GPIO WORKSHOP

GPIO is the General Purpose Input / Output system that enables the Pi to interact with the physical world. The GPIO pins are the bank of 40 pins that runs along the side of the Pi. With these you can control robots to take over the world!

GPIO PIN NUMBERING

To the right you can see how the pins are laid out. The numbers inside the circles are the physical pin numbers. However, these are not used when programming. The tags to the left and right are the "Broadcom" numberings (Broadcom are the company that makes the Pi's processor).

These are how the Raspberry Pi sees the pins, and it is these numbers you'll be using to control the pins in Scratch and, later on, Python.

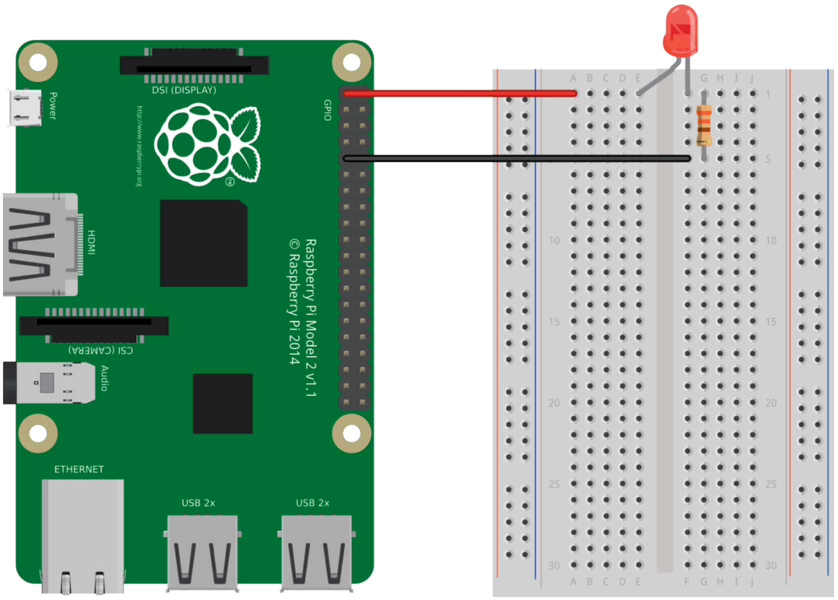
	All Models		
3V3 Power	1	2	**5V** Power
GPIO2 SDA I²C	3	4	**5V** Power
GPIO3 SCL I²C	5	6	**Ground**
GPIO4	7	8	**GPIO14** UART0 TXD
Ground	9	10	**GPIO15** UART0 RXD
GPIO17	11	12	**GPIO18**
GPIO27	13	14	**Ground**
GPIO22	15	16	**GPIO23**
3V3 Power	17	18	**GPIO24**
GPIO10 SPI MOSI	19	20	**Ground**
GPIO9 SPI MISO	21	22	**GPIO25**
GPIO11 SPI SCLK	23	24	**GPIO8** SPI CE0
Ground	25	26	**GPIO7** SPI CE1
ID SD I²C ID	27	28	**ID SC** I²C ID
GPIO5	29	30	**Ground**
GPIO6	31	32	**GPIO12**
GPIO13	33	34	**Ground**
GPIO19	35	36	**GPIO16**
GPIO26	37	38	**GPIO20**
Ground	39	40	**GPIO21**
	A+,B+,2B		

LIGHTING AN LED

Next, take the following components from your CamJam #1 Kit (the smaller tin):

1. A breadboard
2. An LED (any colour, any size)
3. A 330 ohm resistor (the strip of three resistors in the CamJam #1 kit are all 330 ohm; the lone resistor is 4.7 kilo-ohm)

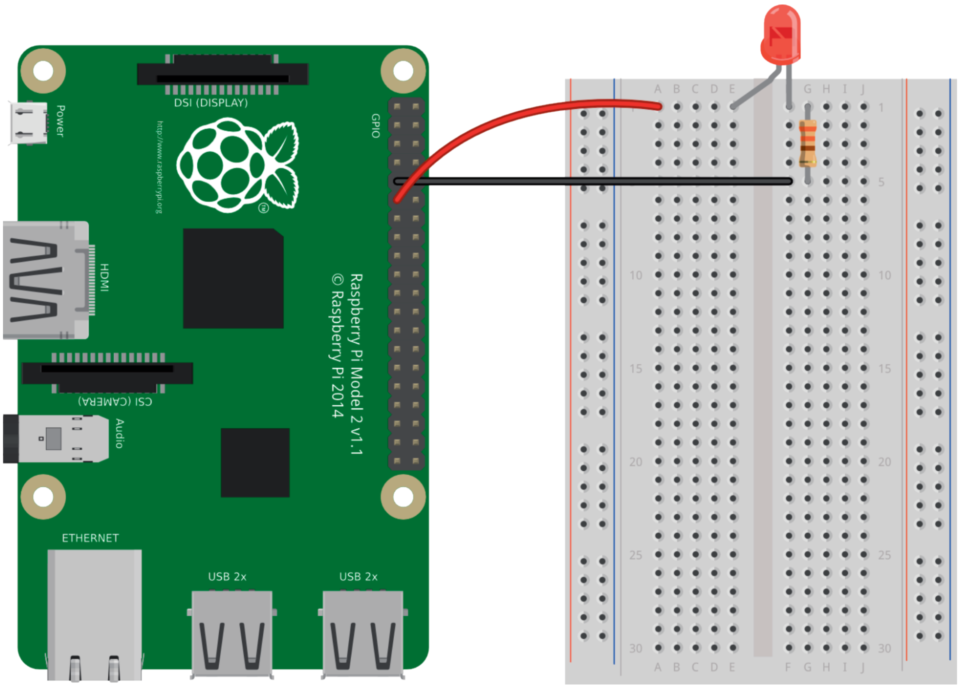
Connect these components as show in the diagram below. The LED should light up once the circuit is complete.



GPIO WORKSHOP

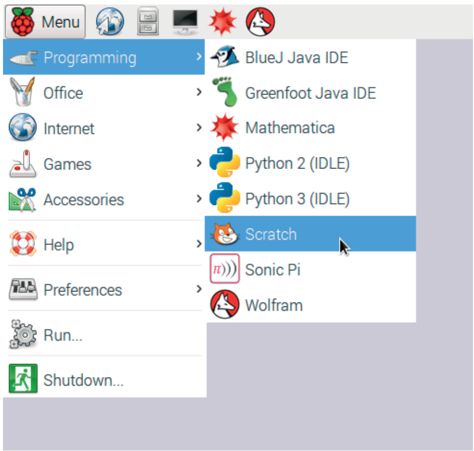
Now, change your circuit so that instead of being connected to the 3v3 power pin, it is connected to the GPIO17 pin instead.

The LED should remain unlit when connected to GPIO17 because this pin is programmable and we haven't yet switched it "on". Next, we're going to use Scratch to control that pin...



LIGHTING LEDS FROM SCRATCH

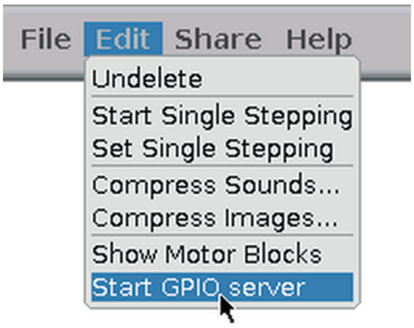
To begin, start Scratch by selecting **Menu > Programming > Scratch**. After a short delay you should see a large window with four distinct sections appear. Maximize this window so it fills the screen.



WARNING

If you see more than one version of Scratch in the Programming menu (e.g. ScratchGPIO7), this means you've got an old version. You'll need to upgrade before continuing.

Before we can begin using the GPIO pins from Scratch, we need to give Scratch access to them. Select the **Edit > Start GPIO Server** menu item to do so.

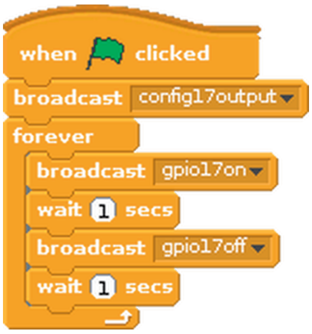


GPIO WORKSHOP

With the GPIO server started, we can use "broadcast" blocks with special messages to control the GPIO pins. The messages that you can use are listed in the table below:

Message	Example	Action
configXoutput	config17output	Configures GPIO17 as an output
configXinput	config26input	Configures GPIO26 as an input
gpioXon	gpio17on	Switches GPIO17 on (must be configured as output)
gpioXoff	gpio17off	Switches GPIO17 off (must be configured as output)
gpioX	gpio16	Reads the state of GPIO16 in a sensor block (must be configured as input)

Set up the following blocks in the script area in Scratch (drag and drop from the blocks palette on the left; all the blocks you'll need are under Control):



```
when ⚑ clicked
broadcast config17output
forever
    broadcast gpio17on
    wait 1 secs
    broadcast gpio17off
    wait 1 secs
```

Now when you click on the green flag at the top right (to run your script), your LED should blink!

EXTRAS

Can you alter your script to:

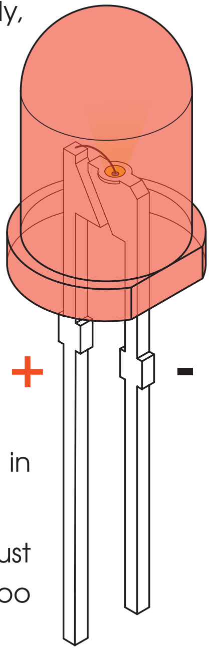
- Make the LED flash faster?
- Flash randomly?

WHAT IS AN LED?

A Light Emitting Diode (LED) is a type of diode that, when connected correctly, produces light. Being a type of diode, it only permits current to flow one way through itself. This means that power must flow from the anode (the positive pin) to the cathode (the negative pin).

The anode can be identified as the longer leg of the LED (the cathode has a shorter leg). The cathode can also be found by feeling the LED body for a slightly flattened edge (as seen in the diagram on the right).

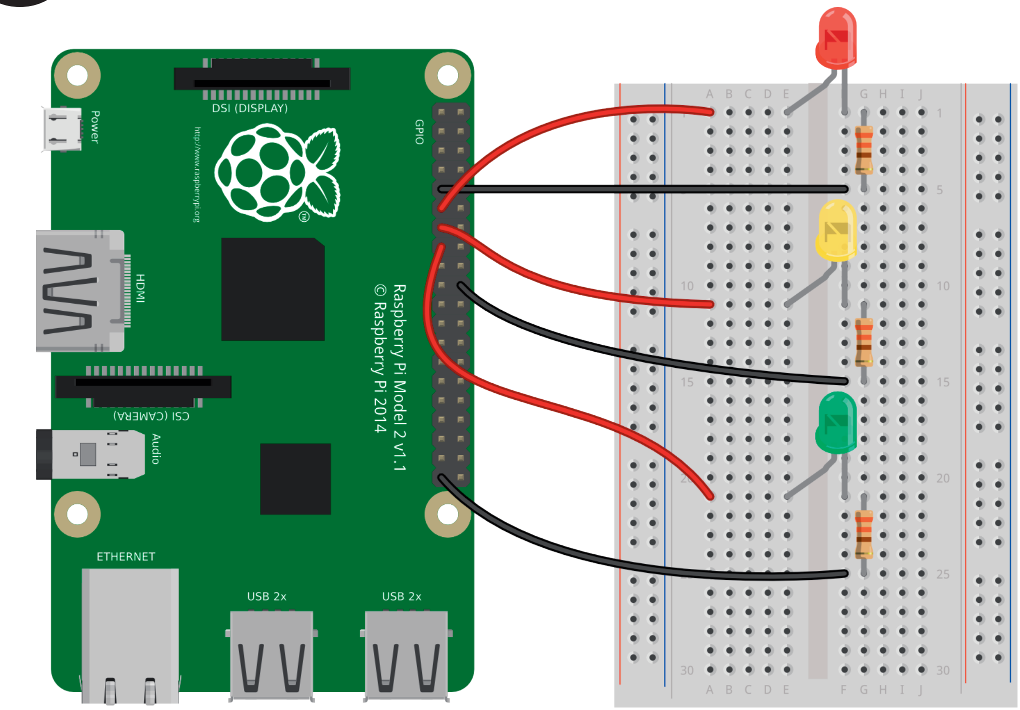
One thing to remember is that a LED must have a resistor in series to prevent too much current from destroying the LED.



GPIO WORKSHOP

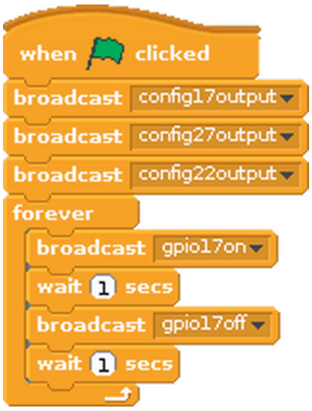
CONTROLLING TRAFFIC FROM SCRATCH

One LED is all well and good, but it's no use for controlling traffic! Add a couple more LEDs to your breadboard with a similar circuit to the red, but shifted down a bit:



Leave some space at the bottom of your board, as shown in the diagram above. You'll need it later!

If you've followed the wiring diagram above, your yellow LED should be connected to GPIO27, and your green LED will be connected to GPIO22. Below is an extended script that configures all the GPIO pins we've used as outputs. Can you extend it to flash the traffic lights sequence (red, red+yellow, green, yellow, and back to the start)?

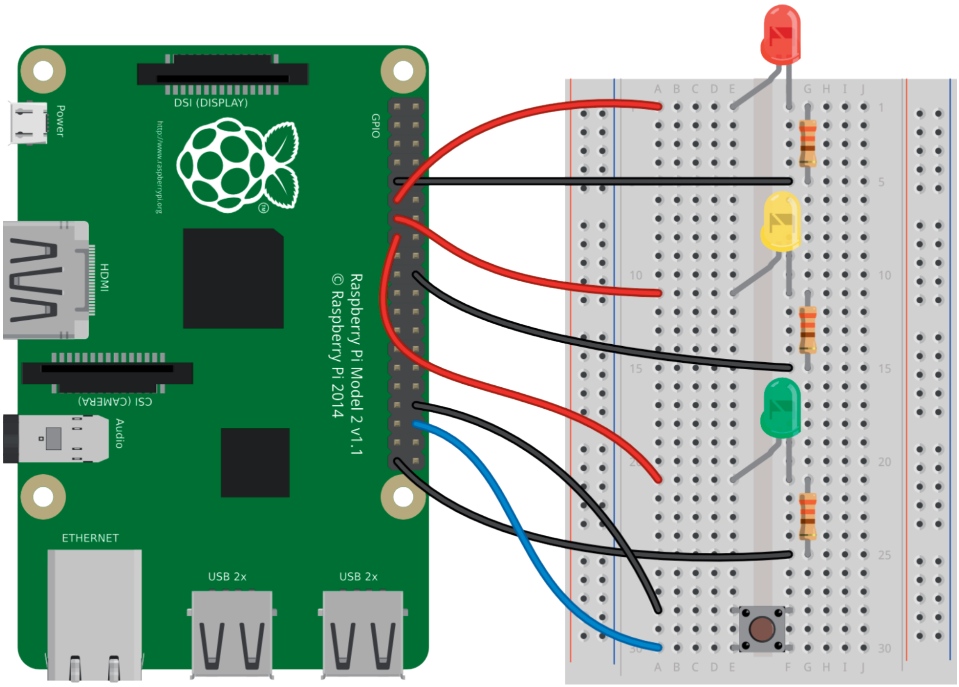


```
when clicked
broadcast config17output
broadcast config27output
broadcast config22output
forever
    broadcast gpio17on
    wait 1 secs
    broadcast gpio17off
    wait 1 secs
```

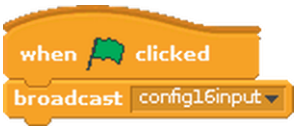
GPIO WORKSHOP

READING BUTTONS FROM SCRATCH

Next we're going to add a button to our breadboard and make the LED react to the button. Firstly, wire the button from your CamJam #1 Kit to the Pi as shown below:

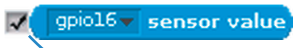


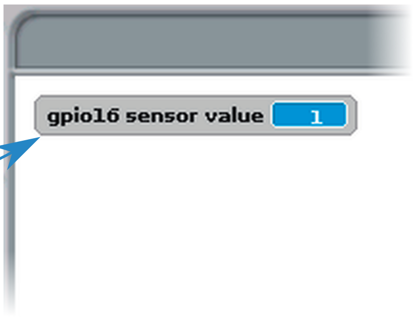
Next, set up the following trivial script in Scratch and run it. This will configure GPIO16 as an input. We need to do this prior to using "sensing" blocks with the button:



```
when clicked
broadcast config16input
```

You can now switch to the "sensing" blocks (in blue), and at the bottom find the "sensor value" block. Select "gpio16" in the drop-down list and then click on the checkbox to the left of the block. You should see "gpio16 sensor value" appear in the stage at the top right.

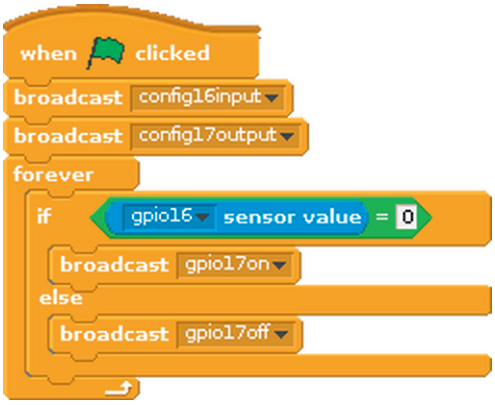




If you press the button on your breadboard (assuming everything is wired correctly) you'll see the "gpio16 sensor value" changing in the stage.

GPIO WORKSHOP

Finally, you can now construct the Scratch blocks shown below. When you click the green flag to run the script you should find that the red LED lights up in response to pressing the button.



```
when [flag] clicked
broadcast config16input
broadcast config17output
forever
    if gpio16 sensor value = 0
        broadcast gpio17on
    else
        broadcast gpio17off
```

LEDS IN PYTHON

We can also control LEDs and read the state of buttons in the Python programming langauge. Save your work in Scratch, if you wish, and then close it down. Leave your breadboard wired to the Pi as it is (with the LED connected to GPIO17, and the button to GPIO16).

Start the Python 3 environment by selecting **Menu > Programming > Python 3**. Once the Python environment appears, select **File > New File** to start a new Python script and save the empty file as something suitable like `gpio_workshop.py`. Now enter the following script:

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

led = LED(17)
while True:
    led.on()
    sleep(1)
    led.off()
    sleep(1)
```

When you select **Run > Run Module** (or press `F5`) you should see the LED blinking as it did with Scratch. You may wish to note the similarities between this script and the Scratch script for blinking the LED.

To stop your script at any time, select **Shell > Restart Shell** in the main Python window (or press `Ctrl+F6` on the keyboard).

EXTRAS

Can you alter your script to:

- Make the LED flash faster?
- Flash randomly?
- Flash two LEDs?

GPIO WORKSHOP

In Python there's an even easier way to blink the LED: using the `blink` method. This takes two parameters: the time to remain on and the time to remain off. The following script uses this method instead.

Note that instead of using an infinite loop which will cause the script to run until stopped, this version executes the `blink` method (which runs in the background) then uses `pause` to wait until the script is stopped:

```
from gpiozero import LED, Button
from signal import pause

led = LED(17)
led.blink(1, 1)
pause()
```

BUTTONS IN PYTHON

As in Scratch, we use a loop to read the state of a button (from the `is_pressed` attribute) and light an LED in response:

```
from gpiozero import LED, Button

led = LED(17)
btn = Button(16)
while True:
    if btn.is_pressed:
        led.on()
    else:
        led.off()
```

Again, compare this script to the Scratch script that lights the LED when the button is pushed. However, we can also use an easier method in Python by connecting events (e.g. `when_pressed`) to handlers (e.g. `on` and `off`):

```
from gpiozero import LED, Button
from signal import pause

led = LED(17)
btn = Button(16)
btn.when_pressed = led.on
btn.when_released = led.off
pause()
```