42 ELECTRONICS PRESENTS:

Intro to Robotics

Level B: Working with Sensors & Intermediate Programming

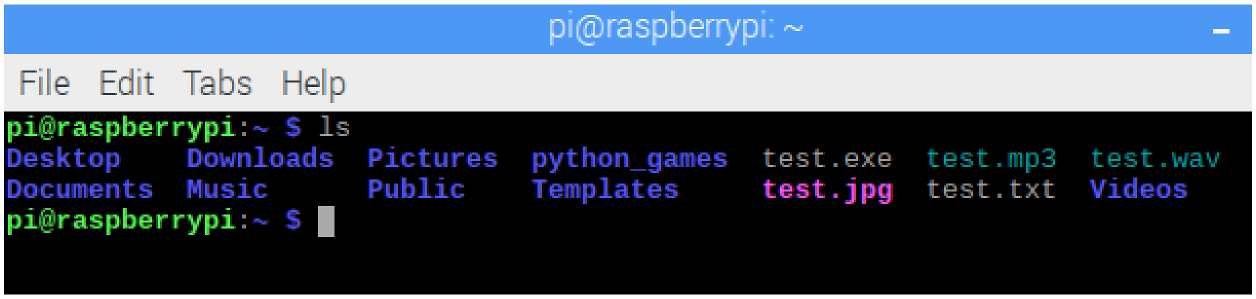
V20029

By Eric Feickert and Julie Feickert

PHOTOCOPYING & DISTRIBUTION POLICY

LISTING DIRECTORY CONTENTS

Now that you know which directory you're in, it would be helpful to know which files or folders are in this directory. Typing **`ls`** and pressing enter will display the contents from your current working directory. Folders will be blue with no file extension listed. Files will be different colors based on their file type which is determined by their file extension.



```
pi@raspberrypi:~ $ ls
Desktop     Downloads  Pictures  python_games   test.exe  test.mp3  test.wav
Documents   Music      Public    Templates      test.jpg  test.txt  Videos
pi@raspberrypi:~ $ 
```

The **`ls`** command is short for the word **`list`**. Even though the first character might look like the number 1 or an upper-case i depending on the font, it's actually a lower-case L.

CREATING A FILE

To create a file, you can use the Nano command-line text editor:

```
sudo nano robot.py
```

This will momentarily escalate your user permissions to a root user, create a new file named robot.py, and open it in Nano. You can make modifications to the file and when you're ready to save, hit CTRL-X (control key and x together), press y to save changes, confirm the filename, and press enter.

You can also use this same command to edit an existing file. If the file already exists then it will open in Nano. If the file does not exist, then the new file will be created, and it will open in Nano.

ACCESSING VALUES IN A LIST USING THEIR INDEX POSITION

The items in a list are also assigned the index value positions:

```
things = ['apple', 42, 'blue', 7]
```

index position 0 of **things** = **'apple'**

index position 1 of **things** = **42**

index position 2 of **things** = **'blue'**

index position 3 of **things** = **7**

These items are accessed just like the strings in the last section. By using the name of the list along with the index position of one of its items, you can get the entire value contained in that position of the list:

things[0] = 'apple'

things[3] = 7

The same format applies for multiple positions in a list as well:

things[2:4] = ['blue', 7]

things[1:] = [42, 'blue', 7]

REPLACING CHARACTERS IN A STRING

At some point you may want to print a string, while replacing one word for another. This can be accomplished by the **.replace()** command. Consider the following string:

```
hello = “Welcome name, it's good to see you name!”
```

It's much better to refer to people by their actual name rather than calling them "name". You can personalize this statement during printing by replacing the word name with someone's actual name:

```
hello = “Welcome name, it's good to see you name!”

print(hello.replace('name', 'Bob'))
```

This print statement will find the string named hello, locate any instances of the word name, replace those with **Bob**, and print the new string. Remember that the original string hello will not actually be modified, only printed with the requested replacements. If you wanted to save a personalized greeting, you could create a new string:

```
hello = “Welcome name, it's good to see you name!”

greeting_bob = hello.replace('name', 'Bob')
```

You now have a personalized greeting saved as a new string that can be used for whatever you need.

NOTE – In Python, a string can be enclosed by single-quotation marks (‘) or double-quotation marks (“). A single-quotation mark happens to be the same character as an apostrophe, so words in your string like it’s or that’s can cause problems if you’re enclosing your string in single-quotation marks. Python will interpret the apostrophe as the end of the string, and it will not know what to do with the remainder of the string. In these cases, enclosing your string in double-quotation marks will allow Python to properly interpret the apostrophe as part of the string.

STEP #7

Now that the **`check()`** function has been completed, it's time to create the main loop for the program. This loop will push a column high, set the col variable equal to the column number, run the row check function, and the return the column low. This column high loop will need to run once for each column.

Create a **`while True:`** loop that contains code to push the following GPIO pins high, set the column value, run the **`check()`** function, and return the GPIO pin low:

GPIO21 will represent column 0

GPIO20 will represent column 1

GPIO16 will represent column 2

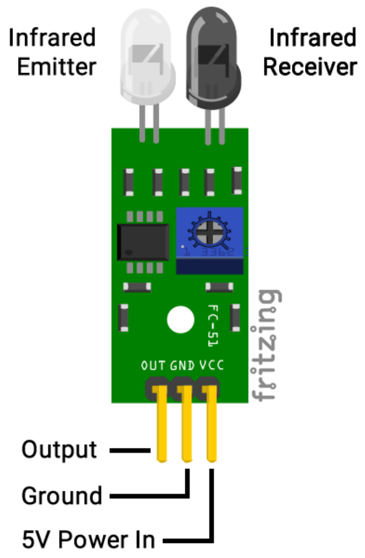
```
while True:

    GPIO.output(21,GPIO.HIGH)

    col = 0

    check()

    GPIO.output(21,GPIO.LOW)

    GPIO.output(20,GPIO.HIGH)

    col = 1

    check()

    GPIO.output(20,GPIO.LOW)

    GPIO.output(16,GPIO.HIGH)

    col = 2

    check()

    GPIO.output(16,GPIO.LOW)
```

INFRARED OBSTACLE SENSOR

An infrared or IR obstacle sensor is a device that uses infrared waves to determine if an object or surface is near the sensor. A 5V output line is used to report whether or not an object is near the sensor. This output line can be level shifted down to 3.3V and connected to an input on your Raspberry Pi, allowing a program to know if something has come close to the sensor.

The obstacle sensor included in your kit has an active low output, which means that when no object is detected the output line will have a high or 5V present. When an obstacle is detected this output line will go to ground or 0V. Your program must be coded to understand that high means no object has been detected, and that low represents that an object is near the sensor.

The sensor contains an infrared emitter and receiver, located next to each other.



The infrared emitter constantly sends out infrared signals, and receiver is constantly waiting to receive those signals back. If an object is close enough to the sensor, the emitter signals will bounce off the object and reflect back to the receiver. The sensor then knows that something is close, and the circuitry on the sensor board will pull the output pin low, indicating an object is near. If an object is not close enough to reflect the emitted signals back to the receiver, then the sensor will know that no object is near the sensor, and the output pin will remain high.

STEP #5

You will use the blue LED color to indicate when the line sensor senses a reflective object like white paper. Sensing a reflective object will cause the output from the obstacle sensor to go low. This low will flow through the level shifter and into **`GPIO20`**.

Add an **`elif`** statement below the **`if`** statement that will check to see if **`GPIO20`** is low or False. If so, that block will turn off the green LED and turn on the blue led. The additional code is highlighted below:

```
try:
   while True:
       if GPIO.input(21) == False:
           GPIO.output(green, GPIO.LOW)
           GPIO.output(red, GPIO.HIGH)
       elif GPIO.input(20) == False:
           GPIO.output(green, GPIO.LOW)
           GPIO.output(blue, GPIO.HIGH)
       else:
           GPIO.output(red, GPIO.LOW)
```

Here is the entire program with the line sensor additions:

```
import RPi.GPIO as GPIO
import time

rgb = [13,19,26]
red = 13
green = 19
blue = 26

GPIO.setmode(GPIO.BCM)
GPIO.setup(rgb, GPIO.OUT)
GPIO.setup(21, GPIO.IN)
GPIO.setup(20, GPIO.IN)

try:
    while True:
        if GPIO.input(21) == False:
            GPIO.output(green, GPIO.LOW)
            GPIO.output(red, GPIO.HIGH)
        elif GPIO.input(20) == False:
            GPIO.output(green, GPIO.LOW)
            GPIO.output(blue, GPIO.HIGH)
        else:
            GPIO.output(red, GPIO.LOW)
            GPIO.output(blue, GPIO.LOW)
            GPIO.output(green, GPIO.HIGH)
        time.sleep(.1)

except KeyboardInterrupt:
    GPIO.cleanup()
```

STEP #3

Its now time to relocate the jumper wires that will connect to the ultrasonic sensor. Move three jumper wires to their new locations, while also adding one long jumper wire for the trigger line:

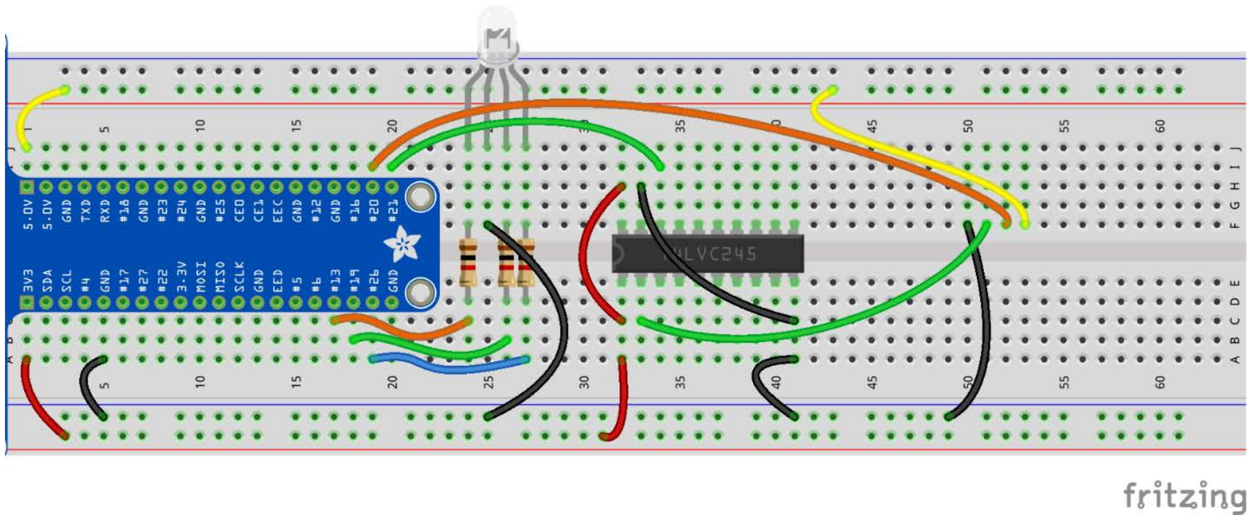
Relocate – 5V – Move connection from C50 to F53

Relocate – Output (Echo)– Move connection from B48 to F51

Relocate – Ground – Move connection from A49 to F50

Add long jumper wire – Input (Trigger) – Add jumper between I19 and F52

The circuit will look like this after your relocations and addition:



QUESTIONS FOR UNDERSTANDING

1. When text is drawn using the **`draw.text`** command, is it sent to the image buffer or written directly to the screen?

2. In (64, 17), what is the value of X and Y? Approximately where is that coordinate located on the 128x64 screen below:

3. What changes must be made to the program from Activity #3 to reorder the lines on the screen to match the layout below?

```
Temp: 80.28
** ALERT **
Switch is OFF
```

Answers can be found on the next page.

STEP #13

The delay is over, and you are now ready to capture the range to the user. You can do this by accessing the **`average()`** function inside your **`ultrasonic.py`** file, and setting the result equal to a variable named **`distance`**.

Next, you will create a variable named **`diff`** that will hold the absolute value of the random **`target`** variable minus the **`distance`** returned from the **`average()`** function. This will be expressed as **`diff = abs(target-distance)`**.

Now you will calculate the range of values that will get a green LED based on the difficulty that's been selected. Create a variable named **`window`** that equals **`10 * skill`**. This means:

Easy mode, **`skill = 2`** so **`window`** will equal 10 X 2 or **`20`**

Hard mode, **`skill = 1`** so **`window`** will equal 10 X 1 or **`10`**

Add these program tasks with the following lines of code:

```
        update_display()
        time.sleep(delay)

        distance = ultrasonic.average()
        diff = abs(target-distance)
        window = 10 * skill
```

STEP #14

It's now time to light the LED using the **`diff`** and **`window`** variables. If **`diff`** is smaller than or equal to **`window`**, the user distance is good, so the LED should turn green. If not, the user distance that round was larger than the window, so the LED should turn red. This can be accomplished by using a simple **`if/else`** block:

```
        distance = ultrasonic.average()
        diff = abs(target-distance)
        window = 10 * skill

        if diff <= window:
            GPIO.output(green, GPIO.HIGH)
        else:
            GPIO.output(red, GPIO.HIGH)
        draw.rectangle((0,0,width,height), outline=0, fill=0)
        draw.text((0, 0), "Target = %s" %target,  font=font, fill=255)
        draw.text((0, 22), "Press pad",  font=font, fill=255)
        draw.text((0, 44), "to start",  font=font, fill=255)
        update_display()


        while GPIO.input(cap1)==GPIO.input(cap2)==GPIO.input(cap3)==GPIO.input(cap4)==False:
            time.sleep(.05)

        if GPIO.input(cap1) == True:
            delay = 1
        if GPIO.input(cap2) == True:
            delay = 2
        if GPIO.input(cap3) == True:
            delay = 3
        if GPIO.input(cap4) == True:
            delay = 4

        draw.rectangle((0,0,width,height), outline=0, fill=0)
        draw.text((0, 0), "Target = %s" %target,  font=font, fill=255)
        draw.text((0, 22), "Capturing range",  font=font, fill=255)
        if delay == 1:
            draw.text((0, 44), "in %i second" %delay,  font=font, fill=255)
        else:
            draw.text((0, 44), "in %i seconds" %delay,  font=font, fill=255)
        update_display()
        time.sleep(delay)

        distance = ultrasonic.average()
        diff = abs(target-distance)
        window = 10 * skill

        if diff <= window:
            GPIO.output(green, GPIO.HIGH)
        else:
            GPIO.output(red, GPIO.HIGH)

        draw.rectangle((0,0,width,height), outline=0, fill=0)
        draw.text((0, 0),  "Target = %s" %target,  font=font, fill=255)
        draw.text((0, 22),  "Player = %.0f" %distance,  font=font, fill=255)
        draw.text((0,44),   "Diff = %.0f" %diff,  font=font, fill=255)
        update_display()
        time.sleep(5)
        GPIO.output(rgb, GPIO.LOW)

except KeyboardInterrupt:
    disp.clear()
    disp.display()
    GPIO.cleanup()
```