Sensing and Actuation Networks and Systems [2021-2022]

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

Theoretical background

This section covers some concepts that will be used during this assignment.

Breadboard

An electronic breadboard (or protoboard), as illustrated in Figure 1, is a prototyping board that helps to electrically connect components and wires quickly and easily (Smart, 2020).

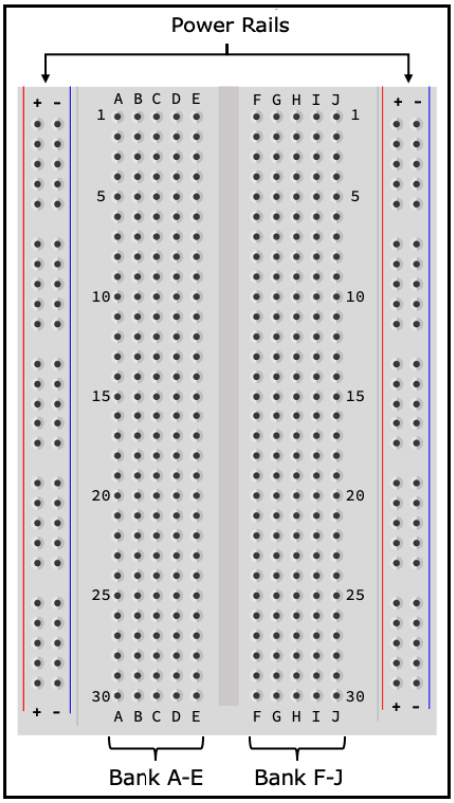

Figure 1. Breadboard example **(Smart, 2020)**

The *holes* in the breadboard are where electrical components and wires are placed to electrically connect them. The holes are electrically connected in the following ways:

- The two outer columns of holes are commonly referred to as *power rails*. There is a positive (+) column and a negative (-) column on either side of the breadboard. Each column of holes is electrically connected and run for the full length of the breadboard. Hence, there are four independent power rails on this breadboard: a + and - rail on the left-hand side of the breadboard and a + and - rail on the right-hand side.

  The power rails are frequently used to help to distribute power around the breadboard to components. Please note that **they do not provide power themselves**. They need a power source such as a power supply or battery connected to them to provide power.

- The centre of the breadboard has two banks of *holes*, which have been labelled *Bank A-E* and *Bank F-J*. Each row of *holes* in a bank is electrically connected. For example, *holes A1* through to *E1* are electrically connected, as are holes *F1* through to *J1*. However, *A1-E1* are not electrically connected to *F1-J1* because they are on a separate bank.

Led light

A LED (Light-Emitting Diode) is a small, yet bright, light made of a tiny crystal that emits a colour when electricity is connected to it. A typical LED is shown in Figure 2. The left-hand side of the diagram shows a physical representation of a LED, while the right-hand side shows the schematic symbol for a LED:

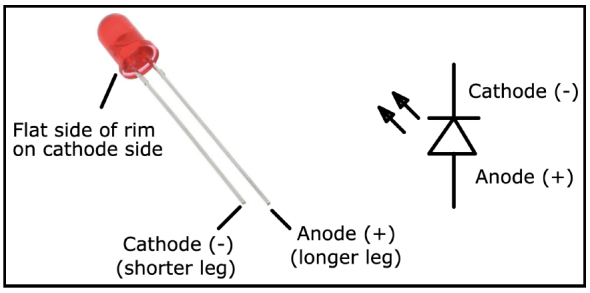

Figure 2. LED and schematic symbol **(Smart, 2020)**

LEDs need to be connected the correct way around into a circuit, otherwise, they will not work. When looking closely at a LED, it is possible to notice a flat side on the LED casing. The leg on this side is the **cathode**, which connects to the **negative** or ground side of a power source. The cathode leg will also be the shorter of the LED's legs. The other leg is known as the **anode** and connects to the **positive** side of a power source.

Sending files to the Raspberry Pi using SCP

To send files to the Raspberry, SCP (Secure CoPy) can be used. SCP is a command-line utility that allows to securely copy files and directories between two locations.

From a Windows PC

In Windows use the app *WinSCP*[1] to transfer the files to your Raspberry Pi. Fill the *hostname* field with the IP address or DNS name of your Raspberry Pi. Fill the user name and password fields as shown in Figure 3.

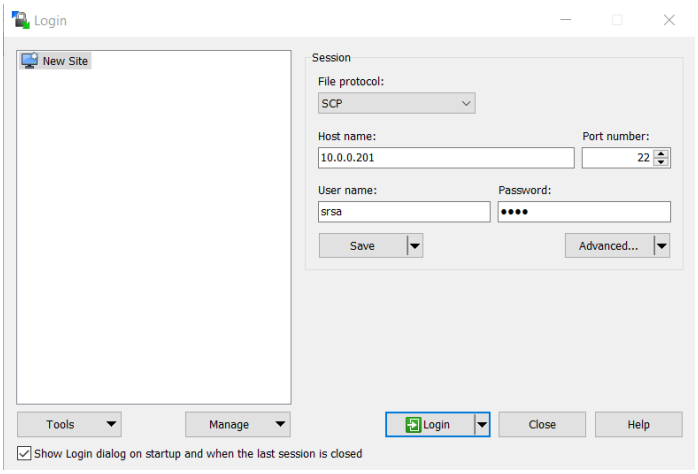

Figure 3. Login screen for WinSCP

[1] https://winscp.net/eng/download.php

Select the *local* directory on the left, and the *remote* directory on the right. Then, drag and drop the files from the local to the remote directory. Then, confirm the transfer as shown in Figure 4.

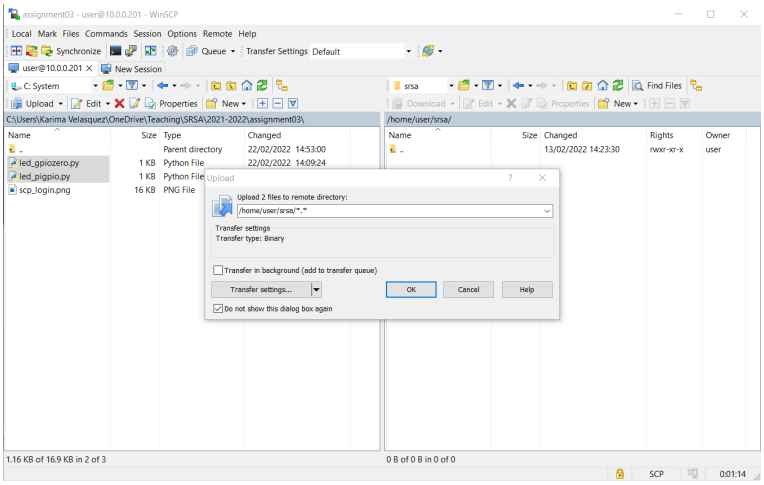

Figure 4. Transfer screen for WinSCP

From a Linux-based PC

Use the following command:

```
scp <file.py> <srsa@remote_address:/remote/directory/>
```

Daemons

A daemon is a process that runs in the background and is not associated with a terminal (Stevens, Fenner, & Rudoff, 2004). Operating systems generally have many daemon processes, running, performing various administrative tasks. There are several ways to start a daemon:

1. Using *startup scripts* during startup;
2. Using a "superserver" such as `inetd`, or by another daemon such as `cron`[2];
3. Using a user terminal, either in the background or not[3].

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:

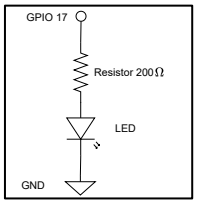

Figure 5. Schematic diagram of the circuit

[2] `inetd` and `cron` are out of the scope of this assignment

[3] This is common in daemon testing scenarios, or when restarting a daemon that was terminated for some reason

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 6.

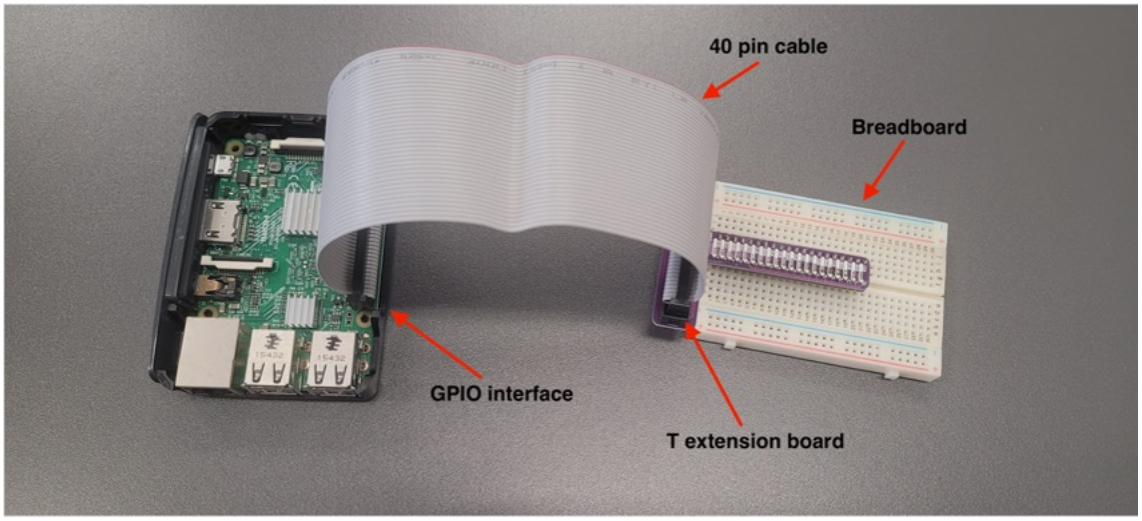

Figure 6. Connection between breadboard and Raspberry Pi via GPIO

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 7 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.**

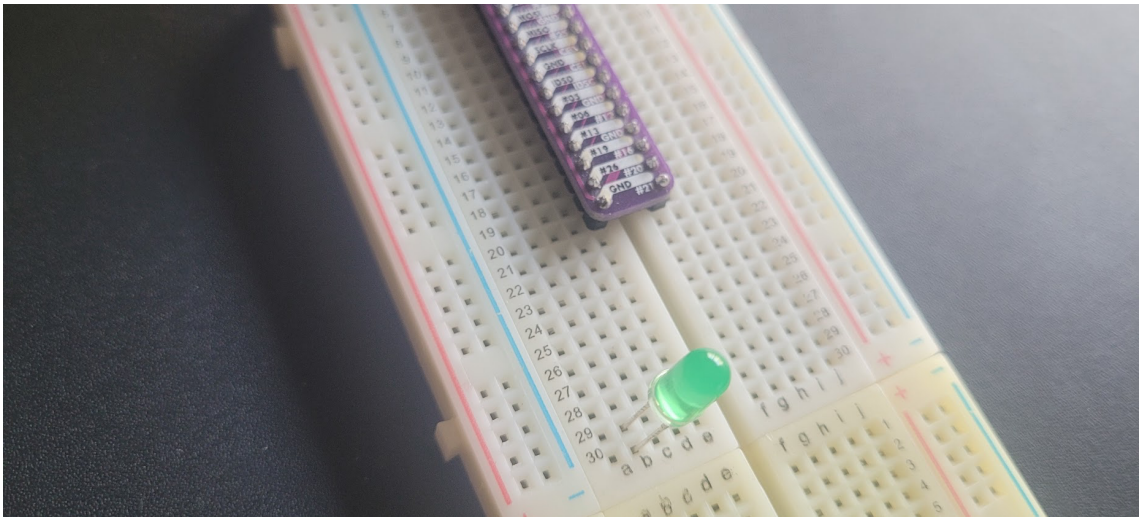

Figure 7. Plugging the LED to the breadboard

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 8. In the example, the resistor is plugged in **c29** and **c23**.

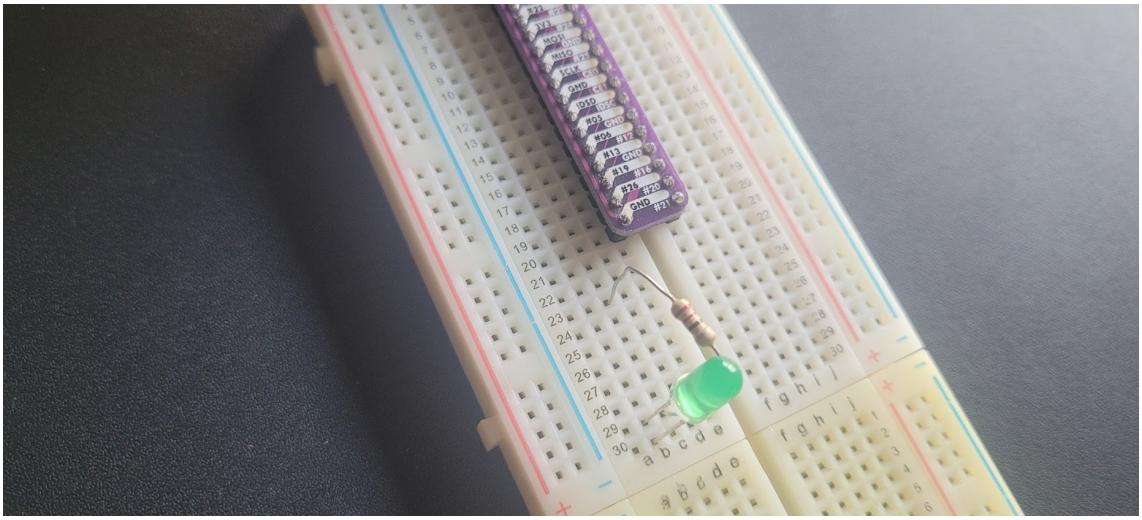

Figure 8. Connecting your resistor to the circuit

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 (**a30** in the example) and to the negative column of your breadboard. See an example in Figure 9.

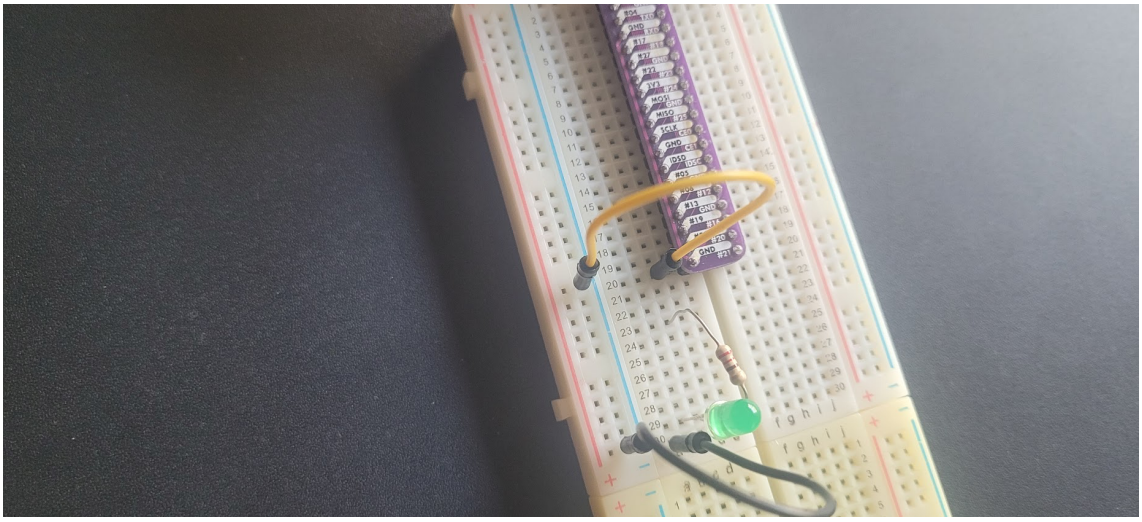

Figure 9. Connecting the jumper cables to your circuit

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 10.

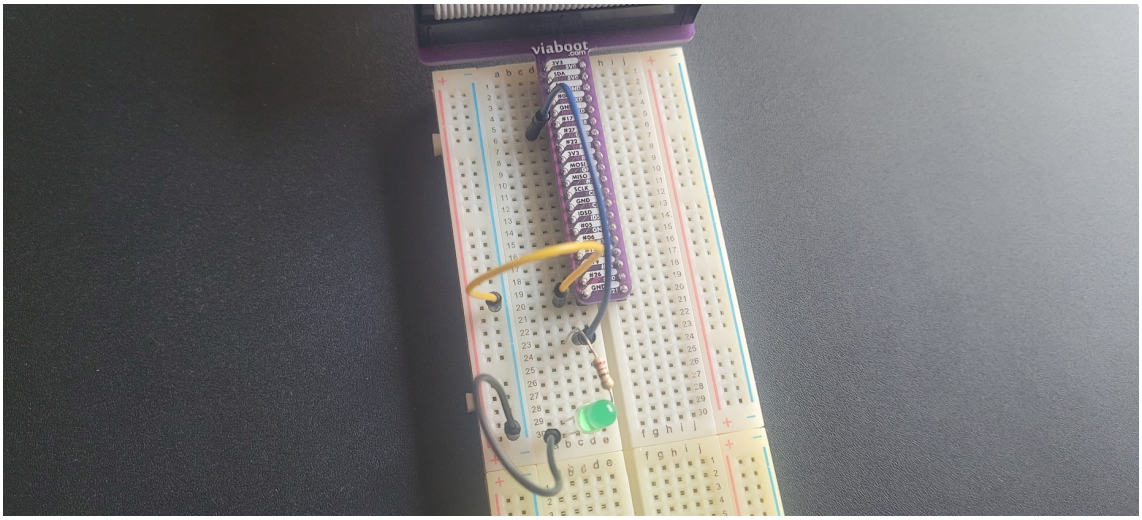

Figure 10. Plugging a final jumper cable to turn on/off your LED

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 GPIO libraries 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. The package `pigpio` is needed for this assignment and should be installed. See previous 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_pigpio.py` given with this assignment (can be downloaded from *UCStudent*) . Analyse the code and the use of the GPIO library. If needed, modify the code according to your circuit (in this assignment GPIO_PIN = **17** was used).

Transfer the `led_pigpio.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_pigpio.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 GPIO libraries 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 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_LED.py <Raspberry_Pi_address> <port>
```

On your PC:

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

4. Adding complexity to the IoT program

Goal: Use the GPIO libraries 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_LED.py <Raspberry_Pi_address> <port>
```

On your PC:

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

python client_udp.py <Raspberry_Pi_address> <port> OFF

python client_udp.py <Raspberry_Pi_address> <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.