Raspberry Pi® CoderDojo Belfast

Python Pi Project #4: Blinking LED

Level: Easy

One of the great things about the Raspberry Pi is its "general purpose input/output" - GPIO, for short.

That's the double row of little pins in the corner of the Pi. You can use these to connect up lots of different electronic components to the Pi, like buttons and other controls, lights, motors, sensors and all sorts of things. And once you've wired them up, you can write your own programs on the Pi to control them. This makes the Pi amazingly versatile. You're only limited by your imagination in what you can do.

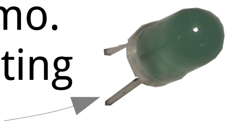
Hacking skills needed

- ➔ wiring components on a breadboard
- ➔ using the GPIO module

Hello World?

One of the first programs that programmers write when they're learning a new language is one that prints out "Hello World". It's a surprisingly good way of getting a feel for what a language is like. Can you write it in Python?

Let's try this out by building the electronic equivalent of the "Hello World" program on the Pi. This is the "flashing LED" demo. An LED is a Light Emitting Diode – one of these

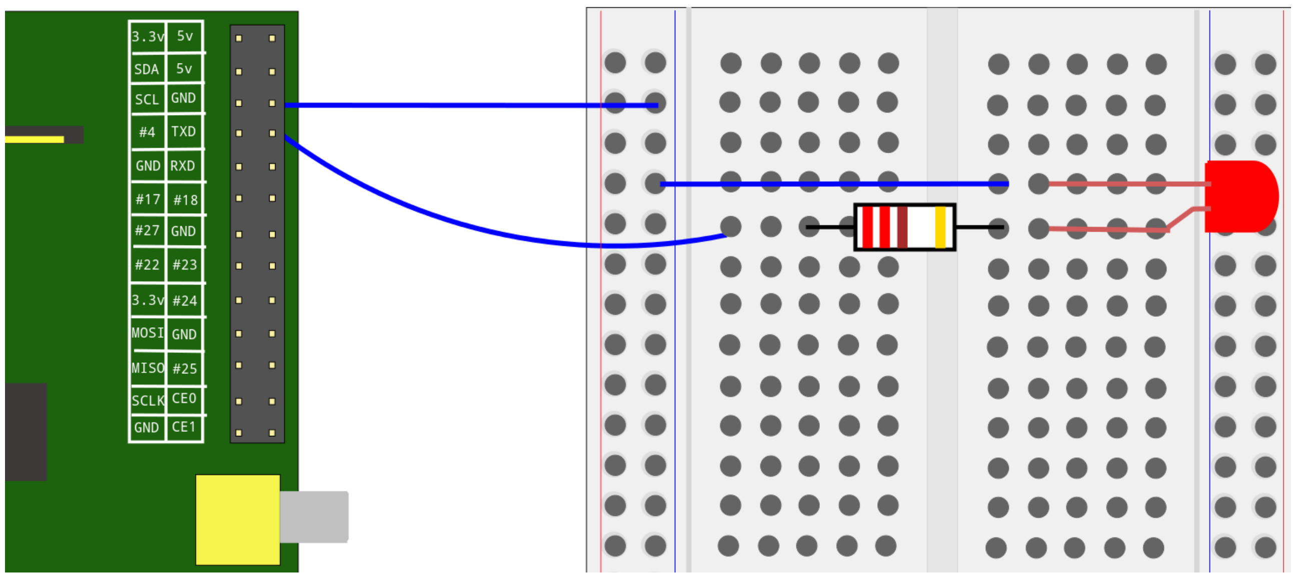


We'll write a program that makes one blink on and off repeatedly.

But before we do that we'll have to build an electronic circuit!

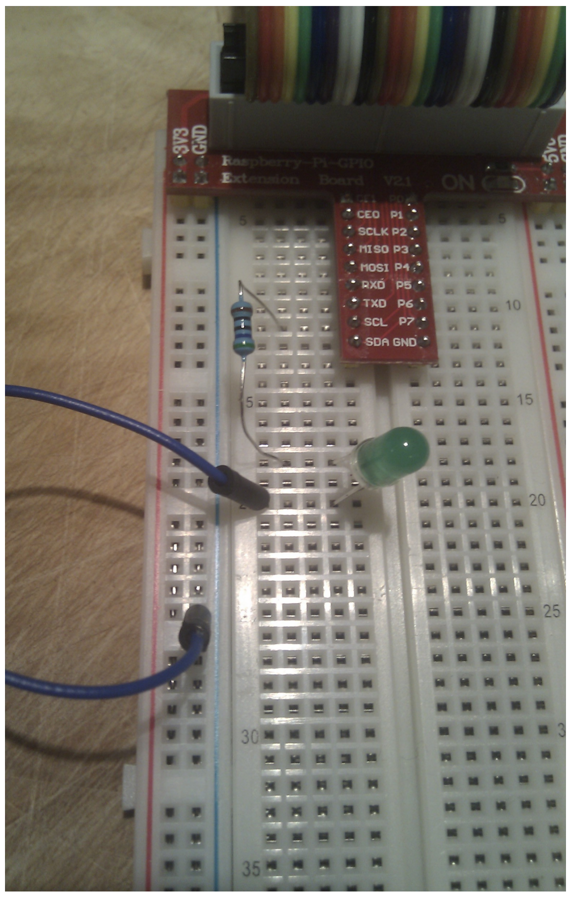
A little electronic circuit

In the picture below is the circuit we're going to build. We'll need just a couple of electronic components – the LED itself, and a resistor; a few wires, and the board to plug it all into – the "breadboard". The breadboard has pinholes to plug our electric connector wires into. There are two columns of pinholes down either side, beside the red and blue lines. The ones beside the red line are all connected, and we can connect them to power output pins on the Pi. The ones beside the blue line are all connected and we can connect them to the electrical 'ground' pins on the Pi.



The other pinholes are arranged in rows on either side of the board. The pins in each row on either side are connected to each other, but not to the rows above or below.

We'll use a 'T' connector to help make thing easier. We'll start by connecting the “TXD” pin on the Pi to the resistor. This pin is pin 8, reading the pin numbers like a book, left to right and top to bottom. It has a voltage of 3.3v. We plug one end of the resistor into the same horizontal row as the TXD pin on the T connector, and the other end into a row further down the board.



How to use a 'T' connector

To make connecting pins and wires easier, we'll use a 'T' connector. This plugs into all the pins on the Pi on one end of a ribbon cable – put the brown edge beside pin 1 at the corner of the Pi. The other end of the cable plugs into the connector, and the connector plugs in to the breadboard. Follow the labels on the connector to know which pin on the Pi you're working with.

Now we'll connect the resistor to the LED - plug the positive terminal (leg) of the LED (the long one, with the bend in it) into the same row as the resistor, and the other end into a different row. Finally we wire up the negative terminal of the LED to 'ground', by connecting a cable to the negative (short) leg of the LED, and the other end of the cable into the 'ground' (blue) column of the breadboard.

Programming the LED

That's it! Our circuit is ready. Now all we need to do is make it do something! We'll write a Python program to make the light flash.

First we need to make sure we have the Python modules that let us use GPIO:

```
$sudo apt-get install python-dev
$sudo apt-get install python-rpi.gpio
```

To use GPIO in a Python program, we need to import the module:

```
import RPi.GPIO as GPIO
```

Now we'll do a bit of setup. We'll turn warnings off to limit the amount of output. Then we set the 'mode' of the GPIO module to 'BOARD'. This just sets up the numbering of the pins the way we described above - left to right, top to bottom.

```
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BOARD)
```

That makes our pin 'TXD' number 8. We'll define a variable so we can refer to the pin with a name; and we'll use that to tell GPIO that we are going to use the pin for output, in order to send a signal to the LED:

```
led_pin = 8
GPIO.setup(led_pin, GPIO.OUT)
```

Now we go into a loop, using 'while True'. To make our LED blink, we'll just turn it on and off again regularly. We turn it on with

```
    GPIO.output(led_pin, True)
```

and off with

```
    GPIO.output(led_pin, False)
```

In between each of these we'll sleep for half a second.

And that's it. See "The complete program" below for the whole thing.

Dojo Challenge:

What else can you make the LED do? How about a countdown, flashing once a second, then an 'explosion', blinking the LED really fast for a couple of seconds.

Can you combine the input statement from Project #3 with the LED, to write a simple guessing game? It can be pretty dumb – pick a random number from 1 to 10, then let the user guess the number until they get it right, and flash the LED to let them know they did.

The complete program - blink.py

```
#!/usr/bin/env python

import RPi.GPIO as GPIO
from time import *
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BOARD)
led_pin = 8
GPIO.setup(led_pin, GPIO.OUT)
while True:
    GPIO.output(led_pin, True)
    sleep(0.5)
    GPIO.output(led_pin, False)
    sleep(0.5)
```