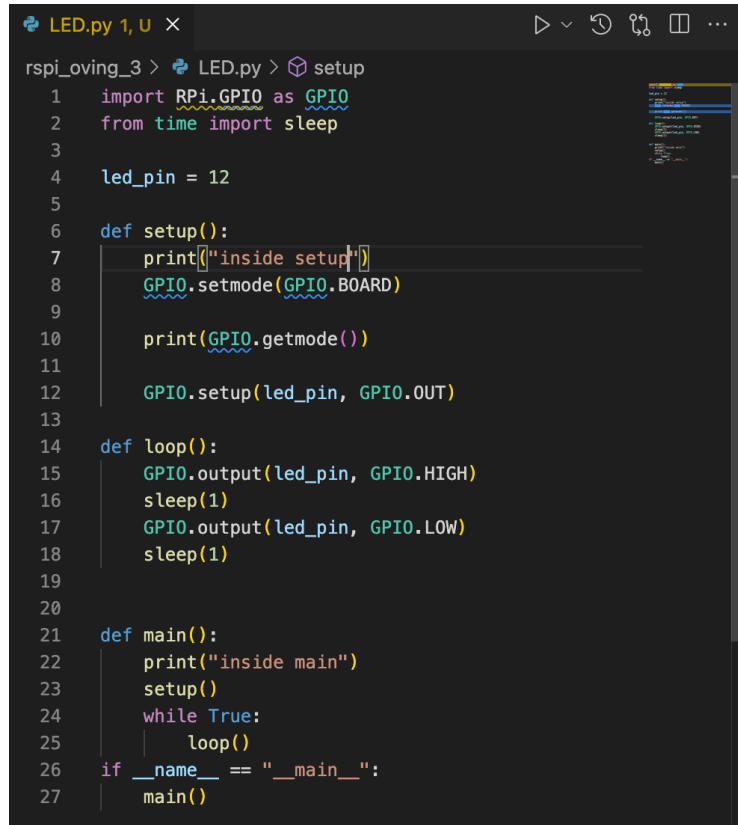


1. Blink en LED (10%)

Ferdig! (se vedlagt kode fra zipfilen)



```
LED.py 1, U x
rspi_oving_3 > LED.py > setup
1  import RPi.GPIO as GPIO
2  from time import sleep
3
4  led_pin = 12
5
6  def setup():
7      print("inside setup")
8      GPIO.setmode(GPIO.BOARD)
9
10     print(GPIO.getmode())
11
12     GPIO.setup(led_pin, GPIO.OUT)
13
14  def loop():
15     GPIO.output(led_pin, GPIO.HIGH)
16     sleep(1)
17     GPIO.output(led_pin, GPIO.LOW)
18     sleep(1)
19
20
21  def main():
22     print("inside main")
23     setup()
24     while True:
25         loop()
26  if __name__ == "__main__":
27     main()
```

Fig 1. 'led.py' som bruker GPIO

Jeg kodet det slik at det så ut som arduino void loop() og void setup() funksjon. Det funker greit!! 😊

Jeg brukte i tillegg sleep() som jeg lærte om tidligere. Det fungerer som delay().

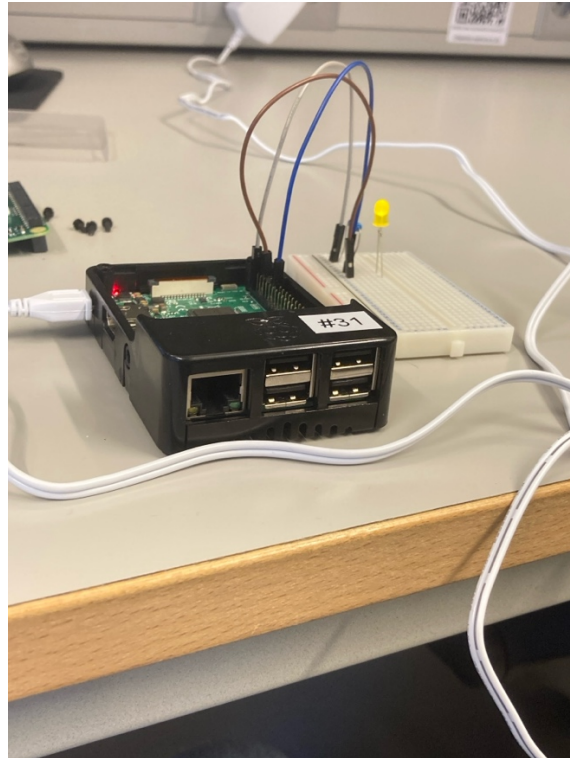
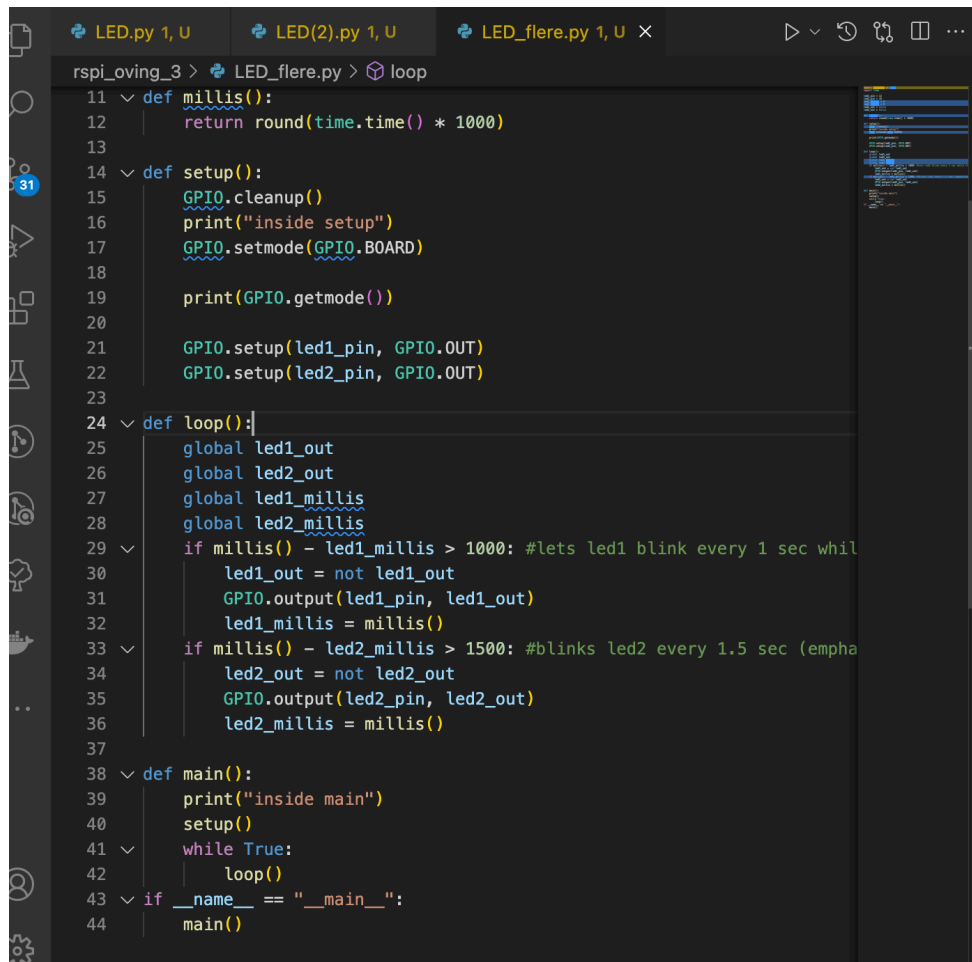


Fig 2. Oppkoblingen samt blink

2. Blink flere LED (20%)

Ferdig! For `millis()`, hadde jeg forskjellige hastighet av blinking 1s og 1.5s for å understreke ikke-blokkerende kode. Koden er gitt under:



```
rspi_oving_3 > LED_flere.py > loop
11 def millis():
12     return round(time.time() * 1000)
13
14 def setup():
15     GPIO.cleanup()
16     print("inside setup")
17     GPIO.setmode(GPIO.BOARD)
18
19     print(GPIO.getmode())
20
21     GPIO.setup(led1_pin, GPIO.OUT)
22     GPIO.setup(led2_pin, GPIO.OUT)
23
24 def loop():
25     global led1_out
26     global led2_out
27     global led1_millis
28     global led2_millis
29     if millis() - led1_millis > 1000: #lets led1 blink every 1 sec while
30         led1_out = not led1_out
31         GPIO.output(led1_pin, led1_out)
32         led1_millis = millis()
33     if millis() - led2_millis > 1500: #blinks led2 every 1.5 sec (empha
34         led2_out = not led2_out
35         GPIO.output(led2_pin, led2_out)
36         led2_millis = millis()
37
38 def main():
39     print("inside main")
40     setup()
41     while True:
42         loop()
43 if __name__ == "__main__":
44     main()
```

Fig 3. `flere_LED.py` som blinker 2 forskjellige LED hastigheter

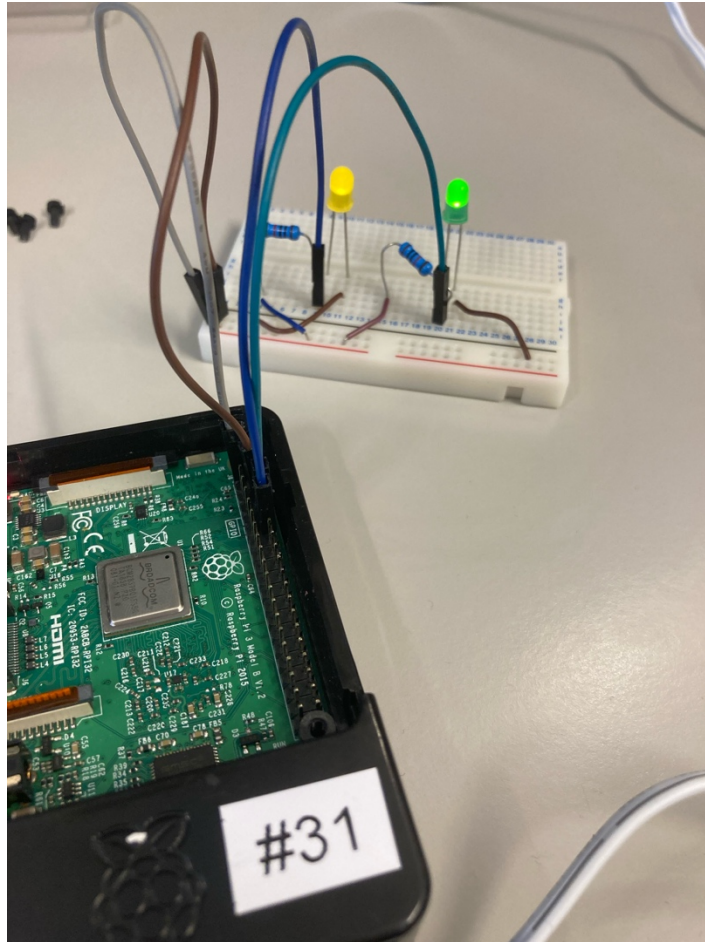


Fig 4. Oppkoblingen av flere LED

3. SW vs. HW PWM (30%)

Her måtte jeg finne en formel for å omgjøre duty cycle til antall rader. ChatGPT hjalp mye her. Koden er vedlagt, det fungerer for software sweeping og slik. Jeg får problemer derimot med hardware.

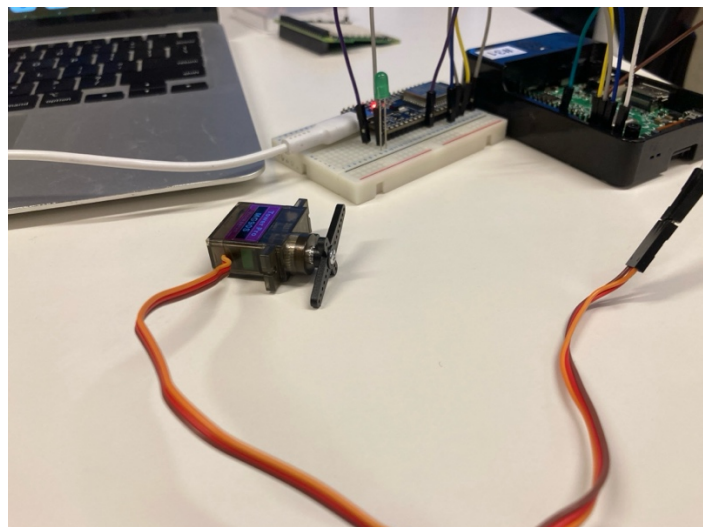
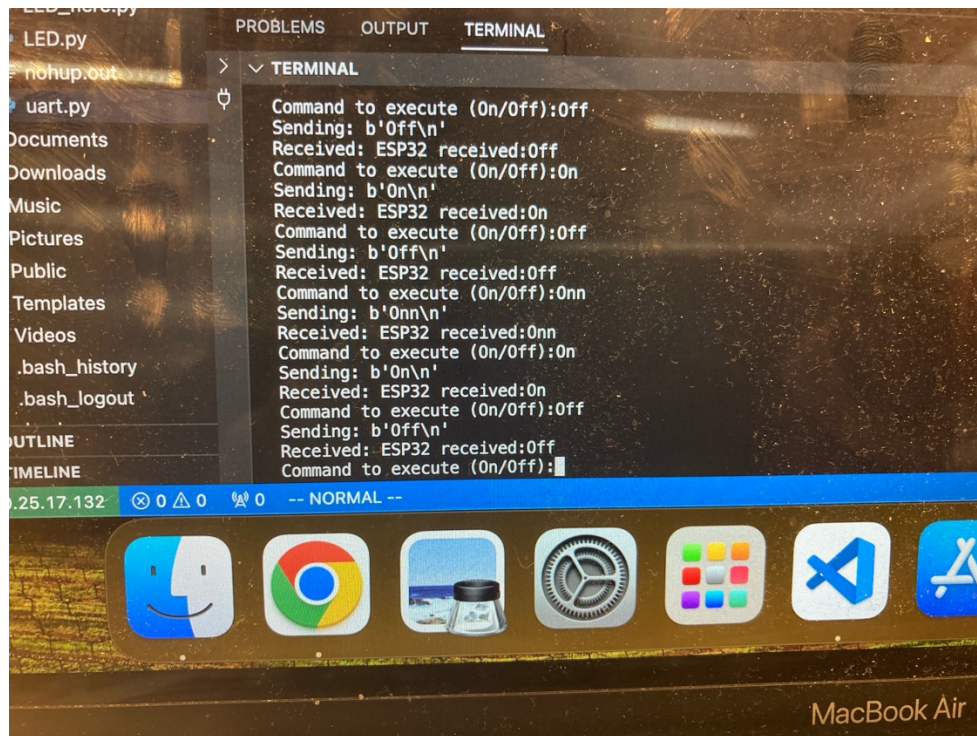


Fig 5. Oppkobling servo

4. Kommunisere med esp32 over UART

Her var to seriellkommunikasjon satt opp i esp32. En for seriell og en seriell2 for uart. TX er koblet til RX og vise versa til raspberry pi. Det er viktig at de har samme **GND** slik at det virker og de får ordrene til og fra hverandre.



```
LED_hare.py
LED.py
nohup.out
uart.py
Documents
Downloads
Music
Pictures
Public
Templates
Videos
.bash_history
.bash_logout
OUTLINE
TIMELINE
0.25.17.132 0 0 0 0 -- NORMAL --
MacBook Air

> ✓ TERMINAL
Command to execute (0n/0ff):0ff
Sending: b'0ff\n'
Received: ESP32 received:0ff
Command to execute (0n/0ff):0n
Sending: b'0n\n'
Received: ESP32 received:0n
Command to execute (0n/0ff):0ff
Sending: b'0ff\n'
Received: ESP32 received:0ff
Command to execute (0n/0ff):0nn
Sending: b'0nn\n'
Received: ESP32 received:0nn
Command to execute (0n/0ff):0n
Sending: b'0n\n'
Received: ESP32 received:0n
Command to execute (0n/0ff):0ff
Sending: b'0ff\n'
Received: ESP32 received:0ff
Command to execute (0n/0ff):
```

Fig 6. UART RSPI gir ordre til ESP32 (seriell)

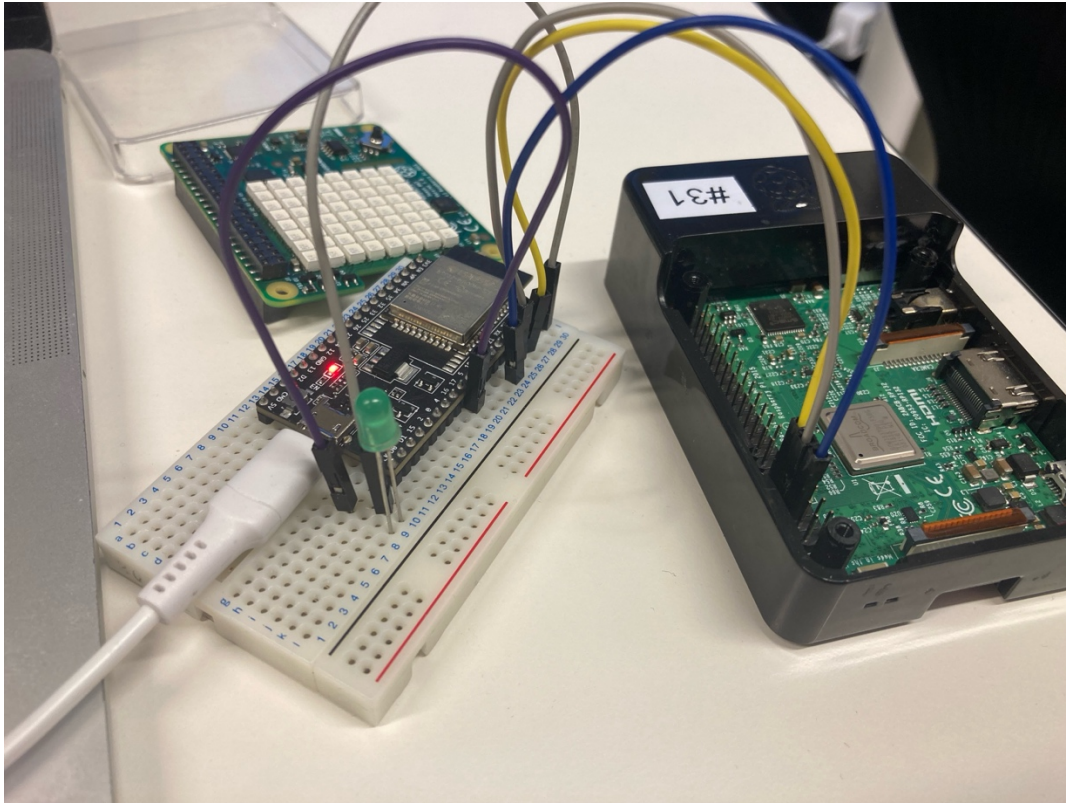


Fig 7. Oppkoblingen (LED er «av»)

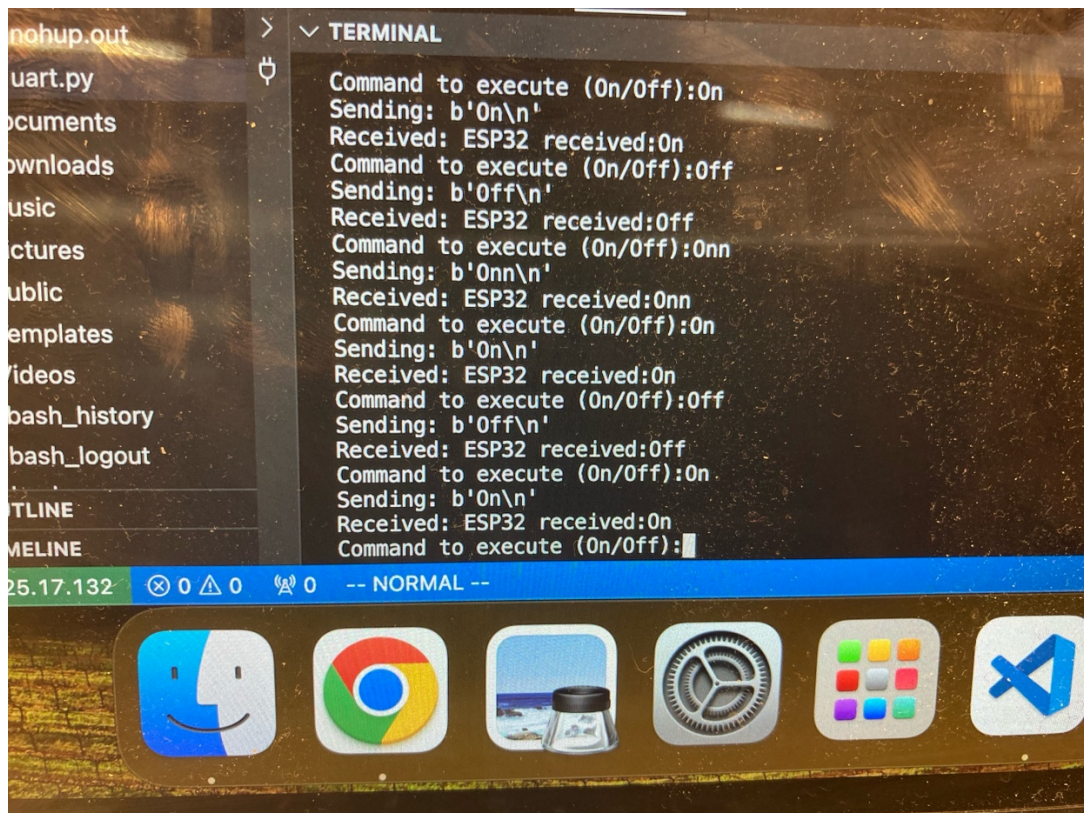


Fig 8. Ordre «ON» gitt av bruker

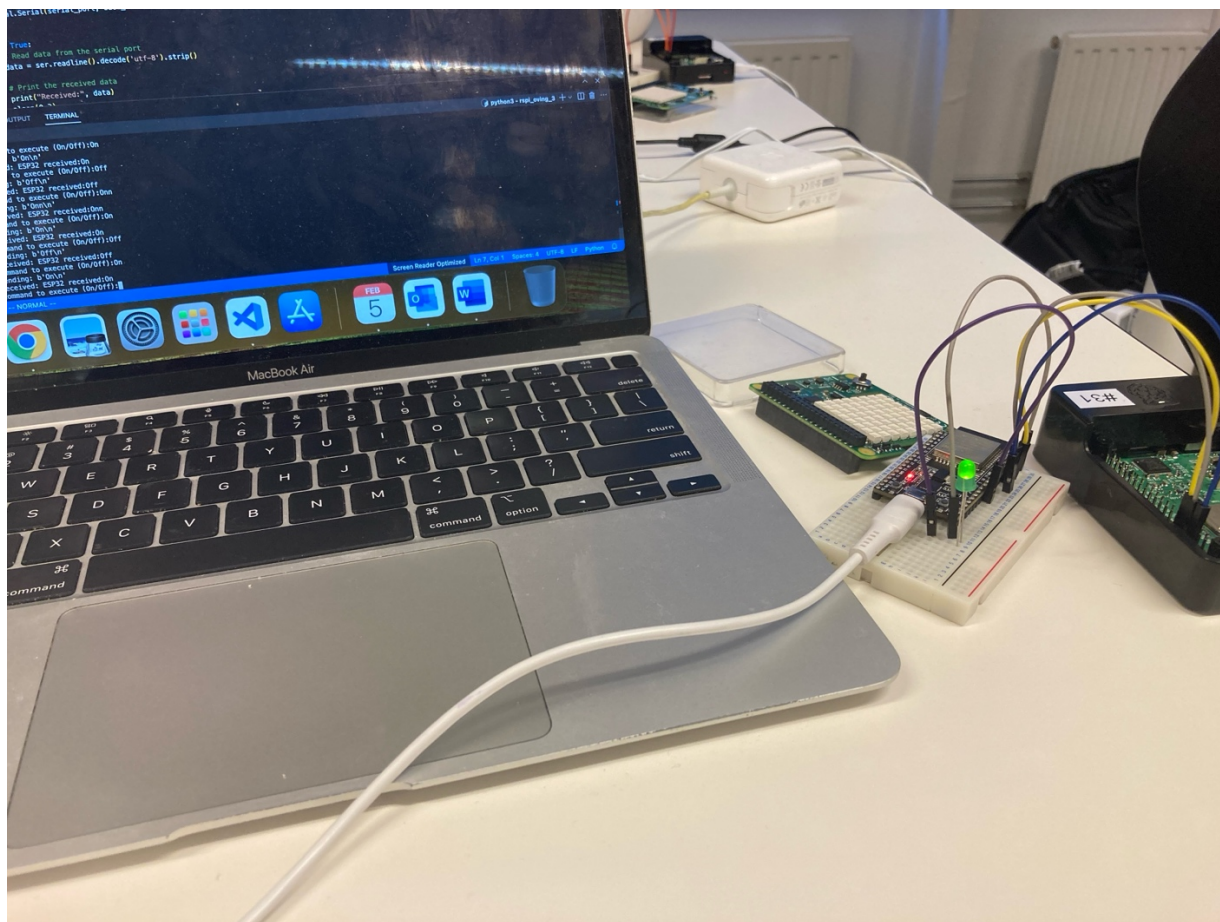


Fig 9. Led-en slår på