How To Create Your Own Mini-Robot

1.Building the Robot:

Robot Items & Parts:

In order to create your very own mini robot you will need the following parts and pieces that are listed below:

- Wheels (x2)
- Robot Frame
- Batteries (x2)
- Battery Holder
- Motors (x2)
- Raspberry Pi
- Bread Board
- Motor Driver Controller Board
- Ultrasonic sensor
- Raspberry Pi camera V2
- Placeholder

Getting Your Build On:

Now that you have acquired all of the items listed above (hopefully) you can now start building your robot. HOORAY! For starters, you should attach the motors to the robot frame. Place one in each hole on both sides of the robot. After you have managed to stick the motors on the frame you should next connect the wheels to the motors as well. One wheel for each side. Afterward, place your Raspberry Pi on the holder that should be located on top of the frame. If you have managed to do all of this then you will

have created the main outline for your mini robot and you can now continue to begin wiring everything up.

2. Wiring The Robot:

Wires: Now you may be thinking, “Wow, I have all these pieces but how do they work together to move my robot?”. Well, To answer your question we will have to use these magical items called wires. Wires can basically connect to anything (that is electronic of course) and help allow two different devices get....from each other. Knowing this you may have figured out that we will probably have to use a variety of wires in order to connect all these devices together. Don’t get overwhelmed by the number of different wires that are needed to attach all the devices together because we will explain them piece by piece.

Moving Motors: So to begin we will teach you how to wire the motors together using the Motor Driver Controller Board. You may use the diagram below as a reference when wiring this. Anyway, to start you will need to place two.... attached to the motors into the Motor A and Motor B outlet MDCB (Motor Driver Controller Board). Afterward, connect the red.... to the …..

- First, you need to connect a keyboard and mouse to the Pi using the USB ports on the Pi.
- You also need to connect the Raspberry Pi to the monitor with an HDMI cable.
- To power the Pi you need to connect a micro USB cord into the Pi in the designated spot then plug the other side into an outlet.

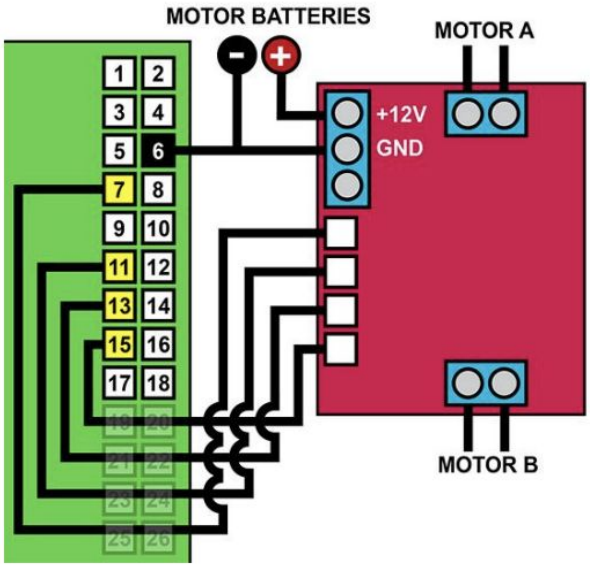
LED SetUp: To use LEDs with the Pi, you will need a breadboard along with resistors, the Pi itself, and of course, the LEDs.
Below is some sample C++ code to run the LEDs. This code would be put in the same file as your Python code:

```
import RPi.GPIO as
GPIO import time
GPIO.setmode(GPIO.BOARD)
GPIO.setup(7,GPIO.OUT)
```

```
GPIO.setup(11,GPIO.OUT)
GPIO.setup(13,GPIO.OUT)
GPIO.setup(15,GPIO.OUT)
GPIO.output(7,True)
time.sleep(1)
GPIO.output(7,False)
GPIO.output(11,True)
time.sleep(1)
GPIO.output(11,False)
GPIO.output(13,True)
time.sleep(1)
GPIO.output(13,False)
GPIO.output(15,True)
time.sleep(1)
GPIO.output(15,False)
GPIO.cleanup()
```

LED Lights Wiring: You will have to connect the pins with the holes in the breadboard using jumpers. Use the six pin to connect resistors to each of the lights. Then use jumpers to connect the breadboard to the GPIO pins on the Pi.

Motor Wiring: In the picture below, pins 7 and 11 control motor A, and pins 13 and 15 control motor B. Pin 6 is the power/battery connection. So 15, 7, 11, and 13 all connect into the control inputs on the H-Bridge.



Ultrasonic Sensor:

```
def
ultrasonic()
:
                #Gnd wire (Green)connects to breadboard/ connects to wire (Black)14

                    GPIO.output(TRIG, True) #Set TRIG as HIGH

                    time.sleep(0.00001) #Delay of 0.00001

                seconds GPIO.output(TRIG, False) #Set TRIG as LOW
```

Camera code:

```
        while GPIO.input(ECHO)==0: #Check whether the

ECHO is LOW pulse_start = time.time() #Saves the last

known time of LOW pulse



        while GPIO.input(ECHO)==1: #Check whether the

ECHO is HIGH pulse_end = time.time() #Saves the last

known time of HIGH pulse



        pulse_duration = pulse_end - pulse_start #Get pulse duration to

a variable



        distance = pulse_duration * 17150 #Multiply pulse

duration by 17150 to get distance

        distance = round(distance, 2) #Round to two decimal

points return(distance);



# uses the ultrasonic sensor to shoot out a sonic pulse then receive

the pulse to tell the distance between the robot and the object in the way.
```

3.Coding the Robot

The Code

Line 1: Sets up the keyboard, GPIO, And time for later use in the code
Lines 2-4: defines values for Echo and Trig for sonar
Lines 5-10: sets up GPIO board to power the motors

Lines 11-18: sets up the Sonar Gpio slots

Lines 19-20: Shuts off GPIO so the motors don't start on

Lines 22-26: defines the camera ON function

Lines 28-30: defines the camera OFF function

Lines 32-47: defines sonar function

```
1.  import keyboard, RPi.GPIO as GPIO, time
2.  TRIG = 16
3.  ECHO = 18
4.  GPIO.setmode(GPIO.BOARD)
5.  GPIO.setup(7,GPIO.OUT)
6.  GPIO.setup(11,GPIO.OUT)
7.  GPIO.setup(13,GPIO.OUT)
8.  GPIO.setup(15,GPIO.OUT)
9.  GPIO.setup(TRIG,GPIO.OUT)
10. GPIO.setup(ECHO,GPIO.IN)
11. pwm7 = GPIO.PWM(7, 60)
12. pwm11 = GPIO.PWM(11, 60)
13. pwm15 = GPIO.PWM(15, 60)
14. pwm7.start(0)
15. pwm11.start(0)
16. pwm13.start(0)
17. pwm15.start(0)
18. dutyCycle = 30
19. GPIO.output(TRIG, False)
20. time.sleep(2)
21.
22. def camON ():
23.     camera = picamera.PiCamera()
24.     camera.start_preview()
25.     camera.resolution = (1296, 730)
26.    camera.start_recording('my_video.h264')
27.
28. def camOFF ():
29.     camera.stop_recording()
30.     camera.stop_preview()
31.
32. def sonar ():
33.     GPIO.output(TRIG, True)
34.     time.sleep(0.00001)
```

```
35.     GPIO.output(TRIG, False)
36.
37.     while GPIO.input(ECHO)==0:
38.             pulse_start = time.time()
39.
40.     while GPIO.input(ECHO)==1:
41.             pulse_end = time.time()
42.
43.     pulse_duration = pulse_end - pulse_start
44.
45.     distance = pulse_duration * 17150
46.     distance = round(distance, 2)
47.     return(distance);
48.
49. def backwards ( int ):
50.     pwm7.ChangeDutyCycle(dutyCycle)
51.     pwm11.ChangeDutyCycle(0)
52.     pwm13.ChangeDutyCycle(dutyCycle)
53.     pwm15.ChangeDutyCycle(0)
54.     return;
55.
56. def forward ( int ):
57.     pwm7.ChangeDutyCycle(0)
58.     pwm11.ChangeDutyCycle(dutyCycle)
59.     pwm13.ChangeDutyCycle(0)
60.     pwm15.ChangeDutyCycle(dutyCycle)
61.     return;
62.
63. def turnleft ( int ):
64.     if dutyCycle > 50:
65.         pwm7.ChangeDutyCycle(0)
66.         pwm11.ChangeDutyCycle(dutyCycle / 4)
67.         pwm13.ChangeDutyCycle(0)
68.         pwm15.ChangeDutyCycle(dutyCycle)
69.     elif dutyCycle <= 50:
70.         pwm7.ChangeDutyCycle(dutyCycle)
71.         pwm11.ChangeDutyCycle(0)
72.         pwm13.ChangeDutyCycle(0)
```

Lines 49-54: defines backward functions

Lines 56-61: defines forwards function

Lines 63-74: Defines left turn function (Note that the turn has two settings for higher and lower speeds)

Lines 76-87: Defines right turn function (Again note that the turn has two settings for higher and lower speeds)

Lines 89-94: Defines stop function

Lines 96-97: Starts active code

Lines 98-99: Does a sort of emergency shutdown when Q is pressed

Lines 101-102: Activates sonar function

Lines 104-119: Takes the output of sonar function and goes forward, backward, or turns left depending on how close an obstacle is (note this basically makes the robot go forward unless it sees a wall) when W is pressed

```
73.         pwm15.ChangeDutyCycle(dutyCycle)
74.     return;
75.
76. def turnright ( int ):
77.     if dutyCycle > 50:
78.         pwm7.ChangeDutyCycle(0)
79.         pwm11.ChangeDutyCycle(dutyCycle)
80.         pwm13.ChangeDutyCycle(0)
81.         pwm15.ChangeDutyCycle(dutyCycle / 4)
82.     elif dutyCycle <= 50:
83.         pwm7.ChangeDutyCycle(0)
84.         pwm11.ChangeDutyCycle(dutyCycle)
85.         pwm13.ChangeDutyCycle(dutyCycle)
86.         pwm15.ChangeDutyCycle(0)
87.     return;
88.
89. def stopped ( int ):
90.     pwm7.ChangeDutyCycle(0)
91.     pwm11.ChangeDutyCycle(0)
92.     pwm13.ChangeDutyCycle(0)
93.     pwm15.ChangeDutyCycle(0)
94.     return;
95.
96. try:
97.     while True:
98.         if keyboard.is_pressed('q'):
99.             break
100.
101.          while keyboard.is_pressed('w'):
102.              test = sonar()
103.
104.              if test < 15:
105.                  backwards(dutyCycle)
106.              elif test <= 25:
107.                  turnleft(dutyCycle)
108.              elif test > 25:
109.                  forward(dutyCycle)
110.
```

Lines 111-112: Starts the backward function when S is pressed

Lines 114-115: Starts the right turn function when D is pressed

Lines 117-118: Starts the left turn function when A is pressed

Lines 120-124: Takes power from motors when no keys are pressed

Lines 126-127: Starts recording when V is pressed

Lines 129-130: Stops recording when C is pressed

Lines 132-160: Sets Motor power to a percentage when the corresponding key is pressed ex. Power set to 70% when 7 is pressed

```
111.        while keyboard.is_pressed('s'):
112.            backwards(dutyCycle)
113.
114.        while keyboard.is_pressed('d'):
115.            turnright(dutyCycle)
116.
117.        while keyboard.is_pressed('a'):
118.            turnleft(dutyCycle)
119.
120.        if keyboard.is_pressed('w') == False:
121.            pwm7.ChangeDutyCycle(0)
122.            pwm11.ChangeDutyCycle(0)
123.            pwm13.ChangeDutyCycle(0)
124.            pwm15.ChangeDutyCycle(0)
125.
126.        if keyboard.is_pressed('c')
127.            camON ()
128.
129.        if keyboard.is_pressed('v')
130.            camOFF ()
131.
132.        if keyboard.is_pressed('1'):
133.            dutyCycle = 10
134.
135.        elif keyboard.is_pressed('2'):
136.            dutyCycle = 20
137.
138.        elif keyboard.is_pressed('3'):
139.            dutyCycle = 30
140.
141.        elif keyboard.is_pressed('4'):
142.            dutyCycle = 40
143.
144.        elif keyboard.is_pressed('5'):
145.            dutyCycle = 50
146.
147.        elif keyboard.is_pressed('6'):
148.            dutyCycle = 60
```

162-163: Shuts down the code and cuts power to GPIO

```
149.
150.        elif keyboard.is_pressed('7'):
151.            dutyCycle = 70
152.
153.        elif keyboard.is_pressed('8'):
154.            dutyCycle = 80
155.
156.        elif keyboard.is_pressed('9'):
157.            dutyCycle = 90
158.
159.        elif keyboard.is_pressed('0'):
160.            dutyCycle = 100
161.
162.    finally:
163.     GPIO.cleanup()
```

Credits:

- https://www.explainingcomputers.com/rasp_pi_robotics.html
- https://github.com/frc2052/MiniRobot/blob/648cbad5f78a3bc5750277e9dd4426d8176b639f/camera.py