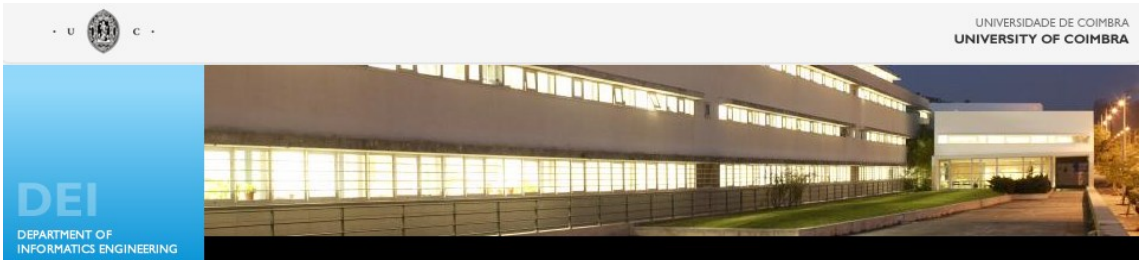

Sensing and Actuation Networks and Systems [2021-2022]

Assignment 04 – Adding complexity to your IoT program

Introduction

This assignment aims to practice network programming using the Python sockets API, while continuing exploring the breadboard and the GPIO interface of the Raspberry Pi.

Objectives

Students successfully concluding this work should be able to:

- Understand the use of the GPIO interface of the Raspberry Pi
- Use the dweet.io messaging service

Support Material

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

Theoretical background

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

The dweet.io platform

dweet.io[1] is a simple, effective, and free messaging service for the Internet of Things (IoT). The goal of IoT messaging is for a device to send a chunk of data to a server where it is stored for some period of time. That data can then be retrieved by one or multiple devices for analysis, display, or storage[2]. There could be many communication errors in a resource-constricted communication platform such as the IoT (e.g., loss of a message, duplicated messages). Dweet ignores these issues and offers a *best effort* service to send and receive messages, thus, the message integrity must be guaranteed by the end devices (i.e., applications in the end must use sequence numbers, timestamps, etc to offer a reliable service).

Exercises

Flashing a LED using Python and GPIOZero

Goal: Use the GPIOZero libraries to make a LED flash using Python.

Activities: Inside your directory, create a *virtual environment* using the command:

```
python -m venv <your_venv>
```

Download and take a look at the example code `led_gpiozero.py`. Analyse the code and the use of the GPIOZero library and notice the differences with the code from the previous assignment, based on `pigpio`. If needed, modify the code according to your circuit (GPIO_PIN = **17**). Run the example code and observe the results. You will need to run the GPIO daemon first. Use the following commands:

```
sudo pigpiod

python led_gpiozero.py
```

If the LED is connected correctly, it should blink each second. Identify the instruction(s) that make the LED blink and the differences with the code from the previous assignment.

Controlling the LED using a button in you circuit

Goal: Use a button to control the LED in your circuit.

Activities: Make sure that your Raspberry Pi is turned off. Connect a button to your breadboard. Add a jumper cable to ground your button (negative column). An example is provided in Figure 1.

[1] https://dweet.io/

[2] https://www.networkworld.com/article/3133738/dweetio-a-simple-effective-messaging-service-for-the-internet-of-things.html

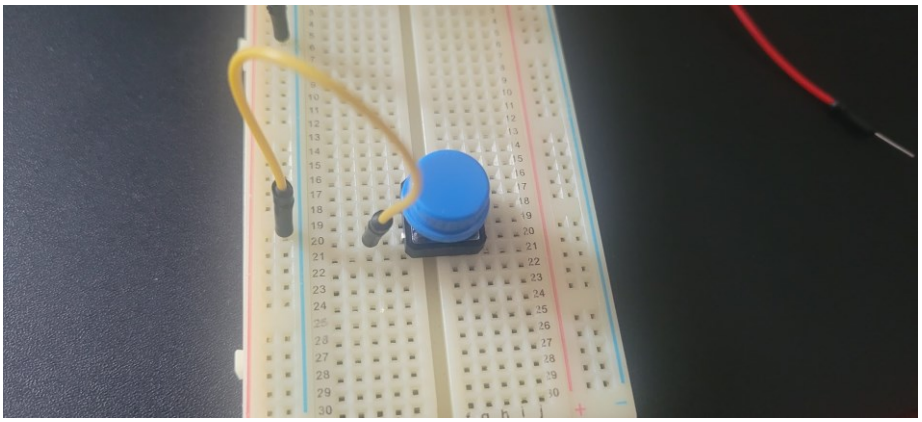

Figure 1. Adding a button to the circuit

Finally, add a jumper cable connecting your button to your GPIO. The example in Figure 2 uses the GPIO pin 23.

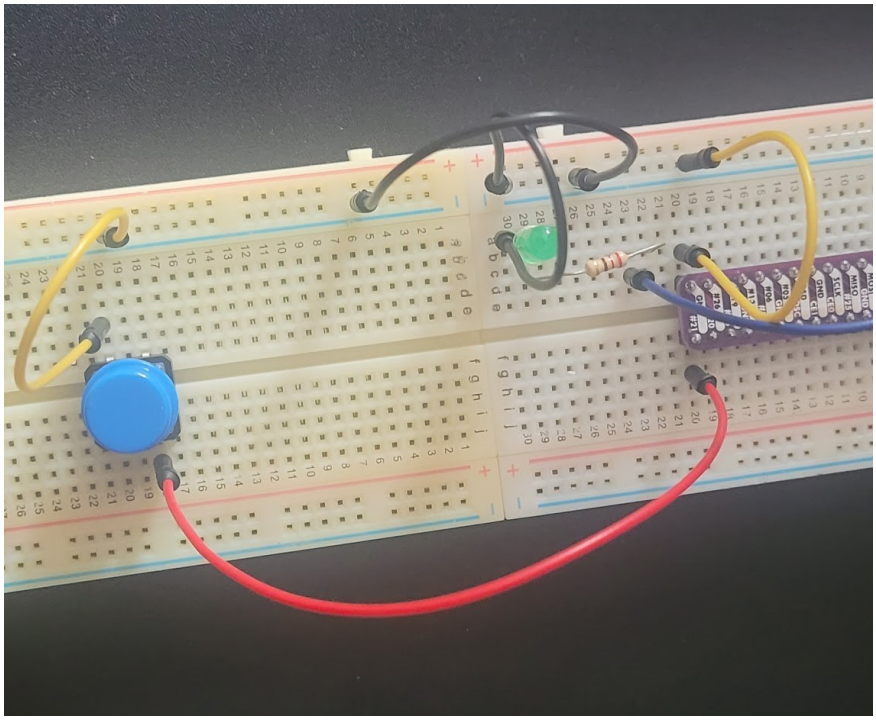

Figure 2. Adding an additional jumper cable to the circuit

Download and analyse the code `button_pigpio.py` and `button_gpiozero.py`. Verify that the code is using the same GPIO pins as your circuit. Run both codes. The LED should turn on and off once the button is pressed.

```
sudo pigpiod

python button_pigpio.py

python button_gpiozero.py
```

Using the dweet service to control the LED

Goal: Use the dweet public service to turn on your LED.

Activities: Complete the code provided (`dweet_led.py`) so that you create a Python program to control a LED using the public dweet.io service. This code monitors and receives *dweets* by polling a dweet.io RESTful API endpoint for data. The data received via *dweets* is passed to instructions specifying if the LED should be turned ON, OFF, or made to BLINK.

You must complete the code by adding the proper instructions that control the led. The missing code is inside the method `process_dweets()`. Use the GPIOZero library primitives. Once you finish the code, run it.

```
python dweet_led.py
```

You should see an output similar to this:

```
INFO:main:Created new thing name a8e38712            # (1)
LED Control URLs - Try them in your web browser:
On : https://dweet.io/dweet/for/a8e38712?state=on    # (2)
Off : https://dweet.io/dweet/for/a8e38712?state=off
Blink : https://dweet.io/dweet/for/a8e38712?state=blink
INFO:main:LED off
Waiting for dweets. Press Control+C to exit.
```

On line (1) the program has created a unique name for your *thing*[3]. You can go to a web browser and use the URLs from line (2) and below to control your LED. Please notice that there could be a short delay to execute the selected action (ON/OFF/BLINK).

- What does the `resolve_thing_name()` method does?
- What does the `get_latest_dweet()` method does?
- What does the `poll_dweets_forever()` method does?
- What does the `process_dweets()` method does?

Using the button to control the LED via dweets

Goal: Use the button to send messages via the dweet service.

Activities: Update the code provided (`dweet_button.py`) so that you create a Python program to control a LED using the public dweet.io service by using a button to post a dweet. You must update the *thing* name using the same value from the previous exercise. Each time the button is pressed, the program cycles through the dweet.io URLs to change the LED's state.

Execute the code. The LED should change its status each time you press the button. For this exercise, you will need the `dweet_led.py` program to be running in a Terminal, otherwise, the LED will not respond to the button presses. Please notice the delay could be longer this time.

```
python dweet_button.py
```

[3] Notice your *thing* name is like a @handle on Twitter. Be careful typing your thing name