LED Exercise

Kyle McCarty and Marc Los Huertos[*]

October 20, 2020

Contents

[*]Acknowledgments: Summer research students, Anna Burns and Alision Joseph contributed to these documents.

1 Introduction

The purpose of this document is to give a brief introduction and familiarization of the components that you will be using in your air quality station. It will also summarily discuss electrical circuits and software coding with the programming language Python.

2 Components

2.1 General-Purpose Input/Output Pins (GPIO)

- The general-purpose input-output pins (GPIO) are one of the main features that makes the Raspberry Pis so versatile. The pin header on the newer Raspberry Pis have 40 pins in total. We use these pins and connect them to devices to control them. In software, the GPIO pins can be designated either as an input pin or output pin.

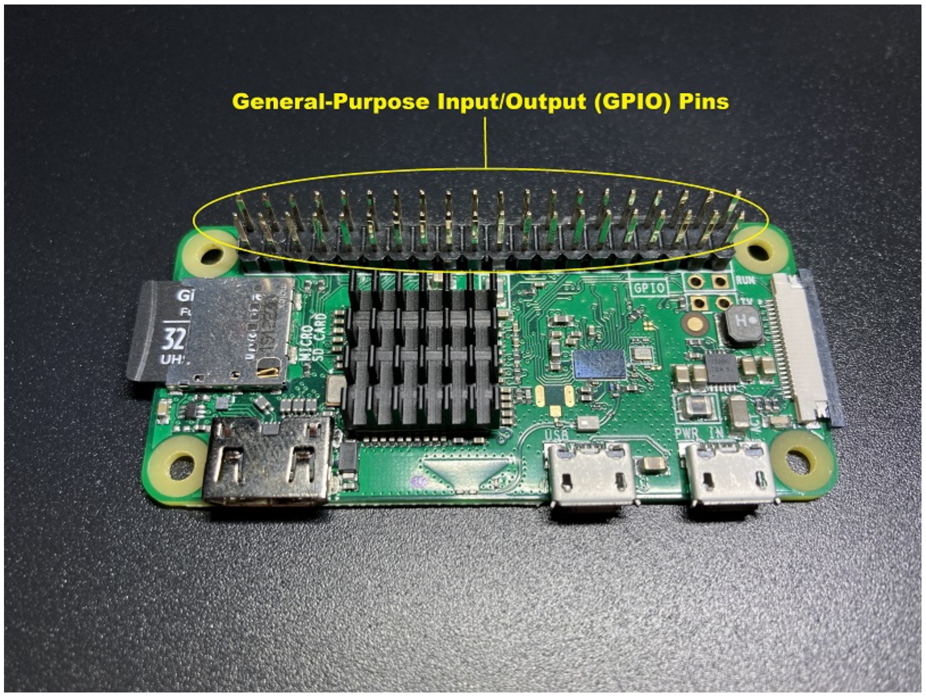

Figure 1: Raspberry Pi Zero W board with pin header soldered on and labled.

- There are, however, a few pins that are not configurable. These are two 5 volt (5V) pins, two 3.3 volt (3V3) pins, and eight ground (0V) pins.

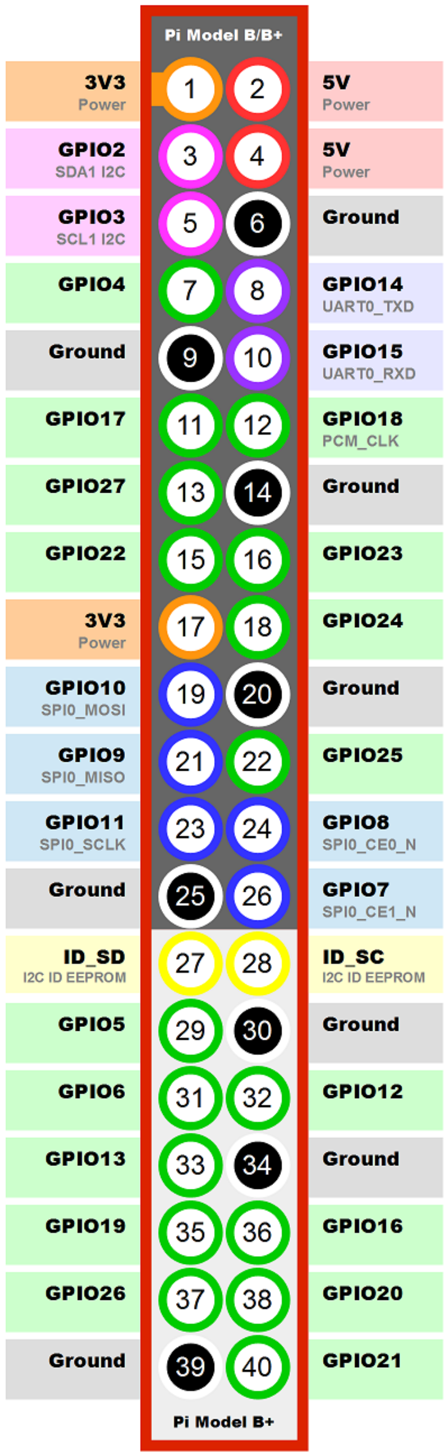

Figure 2: Diagram depicting the physical locations of each pin on the header, the software-assigned label, and function.

- All of the other remaining pins are considered general purpose 3V3 pins. This means you can configure these pins as long as the input and outputs are 3V3.

2.2 Jumper Wires

- A jumper wire is a wire that is used to connect various electronic components. For example, a jumper wire will connect our Raspberry Pi to the breadboard, various components on the breadboard, and various sensors. These are usually found in a variety of colors, which can be tough to keep track of. However, red is almost always for the positive voltage and black is used for ground. It's highly recommneded to follow this convention – could save much heartache.

- It is also important to note, wires come in various shapes and sizes with different end connections. For example, you were supplied with wire that has two bare ends and wire that has one female connector and one male connector. You were supplied these two different wires to use the bare wires on the breadboard and the female/male connectors for the GPIO pins on the Raspberry Pi to the breadboard.

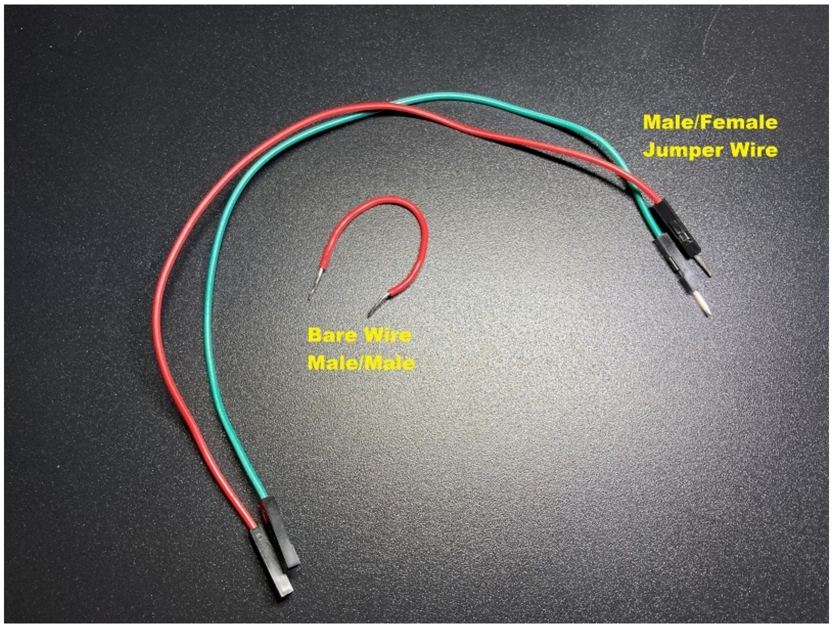

Figure 3: Examples of two types of wire. Bare, male/male wire and male/female jumper wire.

2.3 Breadboard

- Breadboards are testing platforms to build circuts. The have rows and columns which are electrically connected to give design flexibility.
- There are usually two or four rails or bus strips on the long sides of the board that are for power, in our case 3.3 volts or 5 volts, which is usually indicated by a red line. These buses are just a column or row, depending on breadboard orientation, of holes that are electrically connected.
- The ground, usually adjacent to the power bus, is usually blue or black colored.
- The rows in the inside are electrically connected, with a break in the center line. The center line is often strattled by a DIP (dual in-line package), so you can connect to each of the pins from both sides of the breadboard independently.

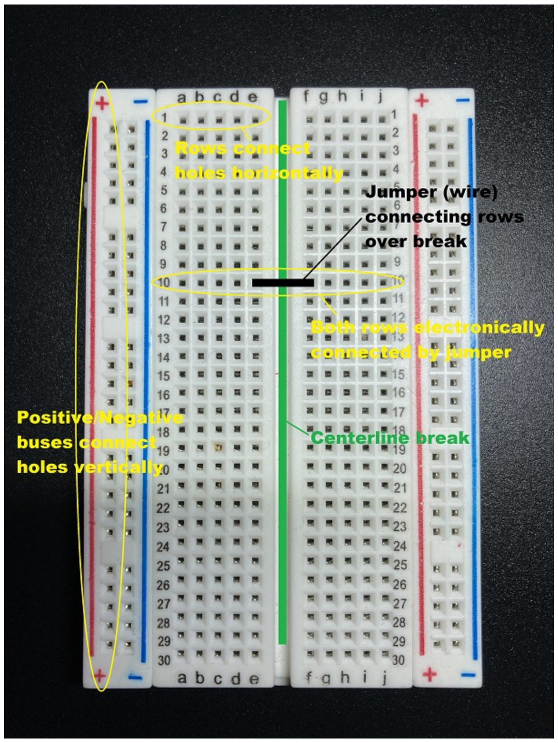
Rows connect holes horizontally
Jumper (wire) connecting rows over break
Both rows electronically connected by jumper
Positive/Negative buses connect holes vertically
Centerline break

Figure 4: Breadboard illustration showing +/- columns are electrically connected while inside rows, with number-letter coordination, are electrically connected albeit being segregated at the centerline.

2.4 LED

- As many of you may know, a light emitting diode (LED) is a diode that emits light when voltage is applied to it. They come in many different colors and sizes, but for this exercise we are using relatively small LEDs.
- LEDs actually have some requirements to work. We need to consider its polarity, forward voltage, and maximum current rating. Luckily, we have selected the right components for your already to use with the Raspberry Pi so you don't have to worry about that!
- We will be using the Raspberry Pi to supply power to the LED so it will turn on. You will also be wiring in a resistor to make sure you don't burn your Raspberry Pi up.

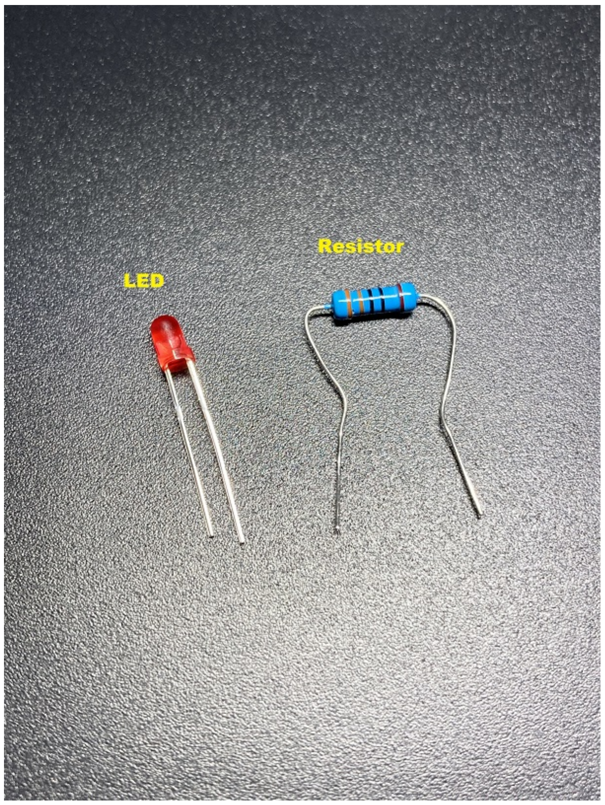

Figure 5: LED and resistor. The LED has two differently sized legs; forward voltage is applied to the LED's long leg. The colored bands on the resister are used to identify the amount of resistance, which is measured in ohms.

2.5 Resistor

- You must **always** use a resistor when wiring an LED to the Raspberry Pi. If you don't use one, you could potentially damage your Raspberry Pi.
- A resistor limits the amount of electricity, or more specifically, current through a circuit. Resistance is measured in Ohms, and the larger the value, the more it limits current to flow.

3 Wiring

3.1 Simple LED Circuit – LED Always On

We are going to create a simple circuit withe the LED using thing Raspberry Pi as a power source. This is really easy and will let you know if you understand how to wire a circuit. The LED should turn on as soon as you power on the Raspberry Pi!

1. First setup you power and ground jumper wires coming from your Raspberry Pi to your breadboard. Use your GPIO pinout diagram to figure out which pin is your 3.3 volt power and which pin(s) are ground. Connect your red wire's (color doesn't really matter but may help) female end to 3.3 volts power on the Raspberry Pi and the male end to the power/-postive/+ bus on the breadboard (Figure **??**). It doesn't really matter which spot on the bus you connect it to, the whole column is connected, remember? Do the same for the ground jumper wire.

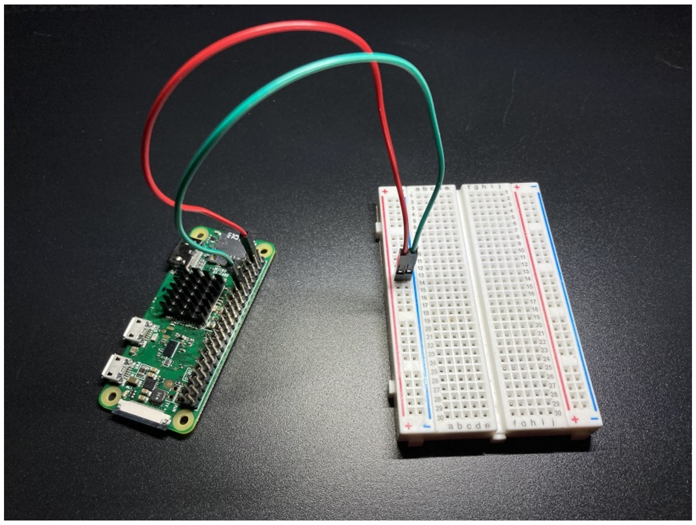

Figure 6: Power and ground connection to breadboard and Pi.]GPIOlayout

2. Next setup the resistor on the breadboard. The resistor should be setup connecting the ground bus on the breadboard to an inside row. Keep in consideration you'll need at least one hole for the LED leg to connect in the same row (Figure 7). Resistors are not directionally dependant, so it doesn't matter which end of it is connected to the ground bus and which end is connected to the inside row.

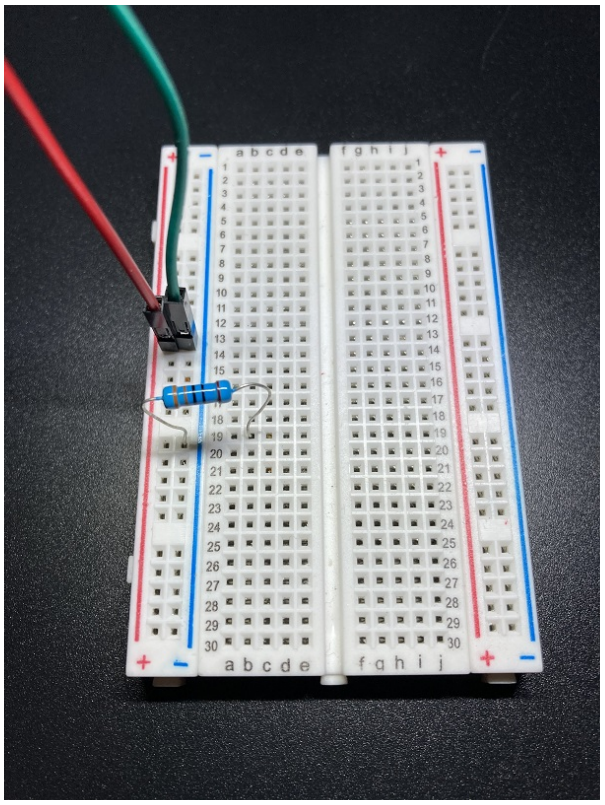

Figure 7: Resistor attached to breadboard.

3. Third, you need to connect the LED to the inside row the resistor is on and a **different** row. LEDs **are** directionally dependant. This is called polarity. If you notice, the LED has a short and long leg. Connect the shorter leg in the hole which is in the same row as the resistor. Connect the longer leg to a hole in another inside row (Figure 8).
4. Lastly, connect a male/male wire from the longer leg of the LED to the power bus.
5. Now, double check your wiring and make sure it's right! It should look something like Figure 9.

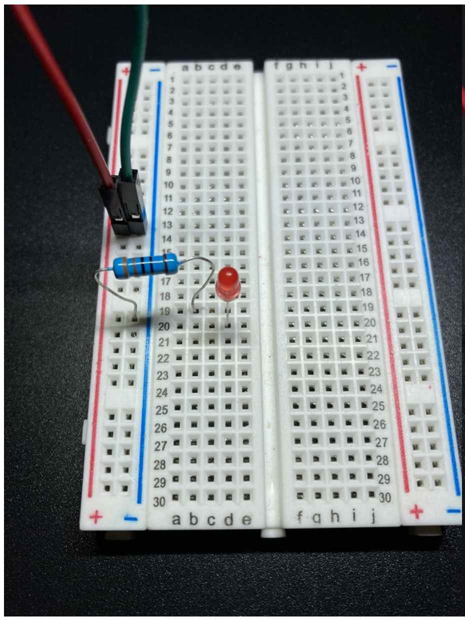

Figure 8: LED and resistor attached to breadboard.

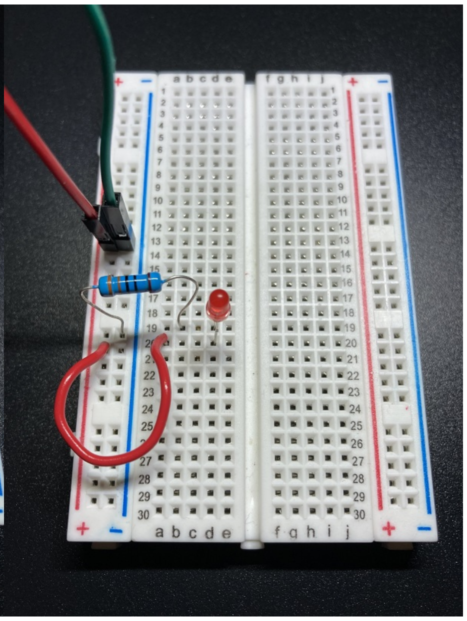

Figure 9: All components hooked up in a complete circuit.

6. Turn on your Raspberry Pi, and as soon as you do this, you should see your LED turn on!

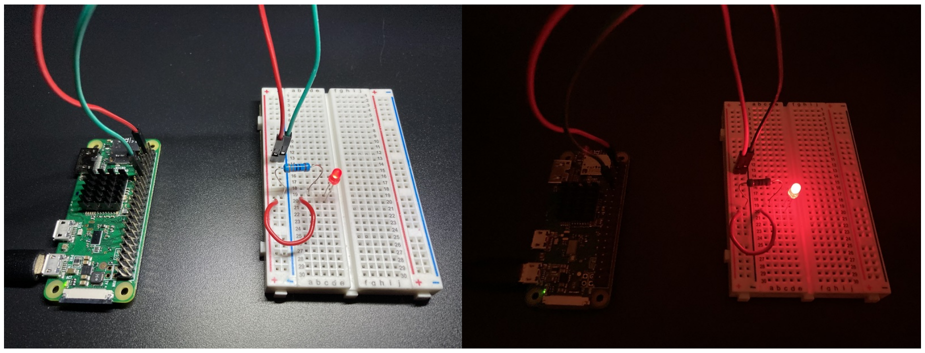

Figure 10: LED is on!

7. If your LED is not turning on, but you are sure you have your wiring correct, check the **troubleshooting** section below.

3.1.1 Simple LED Circuit Troubleshooting

Help! My LED won't turn on!

- Triple check your wiring. Make sure the electrical components are all seated well on, or in, their GPIO pins or breadboard holes.
- It is possible that you wired your LED in reverse (remember polarity). Try switching the leg positions.
- Check your ground wire and its corresponding pin on the Raspberry Pi. Rarely a ground pin will not work. There are several ground pins on the Raspberry Pi, so try connecting the ground wire to a different ground GPIO pin.
- Try switching out the LED. It is possible the LED is broken, this is why we gave you two LEDs.

3.2 GPIO LED Circuit

Next, we are going to change the wiring of the circuit to add a layer of complexity to this exercise. We are going to execute some Python code so that that LED blinks on and off.

- The first step to this is the use a GPIO pin, that isn't direct power, to send a signal that will turn on the LED and then another signal that will turn it off.
- The Raspberry Pi communicates these signals in 2-logic levels, logical high signals or logical low signals. In this case, the Raspberry Pi will send a 3.3 signal, logic level high, or a 0V signal, logic level low. Obviously, 3.3 volts will turn on the LED and 0V will turn it off (no power supplied anymore).
- The wiring essentially stays the same for this setup except that instead of connecting power to the constant 3.3 volt GPIO pin, connect it to **GPIO pin 24.** Refer to your GPIO pin diagram 2.

BE CAREFUL there are two different naming conventions for the GPIO pins. The numbers within the circle on your pinout diagram (1-40) are the **physical** pin locations. The labels on the outside are the **BCM**, or **manufacture**'s designated nomenclature, GPIO 24 etc. You'll run into this issue when trying to code in Python.

4 Flashing LED

4.1 Using code to control the LED

In order to instruct the Raspberry Pi to tell the LED to do more complicated things like turning off and on, we have to communicate with it in the form of a programming language and we'll use Python.

4.2 What is Python?

Python is a general-purpose programming language which its design philosohpy emphasizes readability of code.

In order to communicate instructions via Python, we need to write the code. We do this simply by using a text editor, or some kind of integrated development environment (IDE) like **Thonny**, for example. Thonny is the IDE that comes with Raspberry Pi OS.

A few notes on Python code:

- All the Python code you will be generating will have the file extension of **".py"**.
- Pay very close attention to spacing and indentations! This is very, very important in Python. Python uses particular formatting that represent "code blocks." The code will not run if the indentations are at all off.

4.3 Make your LED flash

1. First, make sure you have connected your LED to the correct GPIO pin as decribed in **3.2**. NOTE: This is not the 24th PIN – look carefully that you are using PIN 18.
2. Boot up you Raspberry Pi and open the IDE Thonny.
3. Once you are in Thonny, write the following code:

```
import RPi.GPIO as GPIO
from time import sleep

GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
GPIO.setup(24,GPIO.OUT, initial=GPIO.LOW)

while True:
   print("LED on")
   GPIO.output(24,GPIO.HIGH)
   sleep(1)
   print("LED off")
   GPIO.output(24,GPIO.LOW)
   sleep(1)
```

   - Keep in mind all of the syntax when you are writing the code. Python is sensitive to capitals, parenthesis, tabs, quotes, etc.

4. While writing the code, using Thonny, it will turn certain words into different colors or turn them bold, etc. This will help you recognize keywords of Python as well as syntax errors. For example, when you don't close your

parenthesis, you will see the text highlighted gray. This is an advantage of writing code in an IDE.

5 Testing Your Code

1. Next, we want to run the script! Press the big green **"Run"** button in the top toolbar of Thonny. Thonny may ask you to save the file. Go ahead and save it as something like **"ledflash.py"**.

2. If you see your LED flashing on and off, congratulations! You may stop the script anytime by pressing the big red **"Stop"** button in the top toolbar of Thonny.

5.1 Debugging

1. If your light isn't flashing, you may have written the code incorrectly. Thonny is smart enough to read your code and point you in the direction of a suspected error. In the **"Shell"** tab you will see some red lettering that will describe where the error occured.

2. Don't forget to **"Stop"** the script if you need to make changes. If you don't stop the script, it won't allow you to edit.