Week 2 Recap

In Week 1 you learnt about some features and how to set up your Raspberry Pi. Then you also wrote your first Python program to make your own simple reaction game.

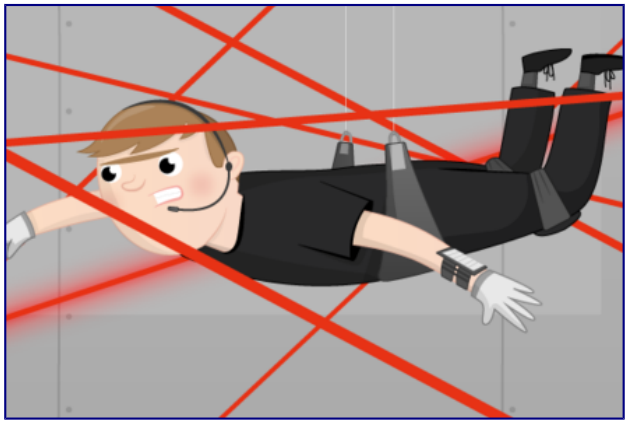
This week you went a little deeper into a key feature of the Raspberry Pi, using the GPIO pins to control other devices.

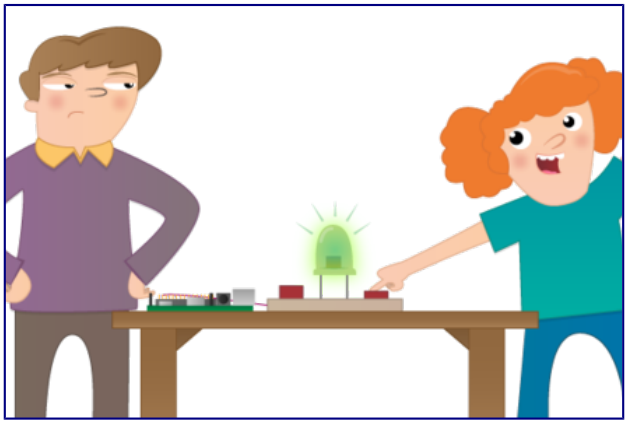
Through this week's activities you have learnt:

- how current flows around a circuit;
- about the purpose of the the GPIO pins on a Raspberry Pi;
- how to build and test a simple circuit using an LED and a Raspberry Pi;
- to turn your LED on and off using your own Python code;
- to apply you ability to control LEDs to a practical activity.

You have taken you first step into the exciting world of physical computing …

Week 3: Press all the buttons!





Sensing the world around you

In this activity you'll be introduced to the to world of sensors and how they can be used to collect data for your programs. From the simple button to environmental sensing, there are sensors for everything!

3.1 Week 3 welcome

3.2 Using sensors with a Raspberry Pi

3.3 Analogue vs digital

3.4 Guess the sensor quiz

3.5 Discussion

Responding to a button press

In this activity you'll wire up a button to the Raspberry Pi and write some code to respond to it being pressed.

3.6 The Highs and Lows

3.7 Wiring up your button

3.8 Detect button presses

3.9 Add an LED

3.10 Test your reflexes

3.11 Applied activity

3.12 Python and buttons quiz

Week 3 welcome

Last week you learnt about controlling an output (LED) using a Raspberry Pi and some Python code.

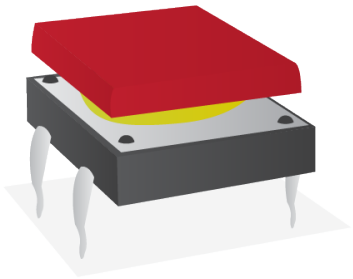
This week you will:

- Connect a button to a Raspberry Pi and write code to respond when it is pressed.
- Learn about other sensors that could be used in place of a button.

Completing this week's exercises is really important, and will help you to identify gaps in your understanding. Don't be discouraged if you don't get everything right on your first try — that's completely normal!

Using sensors with a Raspberry Pi

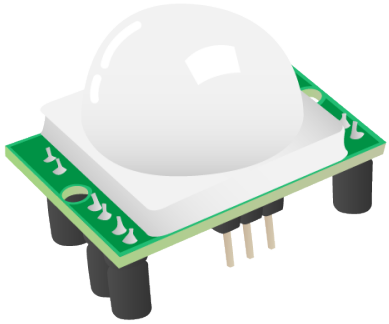
One powerful feature of the Raspberry Pi is the row of GPIO (general purpose input/output) pins along the top edge of the board. These pins are a physical interface between the Pi and the outside world. At the simplest level, you can think of them as switches that you can turn on or off (input) or that the Pi can turn on or off (output).



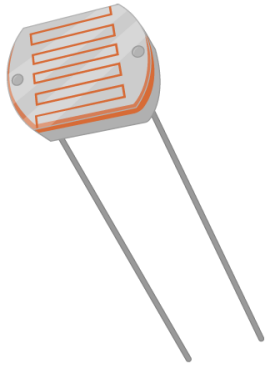
A **push button** is an input component that you can add to the Raspberry Pi GPIO pins. It will complete a circuit when the button is pressed. What that means is that a current will not flow across the button until it is pressed. When it is released, the circuit will be 'broken'. Sensors are another type of input that can be connected to the Raspberry Pi. They may gather data about, for example, light or temperature conditions, but fundamentally they work in a similar way to the push button: they provide different input to the Pi depending on something that is happening externally.

Other sensors can be added to the pins, either as individual components connected to GPIO pins, or through an add-on board called a HAT — the name stands for Hardware Added on Top. Here are a few other popular sensors:

Individual Component Sensors



- A **passive infrared sensor** or PIR detects movement. You have probably seen these before. You will most often find them in the corners of rooms for burglar alarm systems. Every object whose temperature is above absolute zero emits infrared (IR) radiation. The sensor measures the IR signature of the room it's in, and then watches for any changes. Any object moving through the room will disturb the IR signature, and will cause a change to be noticed by the PIR module.



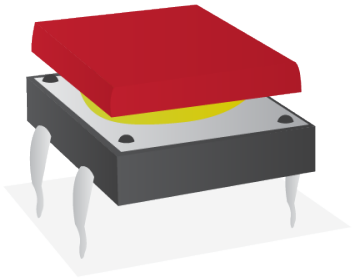
- A **Light Dependent Resistor** or photocell is a component whose resistance will change depending on the intensity of light shining on it. It can therefore be used to detect changes in light. They are commonly used with street lighting, to make streetlamps turn on when it gets dark at night and off when it gets light in the morning.
- A **magnetometer** is used to measure the strength and direction of a magnetic field. Most often they're used to measure the Earth's magnetic field in order to find the direction of North. If your phone or tablet has a compass, it will probably be using a magnetometer to find North. They are also used to detect disturbances in the Earth's magnetic field caused by anything magnetic or metallic; airport scanners use them to detect the metal in concealed weapons, for instance.
- A **temperature** sensor is used to measure hot and cold. It's exactly like the thermometer that you would put in your mouth to take your own temperature, except it's an electronic one built into the Sense HAT and reports the temperature as a number in Celsius.
- A **humidity** sensor measures the amount of water vapour in the air. There are several ways to measure it, but the most common is relative humidity. One of the properties of air is that the hotter it is, the more water vapour can be suspended within it. So relative humidity is a ratio, usually a percentage, between the actual amount of suspended water vapour and the maximum amount that could be suspended for the current temperature. If there is 100% relative humidity, it means that the air is totally saturated with water vapour and cannot hold any more.
- A **pressure** sensor (sometimes called a barometer) measures the force exerted by tiny molecules of the air we breathe. There's a lot of empty space between air molecules, and so they can be compressed to fit into a smaller space; the more air is squeezed into a space, and the smaller the space, the higher the pressure. This is what happens when you blow up a balloon. The air inside the balloon is slightly compressed and so the air molecules are pushing outwards on the elastic skin; this is why it stays inflated and feels firm when you squeeze it. Likewise, if you suck all the air out of a plastic bottle, you decrease the pressure inside it, and so the higher pressure on the outside crushes the bottle.

Analogue vs digital

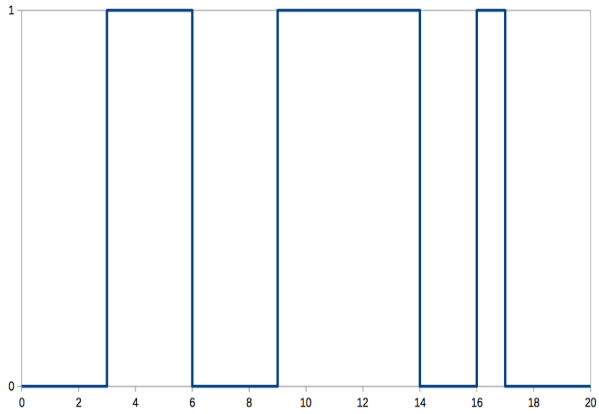
Using the GPIO pins on the Raspberry Pi, it is easy to send a signal to an output component and turn it **on** or **off**. You can also detect whether an input component is on or off. Components that operate in this way are called **digital** components.

An LED is an example of a digital *output* component. It can either be on or off, and there is no value in-between. We can think of the **on** and **off** states as being either **1** or **0**. You can send a **1** to the LED to illuminate it on and a **0** to the LED to turn it off again.

A button is an example of a digital *input* component. It can either be on or off as well. When the button is pressed, it sends a **1** to the Raspberry Pi GPIO pin it is connected to. When the button is released, it sends a **0** to the GPIO pin. There is no other value that can be sent, as you can't **half-press** a button.



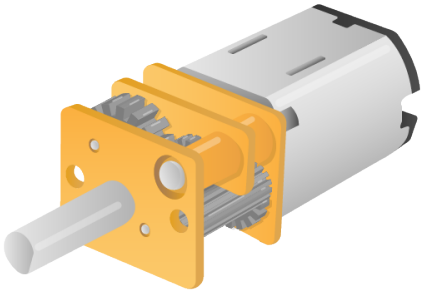
Look at the graph below. It shows the result of a button being pushed and released over time. When it is being pushed it sends a 1, and when it is released it sends a 0.



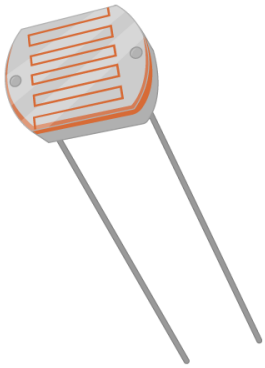
Digital input and output components are easy to use with the Raspberry Pi, as the GPIO pins are all digital. They can only send or receive 1s and 0s.

However, not all components are digital. Some are called analogue components. Analogue components can send and receive values in-between 1 and 0.

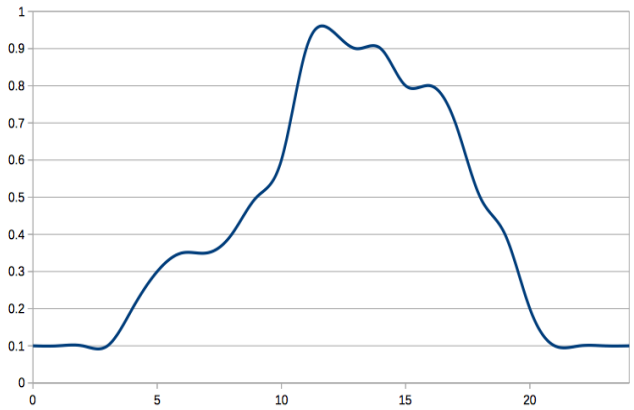
A motor is an example of an analogue *output* component. You can send it values between 1 and 0, which will control the speed of the motor. If you send the motor a 1 it will drive at full speed. If you send it 0.5 it will drive at half speed. Sending a 0 will stop the motor.



An example of an analogue *input* component is a Light Dependent Resistor (LDR). When there is no light shining on the component it will send a 0, and as light increases, the value sent by the LDR will gradually increase until it hits a maximum value of 1.



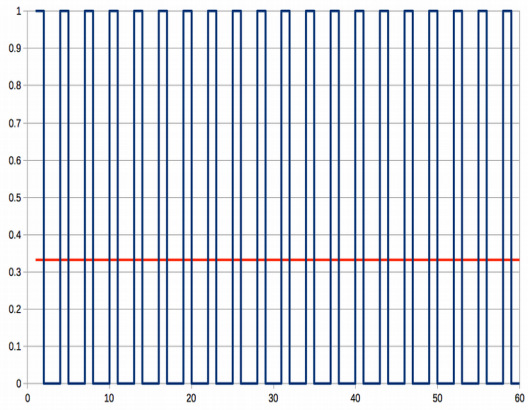
The graph below shows how the signal sent from an LDR will increase and decrease over the course of a 24-hour day, as it goes from dark to light and back again.



Using analogue components with the Raspberry Pi is a little trickier than using digital components.

To use an analogue output component with the GPIO pins, you need to use a technique called Pulse Width Modulation (PWM). This sends very rapid pulses of 1s and 0s to the component, which when taken as an average can be received as values in between 1 and 0.

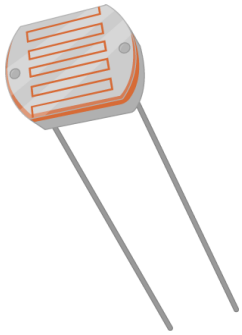
Look at the graph below. The blue line shows the digital signal, over a period of time, moving from 0 to 1 and back again. The signal is 1 for a third of the total time and 0 for the remaining two thirds. This then averages out at around 0.33, which would be the value that is received by the analogue component. You can see this as the red line on the graph.



To use an analogue input component with the GPIO pins, you need to use an Analogue to Digital Converter (ADC), that will turn analogue signals into digital signals. Although you can buy small ADCs for use in your circuits, there are several add-on boards you can buy for the Raspberry Pi with ADCs included, such as the Explorer HAT. Another option is to use a capacitor in your circuits along with the analogue component.

Physical Computing Quiz 4

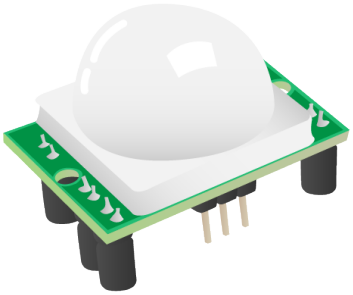
Question 1



Measures the amount of light, Touch, Heat, Fingerprint

This Light Dependant Resistor translates the amount of ambient light into a value. It can be programmed using the gpiozero library.

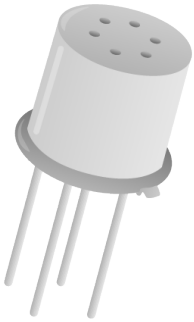
Question 2



Sound sensor, Light sensor, Motion sensor, Moisture Sensor

Yes, this motion sensor uses infrared light to detect motion.

Question 3



Sound sensor, Moisture sensor, Heat sensor, Air quality sensor

Yes, this sensor can measure the quality of the air.

Question 4

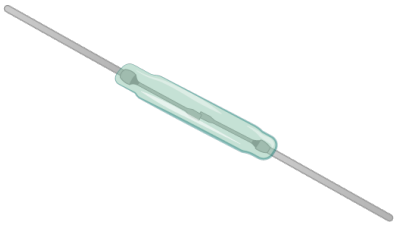
This sensor isn't one we discussed on the previous page. What do you think it measures?

Sound, Infrared, Temperature, Air pressure,

This is an example of a submersible temperature probe.

Question 5

Another new sensor. What do you think it measures?



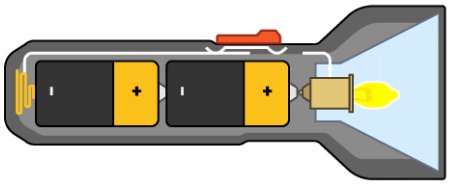
Vibration, Magnetism, Humidity, Infrared

Yes this reed switch contains two separated pieces of magnetic metal which can be influenced by the presence of a magnet.

Discussion of quiz questions

Well done for completing the quiz! In this quiz we tested your knowledge of sensors for getting all kinds of information about the real world.

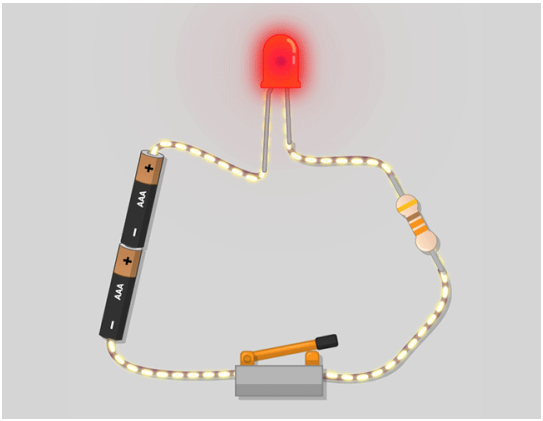
The Highs and Lows



When you reach for your torch in the middle of a power cut, you flick the switch to turn on the light. What happens when you do this is that you complete the circuit between your light bulb and the power supply.

The voltage across the light bulb jumps from 0V to 9V. You might remember from last week's lesson that the voltage is related to how much energy can be supplied to the light bulb. There is no middle voltage that the light bulb can receive: it gets either 0V or 9V.

In the world of electronics, we would describe the light bulb as being sent **high** when the switch is closed and **low** when the switch is open. So the light bulb has two states; high and low.



Look at the circuit here. This is just a simple circuit with an LED and a button (plus a resistor to protect the LED and a cell to provide the voltage). When the button is pushed, the LED turns on as it receives 3.3V (actually a little less than 3.3V, because there is a voltage drop across the resistor). The LED is being sent **high** when the button is pushed, and **low** when the button is released.

Highs and **Lows** are useful to talk about when integrating circuits with computers. This is because we can think of a **high** as representing the number 1, and a **low** as 0. By using this representation of high/low as 1/0, we have a way of communicating between circuits and computers.

Wiring up your button

Buttons and switches are a class of **input** component. They allow a user to have some control over a circuit, or to send signals to a computer. The keys on your keyboard are all examples of buttons; when you press a key it sends a signal to the computer that represents the character that has been pressed.

You can use physical buttons and switches to send signals to your Raspberry Pi, and trigger all kinds of events. Use the discussion below to come up with some different ideas for things that you could make your Raspberry Pi do when a button is pushed.

In the diagram below, a button has been wired up to a Raspberry Pi on GP4 and a GND pin. You'll notice that the button has been pushed into the breadboard across the dividing line. Set up your circuit in the same way as shown.

In this case, GP4 starts off *high* and when the button is pressed and the circuit closed, the pin is sent *low* as it is connected to the ground pin. If you want a more detailed explanation of why this is, then you can have a look at this Wikipedia article on pull-up and pull-down resistors, but it really isn't important for what you are going to do. All you really need to understand is that when the button is pushed, the state of pin 4 is changed.

Detect button presses

1. So how can you detect the **state** of a pin? Again, you can use the *gpiozero* library to help you out with this. In the two lines below, a **Button** object has been created using pin 4.

   from gpiozero import Button btn = Button(4)

2. Now you need to detect when the button is pushed, and give the user some indication of the action that has occurred. `wait_for_press` and `wait_for_release` are two methods that will pause your program until a signal is received from the button.

   while True: btn.wait_for_press() print('You pressed me') btn.wait_for_release() print('You released me')

3. Run your program, press the button and see what happens.

Add an LED

In the next step you're going to create a reaction game much like you did in Week 1 but using physical components to enhance the project.

To start with, add an LED to your setup in the same way you did in Week 1. The video demonstrates this using pin 17, but you can choose any available pin.

Test your reflexes

Now that you can control LEDs and detect button pushes, it's time to build your first game.

1. Create a new Python file and save it as `reaction_game.py`.

The idea of the game will be to have an LED switch on after a random period of time. The player will have to push a button the moment the LED switches on. The time between the LED lighting up and the button press can be recorded, and this reaction time be displayed on the screen.

1. To begin with you'll need to be able to use a few classes and functions from the modules you've already used in this course: the `Button` and `LED` classes from `gpiozero`, and `time` and `sleep` from the `time` module. Additionally you'll need `randint`, which will give you a random number, from the `random` module.

At the top of your file, import the functions you need as shown below.

from gpiozero import Button, LED from time import time, sleep from random import randint

1. The next stage will be to create a new `LED` object on pin 17 and a `Button` object on pin 4.

led = LED(17) btn = Button(4)

1. Now physically connect an LED and a button to your GPIO pins, using the pins you've specified in the code. If you've written it as above, you'll need pins 17 and 4 respectively. Don't forget to use a resistor to protect the LED.

1. It's always a good idea to test your wiring, so let's add a little bit of code to make sure it's all working.

   led.on() btn.wait_for_press() led.off()

2. Save and run your code. The LED should come on, and then turn off again when the button is pressed.

3. Believe it or not, you already have most of the code to make your game now. Let's first make a little change by placing those last three lines into an infinite loop.

   while True: led.on() btn.wait_for_press() led.off()

If you save and run this, the LED should come on, switch off when the button is pressed but then come straight back on again. The loop runs so quickly that you may not even notice the flicker of the LED.

1. You can place a `sleep` into the loop to make the LED pause each time before lighting up, just as you did in the last lesson when you made the LED blink. This time though, instead of making the program sleep for a set amount of time, we can make it sleep for a random number of seconds using the `randint` function. In the code below, the pause will be for a random number of seconds between 1 and 10.

   while True: sleep(randint(1,10)) led.on() btn.wait_for_press() led.off()

2. To measure the time between the led turning on and the button being pushed, we'll need to use the `time` function. In the code below the time is saved in two variables - `start` (when the LED lights up), and `end` (when the button is pressed).

   while True: sleep(randint(1,10)) led.on() start = time() btn.wait_for_press() led.off() end = time()

3. To finish off the basic game, you can subtract `start` from `end` to calculate the reaction time, and then print it out on the screen.

   from gpiozero import Button, LED from time import time, sleep from random import randint led = LED(17) btn = Button(4) while True: sleep(randint(1,10)) led.on() start = time() btn.wait_for_press() end = time() led.off() print(end - start)

4. Save and run your game to see what your reaction times are like. You can quit your program using `Ctrl` + `C`.