)
        move(0, 0)

#The Main method is a common way to start your code
def main(screen):
        selfTest()
```

RUN YOUR CODE

Did it work? Now you have a method you can call with a single line of code to test all your motors. This can be helpful to find a bug, or issue, in your code. If the test method works, then the bug must be in your new code. If the test code stops working, maybe a wire was disconnected.

Keyboard Input

Before we can drive our robot, we need to be able to know when someone pressed a key on the keyboard. Let's make a method to setup our keyboard so we can use it from our code. This is somewhat complicated code. Basically this will let our code keep running without waiting for the user to hit a key, but we can check to see if a key was pressed at

any time. We will call this method when our program starts, and only call it once.

```
#this will setup the keyboard to work well for driving
def initKeyboard(screen):
    curses.cbreak(); #assume the enter key has been pressed after every
key
    screen.nodelay(True); #don't wait for a user to enter a key
    screen.keypad(True) #allow input from the arrow keys on the keyboard
```

Drive!

In this code, we are going to check to see if the arrow keys are pressed, then drive in that direction. We will do this by making a anew method called "Drive."

```
#watch for keys to be pressed, then do stuff
def drive(screen):
    lastMove = 0 #keep track of the last time a key was pressed
    keepGoing = True
    while keepGoing: #program will keep looping while keepGoing is true
        key = screen.getch()
        if key == ord('q'):
            move(0,0) #stop the motors
            keepGoing = False #stop looping
        elif key == curses.KEY_UP:
            lastMove = move(1,1) #both wheels forward
        elif key == curses.KEY_DOWN:
            lastMove = move(-1,-1) #both wheels backward
        elif key == curses.KEY_RIGHT:
            lastMove = move(0,1) #only the right wheel forward
        elif key == curses.KEY_LEFT:
            lastMove = move(1,0) #only the left wheel forward

       #stop motors if no key has been pressed in the past quarter of a
second
       #this allows people to hit the arrow keys quickly, but drive con-
stantly
        if time.time() - lastMove > .25:
            move(0,0)
```

All that is left is to change our main method to use our new driving code. We need to pass the screen value from the main parameter to the initKeyboard and drive methods.

```
#The Main method is a common way to start your code
def main(screen):
        initKeyboard(screen)
        drive(screen)
```

RUN YOUR CODE

Controlling a servo is pretty easy. Let's make a method for it.

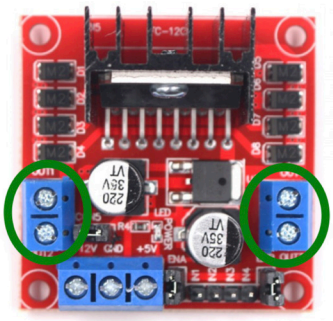
```
def movePopper(pos):
    if pos < 0:
        servo.min() #move pin to left
    elif pos > 0:
        servo.max() #move pin to right
    else:
        servo.mid() #move pin to center
```

Now in our "Drive" method, we need to check for additional values, and then tell our popper to change positions. This shows the updated "Drive" method, which looks for keys 1, 2, and 3 to be pressed, then changes the popper's position. You don't need to change your main method this time.

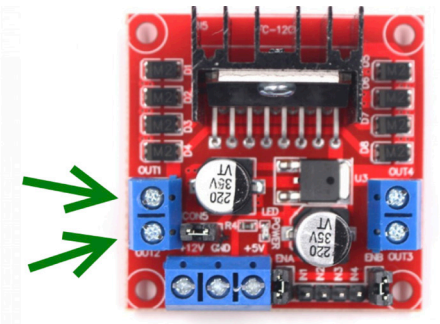
```
#watch for keys to be pressed, then do stuff
def drive(screen):
    lastMove = 0 #keep track of the last time a key was pressed
    keepGoing = True
    while keepGoing: #program will keep looping while keepGoing is true
        key = screen.getch()
        if key == ord('q'):
            move(0,0) #stop the motors
            keepGoing = False #stop looping
        elif key == curses.KEY_UP:
            lastMove = move(1,1) #both wheels forward
        elif key == curses.KEY_DOWN:
            lastMove = move(-1,-1) #both wheels backward
        elif key == curses.KEY_RIGHT:
            lastMove = move(0,1) #only the right wheel forward
        elif key == curses.KEY_LEFT:
            lastMove = move(1,0) #only the left wheel forward
        elif key == ord('1'):
            movePoppper(-1)
        elif key == ord('2'):
            movePopper(0)
        elif key == ord('3'):
            movePopper(1)
        #stop motors if no key has been pressed in the past quarter of a second
        #this allows people to hit the arrow keys quickly, but drive constantly
        if time.time() - lastMove > .25:
            move(0,0)
```

TROUBLESHOOTING

Problem: Your right motor is turning when the left is supposed to turn (or vice versa).
Solution: Swap the wires on the left and right side of the H-Bridge.



Problem: Your left motor is running when it should, but it is driving backward instead of forward.
Solution: Flip your wires but keep them on the same side of the H-Bridge (or vice versa).



Problem: Your motor never turns.
Solutions: Double check all wires are connected as shown in the wiring diagram (page 8). Ensure the batteries are installed and the power switch on the battery pack (if there is one) is on. Review your code and compare it to the instruction code to look for mistakes.

PART 3:

APPENDIX

What Comes in Our Kit:

- 1 - Raspberry Pi
- 1 - SD Card
- 1 - Robot Chassis
- 2 - Wheels (Including Hubs)
- 2 - 6v Motors
- 1 - Mini Breadboard
- 1 - H Bridge
- 1 - USB cable
- 1 - Battery holder
- 1 - Portable Phone Charger Battery Pack
- 1 - Servo
- 1 - Marble Wheel (Marble and 3D-Printed Holder)
- 1 - Thumbtack Holder
- 1 - Thumbtack
- 1 - Snips
- 1 - Phillips Screwdriver
- Jumper Wires

Raspberry Pi
A Raspberry Pi 3 is recommended. However, whichever Raspberry Pi you choose, please ensure you have at least 2 standard USB Ports, WiFi, Full HDMI connector, and a cable that fits the powered USB on your device. A Raspberry Pi Zero will require adapters for the mini HDMI and Micro USB, a USB Hub, a WiFi dongle, and a soldered pin head. With a Raspberry Pi Zero W you will not need a WiFi dongle.

Starter kits are available. Do some research and find the one that looks the most interesting. **Here is one option:**
https://www.amazon.com/dp/B07BCC8PK7
Raspberry Pi only:
https://www.amazon.com/dp/B01LPLPBS8

SD Card
A 16GB card is recommended. However, an 8GB card will also do. You can purchase an SD Card with a USB adapter if your laptop does not have an SD card reader.
https://www.amazon.com/dp/B079H6PDCK

Robot Chassis, Wheels, and Motors
This kit looks like a car:
https://www.amazon.com/dp/B01LXY7CM3
This one looks more like the base for a robot:
https://www.amazon.com/dp/B07DMCWRJ1

There are many robot kits available. Any kit that comes with the yellow 6v motors will work. The robot shown in this book will use only two motors for simplicity. You can purchase a kit with 4 motors and a steering component if you are prepared to do some extra work on your own. **Motors are available here:**
https://www.amazon.com/dp/B07RX7LBJ7

You can also purchase the wheels separately or 3D print them. There are a number of wheel designs and chassis on
https://www.thingiverse.com

Breadboard
Breadboards make life easy. You only need one, but they are very useful and inexpensive. **Here is a 6-pack:**
https://www.amazon.com/dp/B016Q6T7Q4

H Bridge
This is the part that controls your motors. The H Bridge makes it possible for the motors and Raspberry Pi to share power. It will also allow you to use bigger motors with more power in the future, if you like. You only need one.
https://www.amazon.com/dp/B014KMHSW6
A five-pack is also available if you plan to build more projects in the future.
https://www.amazon.com/dp/B078D7DR9V

USB Cable
You will need a USB to Micro USB B. Shorter is better. This link will show you what the ends look like. **Note:** if you are using a Raspberry Pi 4, you will need a different cable to power your Pi.
https://www.amazon.com/dp/B003YKX6WM

Servo
We suggest buying these in multi-packs to lower the cost per unit.
https://www.amazon.com/dp/B07R6LT9P8

Battery Holder and Batteries
The robot motors require 6 volts. That is 4 AA batteries. **Here is a good holder:**
https://www.amazon.com/dp/B00VUM5KX4
More expensive holders with a power switch are also available. Don't forget to buy batteries too.

If you plan to do many projects with your Raspberry Pi, you may want to invest in larger rechargeable batteries. We recommend two 18650 batteries, a dual phone battery pack with power leads, and a charger.

Portable Phone Charger Battery Pack and AC Plug
Any phone battery pack should work. We recommend a battery pack that has at least 1A output, 2A is better. Purchase a small one to ensure it fits on the robot. **Here is a sample:** https://www.amazon.com/dp/B00KG45W08

Jumper Wires
You can never have enough jumper wires.
https://www.amazon.com/dp/B07PD9V4LD

Setting up your Raspberry Pi at Home
If you are new to Raspberry Pi, you should review these instructions for setting up your Raspberry Pi:
https://projects.raspberrypi.org/en/projects/raspberry-pi-setting-up

The getting started instructions under "Setup SD Card", the tutorial recommends installing the Noobs image on your SD card. For this class we used the "Raspian Buster with desktop and recommended software" image. We used this image because it includes the development tools we use to write Python code. You can download the zip for for Raspian Buster here. Raspian also includes many common libraries pre-installed, some of which we use in the robot camp. The steps for setting up the card are the same, but we copied the files from the Buster zip file, instead of the Noobs zip file. You can download the same image we used here:
https://www.raspberrypi.org/downloads/raspbian/

Chromebooks vs. Monitors

For our robot camp, we used chromebooks. This is a somewhat advanced setup. We did this because it was easier to borrow and transport 15 Chomebooks instead of 15 monitors, keyboards, and mice. For doing Raspberry Pi projects at home, we recommend you connect your Raspberry Pi to a keyboard and mouse, as well as a monitor or TV. You can get an old monitor on craigslist or Facebook Marketplace for $10-20. If your monitor only has a VGA connection, you can purchase an HDMI to VGA adapter for around $10.

However, if you really want to connect to your Raspberry Pi with no wires here are the high level steps you should take. This list should help you google for instructions or step-by-step videos on how to setup your Pi, your home network, and your computer. You will need to do the Pi configuration while connected to a keyboard, mouse, and monitor. After completing these steps you should then be able to connect to your Raspberry Pi remotely from any computer on the same WiFi network.

1. Change the default password on your Raspberry Pi
2. Enable SSH and VNC
3. Connect to your WiFi network at home
4. Log-in to your home router, find the connected Pi, and configure your router so it always gives your Pi the same IP Address
5. Reboot your Pi, connect to the WiFi network
6. Check the IP Address of your Pi to make sure it was set the IP you configured
7. Install a VNC Viewer on your PC, Mac or tablet
8. Make sure your computer is connected to the same WiFi network as your Pi
9. Connect to your Pi using the VNC Viewer using the IP Address of your Pi

For more information, visit:

WWW.TEAM2052.COM/ROBOTCAMP