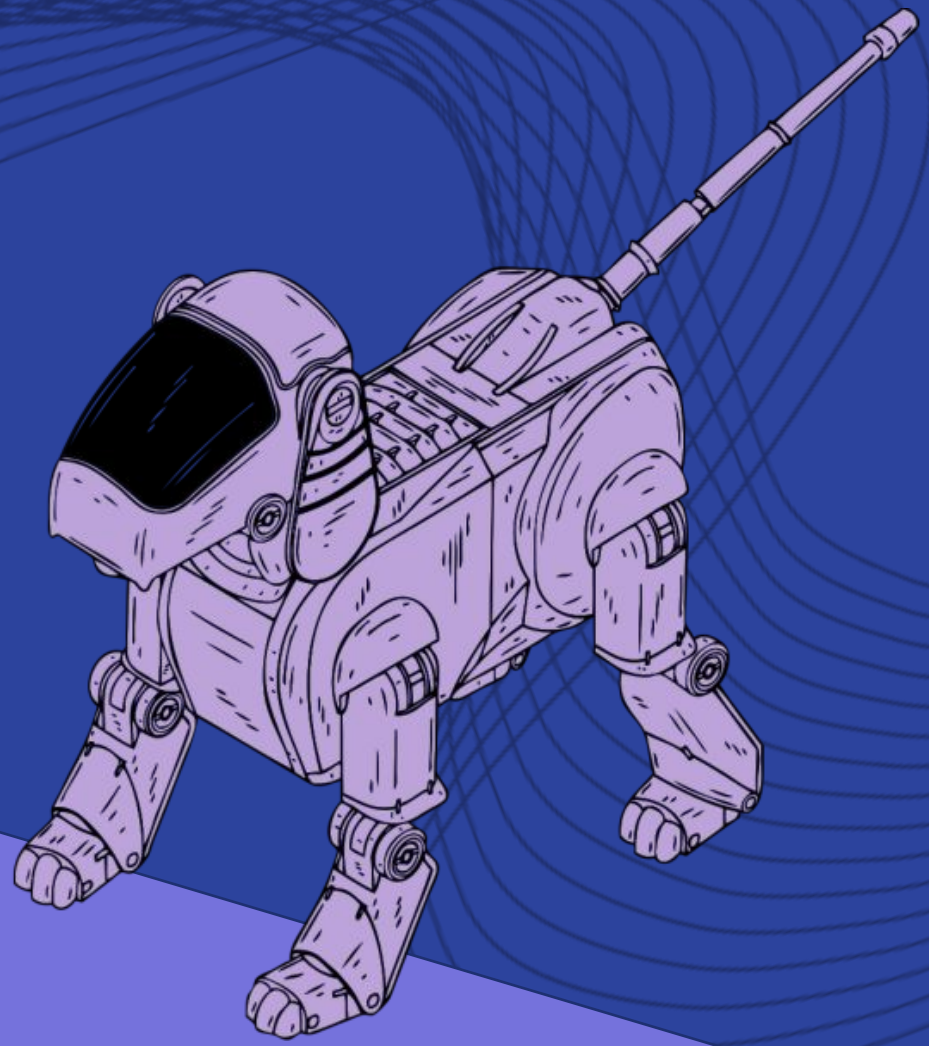
SCHOOL OF CODING

Reaction Time Game

Level 3 – Python

At Play



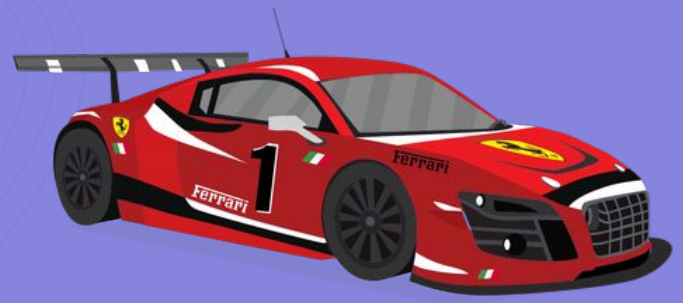
cair 4 YOUTH

Content created by Grace Bennett, Teri Dawkins and Natasha Parbhakar

Introduction

A Raspberry Pi Quick Reaction Game

Quick reflexes can be very useful! Jobs such as goalkeepers or racecar drivers will routinely test and practice their reaction speeds.



cair
4 YOUTH

cair
4 YOUTH

Task

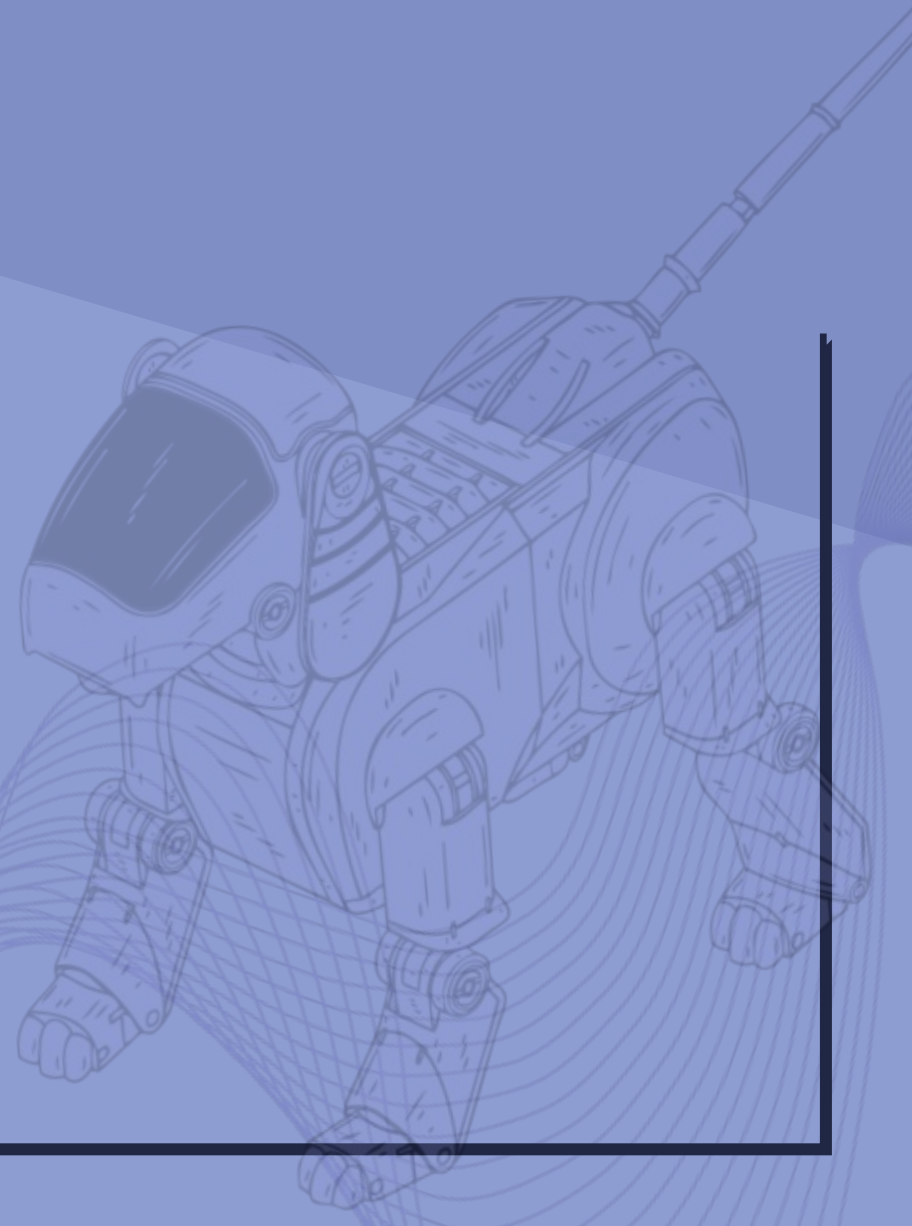
Quick Reaction Game

In this project, you will build a quick reaction time game that you can play against friends and family.

Process

Quick Reaction Game

- Learn how to wire a simple circuit.
- Write a programme to control the circuit.
- How to use variables to store information.
- How to get user information like a player’s name and use it in the game.



cair
4 YOUTH

What you will need

A Quick Reaction Game

- A Raspberry Pi
- A breadboard
- 1 LED
- 1 330 Ohm Resistor
- 4 Male-to-female jumper wires
- 2 Male-to-male jumper wires
- 2 Tactile push buttons

cair
4 YOUTH

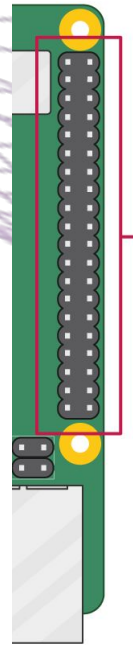
GPIO Pins

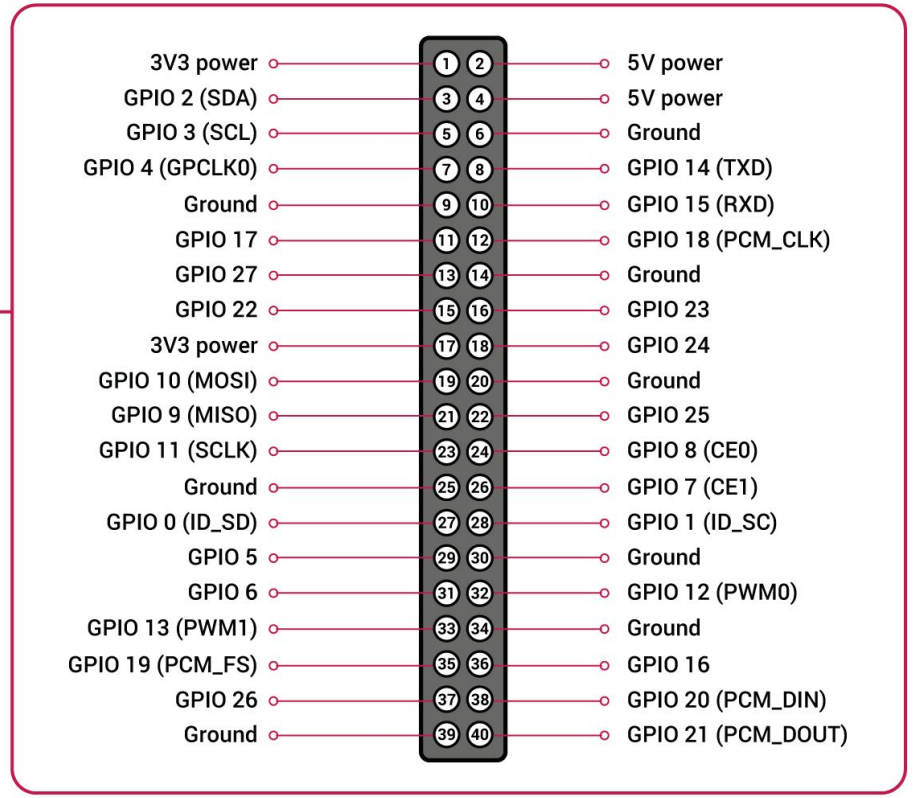
GPIO Pins are general-purpose input/output pins that allow us to interact with electronic components outside of our Raspberry Pi.

Each Raspberry Pi has a unique GPIO pinout, so before you wire up your components, find the appropriate pinout for your Raspberry Pi.

You can find these in the Raspberry Pi Documentation.

On the right is a pinout for the Raspberry Pi 4.



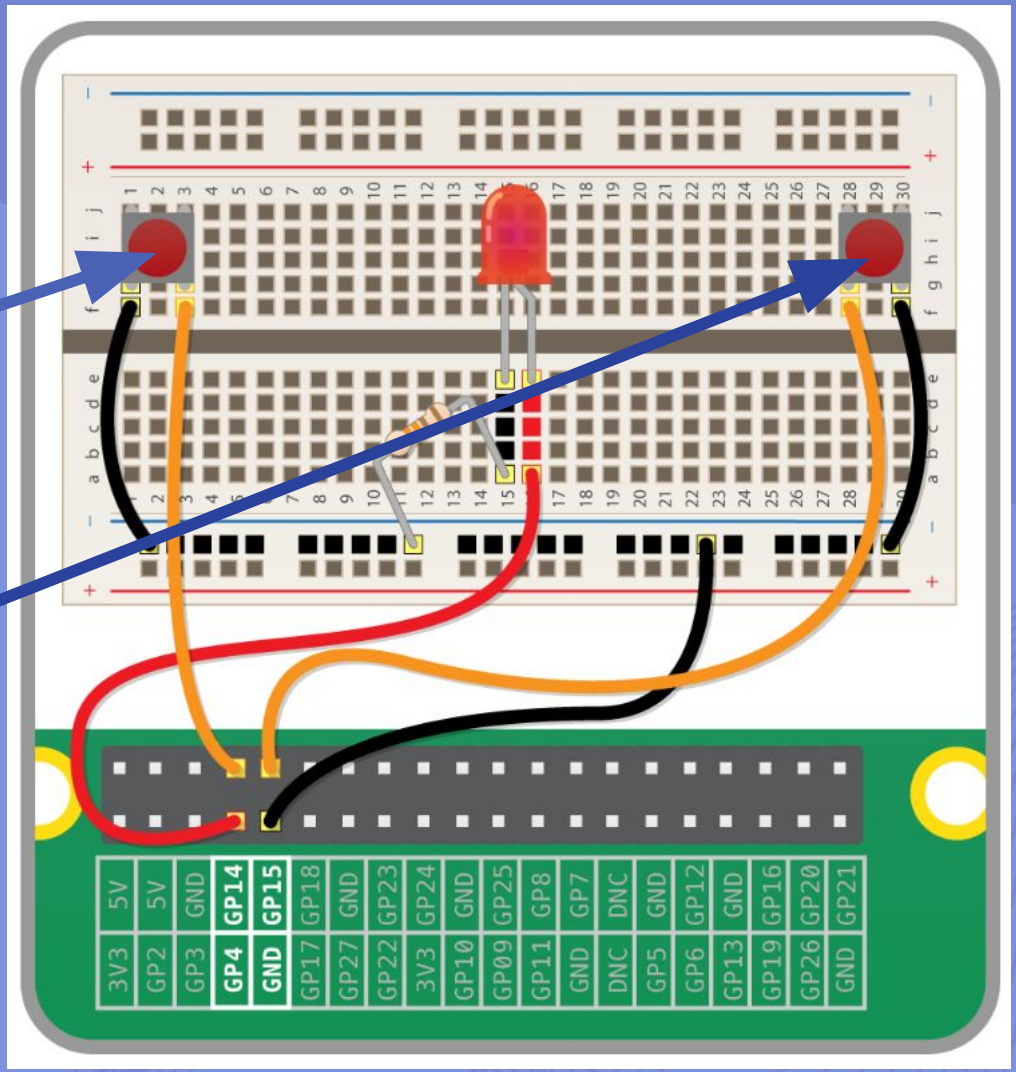


Step 1

Building the circuit

Take a tactile button and push it into the holes on your breadboard, with one set of legs in row H and one set of legs in row J.

Repeat this, but put the second button on the opposite end of the same row.

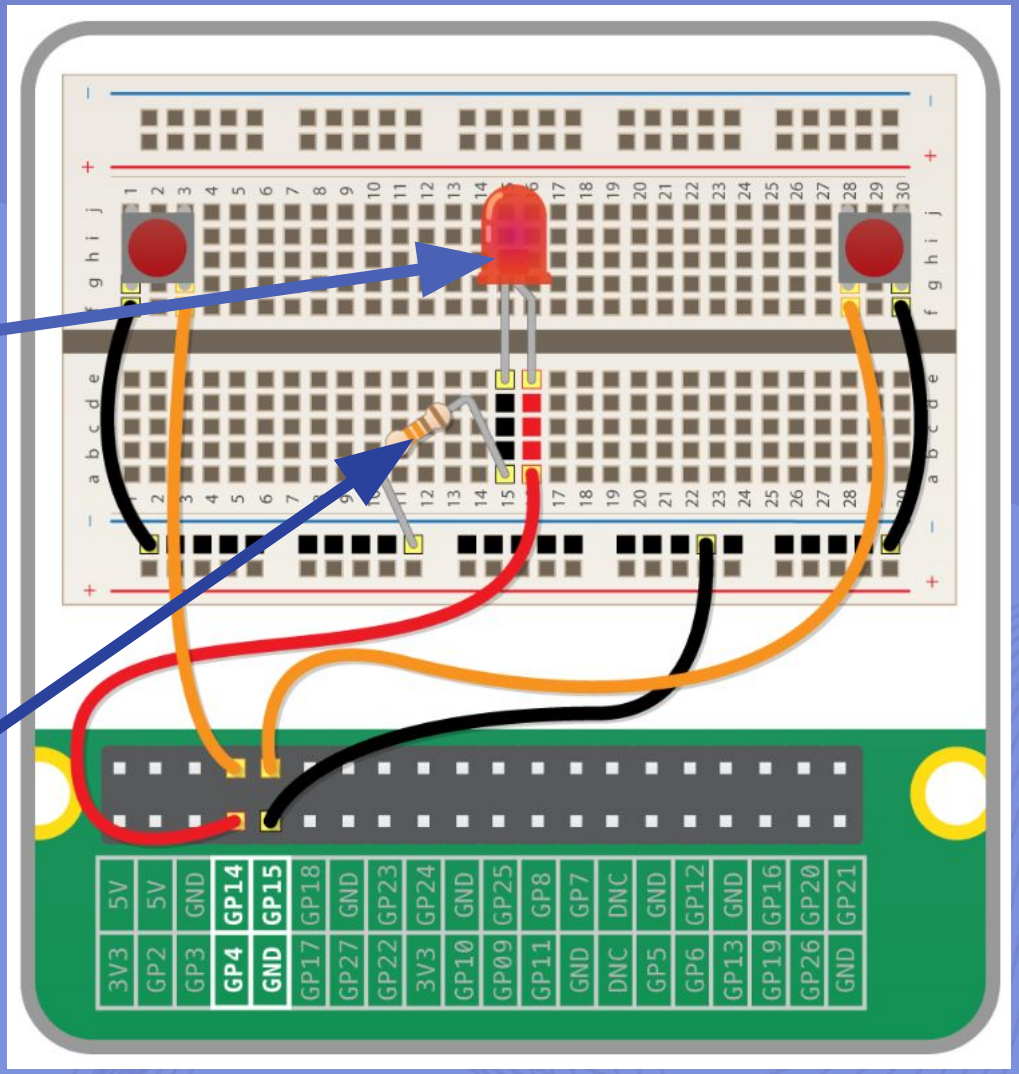


Step 2

Building the circuit

Place an LED with the longer leg in D16 and the shorter leg in D15. Note that the numbering may vary depending on your breadboard, so make sure that you check the diagram below.

Next, push one leg of the resistor into the same column (15) and put the other leg into a hole along the blue strip.

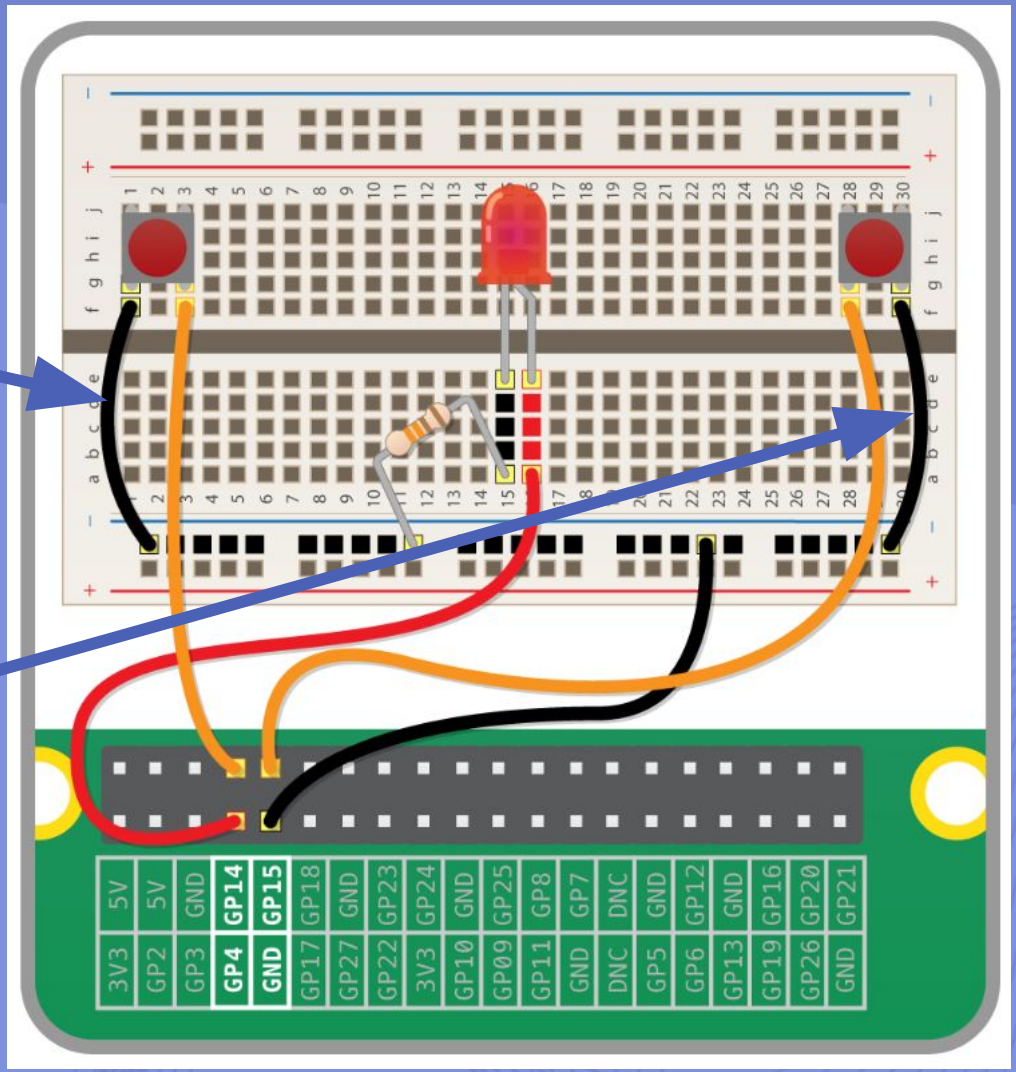


Step 3

Building the circuit

Now we need to add the jumper wires. Take 2 male-to-male jumper wires. Place one end in a hole next to the outside leg of the left-hand button, placing the other end in a hole along the blue strip.

Repeat this step with the right-hand button.

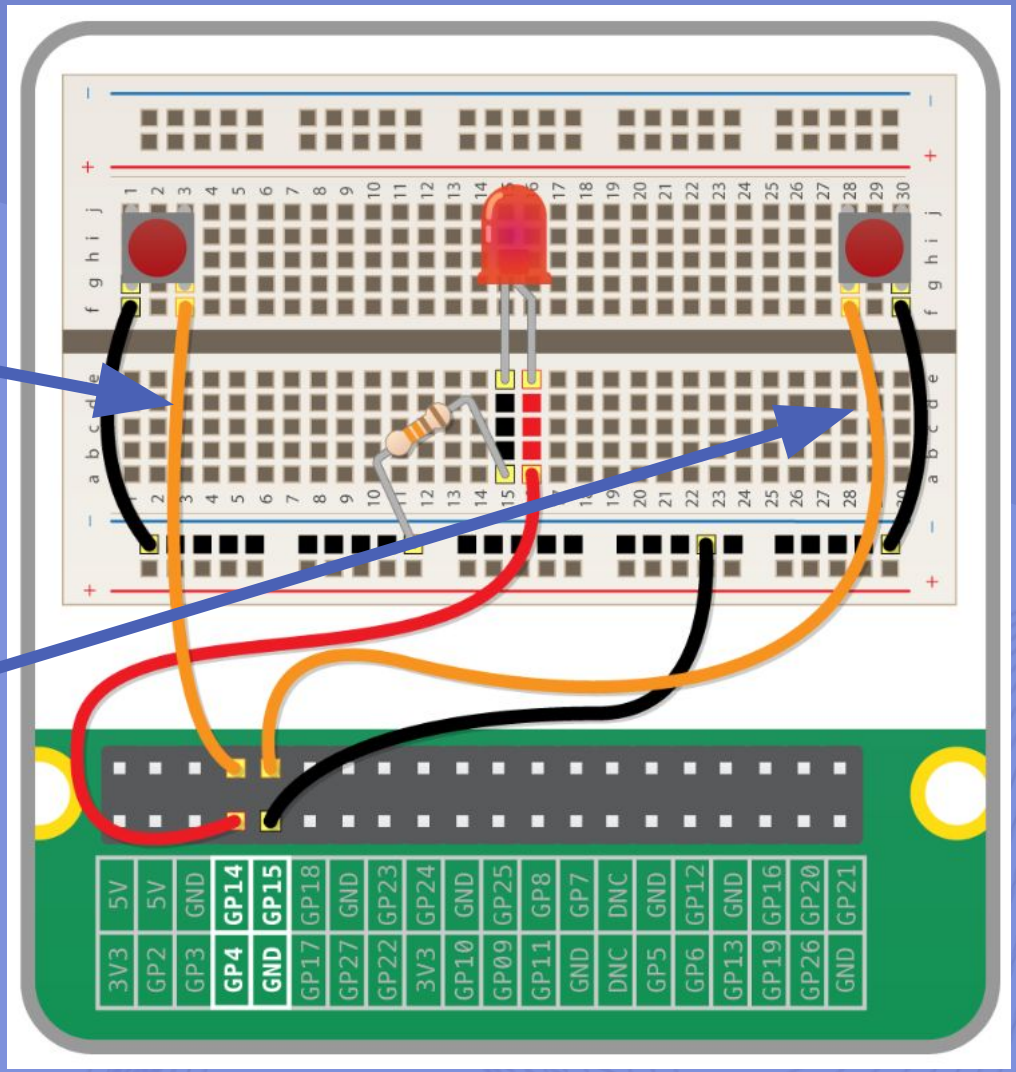


Step 4

Building the circuit

Then with a male-to-female wire, connect GPIO14 to a hole on the breadboard in line with the other leg of the left-hand button.

Repeat this step with the right-hand button, only this time connecting it to GPIO 15

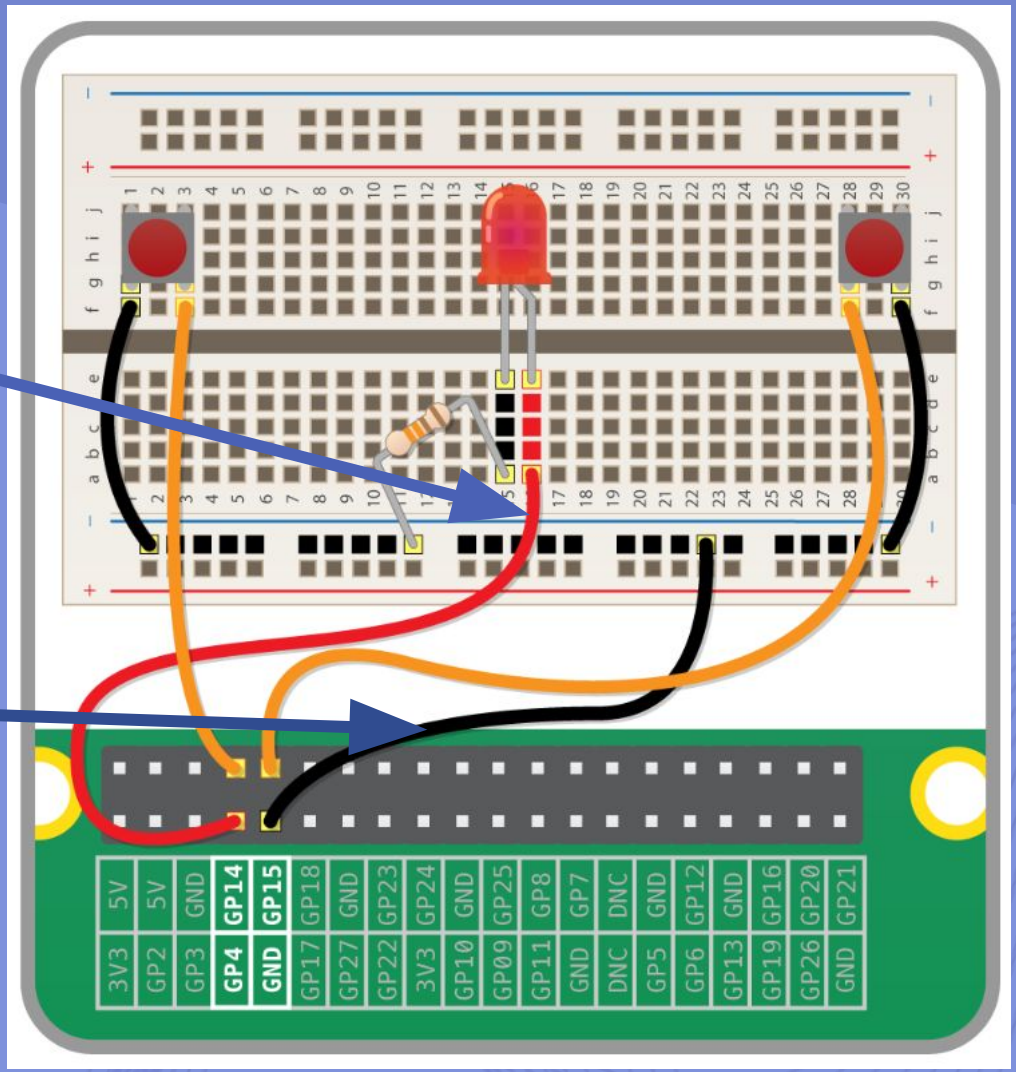


Step 5

Building the circuit

Using another male-to-female jumper wire, connect GPIO 4 to a hole on the breadboard in line with the long leg of the LED.

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



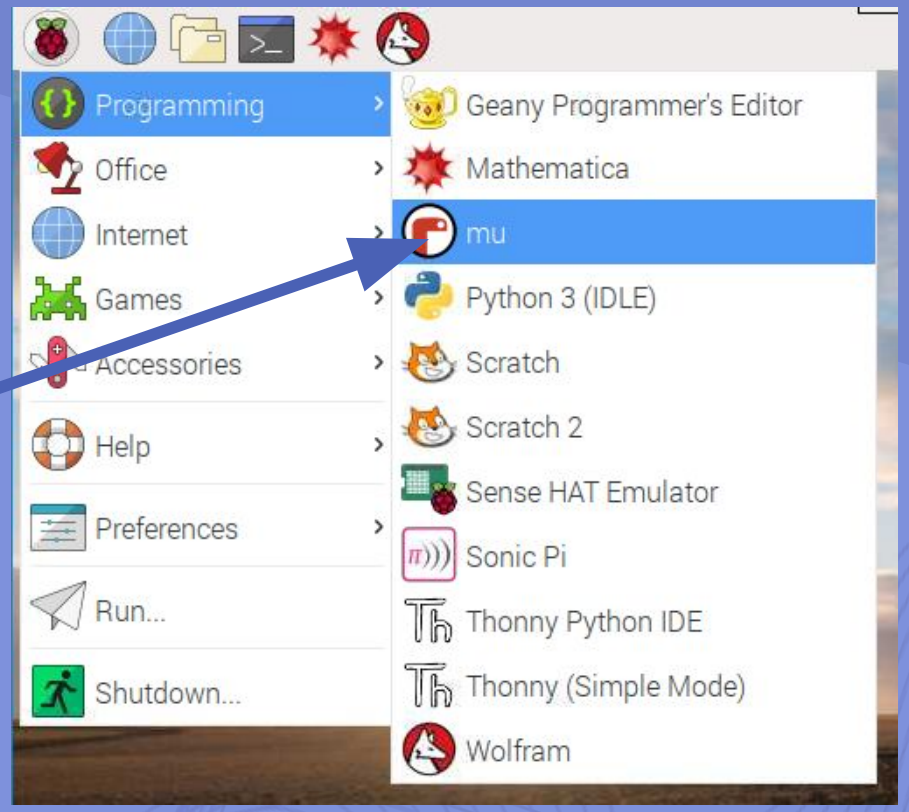
Step 6

Controlling the light

If you haven't already, load up your Raspberry Pi. Open up a python IDE such as Mu.

Click on the Menu > Programming > Mu

Create a new file and save it as reaction.py by clicking on File > Save As



Step 7

Controlling the light

First, import the modules and libraries needed to control the GPIO pins on the Raspberry Pi.

```
1 | from gpiozero import LED, Button
2 | from time import sleep
```

As you are outputting to an LED, we need to set up the pin that the LED connect to on the Raspberry Pi.

Use a variable to name the pin and then set the output.

```
1 | from gpiozero import LED, Button
2 | from time import sleep
3 |
4 | led = LED(4)
```

Step 8

Controlling the light

Next, add a few lines that will turn the LED on, wait for 5 seconds, then turn the LED off

```
1  from gpiozero import LED, Button
2  from time import sleep
3
4  led = LED(4)
5
6  led.on()
7  sleep(5)
8  led.off()
```

Finally, test that it works by clicking Run

Step 9

Adding an element of surprise

Underneath from time import sleep, add a line to import uniform.

```
1  from gpiozero import LED, Button
2  from time import sleep
3  from random import uniform
4
5  led = LED(4)
6
7  led.on()
8  sleep(5)
9  led.off()
```

Here, uniform allows for the random selection of a decimal number from a range of numbers

Step 10

Adding an element of surprise

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

```
1  from gpiozero import LED, Button
2  from time import sleep
3  from random import uniform
4
5  led = LED(4)
6
7  led.on()
8  sleep(uniform(5, 10))
9  led.off()
```

Save your work by clicking Save. Test that everything works by clicking Run.

Step 11

Detecting the buttons

The LED is working now, but we want to add functionality to our program so that when a button is pressed, it is detected. This way, we can record the player's score and see who wins.

```
1  from gpiozero import LED, Button
2  from time import sleep
3  from random import uniform
4
5  led = LED(4)
6  right_button = Button(15)
7  left_button = Button(14)
8
9  led.on()
10 sleep(uniform(5, 10))
11 led.off()
```

Add the following variables underneath led = LED(4)

Step 12

Detecting the buttons

Underneath led.off() we can add a function that will be called whenever a button is pressed, which will tell you which pin the button was on.

```
1  from gpiozero import LED, Button
2  from time import sleep
3  from random import uniform
4
5  led = LED(4)
6  right_button = Button(15)
7  left_button = Button(14)
8
9  led.on()
10 sleep(uniform(5, 10))
11 led.off()
12
13
14 def pressed(button):
15     print(str(button.pin.number) + ' won the game')
```

Step 13

Detecting the buttons

```
1  from gpiozero import LED, Button
2  from time import sleep
3  from random import uniform
4
5  led = LED(4)
6  right_button = Button(15)
7  left_button = Button(14)
8
9  led.on()
10 sleep(uniform(5, 10))
11 led.off()
12
13
14 def pressed(button):
15     print(str(button.pin.number) + ' won the game')
16
17
18 right_button.when_pressed = pressed
19 left_button.when_pressed = pressed
```

Finally, add this code.

If the right_button is pressed, you can send the string ‘right’ to the pressed function

. If the left_button is pressed, then you can send the string ‘left’.

Save your programme and test it with a friend!

Step 14

Get player names

Wouldn't it be better if the programme told you who won instead of which button was pressed? For this, we can find out the user's names using input() statements.

```
1  from gpiozero import LED, Button
2  from time import sleep
3  from random import uniform
4
5  led = LED(4)
6  right_button = Button(15)
7  left_button = Button(14)
8
9  left_name = input('left player name is ')
10 right_name = input('right player name is ')
11
12 led.on()
13 sleep(uniform(5, 10))
14 led.off()
```

Underneath the imported libraries and modules add the following input statements to your code.

Step 15

Get player names

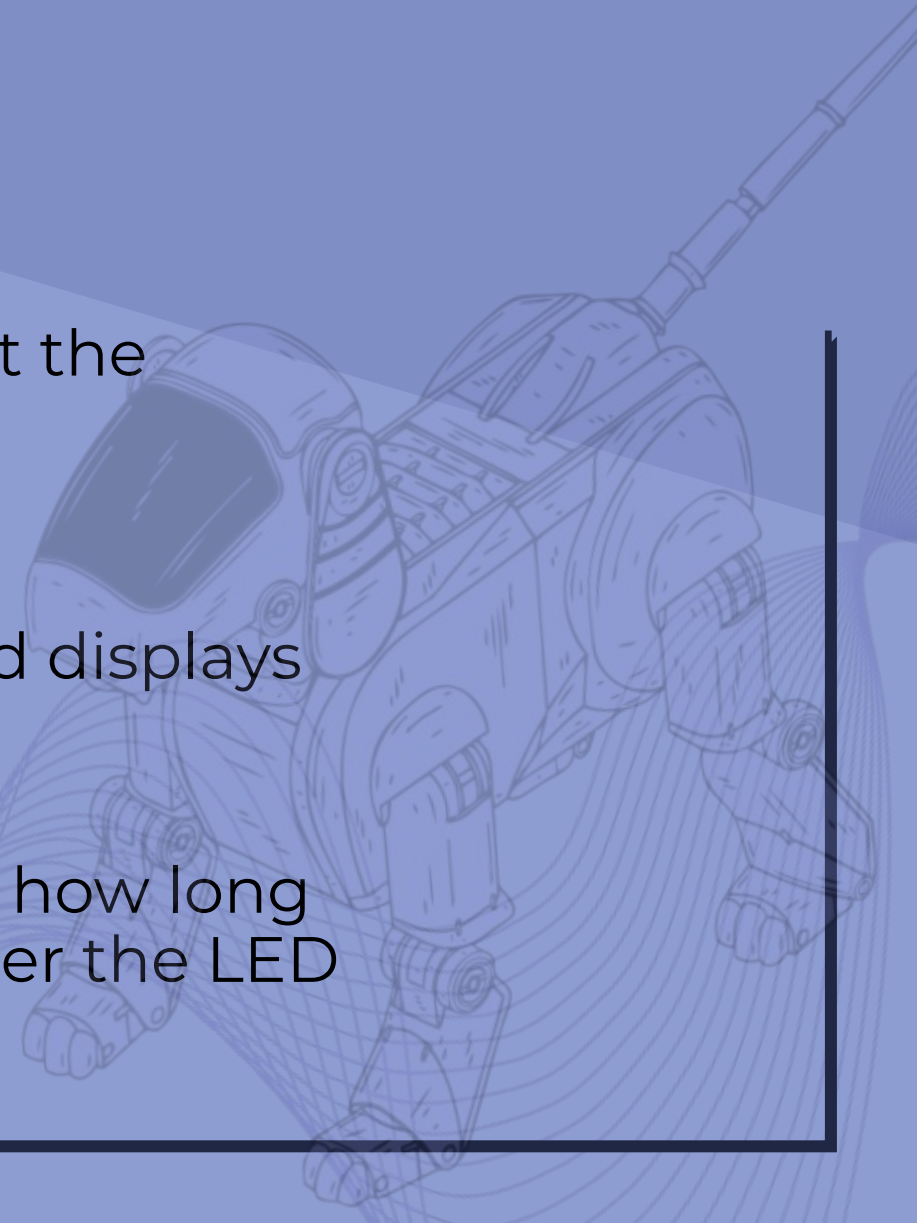
Now rewrite your pressed function, so that it can print out the name of the player who won.

```
13  sleep(uniform(5, 10))
14  led.off()
15
16
17  def pressed(button):
18      if button.pin.number == 14:
19          print(left_name + ' won the game')
20      else:
21          print(right_name + ' won the game')
22
23
24  right_button.when_pressed = pressed
25  left_button.when_pressed = pressed
```

What next?

Ways you could improve your game

- Can you put the game into a loop, so that the game repeats?
- 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 time, to work out how long it took the players to press the button after the LED turned off?



cair
4 YOUTH

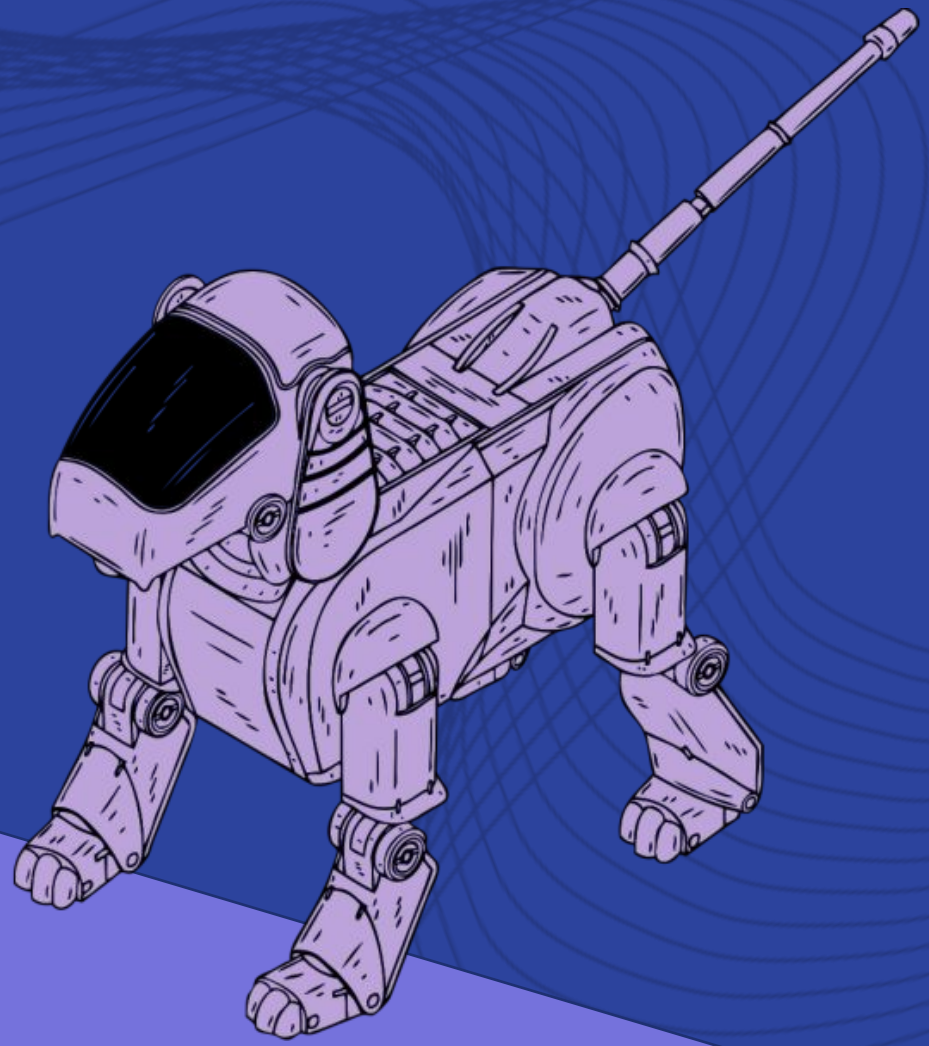
Conclusion

Learning outcomes

- ✔ Learn how to wire a simple circuit.
- ✔ Write a programme to control the circuit.
- ✔ How to use variables to store information.
- ✔ How to get user information like a player's name and use it in the game.

Congratulations!

You have completed the project



cair
4 YOUTH