Word count: 1066

Video, Introduction to Hardware Interfacing on the Raspberry Pi 4

In this video, we will look at the essential tools to start with hardware interfacing, later we will learn about the GPIO Zero Library, and finally, we interface an LED with the Pi and work with it.

Before you start your Pi hardware journey, you will need some essential tools, other than sensors and actuators

.

During the learning phase, you would want to change circuits easily for different projects. Thus a breadboard is an essential tool, which makes circuit dismantling and reconfiguring a breeze. A breadboard cannot obviously be connected to the Raspberry Pi 4 without wires. Thus we need a lot of jumper wires. The next important tool to have is a multimeter. A typical inexpensive multimeter will feature a digital display and a rotary selector switch for its various voltage, current, and resistance ranges. Once you finish this course, you will be off to building your own projects. At that time, you wouldn’t want to use the breadboard still. You will need a more permanent solution. So it’s better to buy a soldering iron kit. If you don’t know how to work with breadboards, multimeters, or soldering iron, please check out the links in the resources and learn.

Now let’s look at some of the software essentials before moving on. GPIO Zero is a friendly python library for physical computing. GPIO Zero started out as a friendly API on top of the RPI.GPIO, RPIO, and pi GPIO library, but later was extended to allow other pin libraries to be used. GPIO Zero comes preloaded with support for many common sensors and actuators. Thus this GPIO library is very user-friendly. This library uses Broadcom pin numbering for the GPIO pins, as opposed to physical numbering. For those who don’t know about the RPI.GPIO library, please check out the links in the resources, to know the importance & impact it had on the Pi community. GPIO Zero is shipped as part of the default Raspbian distribution. Its simple interface takes you straight to the hardware with a minimum of other code required, and it has built-in support for a huge array of sensors and output devices.

The first step is to make sure that your GPIO Zero library is up to date. Go to the terminal and type

sudo apt update and then

sudo apt install python3-gpiozero

So shall we start off with controlling an LED Project on the Pi. In the simplest terms, a light-emitting diode is a semiconductor device that emits light when an electric current is passed through it. Light is produced when the particles that carry the current known as electrons and holes combine together within the semiconductor material.

Inside the semiconductor material of the LED, the electrons and holes are contained within energy bands. The separation of the bands determines the energy of the photons, which determines the wavelength of the emitted light, and hence its color. Different semiconductor materials with different band gaps produce different colors of light. Now we will use a Red LED.

If you look at an LED, you can see two legs. The longer one is called anode or the positive leg, and the shorter one is called cathode, which is the negative leg. See, LEDs are delicate little things. If you force too much current through them, they will sometimes pop quite spectacularly.

To limit the current going through the LED, you should always use a resistor in series with it. This type of resistor is called as a current limiting resistor. The choice for the current limiting resistor also is determined by the forward voltage of the LED, the supply voltage, and the forward current, which is dependent on the color of the LED. Thus by Ohm’s law, with a supply voltage of 3.3 V, a forward current of 20mA & forward voltage of 2 V, any current limiting resistance above 65 ohms is fine. If you want to know more about current limiting resistors and how to choose them, please check out the resource section.

First, connect and fix the LED on the breadboard. Now connect the 180 Ohm resistor in series with the anode of the LED. Connect a jumper wire from the other leg of the resistor to a 3.3 Volts pin on the Board. Finally, connect a jumper wire from the cathode leg to a GND pin on the board. The LED should light up. It will always be on, because it’s connected to a 3V3 pin, which is itself always on. Now move it from 3V3 to GPIO pin 17. The LED should now turn off, but now it’s on a GPIO pin, and can, therefore, be controlled by code.

Now open Thonny Python IDE and type the following in the shell:

from gpiozero import LED

First, we have to import the gpiozero library. As we need only the LED class from the gpiozero library,

next type LED = LED(17)

The parameter “17” inside the LED class tells the Pi which GPIO pin we are using. In this case its 17. By assigning the class to a variable named LED, you have created an instance of this class, also called an object. Now to make the LED turn on, you just need to call the “on” function inside the LED class, which is now accessed via the object “LED”. Thus entering “LED.on()” and pressing enter will light up the LED. Similarly, on entering “LED.off()” it will switch off. But that’s not much fun, right?

With the help of the time library and a little loop, you can make the LED flash. Download the LED\_Blink.py from the resources and open it in the Thonny IDE. You can see that we have imported a new class from the time library called sleep. Also, we have made an infinite loop with the “while True” instruction, inside which we have used the sleep method to make the LED turn on and off at 1-second intervals. Now run the script, and you will see the LED Blinking.

Summary

In this video, we have covered the following

● Essential tools for hardware interfacing

● The GPIO Zero Library

● Interface and Working with an LED using Python

In the next video, we will learn and work with PWM in LEDs and working with Buttons.