Raspberry Pi Activity 5: LED the Way

In this activity, you will implement various circuits, primarily using LEDs, resistors, and push-button switches. The Raspberry Pi will initially be used solely as a power supply; however, as the activity progresses, you will use it to programmatically affect the circuits you create. You will need the following items:

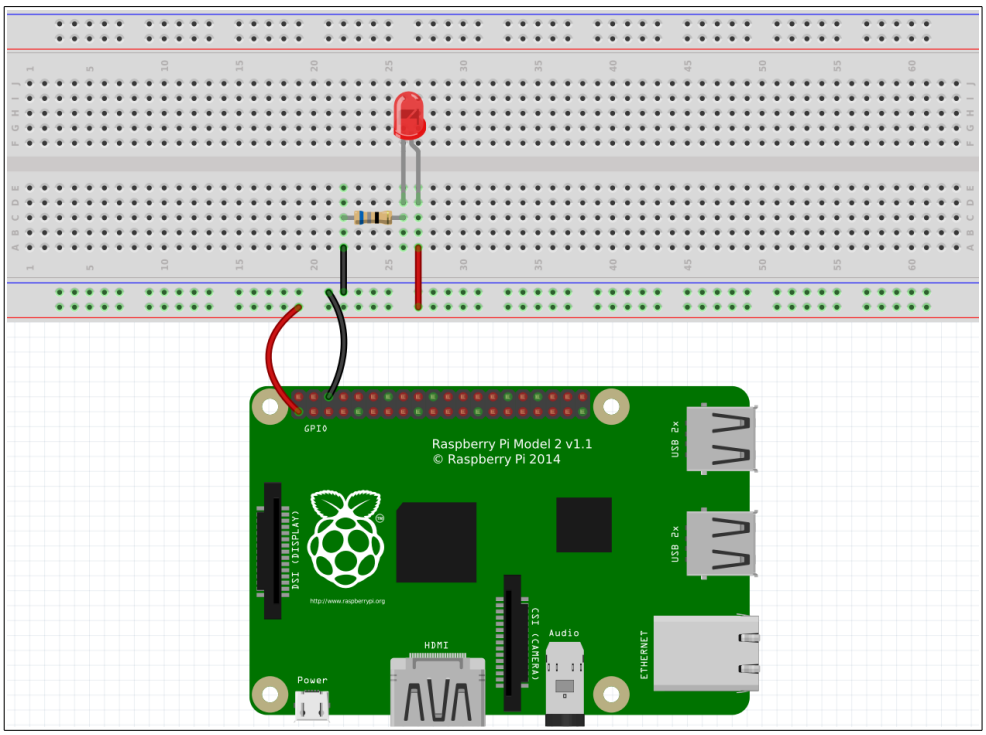
- Raspberry Pi B v2 with power adapter;
- LCD touchscreen with power adapter and HDMI cable;
- Wireless keyboard and mouse with USB dongle;
- USB-powered speakers (optional);
- Wi-Fi USB dongle;
- MicroSD card with NOOBS pre-installed;
- Breadboard;
- T-shaped GPIO-to-breadboard interface board with ribbon cable; and
- LEDs, resistors, switches, and jumper wires provided in your kit.

GPIO

This is the first activity in which the GPIO (General Purpose Input and Output) pins will be used. Recall that these pins allow various inputs to and outputs from the RPi to be utilized in external circuits (typically implemented on a breadboard).

A simple circuit

The first circuit you will implement is the very simple one shown in an earlier lesson (the topic of which was computer architecture). Here was the layout diagram of that circuit:



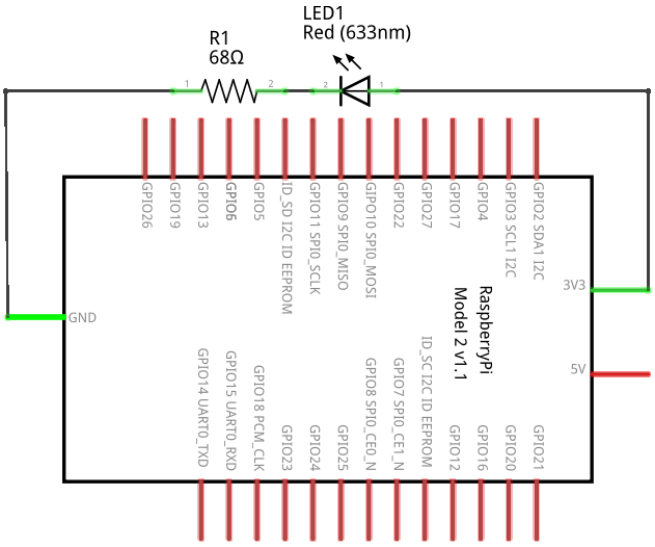
In the diagram, the red wire is connected on one end to a GPIO pin that exposes a 3.3V (3V3) power source. The other end is connected to the bottom row of the breadboard. Recall that this row is internally connected horizontally. Therefore, all holes in that row now have 3.3V.

The black wire is connected on one end to a GPIO pin that exposes ground (GND) or 0V. The other end is connected to the top row in the bottom section of the breadboard. Ground is therefore provided to all holes in that row.

The red LED is inserted so that its legs span across several columns in the center of the breadboard. Recall that these columns are internally connected vertically (however, there is a disconnect across the center gap). Note that the columns are numbered. So for example, the holes in column 20 are internally connected on either side of the center gap (but not to each other across the gap).

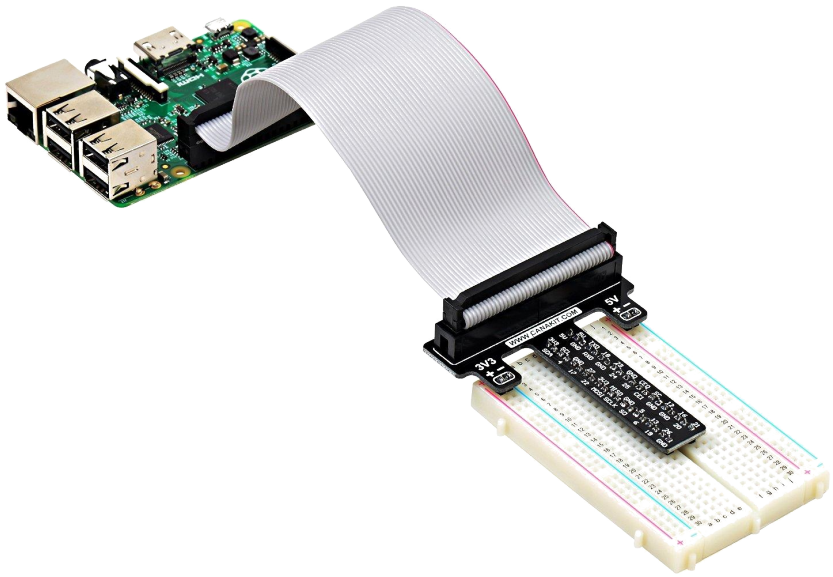
A red wire connected the 3.3V power source from the bottom row to the column that has the positive side of the LED (the long leg – or anode). A 68Ω resistor then connects the negative side of the LED (the short leg – or cathode) to ground. It does so by being placed across several columns (from the negative side of the LED to another column), and then by use of a black wire that brings ground to the other side of the LED.

In all, the circuit requires a red LED, a 68Ω resistor, and some jumper wires (oh, plus the RPi). Here is the circuit diagram:



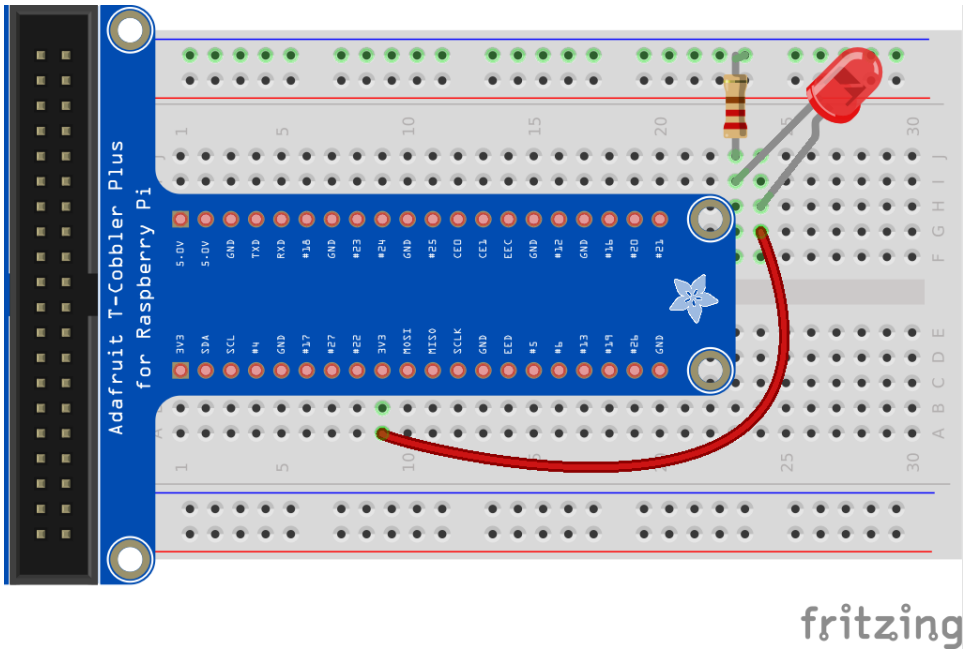
The 68Ω resistor has the colored bands: blue, gray, black. **Note that your kit does not come with 68Ω resistors!** The closest one in your kit is a 220Ω resistor (which will work just fine). It has the colored bands: red, red, brown. For the circuits in this activity, you will use a 220Ω resistor.

Since the power source for the circuit will come from the RPi, we need a way to connect the GPIO pins to the breadboard. One way, as in the layout diagram above, is to connect wires to the GPIO pins and the breadboard. The problem is that the wires in your kit aren't particularly well suited for this. Your kit does, however, include an T-shaped interface board that can extend the GPIO pins to the breadboard using a ribbon cable:



The interface board extends the GPIO pins to the central holes on the breadboard, and also provides 5V and ground to the bottom rows and 3.3V and ground to the top rows of the breadboard. **Connect the interface board and ribbon cable as show above.** Be careful to align the pins on the bottom of the interface board properly before pressing it down into the breadboard. Note that we will need to maximize the space to the right of the breadboard (the part that won't be used by the interface board).

We can now modify the layout diagram a bit to make use of the interface board:



Note that the interface board in the figure above is blue and slightly differs from the one in your kit. The GPIO pins, however, match correctly. We also assume that a ribbon cable connects the interface board to the GPIO pins on the RPi.

Create the circuit shown in the layout diagram above **without connecting the power adapter to the RPi yet**. Make sure that:

- The LED straddles two columns (i.e., does not have both of its legs in the same column of holes);
- You connect a wire from a 3.3V pin to the positive side (long leg) of the LED; and
- You place a 220Ω resistor from the negative side (short leg) of the LED to ground (note that the GPIO interface board brings ground to both blue rows at the top and bottom of the breadboard).

To summarize, use a red wire to connect a 3.3V power source from the interface board to the positive side of the LED. Use a 220Ω resistor to connected the negative side of the LED to ground. When you are certain that your circuit is correct, plug in the RPi. If everything is wired correctly, the LED should light.

Calculations

The resistor used in the circuit is a 220Ω resistor, the source voltage is 3.3V, and the LED has a forward voltage (i.e., voltage drop) of 2V (which we know from reading its data sheet). Using Ohm's Law, we can calculate the current that will flow through the circuit as follows:

$$\begin{aligned} V &= I * R \\ (3.3V - 2V) &= I * 220\,\Omega \\ 1.3V &= I * 220\,\Omega \\ 0.00591\,A &= I = 5.91\,mA \end{aligned}$$

The current through the LED will be 5.91mA. This is much less than the recommended 20mA. Note that the brightness of the LED is directly related to how much current flows through it. The more current, the brighter it will be. Of course, there is a limit (as per the data sheet).

We can calculate the power dissipated by the resistor as follows:

$$\begin{aligned} P &= V * I \\ P &= (3.3V - 2V) * 0.00591\,A \\ P &= 1.3V * 0.00591\,A \\ P &= 0.0077W = 7.7\,mW \end{aligned}$$

The resistor in your kit is a 1/4W (250mW) resistor. It is more than enough. The power dissipated by the LED can be calculated similarly:

$$\begin{aligned} P &= V * I \\ P &= 2V * 0.00591\,A \\ P &= 0.0118\,W = 11.8\,mW \end{aligned}$$

The LED in your kit has a forward current limit of 120mW. Again, it is more than enough.

Increasing the voltage

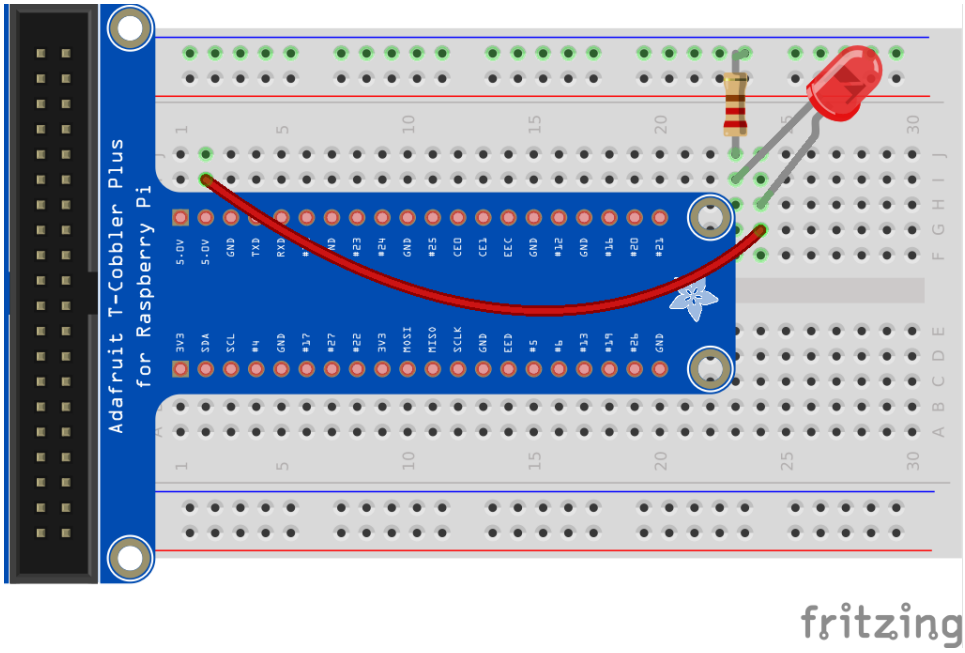
We cannot reduce the resistance in the circuit since there aren't any lesser-valued resistors in the kit. We can, however, change the source voltage! The RPi also has a 5V power source. Suppose you were to, instead, use the 5V power source from the RPi. This would provide more voltage to the breadboard and across the circuit. According to Ohm's Law, if we increase the voltage and keep the resistance in the circuit at 220Ω, the current has to increase. In the space below, calculate the current that would flow through the circuit with a 5V power source and a 220Ω resistor:

Now, calculate the power dissipated by the resistor for the 5V power source:

And finally, calculate the power dissipated by the LED for the 5V power source:

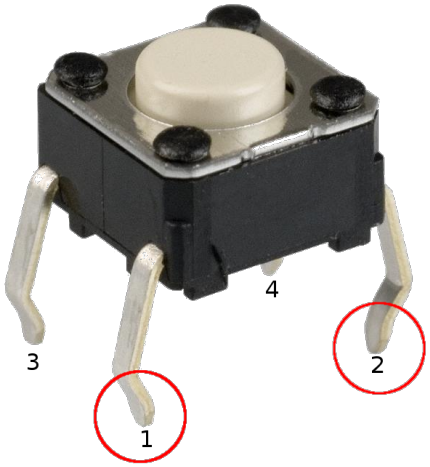
So with the 220Ω resistor and a 5V power source, we increase the current – which should make the LED appear a bit brighter when lit.

Alter your circuit as in the figure below. The only difference is that the positive side of the LED should now be connected to 5V instead of 3.3V. **You may want to unplug the RPi before making modifications and plug it back in when you are certain that your circuit is correct.**



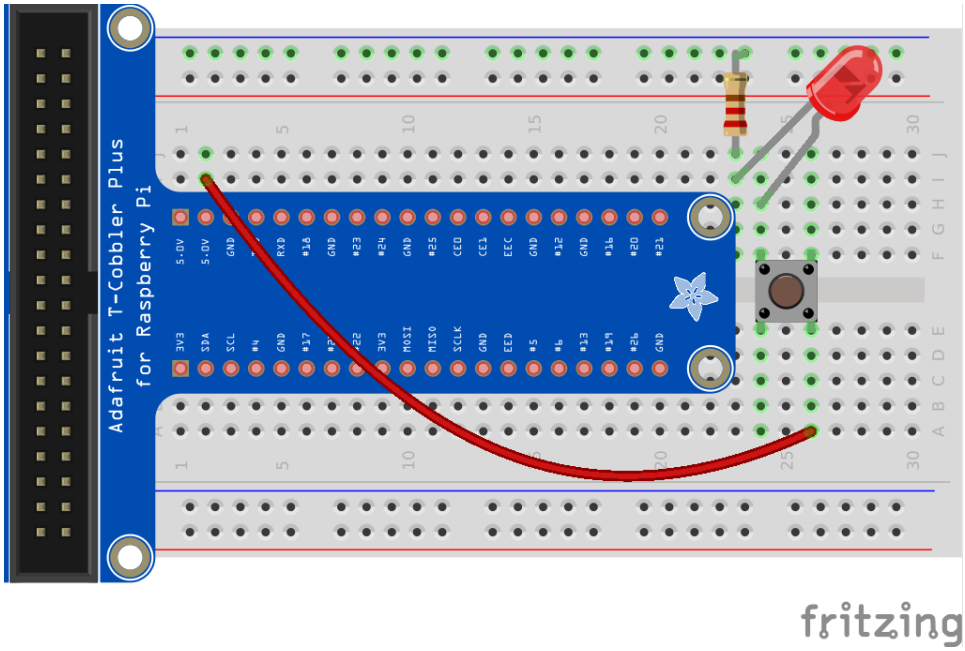
Adding a push-button switch

Let's add a push-button switch to the above circuit. The switch will control the flow of electricity to the circuit. If the button is pushed, current will flow and the LED will light. A push-button switch is a *tactile* switch that usually has four legs, split into two pairs. Each pair shares a common connection. In the figure below, legs 1 and 2 form a pair and share a common connection. That is, they are internally connected. So do legs 3 and 4:

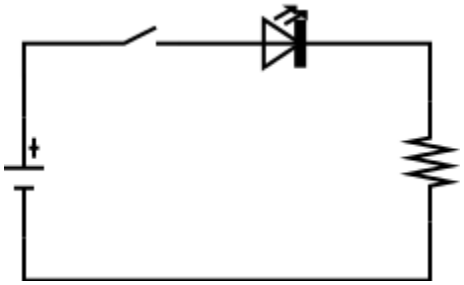


The switch is typically positioned in the center of the breadboard, **across the gap**. Pins 1 and 3 are above (or below) the gap. That is, separate pairs are on either side of the gap. Power is connected to one common pair (e.g., pin 3 or 4). The part of the circuit to be powered when the switch is pushed is connected to the other common pair (e.g., pin 1 or 2).

Modify your circuit by adding a switch as indicated below. Note that the switch in your kit is slightly larger and will take up a bit more space. Try to place it as close to the T-shaped interface board as possible. Since we will place a second switch later in this activity, you can test its placement by aligning the two switches side-by-side to guide you in placing the single switch.

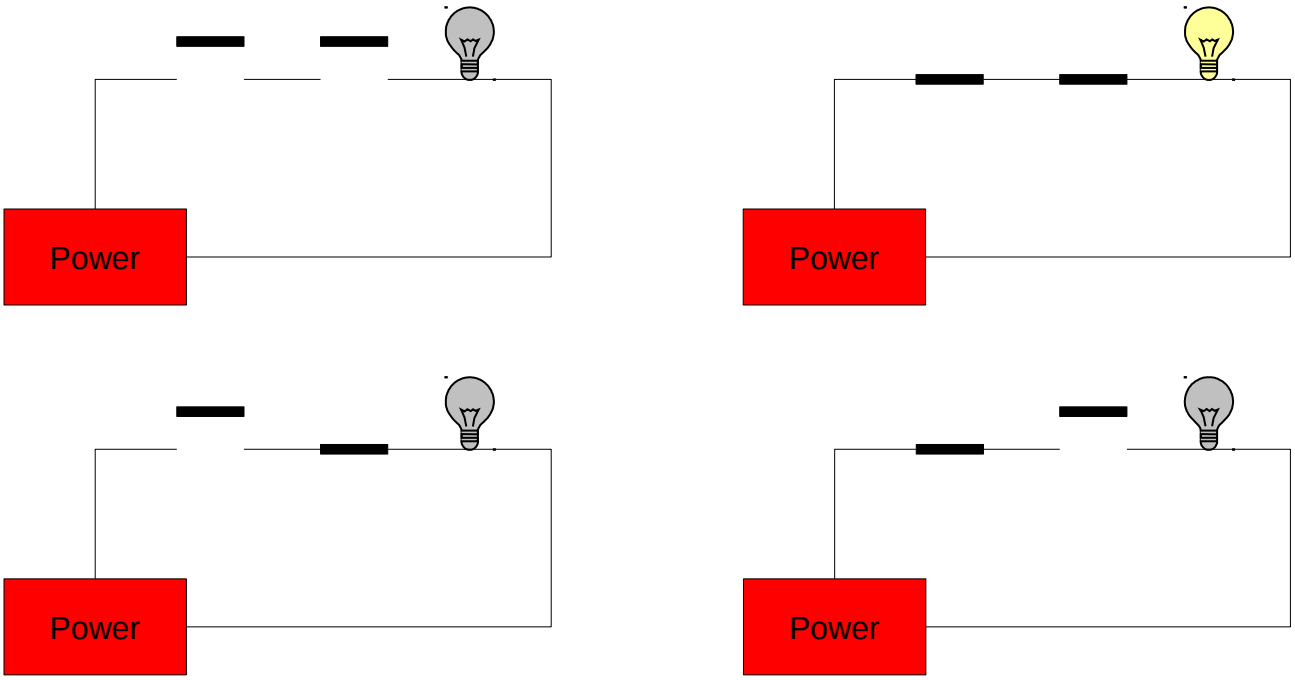


The 5V power source is connected to one side of the switch (i.e., one pair). When the switch is closed, a connection is made to the other pair. Since the positive side of the LED is connected to this other pair, it can light when the switch is closed! Here's what the circuit diagram looks like:



Recall that the circuits in a previous lesson also included versions with multiple switches (both in parallel and in series). We later related this to logic gates. The rest of this activity will have you experiment with various configurations of multi-switch circuits.

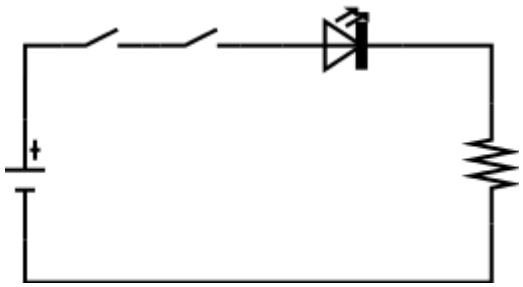
Replicating the *and* gate
Recall the following circuit:



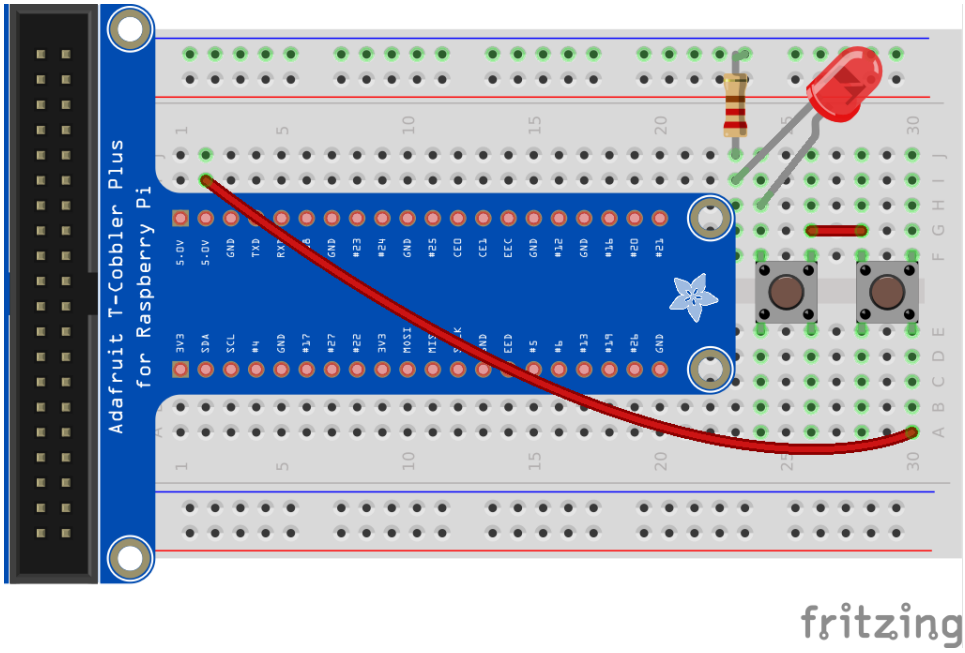
This circuit has two switches in *series*. Placing switches in this configuration in the circuit replicates the functionality of the *and* gate. Fill in the truth table for the *and* gate below:

A	B	Z

The output, *Z*, is only 1 (true) when both inputs, *A* and *B*, are 1 (true). In the circuit above, the light bulb is on (1) when both switches are closed (1). We can create a circuit diagram of this:



To implement this in your LED circuit, modify it as follows:

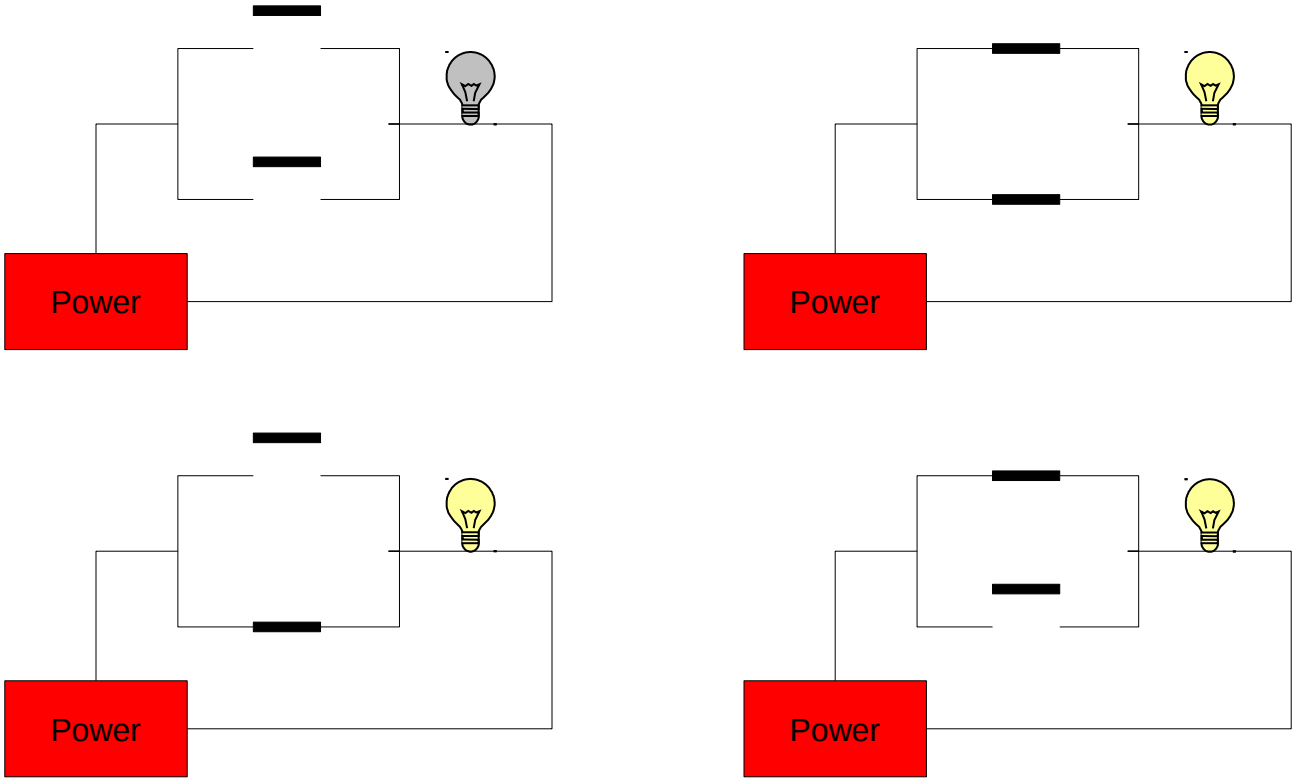


Power is extended to one of the switches (via the long red wire). Note that the connection is only to one of its common pairs (say, pins 1 and 2). This first switch is then connected to the second switch. That is, the first switch's second common pair (say, pins 3 and 4) is connected to one of the second switch's common pairs (say, pins 1 and 2). The second switch's second common pair (say, pins 3 and 4) is connected to the positive side of the LED (they are in the same column). The rest of the circuit (resistor from the negative side of the LED to ground) is the same.

Try the circuit. The LED should only light when *both* switches are closed.

Replicating the *or* gate

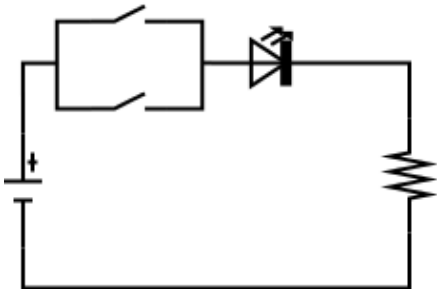
We can also implement the functionality of the *or* gate as follows:



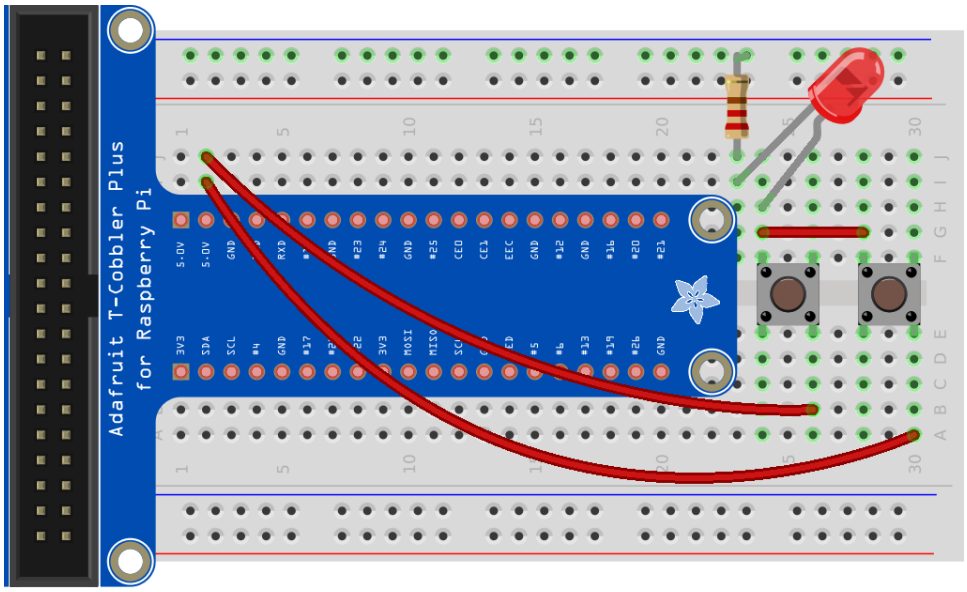
This circuit has two switches in *parallel*. Placing switches in this configuration in the circuit replicates the functionality of the *or* gate. Fill in the truth table for the *or* gate below:

A	B	Z

The output, *Z*, is 1 (true) when any input (*A*, *B*, or both *A* and *B*) are 1 (true). In the circuit above, the light bulb is on (1) when either switch (or both) are closed (1). We can create a circuit diagram of this circuit as follows:



To implement this in your LED circuit, modify it as follows:



Power is extended to a common pair **of both switches** (via the two long red wires). The second common pair **of both switches** is connected to the positive side of the LED.

Try the circuit. The LED should light when either (or both) switches are closed.

Sensing

The previous circuits in this activity have been entirely external of the RPi. That is, the RPi was only used as a power source (basically, a battery). There are many more GPIO pins than the ones we have used so far (3.3V, 5V, and ground). In fact, many of the GPIO pins can be used to provide sensor input to the RPi. Others can be used to provide output capabilities from the RPi. For example, we can programmatically detect when a switch is closed and trigger an LED to light.

Let's start with a simple example: lighting an LED. To do this in Scratch, we will need to install an extension that allows access and provides programmability to the GPIO pins on the RPi.

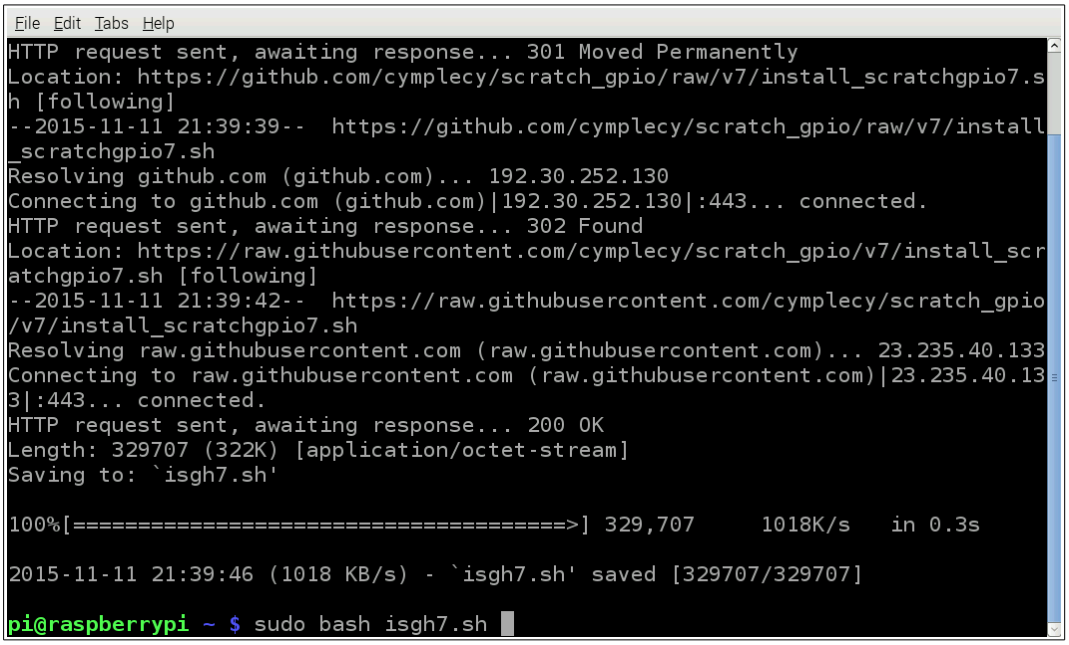
Installing ScratchGPIO

ScratchGPIO is an extension/addon to Scratch that allows us to write scripts that manipulate the RPi's GPIO pins. To install it, you must first open up a terminal and type the following command:

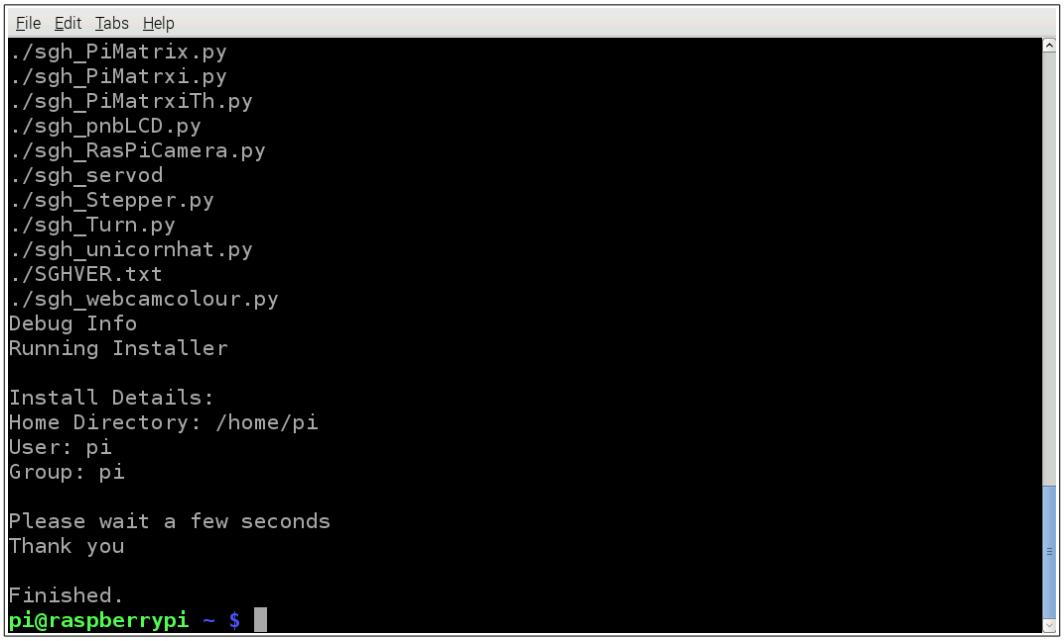
```
wget http://bit.ly/1wxrqdp -O isgh7.sh
```

This command fetches ScratchGPIO from the Internet and saves it on the RPi. **Note that you must be connected to the Internet for this to work!**

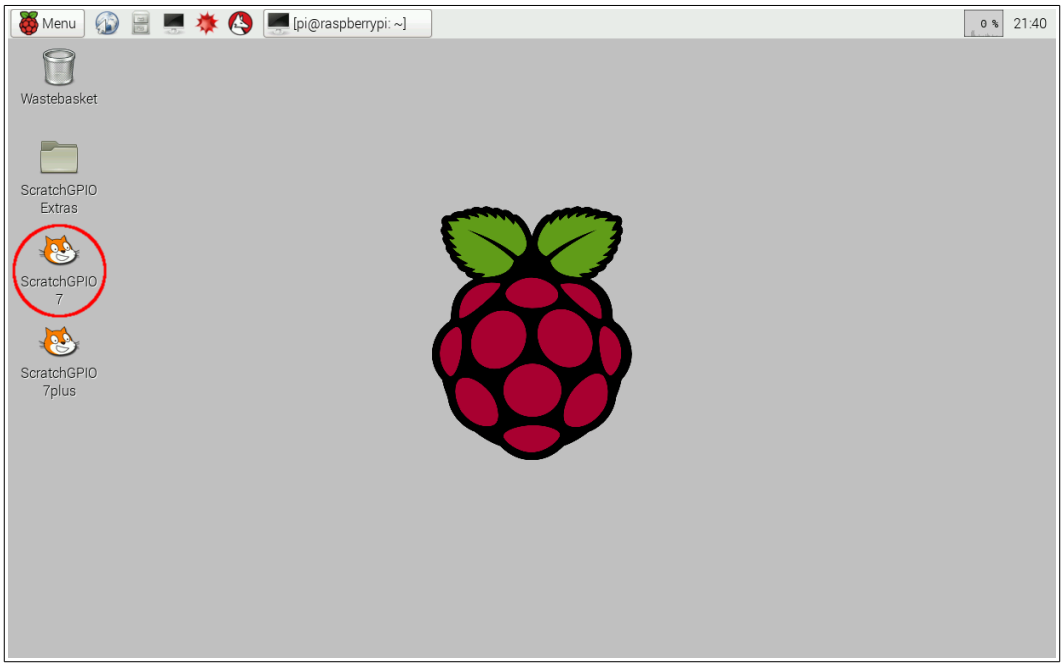
Now, install ScratchGPIO by typing the following command: `sudo bash isgh7.sh`



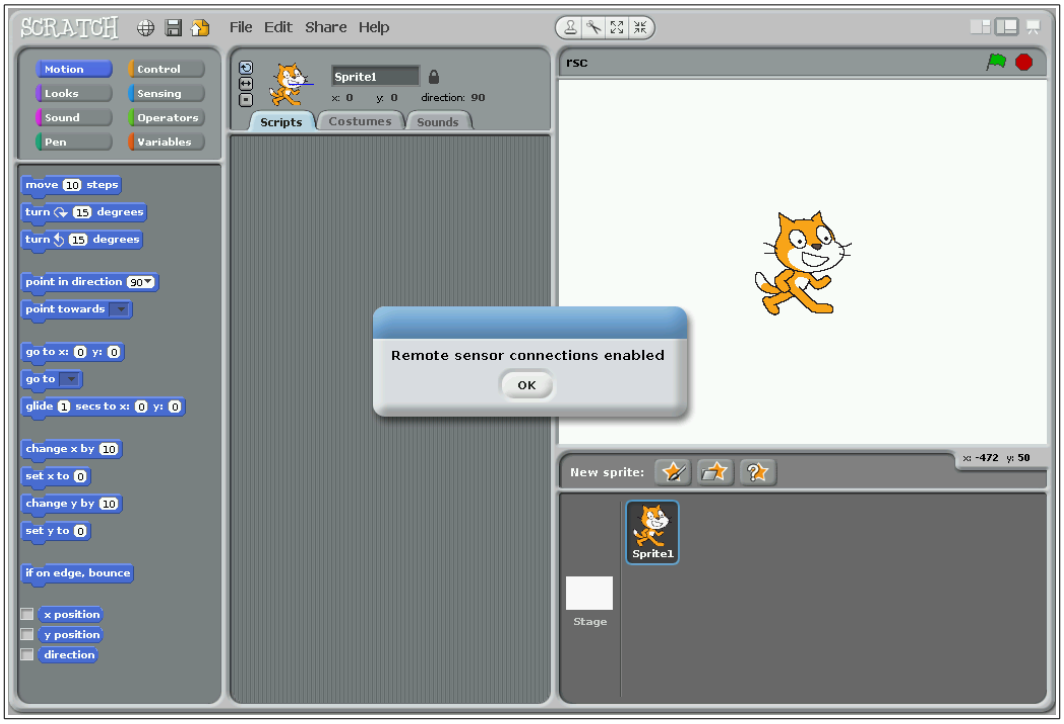
This will install ScratchGPIO and place two new icons on your desktop.



You can now exit the terminal and return to the desktop by typing: `exit`. To launch Scratch with GPIO support, double-click on the ScratchGPIO 7 icon:

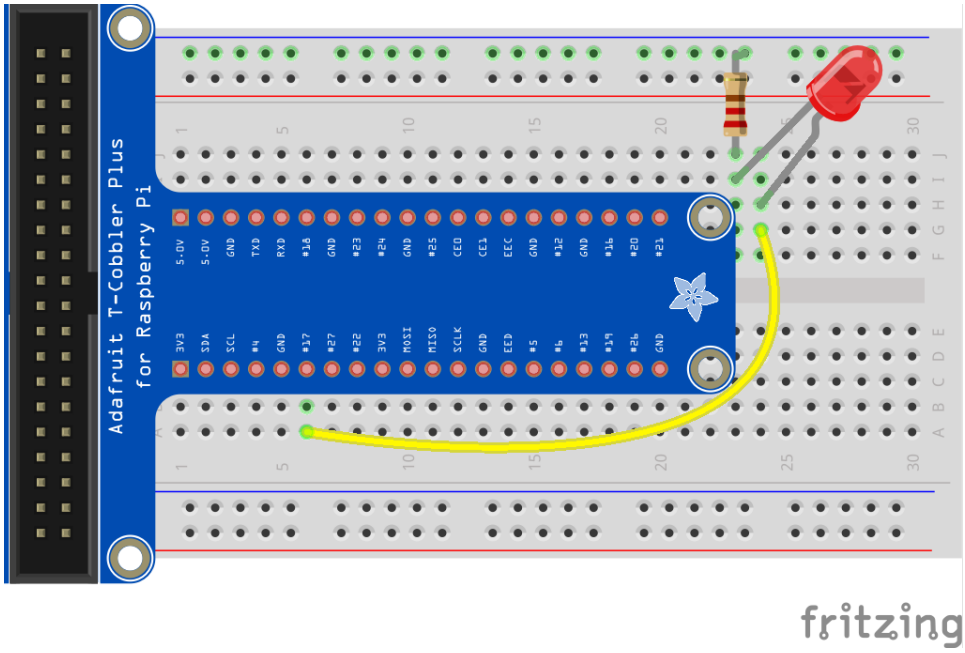


Once loaded, it will provide some note that it has enabled access to program the GPIO pins:



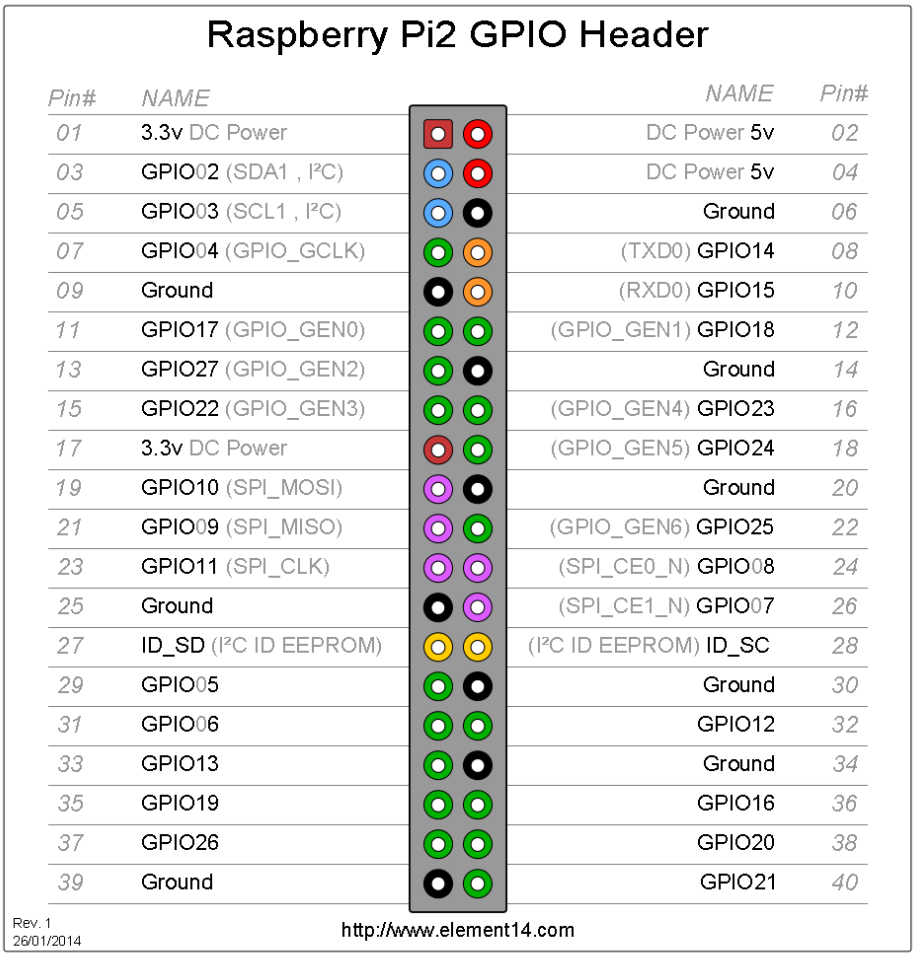
Lighting an LED

Construct the following circuit (which slightly differs from the first part of this activity):



Note that the yellow wire connects the positive side of the LED to GPIO 17. On the T-shaped GPIO interface board, it is shown as GPIO 17; however, on the RPi it is actually pin 11.

There are several ways to refer to the GPIO pins on the RPi. Here's some detail on matching RPi pins to their GPIO numeric or functional equivalents:



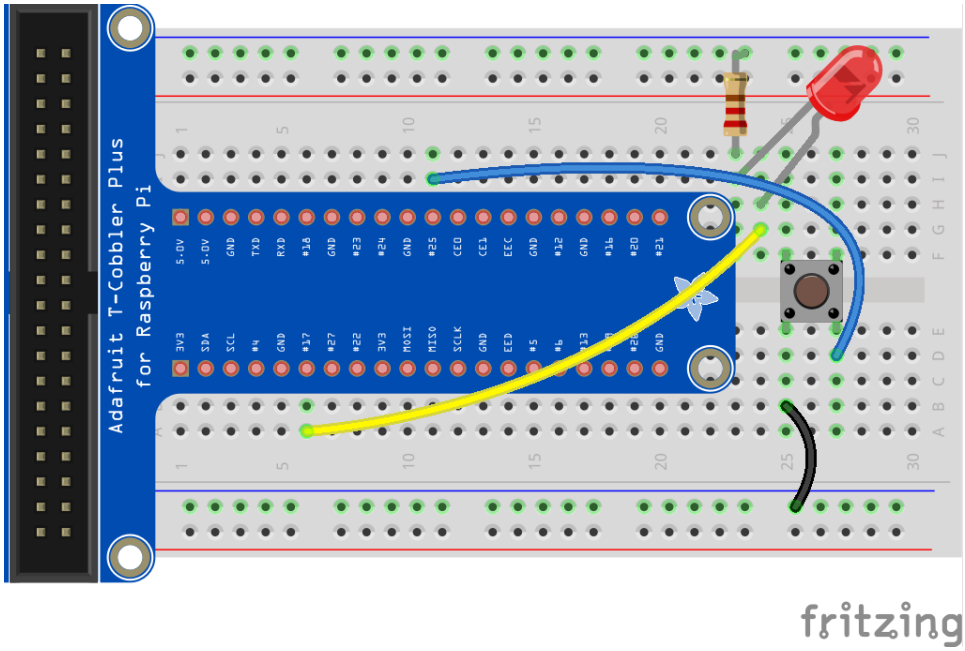
Another example: GPIO 25 is pin 22 on the RPi, and GPIO 5 is pin 29 on the RPi. In ScratchGPIO, we can specify output GPIO pins either way (e.g., as pin11 or gpio17). Input GPIO pins must be specified by their pin number (e.g., pin22 or pin29). We'll see this in coming examples.

To make the LED in the circuit above turn on, we need to turn pin11 (or gpio17) on. In ScratchGPIO, this is accomplished by broadcasting as follows:

Notice that we first broadcast **gpio17on** and wait for one second. This effectively turns on GPIO 17 by providing 3.3V to the pin. After one second, we turn it off. So this script should make the LED blink on and off, each of which should last one second (i.e., on for one second, off for one second, on for one second, and so on).

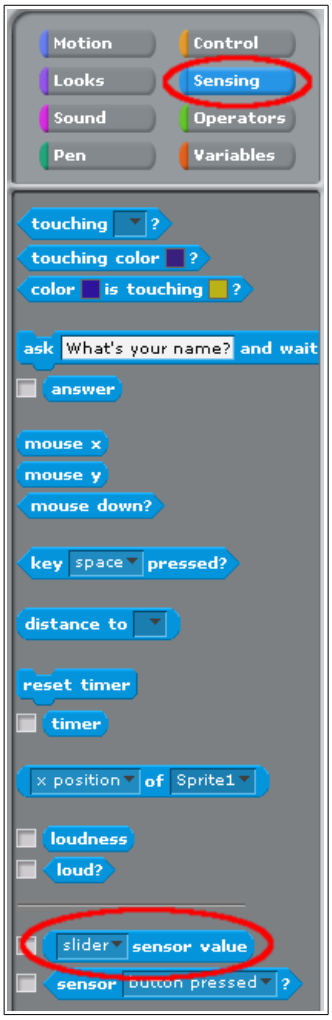
Adding a switch

Let's add a switch to the mix. To make this work, we want to programmatically detect the status of the switch (open or closed). If the switch is closed, then the LED should turn on; otherwise, it should remain off. To begin, implement the following circuit:



The LED portion is unchanged from the last circuit. The only difference is the addition of a switch. One common pair is connected to ground (the upper horizontal row in at the bottom of the breadboard). The other common pair is connected to GPIO 25 (pin 22 on the RPi according to the image shown earlier).

Detecting the state of the switch is not particularly difficult. We can poll (or check) the state of input pins using the **sensor value** block in the sensing blocks group:

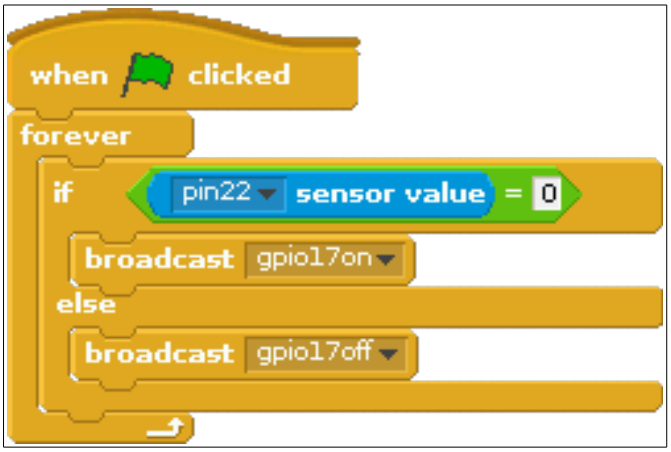


The drop-down menu allows us to select an input pin to *sense*. In the case of the circuit shown earlier, we want to detect the state of the switch that is connected to pin 22. Therefore, we will select pin22 as the sensor value.

Input GPIO pins (like pin 22) are high (1) by default. That is, 3.3V is provided to the pin **by default**. To detect a switch press, we must bring the input pin down to 0V (or ground). In the circuit, the switch, when pressed, connects the input pin to ground (since the other common pair of the switch is wired to ground). Although this method of detecting may appear to be backwards, it is actually used often. In ScratchGPIO, a pin that is on (or high) will read as 1, and a pin that is off (or low) will read as 0. Here's a neat way to observe the behavior of an input pin changing state:

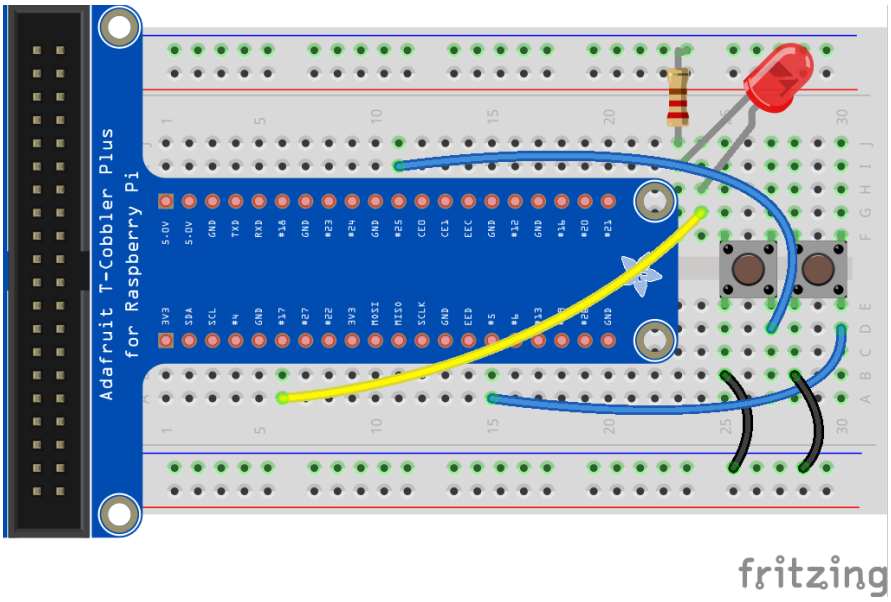
For this script, first create a variable called `switch` (in the variables blocks group). In the script above, since the push-button switch is connected to pin22 (GPIO 25), the variable `switch` will be set to the state of the pin. Try it. Notice that when the switch is open, the value is 1 (high); and when the switch is closed (pushed), the value is 0 (low).

To *connect* the switch to the LED (i.e., to make the switch control the LED), implement the following script:

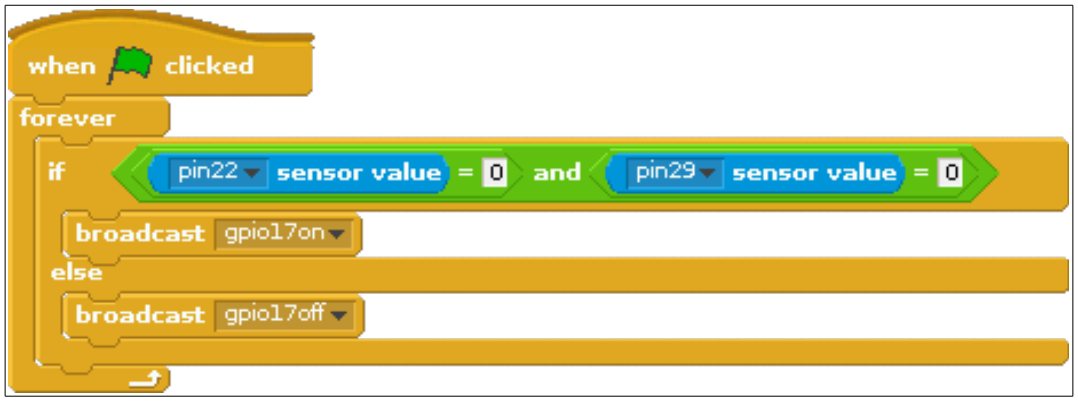


Here, we are simply either turning the LED on or off depending on the state of the pin that the switch is connected to. Since the switch is connected to pin22 (GPIO 25), then we want to turn the LED on (via the **broadcast gpio17on** block) when the switch is pressed (i.e., pin22 is low – or 0). Otherwise, we turn the LED off.

Two switches (to implement *and* and *or*)
Recall that to manually implement *and* and *or*, you had to wire two switches either in series or parallel. Programmatically doing so precludes this. The logic can be done in ScratchGPIO! Let's try this by first implementing the following circuit:

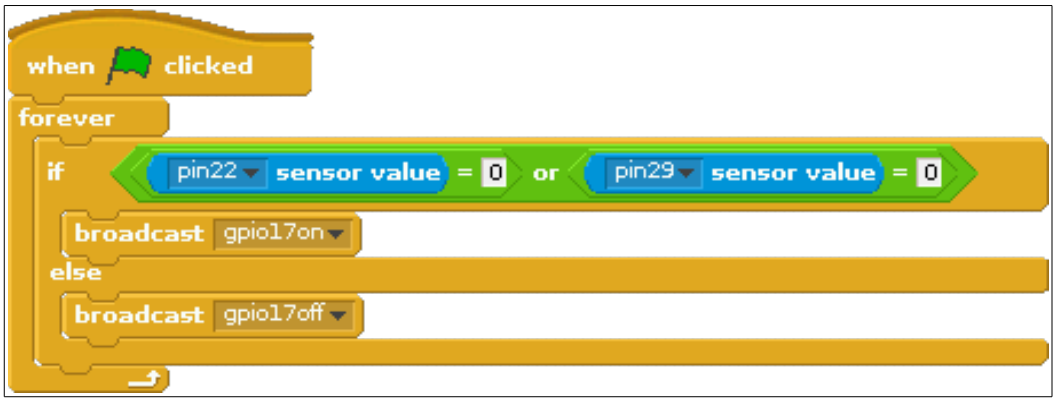


The only difference in this circuit is the addition of the second switch. It is connected to ground and to pin29 (GPIO 5). To implement the functionality of *and* (i.e., replicating two switches in series), we simply need to turn the LED on when both switches are low (0). We can do this by implementing the following script:



The logic is actually quite clear: the LED on GPIO 17 is turned on if both pin22 **and** pin29 are low (0). Recall that they are brought down to ground (0V, or low) when the switches are closed. The control block in Scratch **[] and []** is an implementation of the *and* gate. Both conditions (on the left and right) must be true in order for the entire block to be true. In this case, both pin 22 and pin 29 must be low (0) in order for the true part of the **if-statement** to be executed. Otherwise, the false part is executed. This means that both switches must be closed in order for GPIO 17 to be turned on (i.e., to turn on the LED).

Of course, implementing the *or* gate is just as easy. We simply switch the *and* block with an *or* block. The rest of the logic is exactly the same:



Homework: Blink!

Implement a single LED, single switch circuit and write a script that does the following:

(1) The LED should blink continuously such that it is on for 0.5s and off for 0.5s;
(2) When the switch is pressed (closed), it should change the LED's blink rate so that it is on for 0.1s and off for 0.1s (i.e., it should make the LED blink faster); and
(3) When the switch is released (open), the LED should go back to blinking at the original rate of 0.5s on and 0.5s off.

You are to submit your Scratch v1.4 (not v2.0!) file (with a .sb extension) through the upload facility on the web site.

Did you know?

When circuits are continuously toggled (such as when an LED is turned on and off, over and over), we can refer to the portion of time that the circuit is on as a duty cycle. Formally, a **duty cycle** is the percentage of one period in which a signal is active. A period is the time it takes for a signal to complete an on-and-off cycle. In a simple LED circuit, a duty cycle of 50% means that the LED turns on and off for the same amount of time (e.g., the LED turns on for one second, off for one second, and so on). A duty cycle of 25% means that the LED turns on 25% of the time (e.g., the LED turns on for 0.25s, off for 0.75s, and so on).

Homework

Complete and turn in the questions on the following two pages.

Raspberry Pi Activity 5 Assignment: LED the Way

Name: _____

Calculations

Suppose that a circuit has a power source voltage of 9V, and an LED with a forward voltage (i.e., voltage drop) of 2.5V that requires 25mA of current for optimum brightness. In the space below, calculate the resistance of the series resistor required. State the formula used as the basis for your answer, identify all units, and show all work!

In the space below, calculate the power dissipated by the resistor. State the formula used as the basis for your answer, identify all units, and show all work!

In the space below, calculate the power dissipated by the LED. State the formula used as the basis for your answer, identify all units, and show all work!

Truth tables
Fill in the truth table for the *and* gate below:

A	B	Z

Fill in the truth table for the *xor* (exclusive *or*) gate below:

A	B	Z

Circuits
Using the figure below, draw the single-switch, single-LED circuit in which the RPi was responsible for detecting the state of the switch (as input) and accordingly affecting the blink rate of the LED (as output):