Sensing and Actuation Networks and Systems [2022-2023]

Assignment 03 – Creating your first IoT program

Introduction

This assignment aims to create a basic Internet of Things (IoT) program that flashes a LED. During this assignment, students will get familiarised with a breadboard and will be able to analyse and create simple IoT programs to manipulate electronic components using a Raspberry Pi and a breadboard.

Objectives

Students successfully concluding this work should be able to:

- Understand the use of a breadboard
- Understand the concept of daemons
- Practice with a breadboard by creating a simple circuit

Support Material

- Example code
- Raspberry Pi 2 Model B with net cable (for use during PL)
- Breadboard
- LED light
- Jumper cables
- Resistors
- T extension board and serial cable

Exercises

1. Connecting the electronic components to the breadboard

Goal: Learn how to use the breadboard to connect extra devices to the Raspberry Pi.

Activities: In this activity you will create the following circuit:

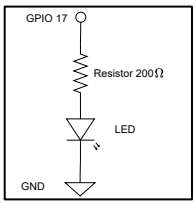

Figure 1. Schematic diagram of the circuit

You will need the following components:

- 1 x 5 mm LED
- 1 x 220 Ω resistor: colour bands red, red, brown, and gold
- Male-to-male jumper cables
- A breadboard/protoboard
- A 40 pin (serial) cable
- T extension board

Turn off your Raspberry-Pi before starting this exercise. **It will only be turned on after the circuit is completed**.

Plug the T extension board to the breadboard and to the GPIO interface of your Raspberry Pi, as depicted in Figure 2.

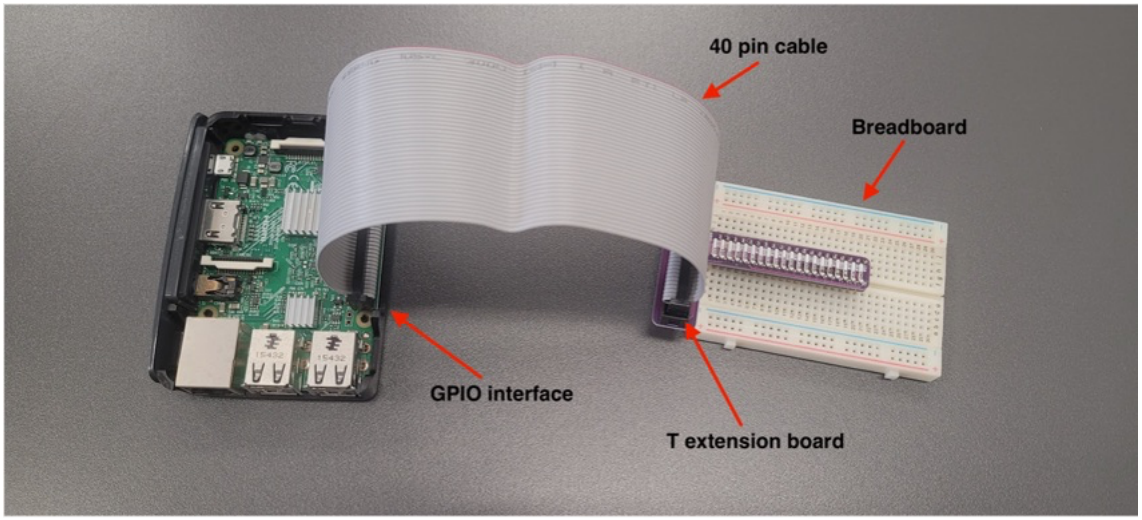

Figure 2. Connection between breadboard and Raspberry Pi via GPIO

Connect the LED to your breadboard vertically (i.e., two legs on the same column). Notice the row you are using for the cathode and anode. See Figure 3 for an example. In the example, the anode (longer leg) is connected on **b29** and the cathode (shorter leg) on **b30**. **DO NOT USE THE**

PORTS USED BY THE T EXTENSION BOARD! Those will be reserved to interact your Raspberry Pi via GPIO.

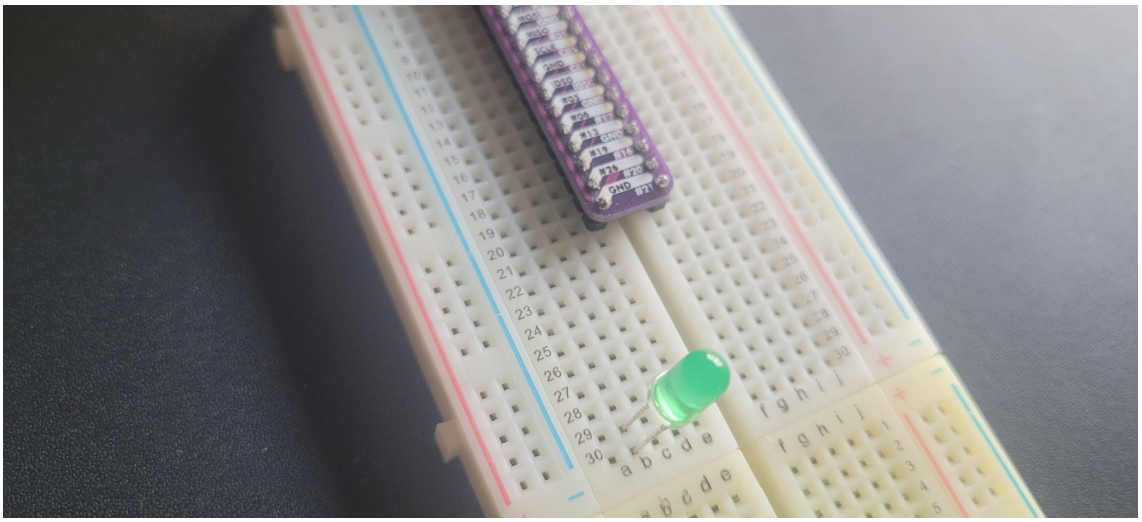

Figure 3. Plugging the LED to the breadboard

Now, we will plug the resistor. We need to match the same row than the anode of the LED (positive), and the other end should be on the same column. An example is provided in Figure 4. In the example, the resistor is plugged in **c29** and **c23**.

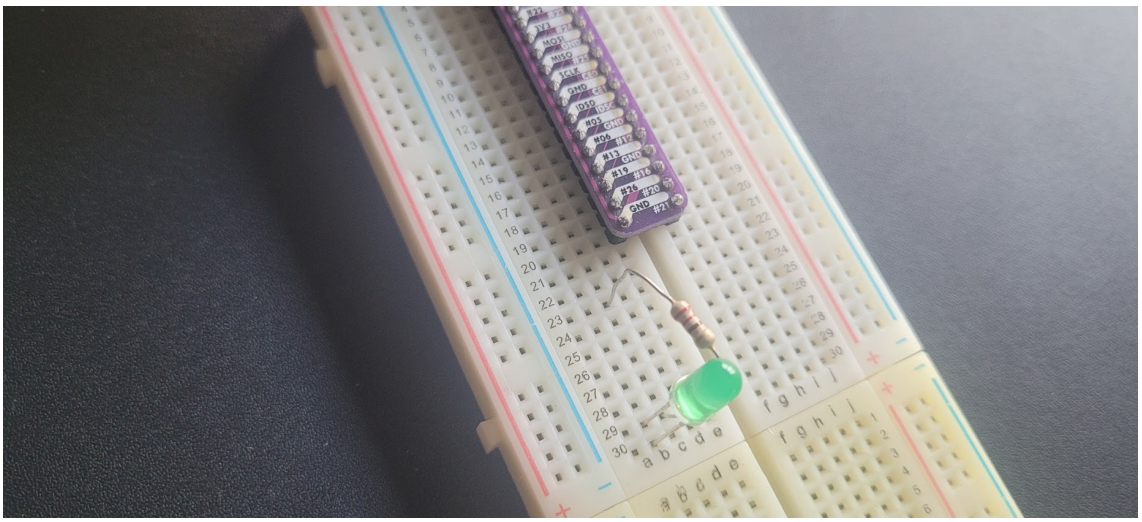

Figure 4. Connecting your resistor to the circuit

We will use two jumper cables at this point. One should come from a GROUND (GND) pin from your GPIO and to the negative column of the same bank. The second jumper cable will connect to the same row as the cathode (negative) of your LED (**b30** in the example) and to the negative column of your breadboard. See an example in Figure 5.

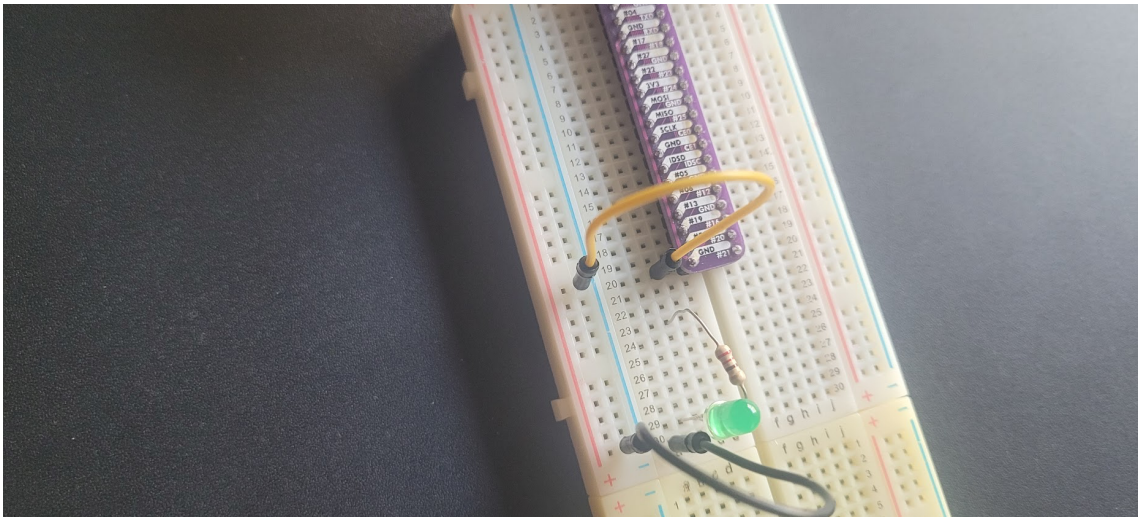

Figure 5. Connecting the jumper cables to your circuit

On the same row as the resistor (e.g., **d23**), we will plug a final jumper cable. The other end of the cable will be connected to the port number **17** of your GPIO interface. An example is provided in Figure 6.

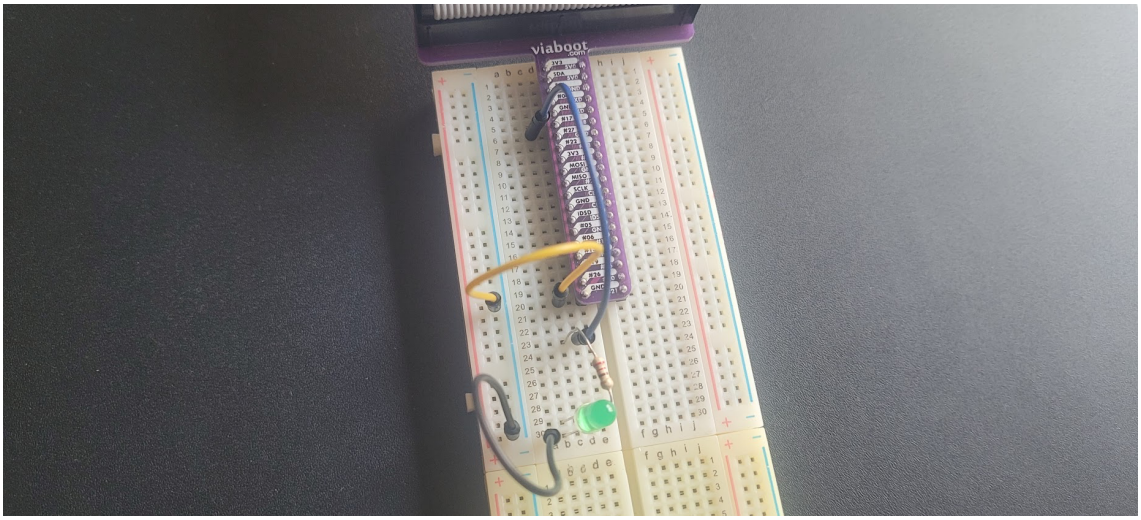

Figure 6. Plugging a final jumper cable to turn on/off your LED

After the circuit is completed **and verified**, you can turn on your Raspberry Pi.

DISCLAIMER: It is a good practice to leave the connection to the energy source to the final part of your circuit montage.

2. Flashing a LED using Python and GPIO

Goal: Set up the environment and use the GPIOZero library to make a LED flash using Python.

Activities:

Connect to the Raspberry Pi using the SSH protocol. Inside your directory, create a *virtual environment* and activate it. If the *virtual environment* is already created, you only need to

activate it. The package `pigpio` is needed for this assignment and should be installed. See slides from this assignment for details.

We also need to start the PiGPIO daemon, necessary to use the PiGPIO GPIO client library. To start it, use the following command:

```
sudo pigpiod
```

`pigpiod` is a utility which launches the `pigpio` library as a daemon. Once launched, the `pigpio` library runs in the background accepting commands from the pipe and socket interfaces.

Take a look at the example code `led_gpiozero.py` given with this assignment (can be downloaded from *UCStudent*). Analyse the code and the use of the GPIO Zero library. If needed, modify the code according to your circuit (in this assignment GPIO_PIN = **17** was used).

Transfer the `led_gpiozero.py` file to your Raspberry Pi directory, using SCP.

Now you can execute the example code in your *virtual environment* using the following command:

```
python led_gpiozero.py
```

If the LED is connected correctly, it should blink each second. Identify the instruction(s) that make the LED blink (including the correct connection with the GPIO interface).

3. Flashing a LED remotely using Python sockets and GPIO

Goal: Use the GPIOZero library to make a LED flash when receiving a message from a UDP socket in Python.

Activities:

For this activity two python files are given: `server_udp_BLINK.py` and `client_udp_BLINK.py`. Analyse the code in both files and the use of the GPIO Zero library. If needed, modify the code according to your circuit (in this assignment GPIO_PIN = **17** was used). Complete the code given in file `server_udp_BLINK.py` such that when the server receives the message '*BLINK*' from the client, it blinks the LED. The code in `client_udp_BLINK.py` does not need any change.

After all the changes are made, transfer the file `server_udp_BLINK.py` to your Raspberry Pi directory, using SCP.

In this activity the server code (`server_udp_BLINK.py`) will be executed on your Raspberry Pi and the client code (`client_udp_BLINK.py`) will be executed on your PC.

On the Raspberry Pi:

```
python server_udp_BLINK.py <Raspberry_Pi_address> <port>
```

On your PC:

```
python client_udp_BLINK.py <Raspberry_Pi_address> <port> BLINK
```

4. Adding complexity to the IoT program

Goal: Use the GPIO Zero library to control the LED when receiving a message from a UDP socket in Python.

Activities: Modify the code from the previous exercise, such that the server can receive three commands:

1. ON: turns ON the LED;
2. OFF: turns OFF the LED;
3. BLINK:n:i: makes the LED BLINK 'n' times, turning it ON/OFF for 'i' seconds.

On the Raspberry Pi:

```
python server_udp_BLINK.py <Raspberry_Pi_address> <port>
```

On your PC:

```
python client_udp_BLINK.py <Raspberry_Pi_address> <port> ON
```

```
python client_udp_BLINK.py <Raspberry_Pi_address> <port> OFF
```

```
python client_udp_BLINK.py <Raspberry_Pi_addr.> <port> BLINK:n:i
```

References

Smart, G. (2020). *Practical Python Programming for IoT: Build advanced IoT projects using a Raspberry Pi 4, MQTT, RESTful APIs, WebSockets, and Python 3.* Packt Publishing.

Stevens, R., Fenner, B., & Rudoff, A. M. (2004). *UNIX Network Programming Volume I: The Sockets Networking API.* . Addison Wesley Professional.