4.4.1 Sending and Receiving Signals Using GPIO Pins

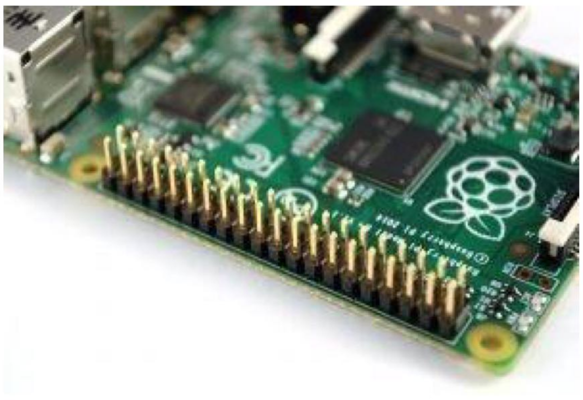
In recent years, the Raspberry Pi has become popular largely as an inexpensive, compact Linux box for media and retro video games, as well as a network device. Some hobbyists go on to use their Pi these ways for years, all without knowing what the pins on the side of their device really do.It's these pins that hold the true power of the Pi. They can control homes, machines, new inventions, and even robots, so why is it so many people don't know what they really are?

What Do These Pins Actually Do?

These 40 (or 26, depending on your Pi model) pins are part of what's known as the "GPIO Header". Within this header, there are four main kinds of pin;

- **Power**: These provide DC power at 3.3 and 5 volts
- **Ground (GND)**: These connect to ground, to close your circuit
- **DNC**: This stands for "do not connect", so don't worry about them
- **GPIO**: These can be set to either send, or receive control voltages

GPIO stands for 'General Purpose Input/Output', and it's these pins that let the Raspberry Pi do its magic. This is because the pins have no specific function, and can be set to a dedicated purpose, such as controlling a signal.



A GPIO pin set to output can provide either 3.3 volts, known as a HIGH signal, or 0 volts, known as a LOW signal. When set to input, it can read these same voltages.

GPIO Pins Don't Provide Much Power

It's important to remember that GPIO pins (and the 3.3-volt power pins) are meant to control and communicate with other components.

You can get about 51mA from all 3.3 volt pins combined, but you'll want to take care when connecting; if your circuit tries to pull too much current through these 3.3 volt pins, you can fry the whole board.

The 5-volt power pins, on the other hand, give you all the power available from the power supply, minus the bit used by the Raspberry Pi itself.

Connecting Your GPIO Pins to a Breadboard

When you first start using these GPIO pins, it's wise to use a breadboard. This makes it easy to build circuits without solder and to modify them.

If you've never used a breadboard before, familiarize yourself with the basics here:

A GPIO extension board also helps immensely. This connects the GPIO header via a ribbon and places the pins directly on the breadboard, in a clearly labeled manner.

It requires some real estate though: 20 rows each side of the breadboard. On a small board, that's nearly the whole thing! A breadboard with 40 rows or so leftover gives plenty of space for beginner projects.

GPIO Pins With Special Uses

Every GPIO pin can be set to send and receive HIGH and LOW signals. Some have special uses too.

Hardware PWM

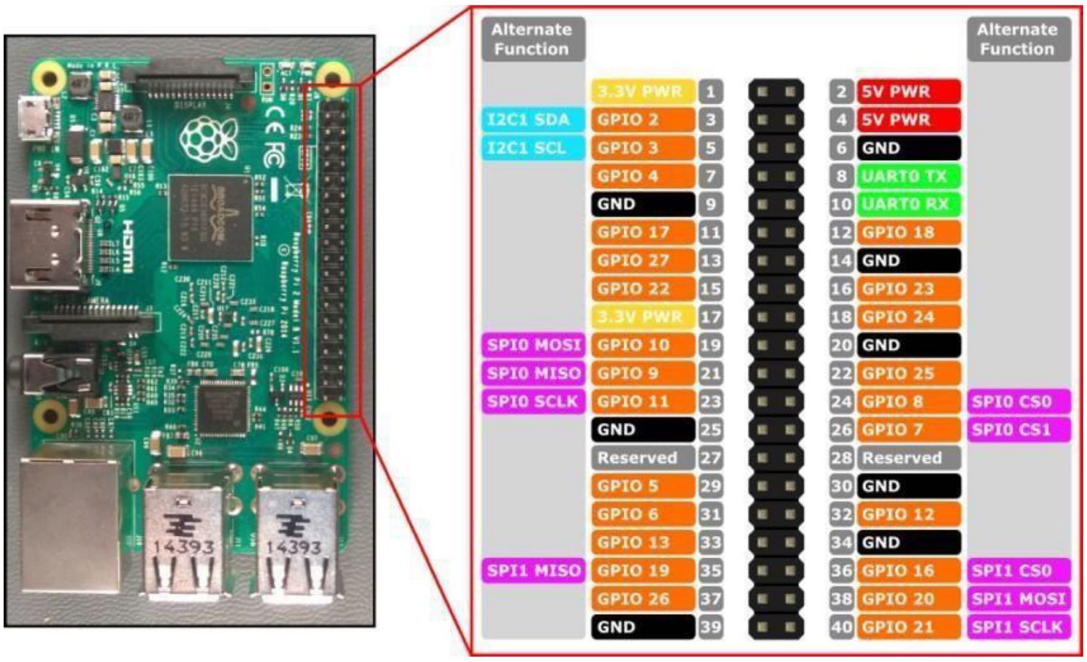
GPIO pins output either 3.3 or 0 volts: a HIGH or LOW signal. Pulse width modulation, or PWM, is a way of simulating the range of voltages in between by flickering the pin on and off rapidly. This isn't a true analog signal, but it's fine for something like dimming an LED. It will flicker faster than you can see and simply appear dimmer.

You can also use a low pass filter to smooth a PWM into an analog signal. This can be used for analogue audio, if you aren't fussy about quality. It's fine for a doorbell or toy.

You can generate a PWM signal from any GPIO pin using software, but the operating system juggles this with other tasks, so this signal can jitter.

Serial Bus Pins

If you take a look below at the diagram (known as a Raspberry Pi 'Pinout') you'll see that some pins are I2C, SPI, and UART serial. These are serial bus protocols that can be used to send and receive data from other components.



You can combine these with a digital-analog converter, or DAC, to output an analogue signal. This can be preferable to PWM for high quality audio, or to control many components.

Pull Up and Pull Down Resistors

Often you will want your Raspberry Pi GPIO pin to read the position of a button or a switch. That's easy to do by wiring it so that it closes a circuit attached to the control voltage to read HIGH, or to ground to read LOW.

The problem is that when this circuit is open and nothing else is attached to the pin, it might return any value. This is known as "floating," and it's extremely unhelpful.

You can prevent floating with "pull up" or "pull down" resistors.

A pull up resistor is wired to your control voltage; when nothing else is attached, the pin will read HIGH. A pull down resistor is wired to ground; the pin will read LOW. Use whichever provides the opposite value to your switch or button.

You don't need to wire these resistors into your circuit. They're inside the Raspberry Pi already and you can control them from software.

Using Software to Control GPIO Pins

Among the easiest ways to control GPIO pins is by using the GPIO Zero library in Python. If you've written any Python before, you'll pick this up easy.
If this is your first time using Python, If you don't, the commands below will still work; you'll just be less able to follow along. The web version of 'Automate the Boring Stuff With Python' is excellent and costs nothing.

GPIO Zero is installed by default on Raspbian Desktop images. If you are using Raspbian Lite or a different operating system, you may need to install it.

Let’s Use All This to Switch a Light On

Let’s have a go at turning on an LED! A job this simple doesn’t really require a computer, but we’ll involve the Raspberry Pi in the GPIO pins.

Connecting the Power Rails
If you’re using the extension board, connect it to the Raspberry Pi and to the breadboard. Then attach the 3.3-volt power pin to the positive power rail running across the bottom of the breadboard, and the ground pin to the negative power rail.

Connecting and Testing the Button
Now add your button to the middle of the breadboard. Connect one pin of the button to a Raspberry Pi GPIO pin. I’m using 13, because it’s my lucky number.

Then, connect the diagonally opposite pin of the button to the negative power rail. When you push this button down, the circuit closes.
If you get a message saying ‘ImportError’, make sure you capitalized it correctly. If it says ‘ModuleNotFoundError’, you need to install GPIO Zero. Otherwise, it’s time to assign our pin to the button:

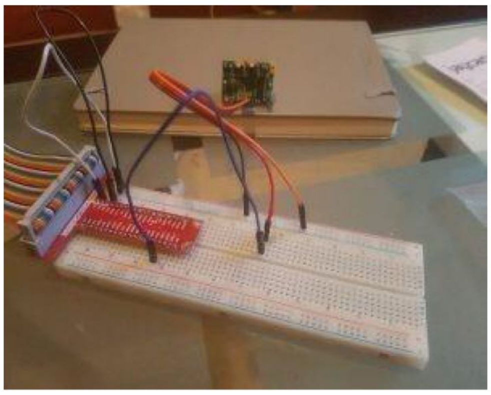
```
button = Button(13)
```
This Button class takes care of assigning the pull up resistor. Now let’s test that it works by typing the following lines:

```
while True:
if button.is_pressed:
print(‘Sweet, the button works!’)break
```

Python is fussy about indentation, so be sure to copy the spaces too. Then press Enter again torun the loop.

This loop will run until someone presses the button, so press it. It should produce a message saying the button works. This means you’ve successfully built a simple circuit that sends a message to your Raspberry Pi. If this doesn’t work, check that everything is connected properly
and try again.

Connecting and Testing the LED
The D in LED stands for “diode”, which means electricity only runs in one direction through it. You will notice that one leg of the LED is slightly longer: this connects to positive. Here, that means connecting to the GPIO pin. I’m using pin 26, for no particular reason. Place the LED in the breadboard, making sure that the legs are spaced horizontally so that you aren’t shorting the connection out. Now connect this positive leg to your GPIO pin.An LED should be wired in series with a resistor, so attach one end of your resistor to the short leg of the LED, and the other end to the negative power rail. Resistors can go in either way around.



Now let’s go tell the Raspberry Pi what’s going on. Type: Image: Pi & Breadboard

```
from gpiozero import LEDled = LED(26)
```

If everything’s wired correctly, you can switch it on and off with these commands:

```
led.on()
led.off()
```

Controlling the LED With the Button

Now that everything’s connected and you’ve checked they work, type:

```
button.when_pressed = led.on
```

Now press the button. The LED should switch on and stay on. Now type:

```
button.when_released = led.off
```

Press the button again; the LED should switch off when the button is released.