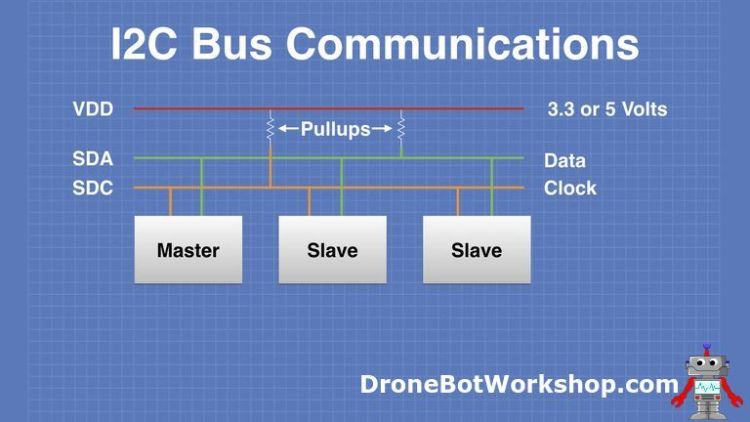
I2C operation

As a quick recap I2C, or the ”Inter-Integrated Circuit Bus”, is a method of exchanging serial data between two or more devices. An I2C circuit consists of one bus “Master” and one or more bus “Slaves”.

The bus uses four connections:

* **SDA** – the serial data line,. This is a bidirectional data line, handling all communications between the master and slave(s).
* **SCL** – The clock line. This provides the clock signal to synchronize the data on the SDA line.
* **VCC** – This is the logic-level voltage reference, typically either 5-volts or 3.3.-volts.  In many arrangements, this voltage is also used to power the slave device.
* **GND** – A ground reference.

The arrangement of master and slaves(s) is illustrated below.



Notice the use of pull-up resistors. These resistors pull the logic and clock levels up to the level of the VCC reference voltage. This concept is very important to understand when interfacing devices with mixed logic levels.

Another important concept is that it is the Master that determines the logic voltage level. Keep that in mind when we start hooking up our Raspberry Pi and Arduino.

Mixing Logic Levels

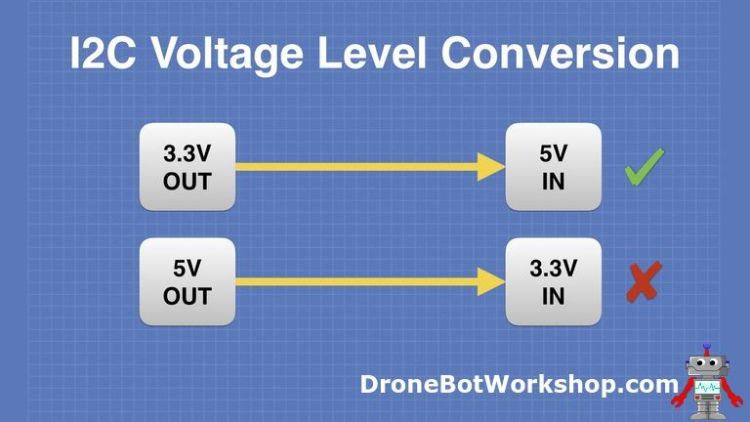
5-volt logic, also sometimes referred to as “TTL logic”, has been around for many decades.  The original microprocessors of the 1970s used this type of logic, as did the discrete CPU designs before them.

Despite its age 5-volt logic is still very common, and it’s no coincidence that the standard USB voltage is 5-volts (although the newer USB-C can make use of multiple voltages).

3.3-volt logic devices have also been around for many years, they are popular as they consume less current and are therefore ideal for battery-powered devices.

The rule for connecting the two logic families together is pretty simple:

* You may connect a 3.3-volt output to a 5-volt input.
* You may NOT connect a 5-volt output to a 3.3-volt input.



Interestingly, a 3.3-volt logic signal is capable of working properly when connected to a5-volt input. This is because most 5-volt logic chips have a threshold of a bit less than 3-volts, in other words, a logic signal of 3-volts or more will be recognized as a valid signal.

An exception to this is a Schmitt Trigger, a logic gate that has very narrow thresholds for zero and one. This type of gate is often used to “clean up” a noisy logic line and will not be triggered by a 3.3-volt signal.

Connecting Directly

The most common scenario for interfacing a [Raspberry Pi](https://www.raspberrypi.org/) with an [Arduino](https://www.arduino.cc/) is to have the Raspberry Pi assume the role of Master.

In this configuration, it is possible to connect the Raspberry Pi and Arduino directly together, as the Master is determining the logic levels. However, you need to be careful when doing this.

The most important thing to pay attention to when interfacing 3.3-volt logic to 5-volt logic is the arrangement of the pull-up resistors. It is critical that in this arrangement the pull-ups are connected to the 3.3-volt reference. Any pull-up connected to 5-volts will raise the logic level, possibly destroying the 3.3-volt device(s).

The Raspberry Pi has internal pull-up resistors on the I2C lines, which pul the bus up to 3.3-volts. As long as you don’t connect any devices that pull the levels up to 5-volts you will be OK.

You should also note that Arduino has open-collector outputs. Because of this, the Arduino logic levels on its I2C bus will be set to the levels of the pull-ups, which in this arrangement are in the Raspberry Pi.

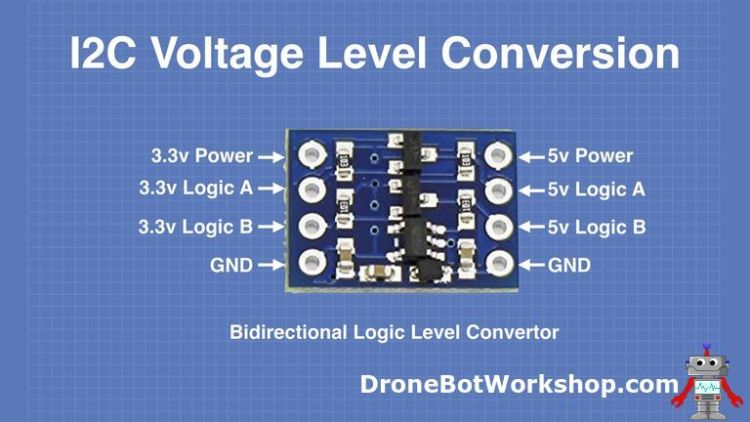
If you attempt to interface to other 5-volt I2C devices you need to ensure that none of them have pull-ups that would bring the logic levels up to 5-volts.

Of course, if you decide to use the Arduino as a Master then you won’t be able to connect it directly to a Raspberry Pi (unless you use a 3.3-volt Arduino). For that job, you’ll need to add one more component.

Bidirectional Logic Level Converters

A Logic Level Converter is exactly that, a device that accepts logic signals of one voltage level and converts them into signals of another logic voltage level.

A logic level converter can be used to resolve issues with using I2C devices that operate on different voltage levels. Since the SDA (data) line is used to both send and receive data you will need to use a bidirectional logic level converter.



These devices are very inexpensive and are available in many configurations, common configurations are 2, 4 or 8-channel devices.  Fo use with I2C a 2-channel device will suffice.

The bidirectional logic level converter is very easy to use. On one side you have the 3.3-volt logic signals, plus power and ground. The other side has equivalent connections using 5-volts.

Even is a scenario using a Raspberry Pi as Master it is still advantageous to use a bidirectional logic level converter.  It provides a more reliable signal on both sides, and it prevents disaster caused by accidentally connecting a 5-volt logic device that has internal pull-up resistors.

All you need to do is ensure that all 3.3-volt devices are kept on the 3.3-volt side and that 5-volt devices are connected to the 5-volt side.

Today we will look at both methods of connecting the Raspberry Pi to the Arduino via I2C.

Raspberry Pi I2C Setup

Using I2C with an Arduino is pretty simple. You just look at the specifications to see which two pins on your Arduino are used for SDA and SCL, hook them up accordingly and then use the [Wire Library](https://www.arduino.cc/en/reference/wire) (which is built into your Arduino IDE) in your sketch.

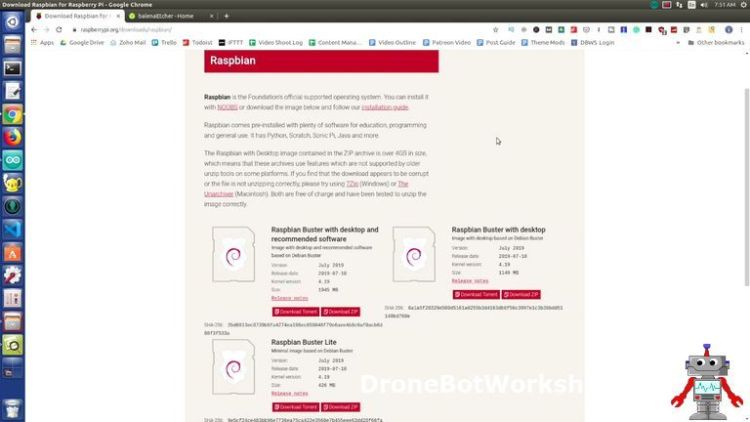
The Raspberry Pi requires a couple of extra steps, however. In fact, by default I2C is not even enabled.

Obtaining Raspbian

Unlike the Arduino, you need to install an operating system in order to start using your Raspberry Pi, which makes sense as the Raspberry Pi is essentially a small computer. The operating system, as well as all of your programs, are stored on a small MicroSD card.

While there are a number of operating systems that will work with the Raspberry Pi, the most common choice is Raspbian.  This is a Linux-based OS especially made for the Raspberry Pi.

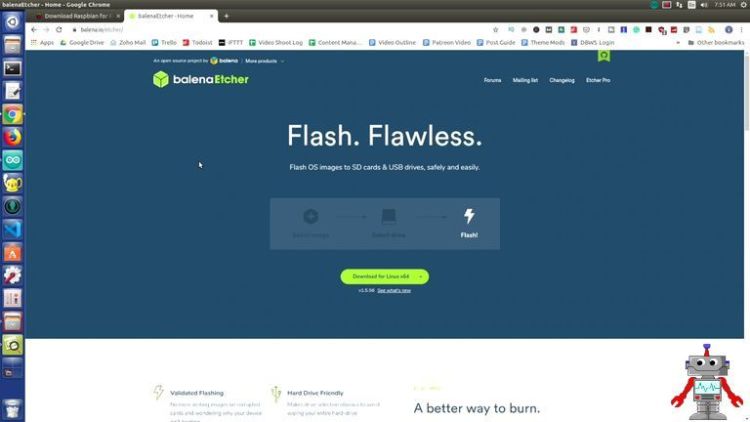
You’ll need to [download the latest version of Raspbian](https://www.raspberrypi.org/downloads/raspbian/) from the Raspberry Pi website. As of this writing that is Raspbian Buster, released in July 2019.



There are a few variations of Raspbian to choose from, I recommend that you grab one of the versions that include the desktop.

Once you have the ZIP file downloaded you’ll need to burn the image onto a blank MicroSD card. Make sure you choose a card that is at least 8GB, I recommend 16GB or more, and pick a brand-name card with decent transfer rate specifications. This is especially important if you are using one of the newer Raspberry Pi 4 models.

A great utility for burning MicroSD cards is [Etcher](https://www.balena.io/etcher/), it’s a free program that is available for Windows, Linux and Mac operating systems.



Once you have the MicroSD card burned, insert it into your Raspberry Pi, with the contacts facing towards the printed circuit board.  You’ll also need to hook up a USB keyboard, USB mouse and HDMI video monitor. Then power up the Raspberry Pi with a suitable power supply and watch the screen.

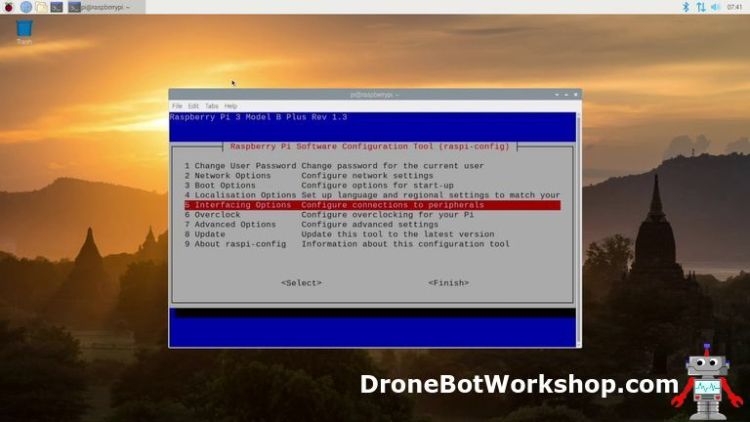
Enable I2C

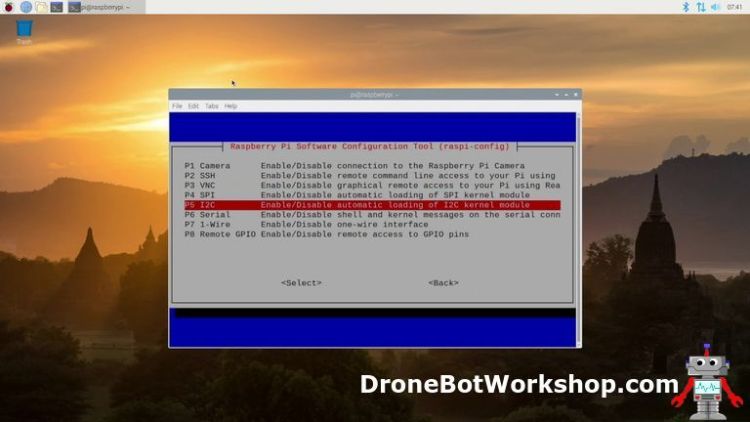
When you first boot up Raspbian it will partition the MicroSD card, which is it equivalent of a hard drive, and then restart. It will then take you to a Raspbian desktop, where you’ll be prompted to answer a number of questions about your locale, keyboard, and monitor.  You also can set up WiFi here, assuming you’re using a Raspberry Pi that supports WiFi.

The system will then reboot and you’ll arrive back on the Raspbian desktop.

Now that the operating system is installed you’ll need to configure Raspbian to enable the I2C functions.  You can do it as follows:

* Open a Terminal window, you’ll see an icon for this on the top menu bar.
* At the command prompt type *sudo raspi-config*. This will  bring up a configuration menu
* Use your arrow keys to navigate to *Interfacing Options*.
* Select *I2C.*You wil be asked if you wish to enable the I2C interface.
* Use your keyboard to move to the*Yes* key and press *Enter*.
* Use the Tab key to navigate to *Exit*and press *Enter*.







I2C should now be operational on your Raspberry Pi. To confirm this, type the following at the Terminal command prompt:

*ls /dev/\*i2c\**

This should bring back this response:

*/dev/i2c-1*

This indicates that the I2C-1 port is active. This is the port we will be using for our experiments.

Incidentally, there is also an I2C-0 port on the Raspberry Pi GPIO, however, it is only used to communicate with Raspberry Pi HATs (Hardware Attached on Top) and cannot be used to communicate with external devices.

Now that your Raspberry Pi is configured correctly we can begin our experiments.

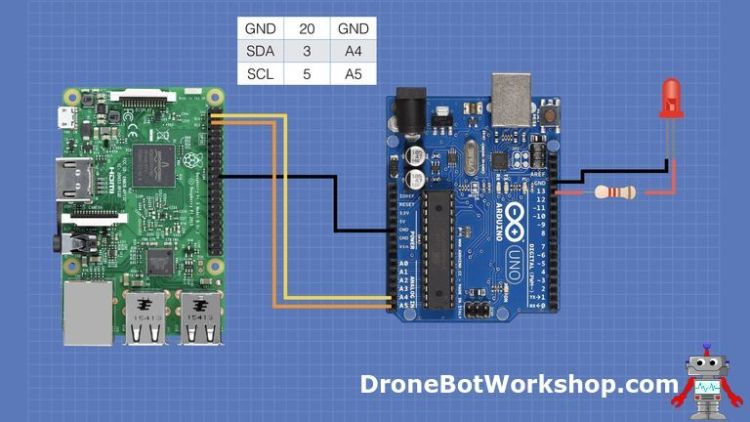
## Direct Connection

The first experiment we will perform is to hook up the Arduino and Raspberry Pi directly using the I2C bus.  Remember, this only works because the Raspberry Pi is master.

### Hookup Diagram – Direct Connection

The simple hookup for the Arduino and Raspberry Pi is shown below. Make sure you power down both units before doing the wiring.

Note that any model of Raspberry Pi will work. I’m using an Arduino Uno, you could also use a Nano or Pro Mini.  The Mega2560 has a different I2C arrangement, you can rewire for that if you wish but it is easier just to use an Uno.



Make sure that you connect the I2C bus up correctly. Although I used pin 20 on the Raspberry Pi GPIO you could use any of the ground pins. The Arduino also has multiple ground connections and any of these can be used as well.

The LED and dropping resistor are optional, as we are using pin 13 which has an onboard LED.  The external LED can be a bit easier to see.

Once you are all hooked up we can begin the test.

## Running the Test

Power up both the Raspberry Pi and the Arduino. You’ll need a keyboard, mouse, and monitor on the Raspberry Pi, but all you’ll need for the Arduino is a power source.  This can be a USB cable or 9-volt adapter.

On the Raspberry Pi desktop open a Terminal window.

The first thing you’ll need to do is to navigate to the folder that contains your Python program. I placed mine in the Documents folder, so at the command prompt I typed the following:

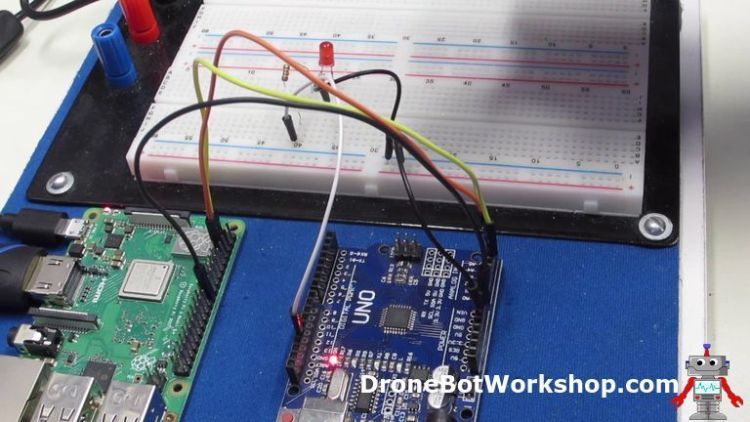
*cd /Documents*

If you elected to use a different folder substitute its name, be sure to observe capitalization as all Linux commands are case-sensitive. If you kept the file in your Home folder then there is no need to change directories as the Terminal opens up in your Home folder.

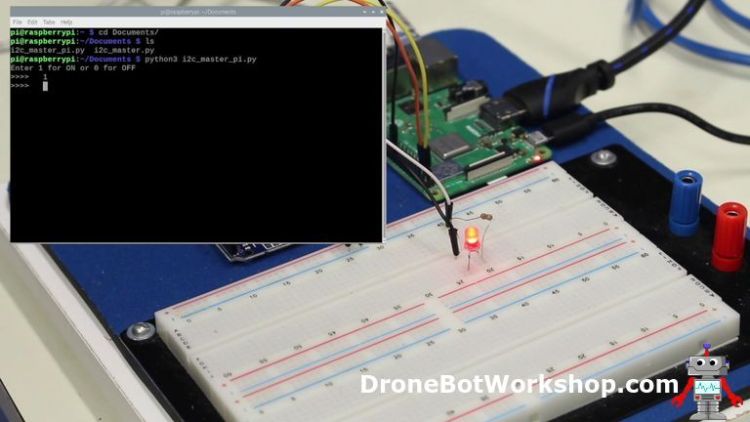
Now run the program by typing the following command:

*python3 i2c\_master\_pi.py*

Press *Enter* and the program should start running.  You’ll see the prompt “Enter 1 for ON or 0 for OFF”, followed by a series of greater-than symbols which act as a command-prompt within the program.



Pres the “1” key on your keyboard and then press Enter. You should see the LED on the Arduino illuminate.  Pressing the “0” key should extinguish the LED. you may continue to do this as much as you like. When you are finished type any other key and press *Enter*, and the program will end.



It is a very simple test, and obviously it has no real value – after all, you could just attach an LED directly to the Raspberry Pi GPIO if all you wanted to do was to control it using a Python program!

But the program does illustrate the process of sending commands via I2C to an external device.  And that’s all it needs to do.

## Connection with Converters

For our second experiment, we will use a bidirectional data level converter to connect the Raspberry Pi I2C bus to the Arduino.

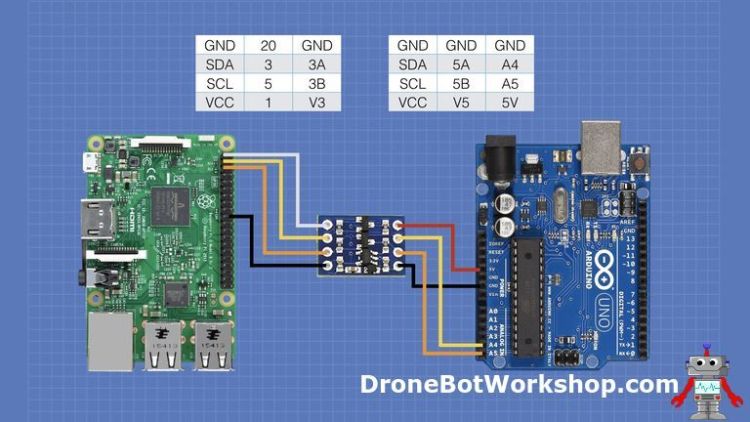
While this is a bit more complex it is a lot safer and more reliable over distance than the previous hookup. By keeping all of the 5-volt logic devices on one side of the converter and the 3.3-volt logic device on the other side you eliminate the possibility of damaging the 3.3-volt devices.

This is also the only method of setting up the 5-volt Arduino Uno as an I2C Master, however, we will be keeping the Arduino as the I2C Slave device.

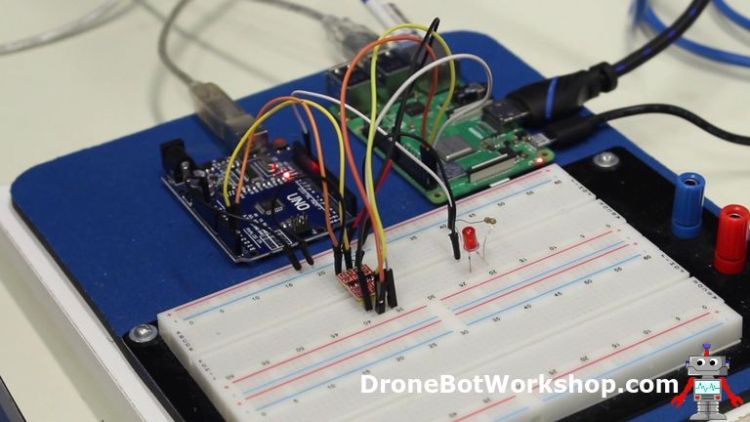
In fact, all we will do is to hook everything up and then run the same program we did earlier.

### Hookup Diagram – Using Logic Converter

Here is the hookup diagram for this experiment:



You will be powering the 3.3-volt side of the bidirectional data level converter using pin 1 on the Raspberry Pi GPIO.  The 5-volt power for the other side comes from the Arduino 5-volt output. Make sure that you power off both units when making the initial connections.



After you have everything hooked up the instructions are exactly the same as the previous experiment.  There should be no differences observed, you will once again be able to toggle the LED using the keyboard on the Raspberry Pi.