Tien Nguyen **IOT ESSENTIALS**

**Report Project 2**

Information Technology

ITMI17SP 21/04/2019

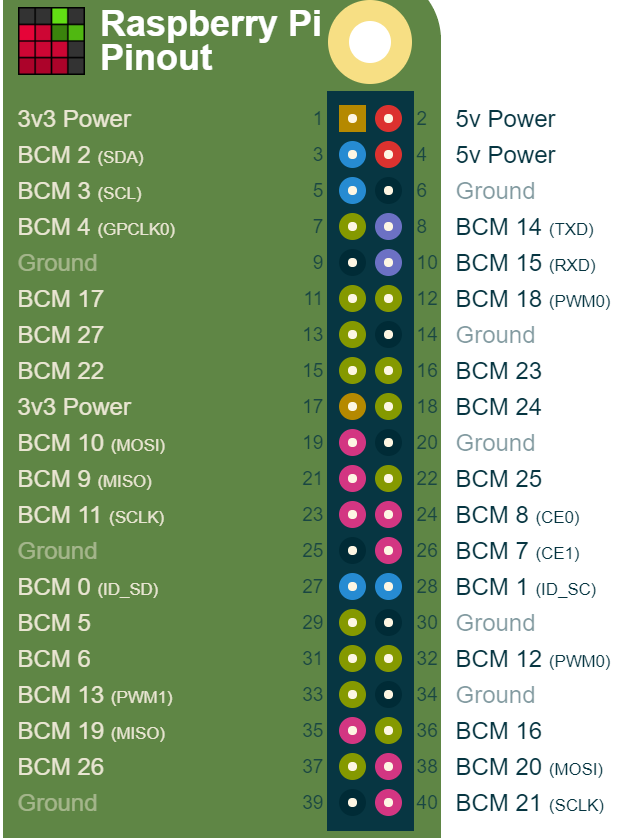
Timo Hynninen

**HELLO WORLD – I AM BLINKY!**

First, I explore the General Purpose Input/Output (GPIO) pins of the RPi, which allows connection to electronic hardware (LEDs, sensors). The pins can read and send information. The first task was to connect a LED to the RPi and write simple code to light up the LED. I went to terminal and installed the software that connected python to code for the GPIO using the command: “sudo apt-get install python-rpi.gpio python3-rpi.gpio”.

Of the 2 legs of the LED, the longer lead is the anode and the shorter lead is the cathode. To be able to connect the LED to the RPi, I needed a breadboard, some jumper wires and a resistor (around 470Ω to prevent the LED from being burnt.

I started to connect the LED to the RPi through the breadboard. According to the pinout:

I connect the ground (pin 6) to the cathode lead of the LED. I put one of the resistor legs on the same line as the anode lead of the LED, and I connect the other leg of the resistor with any of the BCM pins.

Next, I insert the code to IDLE editor that comes with Python 3 to light up the LED as follow:

from gpiozero import LED

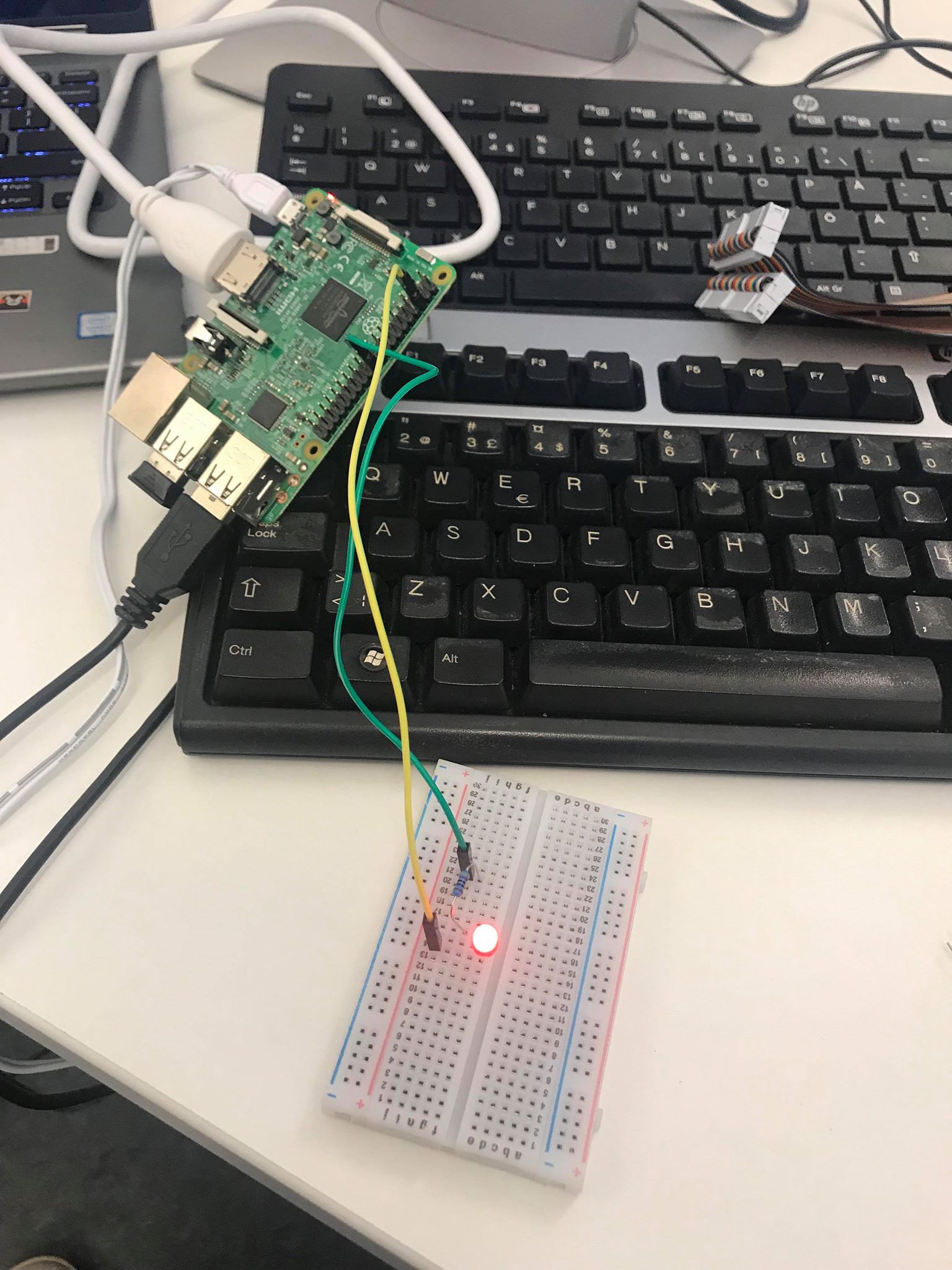
from time import sleep

led = LED(25)

led.on()

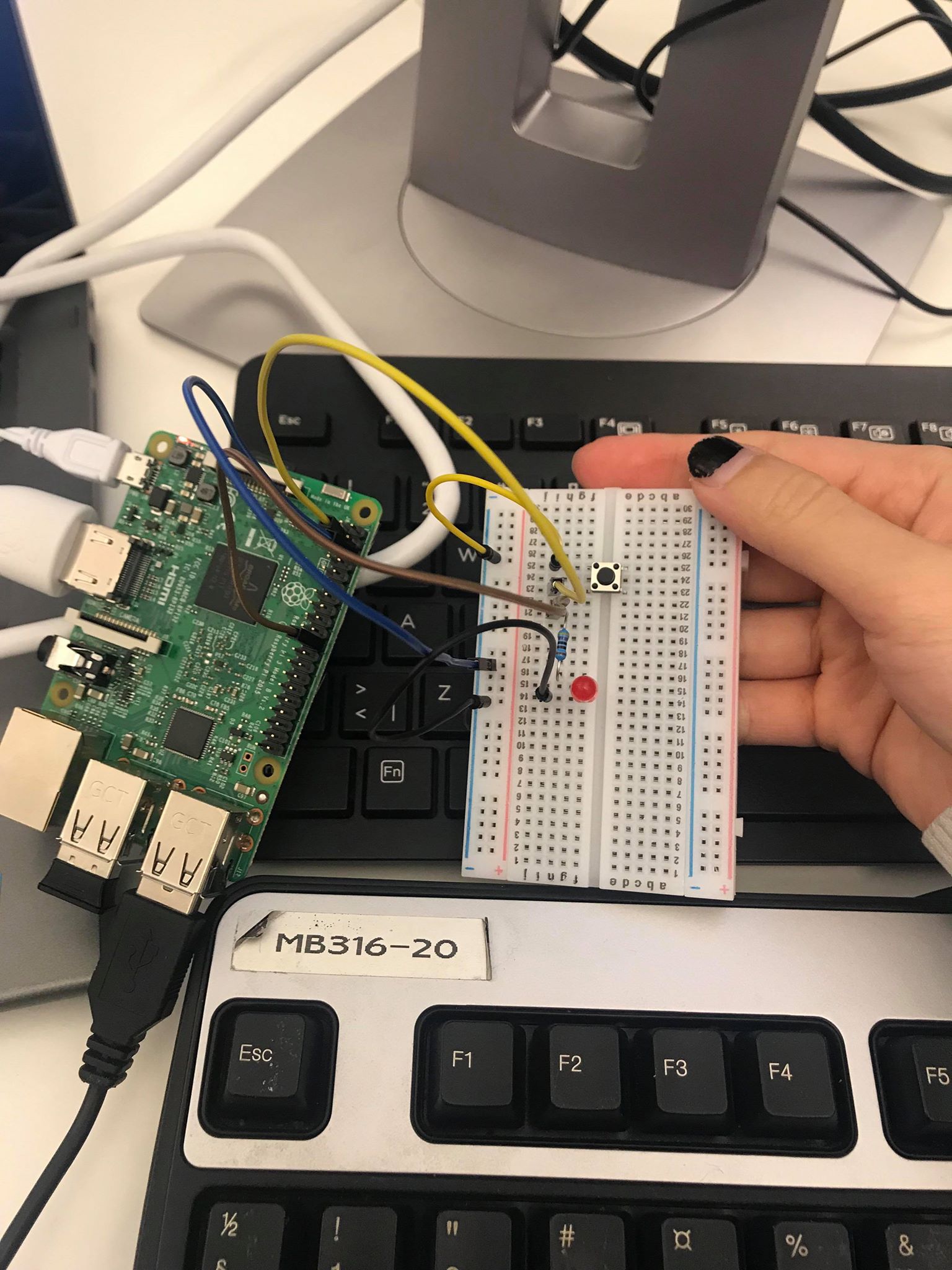
Sleep(10)

led.off()

The circuit looked like this when I run the code:

The first 2 lines of the codes were to make the LED and time recognized to Python. The third line was to assign the LED connected to BCM 25 to variable “led”. This made it easier to call it later. The last 3 lines were the operation/action: led turned on, waited for 10s, then turned off.

The next task was to turn on the LED using a button. I then the connected the “ground” to the negative power rail of the breadboard for more convenient use. I connected one leg of the button to the ground and the other leg to any of the BCM. The circuit and the codes look like this:



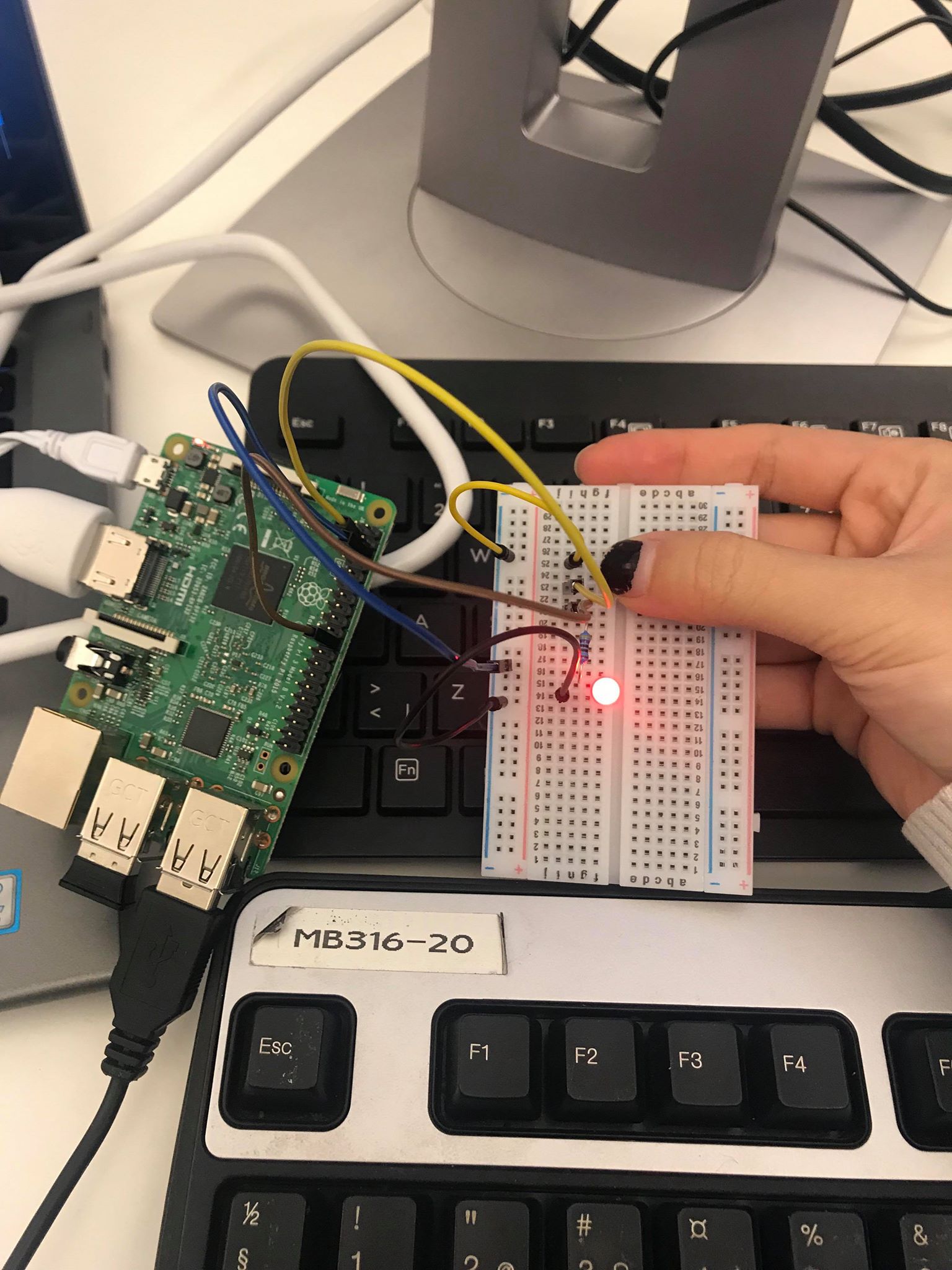
from gpiozero import LED, button

from signal import pause

led = LED(25)

button = Button(2)

button.when\_pressed = led.on()

button.when\_released = led.off()

pause()

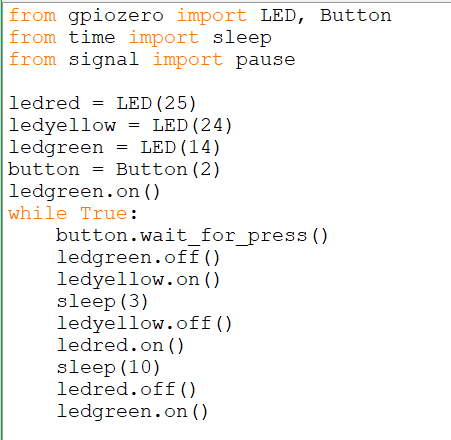
The codes look pretty self-explained. The “pause()” operation at the end was to stop the script from terminating without an infinite loop.

**PROJECT CHALLENGE**

The challenge was to design the traffic lights, where there would be a push-button to change the state of the traffic light. The light cycles from green to red by changing into yellow in between.

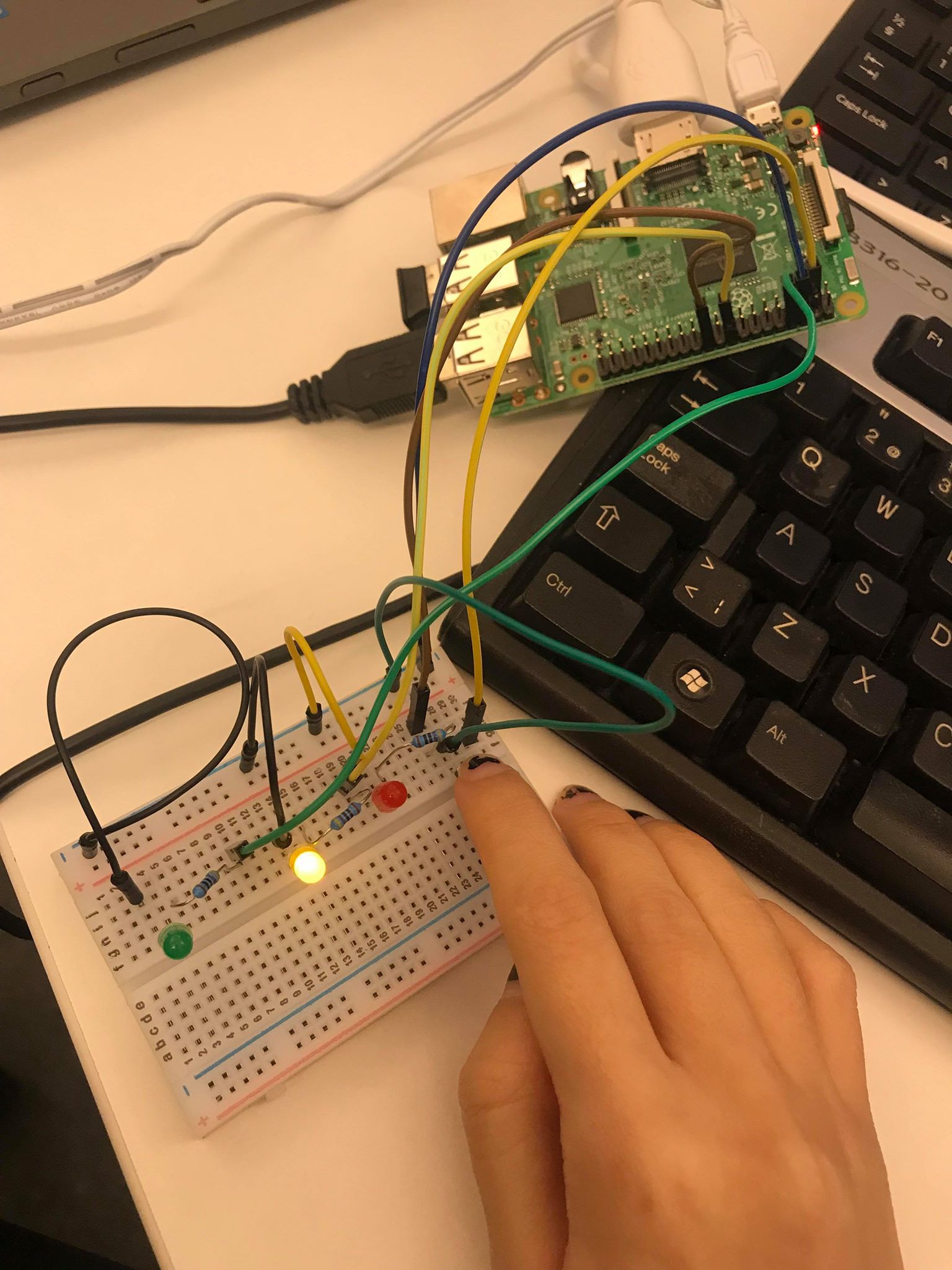
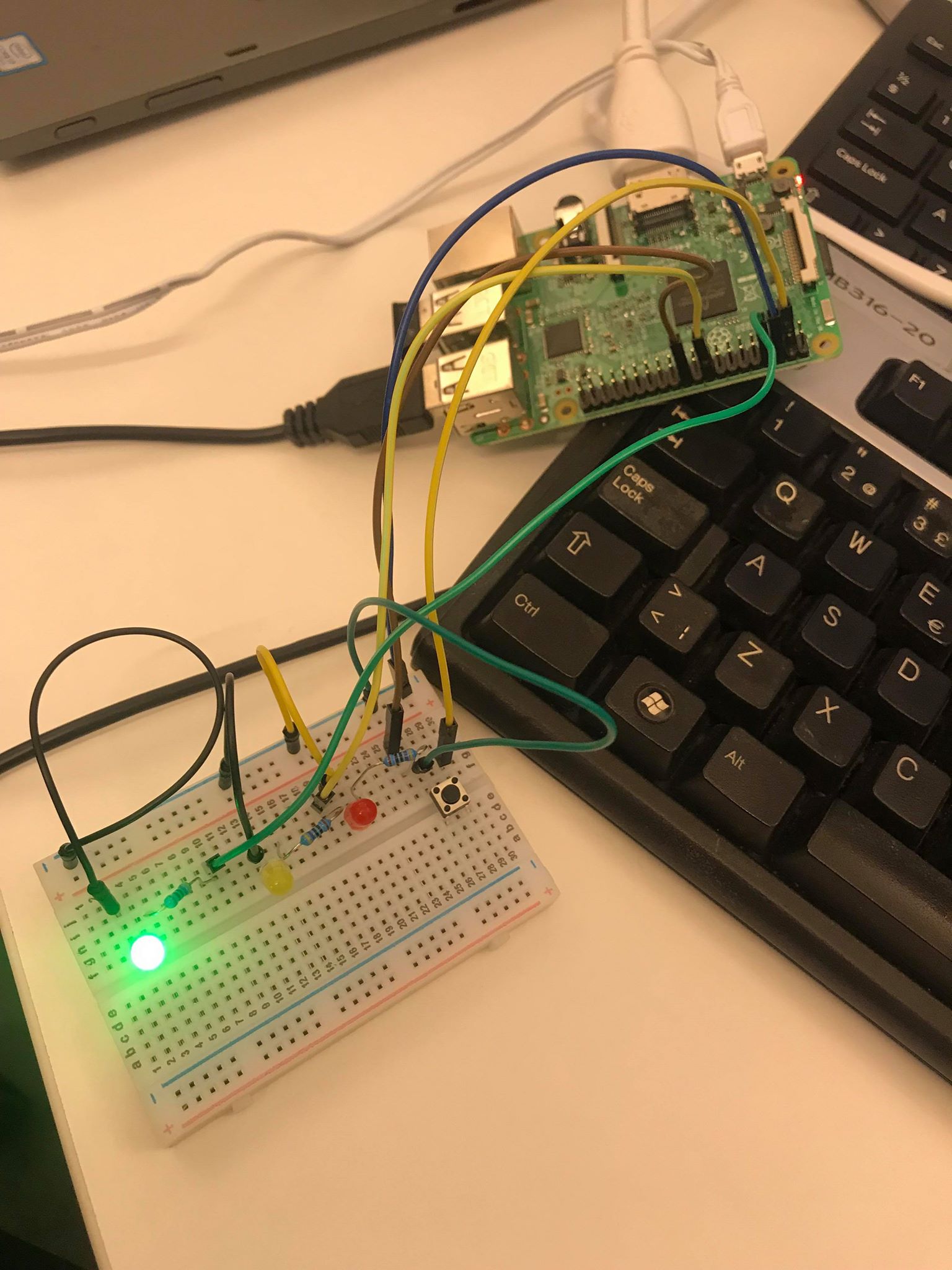
I used a button and 3 LEDs in this challenge: red, yellow and green. Accompanied the leds would be 3 resistors of 470 Ω. I connect the LEDs (with resistors) and button separately, as I did with the previous task: one lead to the ground, and on lead to the BCM. Each LED was assigned to a different BCM.

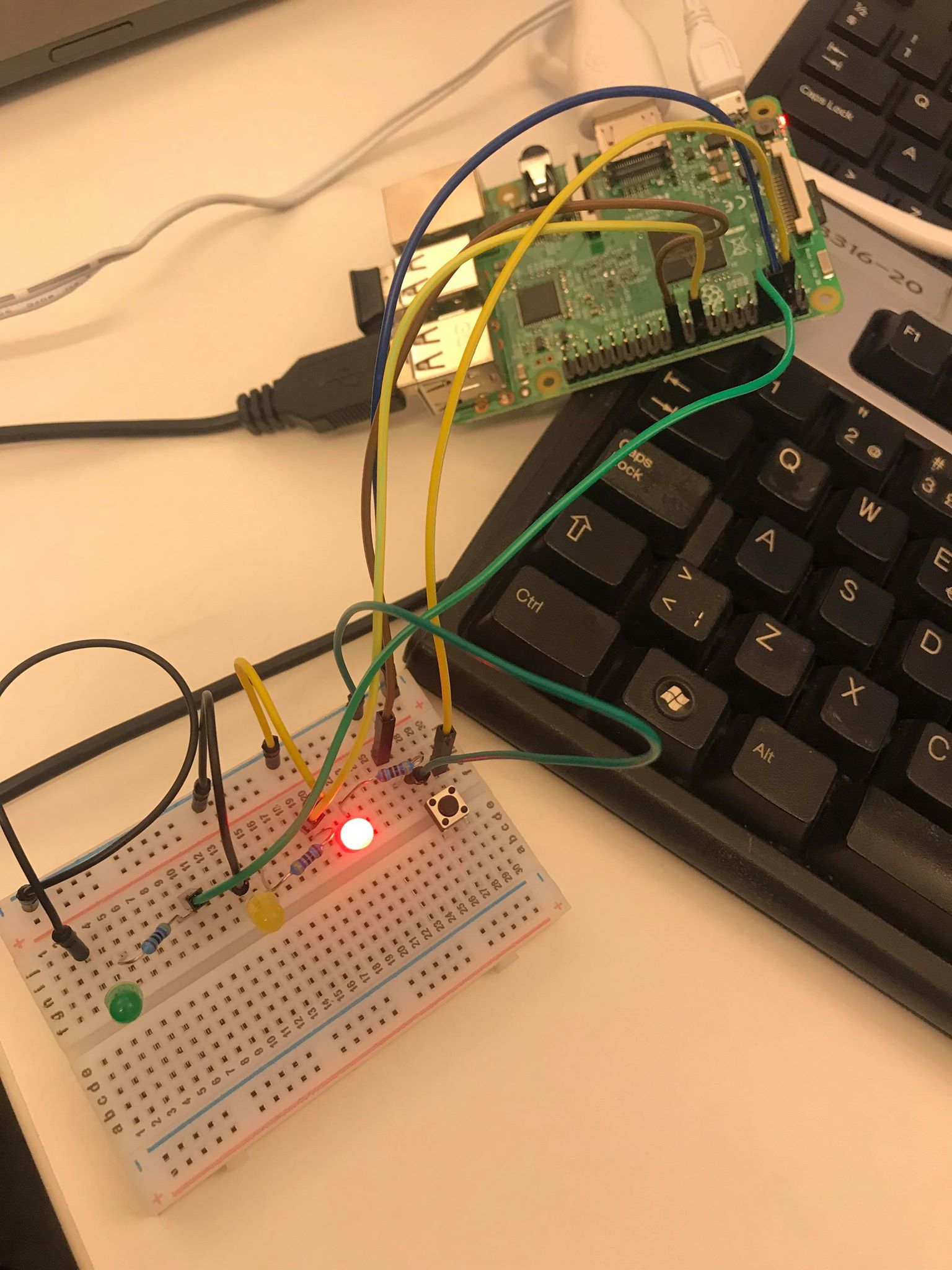
The codes to make the traffic lights operated as wish were as follow:

After assigning the LEDs and button to the variables, I turn on the green light and it will always be the default. Next I created a loop with statement “while True:”. The part in the loop is changing the state between the lights. Operation “wait\_for\_pressed” was used as I only wanted to press the button once instead of having to hold the button like in the previous task. After I pressed the button, there would be a sequence of actions:

1. The green led turned off
2. The yellow led turned on and waited for 3s
3. The yellow led turned off
4. The red led turned on and waited for 10s
5. The red led turned off
6. The green led turned back on

The circuit looked and functioned like this:





There is an attached video filming the fuction of my traffic lights.

The link to the repository of the code for the project challenge is <https://github.com/ttienism/IoT-projects>

**CONCLUSION**

I learnt how to build a simple circuit with breadboard and jumper wires. I also learnt how to implement Python codes with the GPIO pins of RPi to do basic action like making a LED blink.

The project challenge is very interesting to do and the result is satisfying.