Quick Reaction Game

In this resource, you're going to make a quick reaction game using a few electronic components and a Python script. If you've never before used a breadboard, some buttons, and an LED, you might find it helpful to work through some of the exercises in Physical Computing with Python first. This will give you a better understanding of how to control components with the Raspberry Pi's GPIO pins.

Building the circuit

This is the circuit you are going to build, consisting of two push-to-make buttons and an LED.

1. Take one of your tactile buttons and push it into the holes on your breadboard, with one set of legs on row **H** and one set of legs on row **J**.

2. Repeat the last step with the second button, placing it at the other end of the breadboard on the same row.

3. Place an LED with the longer leg above the ridge in the breadboard in **D16** and the shorter leg in **D15**. The numbering will depend on your breadboard so make sure that you check the diagram below.

4. Next push one leg of the resistor into the same column (**15**) as the short leg of the resistor and the other leg into a hole along the blue strip.

5. Now it's time to add the jumper wires. Start by taking two male-to-male jumper wires and placing one end in a hole next to the outside leg of the left hand button, and the other end in a hole along the blue strip. Repeat this step with the right hand button.

6. Then with a male-to-female jumper wire, connect **GPIO14** to a hole on the breadboard in line with the other leg of the left hand button. Repeat this step for the right hand button, only this time connecting it to **GPIO15**.

7. Using another male-to-female jumper wire, connect **GPIO4** to a hole on the breadboard in line with the long leg of the LED.

8. Finally, connect a **GND** GPIO pin to the blue strip on the breadboard with the remaining male-to-female jumper wire.

Controlling the light

When programming, it makes sense to tackle one problem at a time. This makes it easier to test your project at various stages.

1. Click on the `Menu` > `Programming` > `Python 3 (IDLE)`

2. Create a new text editor file by clicking on `File` > `New File`

3. Save this file as **reaction.py** by clicking on `File` > `Save As`

4. First you will need to import the modules and libraries needed to control the GPIO pins on the Raspberry Pi. Type:

```
from gpiozero import LED, Button
from time import sleep
```

5. As you are outputting to an LED, you need to set up the pin that the LED connects to on the Raspberry Pi as an output. First use a variable to name the pin and then set the output:

```
led = LED(4)
```

6. Next add a line to turn the LED on:

```
led.on()
```

7. Now add a line to wait 5 seconds by typing:

```
sleep(5)
```

8. Then add a line to turn the LED off like this:

```
led.off()
```

9. Save the file by clicking on `File` > `Save`.

10. Finally, test that it works by click on `Run` > `Run Module` or by pressing `F5` on the keyboard.

If the LED does not come on for five seconds, go back and see if you can work out what went wrong. This is a very important skill in computing called **debugging**, which means finding and fixing errors or bugs in your code.

Adding an element of surprise

The object of the game is to see who can press the button first when the light goes out, so it would be better if the length of time it stayed on were random. You need to add and amend some lines of code in your Python program to make this happen.

1. If the file **reaction.py** is not already open in IDLE3 then open it by clicking on **File** and **Open**.

2. Underneath `from time import sleep` add the following line:

```
from random import uniform
```

Here, `uniform` allows for the random selection of a decimal (floating point) number from a range of numbers.

3. Then locate the line `sleep(5)` and amend it so that it reads:

```
sleep(uniform(5, 10))
```

4. Save your work by clicking on **File** and **Save**. Test that everything works by pressing `F5` to run your code.

Detecting the buttons

The LED is working; now you want to add functionality to your program so that when a button is pressed it is detected. That way you can record the players' scores to see who wins.

As with the last step, some code needs to be added to your current program.

1. With the file **reaction.py** open add the following variables underneath `led = LED(4)`:

```
led = LED(4)
right_button = Button(15)
left_button = Button(14)
```

2. Then underneath `led.off()` you can add a function that will be called whenever a button is pressed, which will tell you which **pin** the button was on:

```
def pressed(button):
    print(str(button.pin.number) + ' won the game')
```

3. To finish off, when either button is pressed, the function will be called. If the `right_button` is pressed, then you can send the string `'right'` to the `pressed` function. If the `left_button` is pressed, then you can send the string `'left'`.

```
right_button.when_pressed = pressed
left_button.when_pressed = pressed
```

Your completed code should now look like this

```
from gpiozero import LED, Button
from time import sleep
from random import uniform

led = LED(4)
right_button = Button(15)
left_button = Button(14)

led.on()
```

```
sleep(uniform(5, 10))
led.off()

def pressed(button):
    print(str(button.pin.number) + ' won the game')

right_button.when_pressed = pressed
left_button.when_pressed = pressed
```

Save your program and test it with a friend.

Get player names

Wouldn’t it be better if the program told you who has won instead of just which button was pressed? For this, you need to find out the players’ names. In Python, you can use **input** for this.

1. To find out the names of the players you can use `input` to ask the players to type in their names. Underneath the imported libraries and modules, type:

   ```
   left_name = input('left player name is ')
   right_name = input('right player name is ')
   ```

2. Now you can rewrite your pressed function, so that it can print out the name of the player who won.

```
def pressed(button):
    if button.pin.number == 14:
        print(left_name + ' won the game')
    else:
        print(right_name + ' won the game')
```

3. Save **reaction.py** and test your game to see if it works.

4. You might notice that the game doesn't quit when the button has been pushed. This can be fixed by adding an exit into the `pressed` function. First, add the following line to your imports.

```
from sys import exit
```

5. Then you can call `exit()` within your `pressed` function, once the prints have been completed.

```
def pressed(button):
    if button.pin.number == 14:
        print(left_name + ' won the game')
    else:
        print(right_name + ' won the game')
    exit()
```

What next?

- Can you put the game into a loop (you'll need to remove the `exit()`), so that the LED comes on again?
- Can you add scores for both players that accumulate over a number of rounds, and displays the players' total scores?
- How about adding in a timer, to work out how long it took the players to press the button after the LED turned off?