01 Blink

Nicholas Bruzzese

November 12, 2024

To calculate the resistor value, we need to examine the specifications of the LED. Specifically, we need to find the forward voltage (V_F) and the forward current (I_F). A regular red LED has a forward voltage (V_F) of 1.7V and forward current of 20mA (I_F). Additionally, we need to know the output voltage of the Raspberry Pi, which is 3.3V.

We can then calculate the resistor size needed to limit the current to the LED's maximum forward current (I_F) using Ohm's law like this:

$$R_\Omega = \frac{V}{I} = \frac{3.3 - V_F}{I_F} = \frac{3.3 - 1.7}{20\,\text{mA}} = 80\,\Omega \tag{1}$$

Unfortunately, 80 ohm is not a standard size of a resistor. To solve this we can either combine multiple resistors or round up to a standard size. In this case, we would round up to 100 ohm.

Important information: Since Ohm's law tells us that I (current) = V (voltage) / R (ohm), rounding up the resistor value will lower the actual current being drawn slightly. This is good because we don't want to run our system at the maximum current rating. Rounding down instead of up would be dangerous, as it would increase the current being drawn. Increased current would result in running our system over the maximum rating and potentially damaging components.

With the value calculated for the current limiting resistor, we can now hook the LED and resistor up to GPIO pin 8 on the Raspberry Pi. The resistor and LED need to be in series like the diagram below.

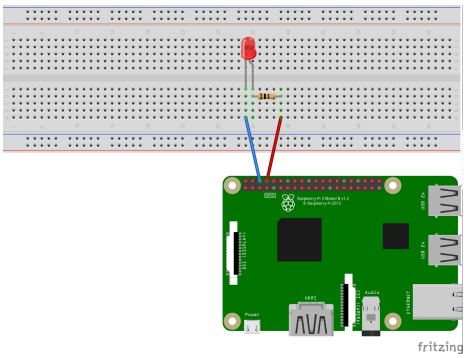

Figure 1: Wiring Diagram

When hooking up the circuit, note the polarity of the LED. You will notice that the LED has a long and short lead. The long lead is the positive side, also called the anode; the short lead is the negative side, called the cathode. The long lead should be connected to the resistor, and the short lead should be connected to ground via the blue jumper wire and pin 6 on the Raspberry Pi, as shown in the diagram.

To find the pin number, refer to this diagram showing the physical pin numbers on the Raspberry Pi.

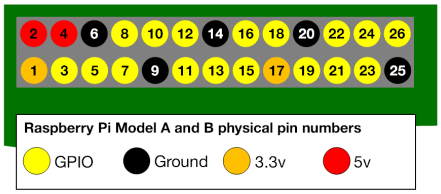

Figure 2: Raspberry Pi Pin Layout

With the library installed, now open the Thonny Python IDE. Our script needs to do the following:

- Initialize the GPIO ports
- Turn the LED on and off in 1-second intervals

To initialize the GPIO ports on the Raspberry Pi we need to first import the Python library, the initialize the library and setup pin 8 as an output pin.

```
# Import Raspberry Pi GPIO library
import RPi.GPIO as GPIO
# Import the sleep function from the time module
from time import sleep
```

```
# Ignore warning for now
GPIO.setwarnings(False)
# Use physical pin numbering
GPIO.setmode(GPIO.BOARD)

# Set pin 8 to be an output pin and set initial value to low (off)
GPIO.setup(8, GPIO.OUT, initial=GPIO.LOW)
```

Next we need to turn the LED on and off in 1 second intervals by setting the output pin to either high (on) or low (off). We do this inside a infinite loop so our program keep executing until we manually stop it.

```
# Run forever
while True:
# Turn on
GPIO.output(8, GPIO.HIGH)
# Sleep for 1 second
sleep(1)
# Turn off
GPIO.output(8, GPIO.LOW)
# Sleep for 1 second
sleep(1)
```

With our program finished, save it as blinkingled.py and run it inside the Thonny IDE.