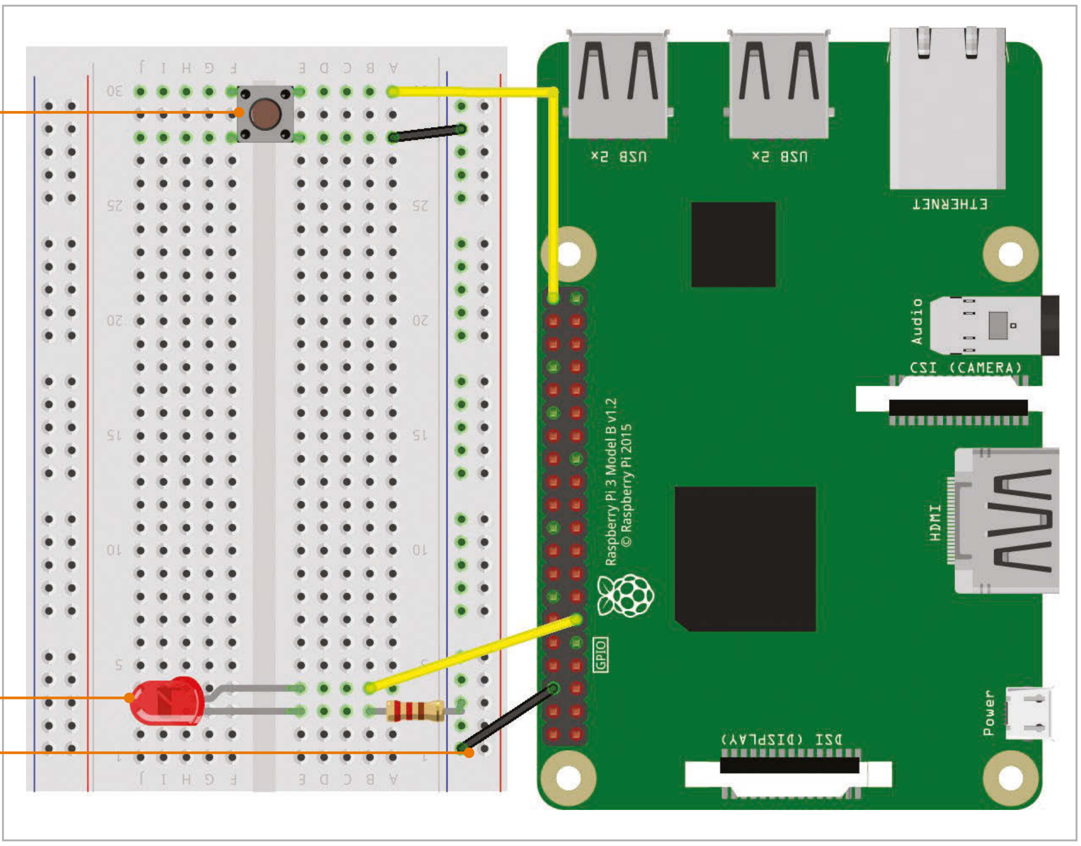
[CHAPTER SEVEN]

LIGHT AN LED

Scratch can be used with the Pi's GPIO pins for physical computing projects. Here, we'll hook up a button-activated LED



When the button is pressed, the circuit is broken and Scratch senses a zero value from GPIO pin 21

The LED's longer leg is wired to GPIO 17, while the other is connected via a resistor to the ground rail

By wiring the '–' row, or ground rail, to a GND pin, multiple components can share the connection

You'll Need

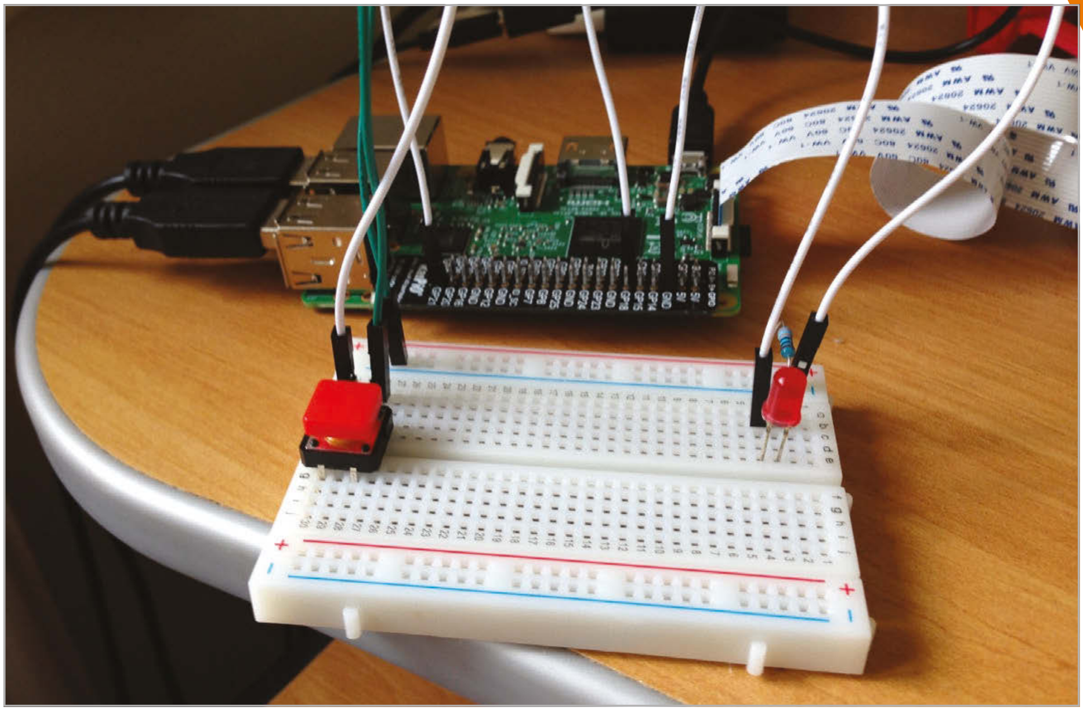
- Solderless breadboard
- LED
- 333Ω resistor
- Push button
- 3× male-to-female jumper wires
- Male-to-male jumper wire

In the latest version of Raspbian Jessie, Scratch features a built-in GPIO server to make it easier to control electronic components or add-on boards. In this first GPIO tutorial, we'll be creating a simple circuit with a button that, when pressed, causes an LED to light up. Take a look at the 'You'll Need' box to see which electronic components are required; you can buy them separately, but they're all in the CamJam EduKit #1 (**magpi.cc/1OcXtim**).

>STEP-01

Connect the LED

It's best to turn the Pi off when building your circuit. The breadboard features numbered columns, each comprising five connected holes. Place your LED's legs in adjacent numbered columns, as shown in the diagram. Note that the shorter leg of the LED is the negative end; in its breadboard column, insert one end of the resistor, then place the other end in the outer row marked '–' (the ground rail). Use a male-to-female jumper wire to connect another hole in that ground rail to a GND pin on the Pi. Finally, use a jumper wire to connect a hole in the column of the LED's longer (positive) leg to GPIO pin 17.



Above: This project is simple to wire up using a solderless breadboard and some jumper wires

>STEP-02

Configure Scratch GPIO

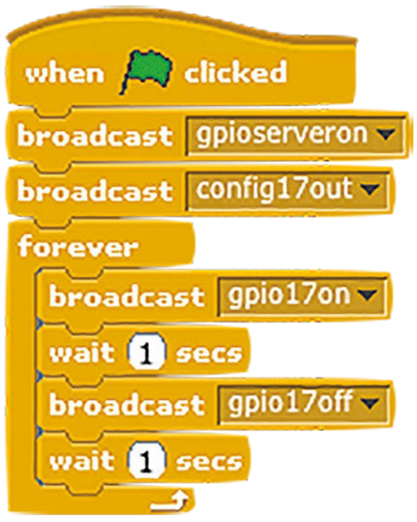
Before we can use the GPIO pins from Scratch, we need to turn its GPIO server on. While this can be done from the Edit menu, instead we'll get our code to activate it. Under a **`when green flag clicked`** block, add a **`broadcast`** Control block, click its arrow, select new/edit, and enter **`gpioserveron`**. We also need to configure GPIO pin 17 as an output pin (to trigger the LED), so add another **`broadcast`** block and change it to **`config17out`**.

>STEP-03

Light the LED

We'll now test our circuit by using a loop to make the LED blink. Add a **`forever`** block to the bottom of your code. Within it, add the following blocks: **`broadcast gpio17on`**, **`wait 1 secs`**, **`broadcast gpio17off`**, and **`wait 1 secs`**. Now try running the code (**Listing 1**) and your LED should blink on and off continually.

.01



```
when green flag clicked
broadcast gpioserveron
broadcast config17out
forever
    broadcast gpio17on
    wait 1 secs
    broadcast gpio17off
    wait 1 secs
```

>STEP-04

Connect the button

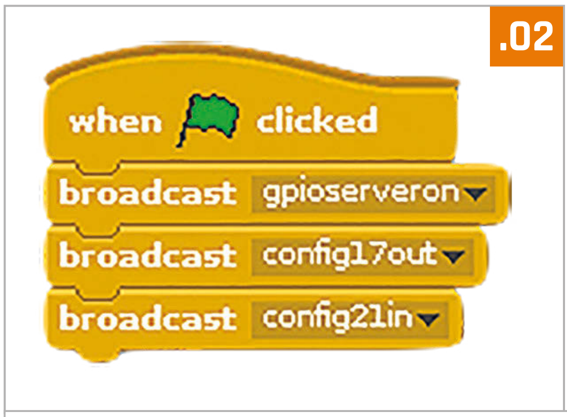
We can control our LED by adding a push button. Again, we'd advise you to turn the Pi off while connecting new components. Add the push button to the breadboard, with its pins straddling the central groove (as shown in the diagram). Connect a male-to-female jumper wire from one pin's column to GPIO pin 21 on the Pi. Connect a male-to-male jumper from the other pin (on the same side of the groove) to the ground rail you're using for the LED circuit (to share its connection to the GND pin).

>STEP-05

Configure button

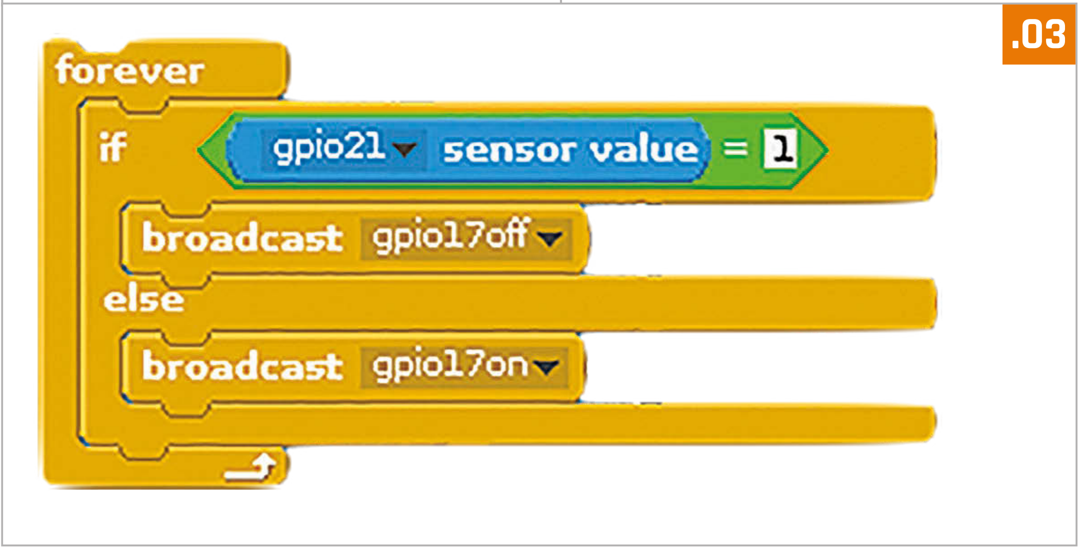
Before Scratch can react to your new button, it needs to be told which pin is its input. Delete the **forever** loop from your blinking LED code, by dragging it out of the area. Add another **broadcast** block with **config21in** to configure GPIO pin 21 as an input – see **Listing 2**. Run and stop the code. Now, click the Sensing category in the top-left pane. Find the **sensor value** block and change it to **gpio21**. Click its tickbox to show its value on the stage: whenever the button is pressed, it should change from 1 to 0.

.02

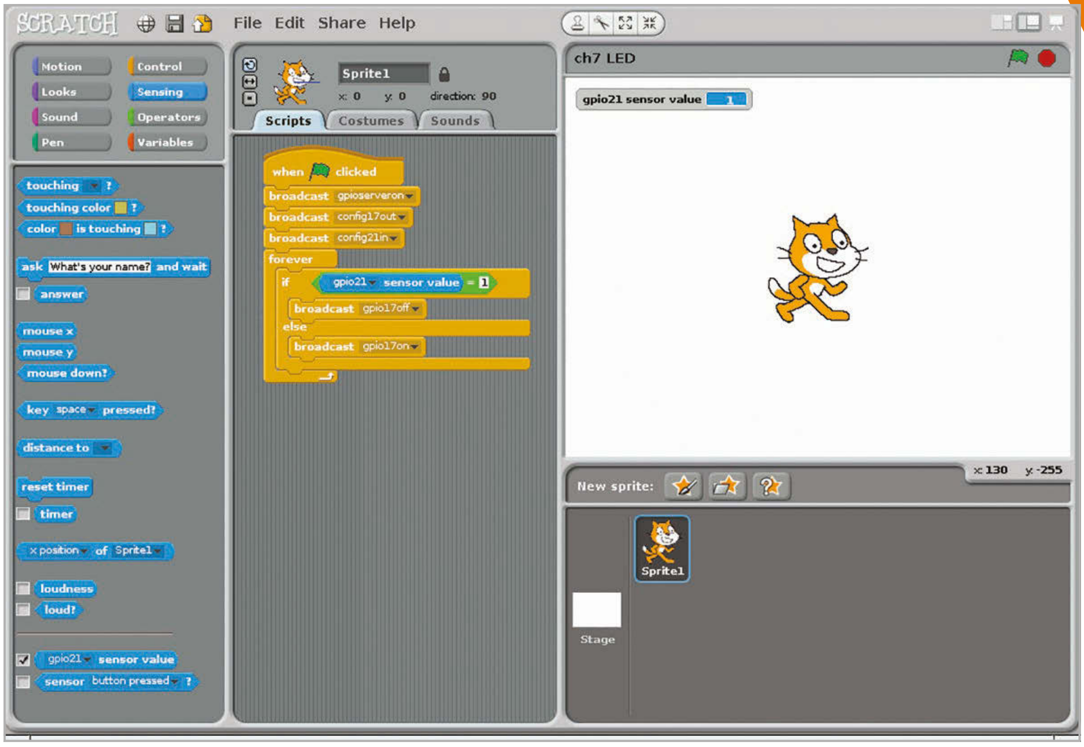


```
when [flag] clicked
broadcast gpioserveron
broadcast config17out
broadcast config21in
```

.03



```
forever
  if gpio21 sensor value = 1
    broadcast gpio17off
  else
    broadcast gpio17on
```



```
when [flag] clicked
broadcast gpioserveron
broadcast config17out
broadcast config21in
forever
    if gpio21 sensor value = 1
        broadcast gpio17off
    else
        broadcast gpio17on
```

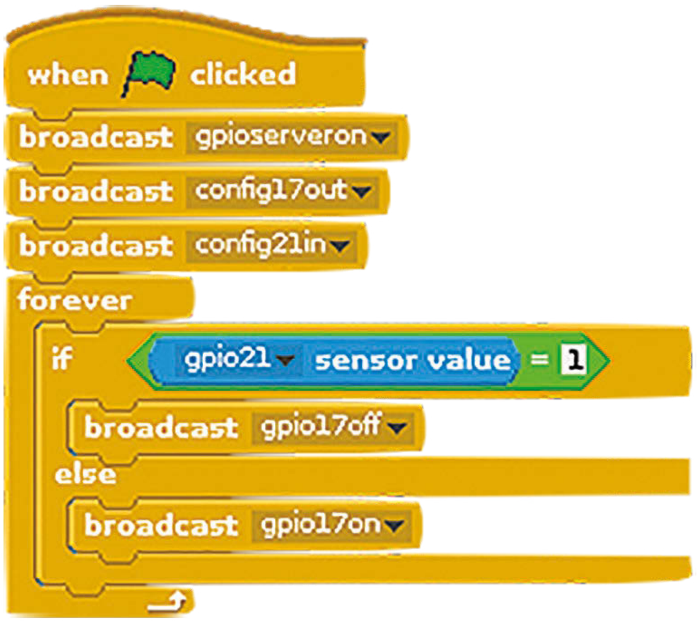
Left: Ticking the button's `gpio21 sensor value` will show it on the stage, which is handy for testing

>STEP-06

Link to LED

With the button working, it's time to make it trigger the LED. Add the code from **Listing 3** to the end of yours. Again, we're using a **`forever`** block for a continual loop. Inside it we add an **`if…else`** block. In the **`if`** field, we place an **=** Operator block; in its left field, we add **`gpio21 sensor value`**, with 1 in the right field. Underneath, we insert **`broadcast gpio17off`**. This way, when the button isn't pressed, the LED will be off. Under **`else`**, we insert **`broadcast gpio17on`**, to light the LED when the button is pressed. Run the code (as in **Listing 4**), press that button, and watch your LED! In the next chapter, we'll add more LEDs to the circuit to make a pedestrian crossing.

.04



```
when [flag] clicked
broadcast gpioserveron
broadcast config17out
broadcast config21in
forever
    if gpio21 sensor value = 1
        broadcast gpio17off
    else
        broadcast gpio17on
```