

# Hochschule Ravensburg-Weingarten

## University of Applied Sciences

### Task - 1 : Verification of Raspbian Image Using the VNC Viewer

| Sr No | Student Names             | Matriculation Numbers |
|-------|---------------------------|-----------------------|
| 1     | Arjun Ramwshbhai Beladiya | 28744163              |

Guided by:  
Prof. Markus Pfeil

---

**Task a) Connect the ET1 using the details given in the info pdf.**

## **Answer**

- SSID: raspi-webgui\_23
- Password: ChangeMe
- IP-Address: 10.3.141.1
- VNC Password: pivncet1

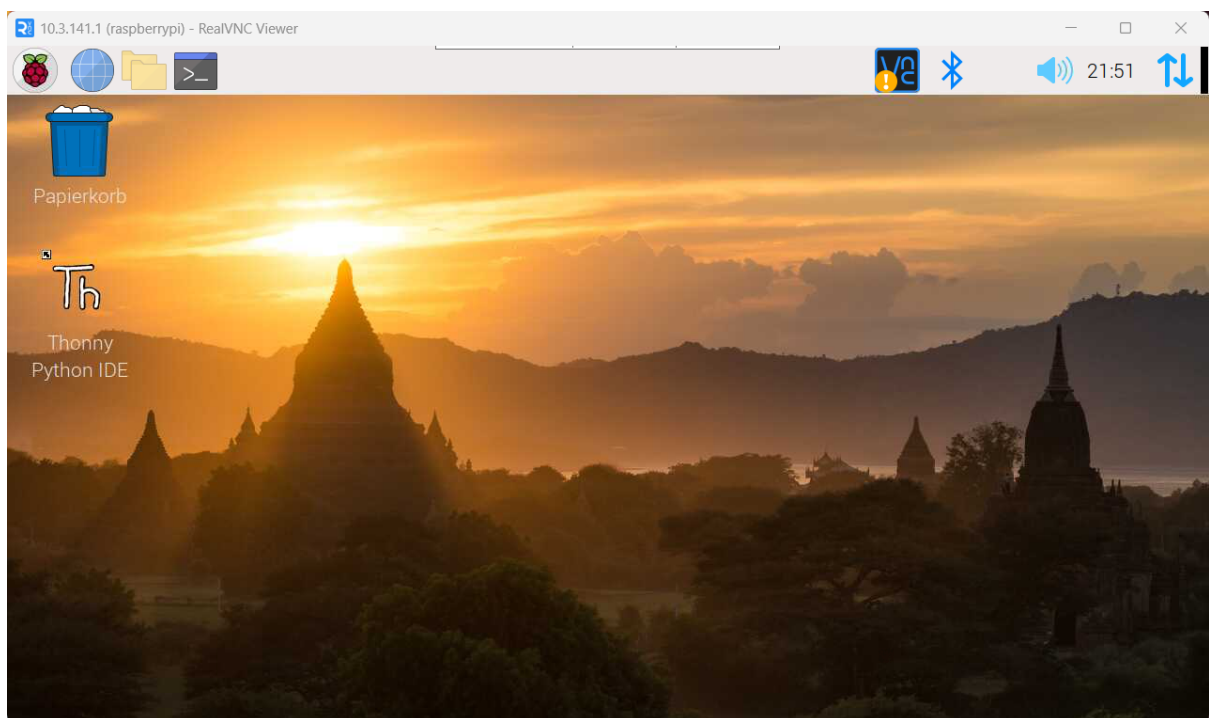


Figure 1: Raspberry Pi Desktop Environment Accessed via VNC

## Task b) Using the VNC Viewer and verify the image has the necessary programs on.

- Thonny (Python IDE)

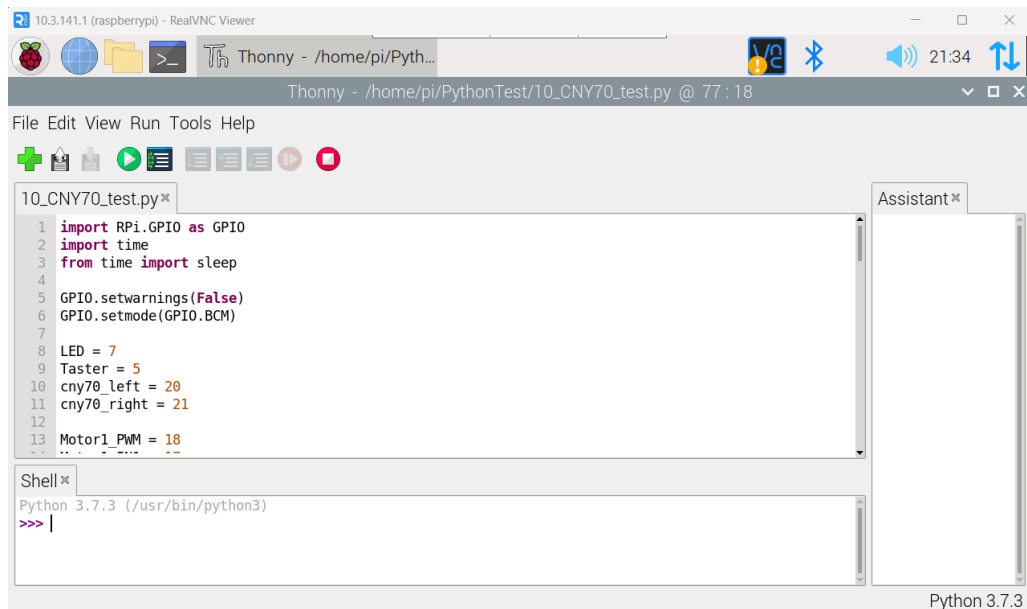


Figure 2: Python IDE

- C-Development Environment

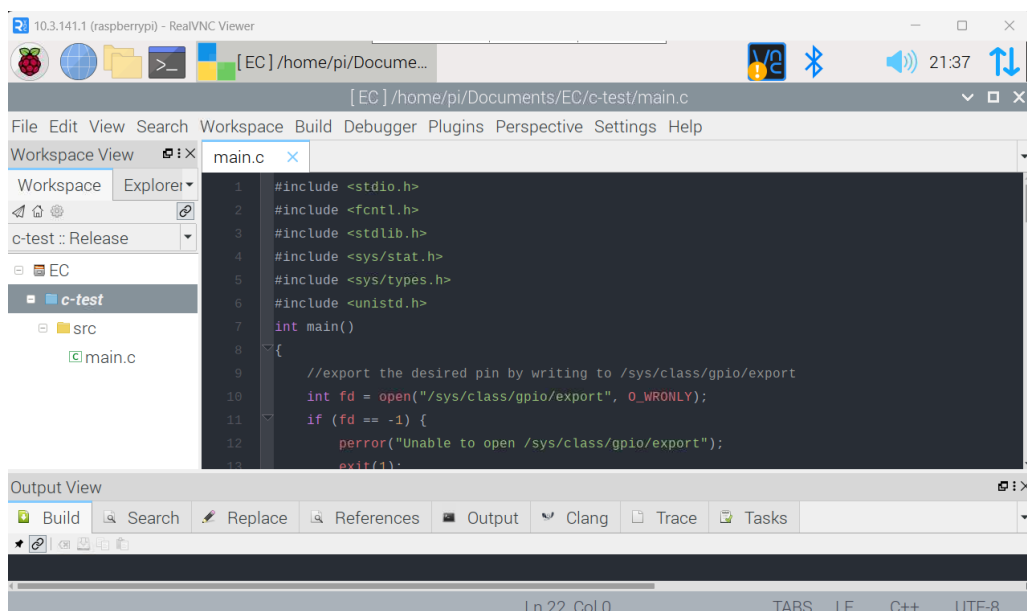


Figure 3: C-Development Environment

- Node.JS Installation

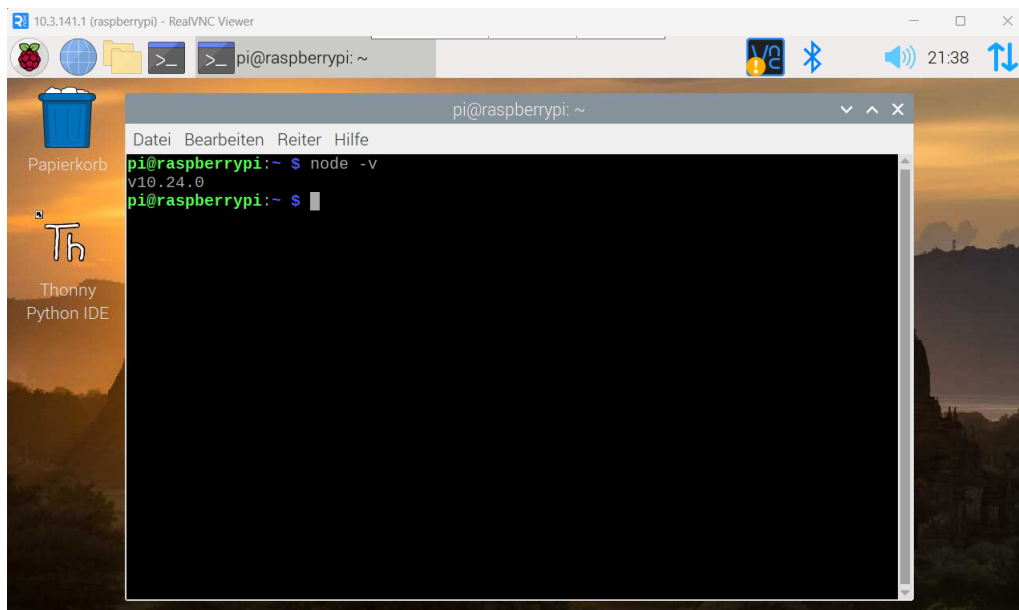


Figure 4: Node.JS Installation

- Firefox

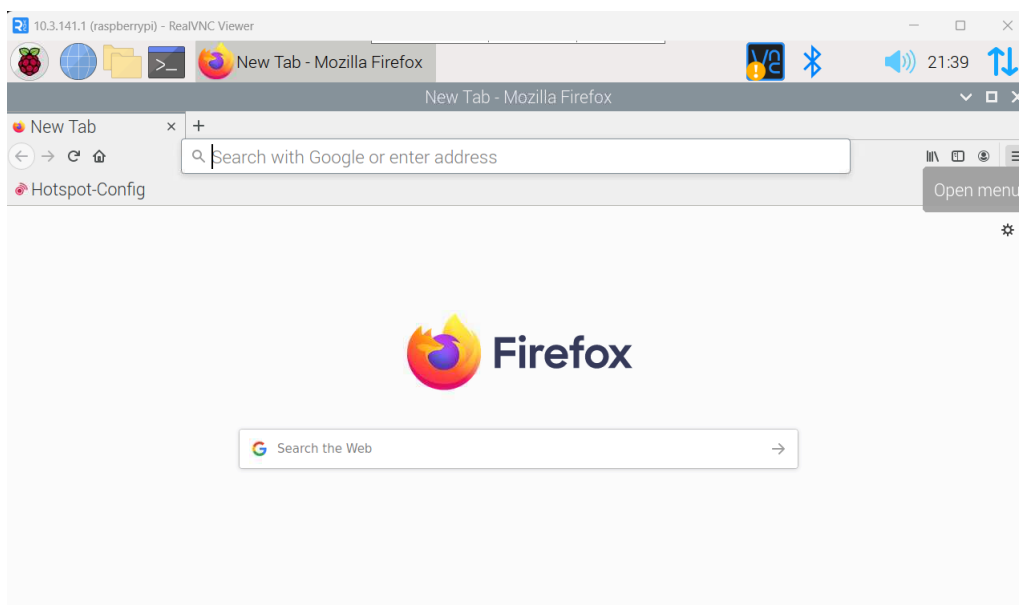
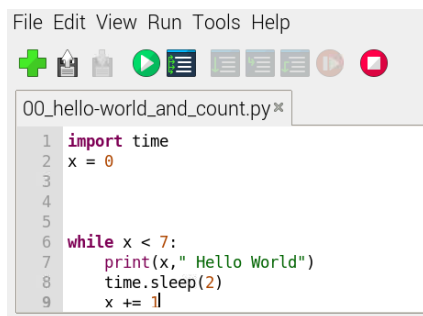


Figure 5: Firefox

## Task c) Using the Thonny Python IDE find the test Python scripts and evaluate what they do.

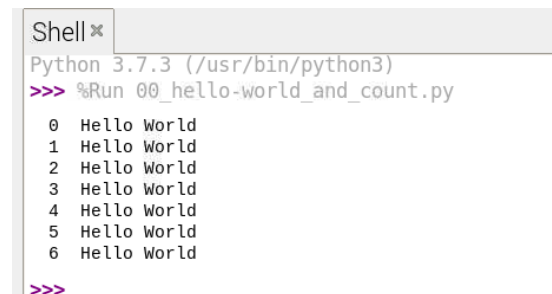
### 00: Hello World

- This code simply prints "Hello World" seven times. It starts at zero, waits two seconds between each message, and keeps counting up until it hits seven.



```
File Edit View Run Tools Help
00_hello-world_and_count.py
1 import time
2 x = 0
3
4
5
6 while x < 7:
7     print(x, " Hello World")
8     time.sleep(2)
9     x += 1
```

Figure 6: Code



```
Shell x
Python 3.7.3 (/usr/bin/python3)
>>> %Run 00_hello-world_and_count.py
0 Hello World
1 Hello World
2 Hello World
3 Hello World
4 Hello World
5 Hello World
6 Hello World
>>>
```

Figure 7: Result

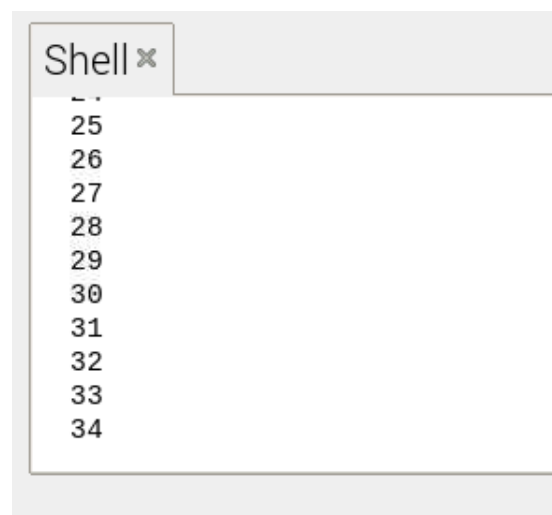
### 01: Blink LED

- This code makes an LED blink on and off every second. It starts counting from zero and prints the current count each time the LED changes state. The LED is connected to pin 7 on the Raspberry Pi.



```
File Edit View Run Tools Help
01_blink.py
1 import RPi.GPIO as GPIO
2 import time
3
4 GPIO.setwarnings(False)
5 GPIO.setmode(GPIO.BCM)
6
7 LED = 7
8
9 GPIO.setup(LED, GPIO.OUT)
10 i = 0
11
12 while (1):
13     print(i)
14     i += 1
15     GPIO.output(LED, GPIO.HIGH)
16     time.sleep(1)
17     GPIO.output(LED, GPIO.LOW)
18     time.sleep(1)
19
20
```

Figure 8: Code



```
Shell x
25
26
27
28
29
30
31
32
33
34
```

Figure 9: Result

## 02: Blink Push Button

- This code controls an LED using a pushbutton. When you press the button, the LED turns off. When you let go, the LED turns back on.

```
File Edit View Run Tools Help
+ [Icons]
02_blink_pushbutton.py*
1 import RPi.GPIO as GPIO
2 import time
3
4 GPIO.setwarnings(False)
5 GPIO.setmode(GPIO.BCM)
6
7 LED = 7
8 Taster = 5
9
10 GPIO.setup(LED, GPIO.OUT)
11 GPIO.setup(Taster, GPIO.IN, pull_up_down = GPIO.PUD_UP)
12
13 # Pressing the pushbutton (Taster) LED is off
14 while (1):
15     if GPIO.input(Taster) == GPIO.HIGH:
16         GPIO.output(LED, GPIO.HIGH)
17     if GPIO.input(Taster) == GPIO.LOW:
18         GPIO.output(LED, GPIO.LOW)
19
```

Figure 10: Code

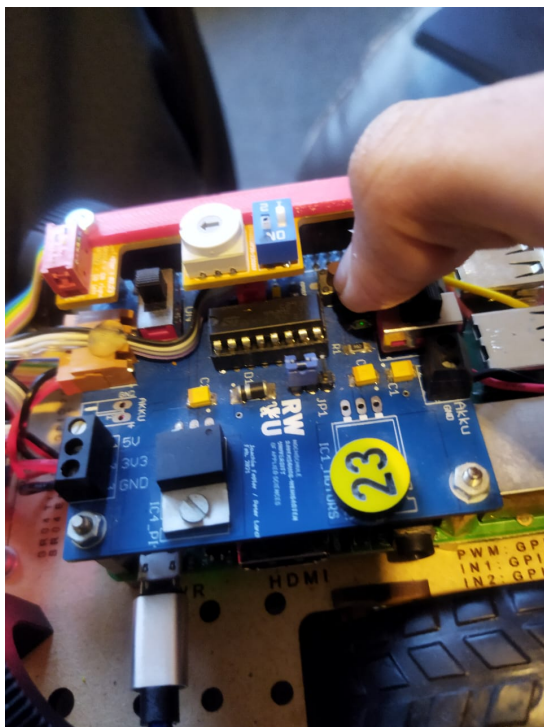


Figure 11: Pushbutton is Pressed

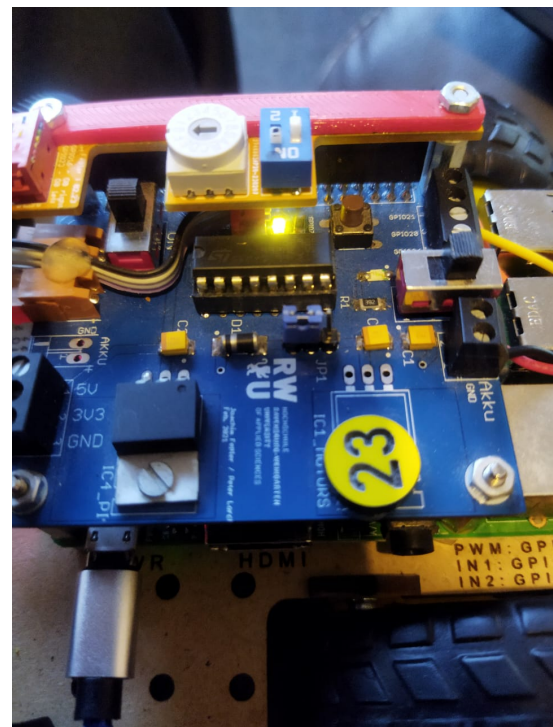
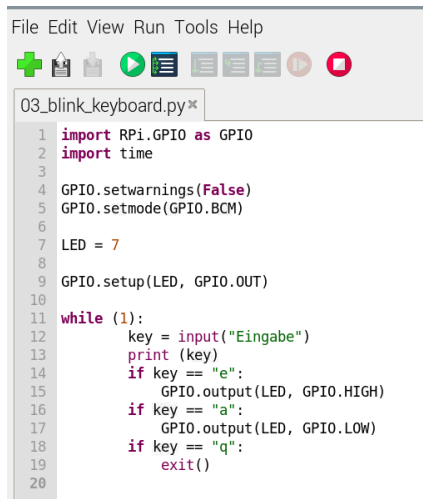


Figure 12: Pushbutton without pressing

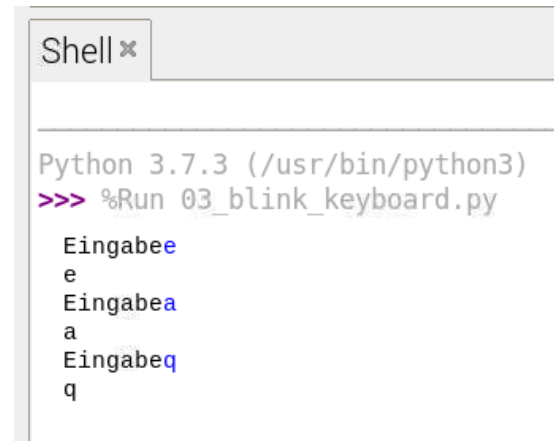
### 03: Blink Keyboard

- This code waits for keyboard input in a loop. If you type 'e' and press Enter, the LED turns on. Typing 'a' turns it off. Typing 'q' exits the program.



```
03_blink_keyboard.py*
1 import RPi.GPIO as GPIO
2 import time
3
4 GPIO.setwarnings(False)
5 GPIO.setmode(GPIO.BCM)
6
7 LED = 7
8
9 GPIO.setup(LED, GPIO.OUT)
10
11 while (1):
12     key = input("Eingabe")
13     print(key)
14     if key == "e":
15         GPIO.output(LED, GPIO.HIGH)
16     if key == "a":
17         GPIO.output(LED, GPIO.LOW)
18     if key == "q":
19         exit()
20
```

Figure 13: Code



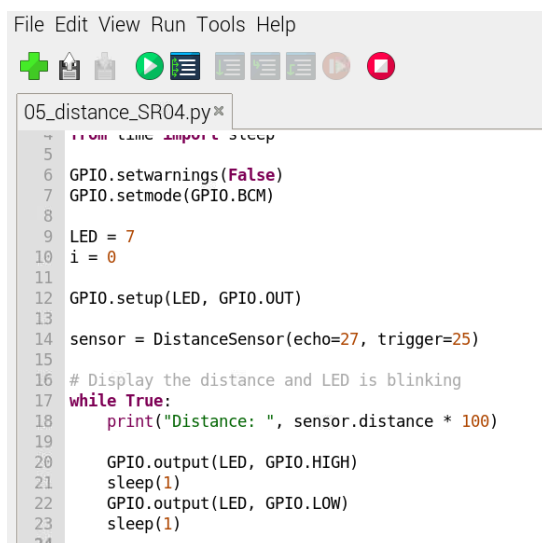
```
Shell x
Python 3.7.3 (/usr/bin/python3)
>>> %Run 03_blink_keyboard.py

Eingabee
e
Eingabea
a
Eingabeq
q
```

Figure 14: Result

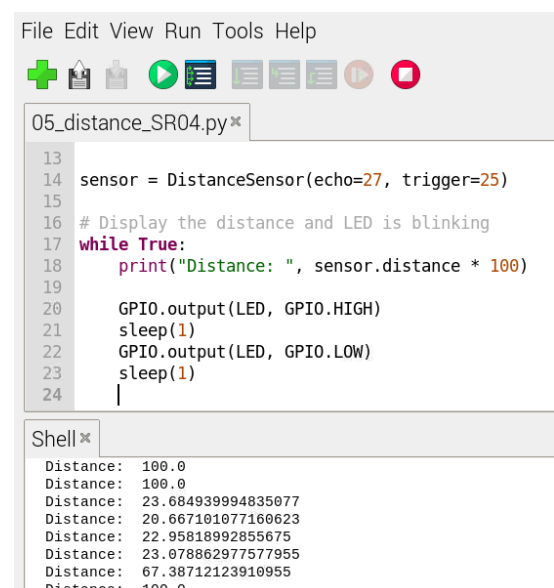
### 05: Distance SR07

- This code constantly measures distance using the SR07 sensor and prints the result. At the same time, it blinks an LED on pin 7 every second, giving a visual cue while displaying the distance.



```
File Edit View Run Tools Help
05_distance_SR04.py*
1 from time import sleep
2
3 GPIO.setwarnings(False)
4 GPIO.setmode(GPIO.BCM)
5
6 LED = 7
7 i = 0
8
9 GPIO.setup(LED, GPIO.OUT)
10
11 sensor = DistanceSensor(echo=27, trigger=25)
12
13 # Display the distance and LED is blinking
14 while True:
15     print("Distance: ", sensor.distance * 100)
16
17     GPIO.output(LED, GPIO.HIGH)
18     sleep(1)
19     GPIO.output(LED, GPIO.LOW)
20     sleep(1)
21
```

Figure 15: Code



```
File Edit View Run Tools Help
05_distance_SR04.py*
13
14 sensor = DistanceSensor(echo=27, trigger=25)
15
16 # Display the distance and LED is blinking
17 while True:
18     print("Distance: ", sensor.distance * 100)
19
20     GPIO.output(LED, GPIO.HIGH)
21     sleep(1)
22     GPIO.output(LED, GPIO.LOW)
23     sleep(1)
24
Shell x
Distance: 100.0
Distance: 100.0
Distance: 23.684939994835077
Distance: 20.667101077160623
Distance: 22.95818992855675
Distance: 23.078862977577955
Distance: 67.38712123910955
Distance: 100.0
```

Figure 16: Result

## 06: Motors Test

- This Python script makes two motors move forward and backward, using PWM to control their speed. It also uses an LED to show when the motors are running. The direction of each motor is managed using GPIO pins.

## 07: Motors PWM

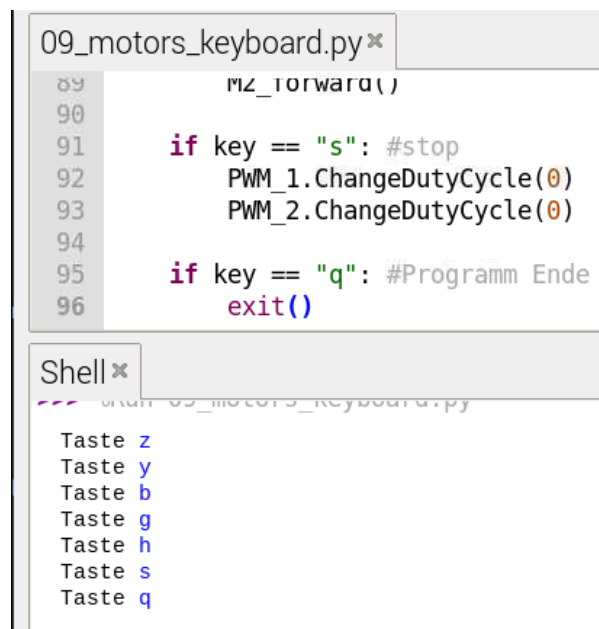
- This Python script runs two motors using PWM to adjust their speed and GPIO pins to set their direction. An LED lights up to show that the motors are active, and the motors move forward and backward during the program.

## 08: Cycle Drive

- This Python script manages two motors, using PWM to control their speed and GPIO pins for direction on a Raspberry Pi. Additionally, it toggles an LED (connected to GPIO pin 23) to indicate activity during operation.

## 09: Motors Keyboard

- This Python script controls two motors using a push button (connected to GPIO pin 5) and keyboard inputs. Different key presses are mapped to motor actions like moving forward, backward, left, right, and stopping. The motor speed is adjusted using PWM signals.



```
09_motors_keyboard.py
89         m2_forward()
90
91     if key == "s": #stop
92         PWM_1.ChangeDutyCycle(0)
93         PWM_2.ChangeDutyCycle(0)
94
95     if key == "q": #Programm Ende
96         exit()

Shell x
-- python 09_motors_keyboard.py
Taste z
Taste y
Taste b
Taste g
Taste h
Taste s
Taste q
```

Figure 17: Result



## 10: CNY70 Test

- This Python script reads data from two CNY70 sensors used for line tracking (left and right). It controls an LED based on the sensor readings: when a sensor detects a black line, it prints a message, and when it detects a white surface, it turns on the LED. Meanwhile, motor 1 keeps moving forward continuously.

```
Shell x
left sensor on black
right sensor on black
left sensor on black
right sensor on black
left sensor on black
right sensor on black
left sensor on black
right sensor on black
left sensor on black
```

Figure 18: Result

```
Shell x
left sensor on white
right sensor on white
left sensor on white
right sensor on white
left sensor on white
right sensor on white
left sensor on white
right sensor on white
left sensor on white
```

Figure 19: Result

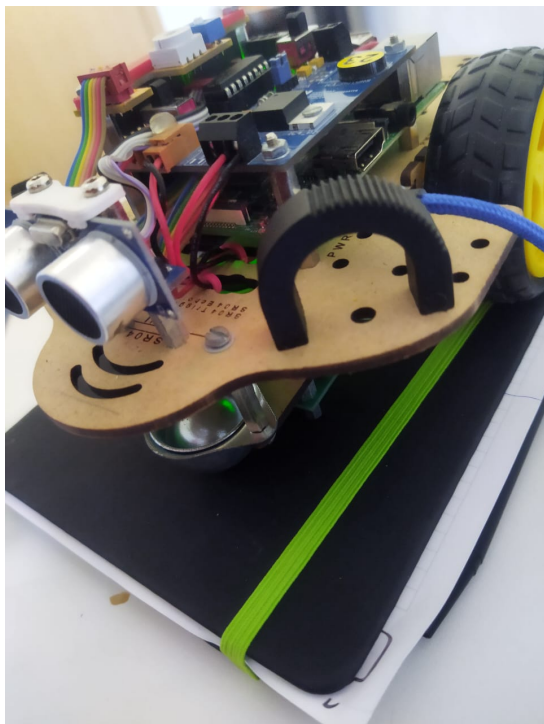


Figure 20: Black surface

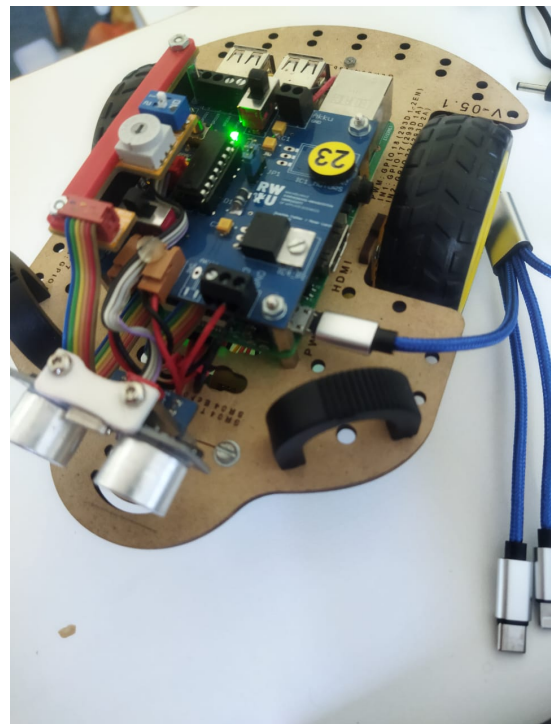
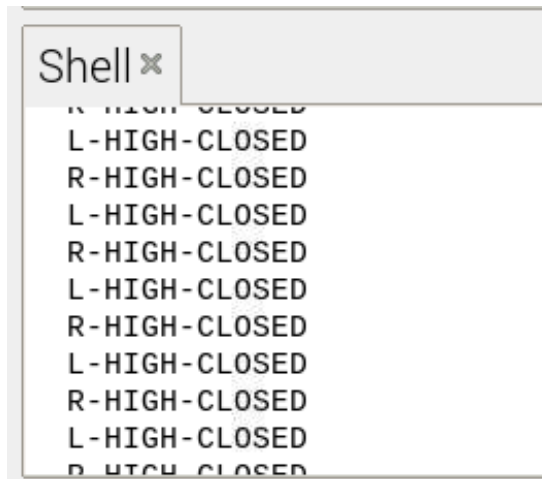


Figure 21: White surface

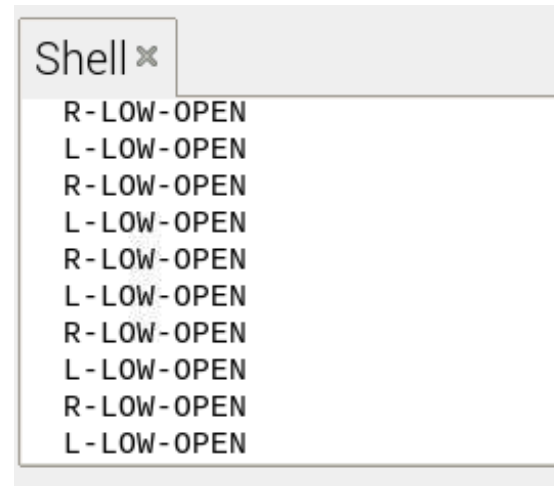
## 11: Distance Sensor Test

- This Python script reads data from two distance sensors connected to GPIO pins and prints messages based on their states. If a sensor reads HIGH, it is considered closed; if it reads LOW, it is considered open. The script continuously checks the sensor states, updating every second.



```
Shell x
R-HIGH-CLOSED
L-HIGH-CLOSED
R-HIGH-CLOSED
L-HIGH-CLOSED
R-HIGH-CLOSED
L-HIGH-CLOSED
R-HIGH-CLOSED
L-HIGH-CLOSED
R-HIGH-CLOSED
L-HIGH-CLOSED
R-HIGH-CLOSED
L-HIGH-CLOSED
R-HIGH-CLOSED
```

Figure 22: Result

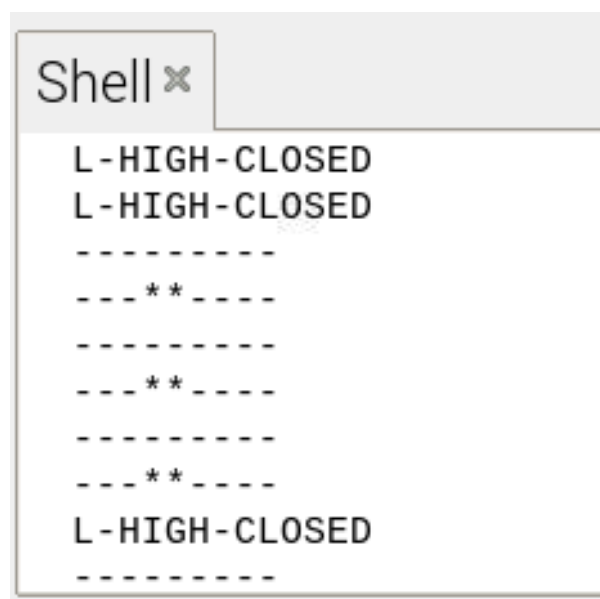


```
Shell x
R-LOW-OPEN
L-LOW-OPEN
R-LOW-OPEN
L-LOW-OPEN
R-LOW-OPEN
L-LOW-OPEN
R-LOW-OPEN
L-LOW-OPEN
R-LOW-OPEN
L-LOW-OPEN
R-LOW-OPEN
L-LOW-OPEN
R-LOW-OPEN
L-LOW-OPEN
```

Figure 23: Result

## 12: Distance Sensor Interrupt

- This Python script sets up event detection for two distance sensors. When a rising edge (a change from LOW to HIGH) is detected on either sensor pin, it triggers a callback function that prints a message indicating the sensor has closed.



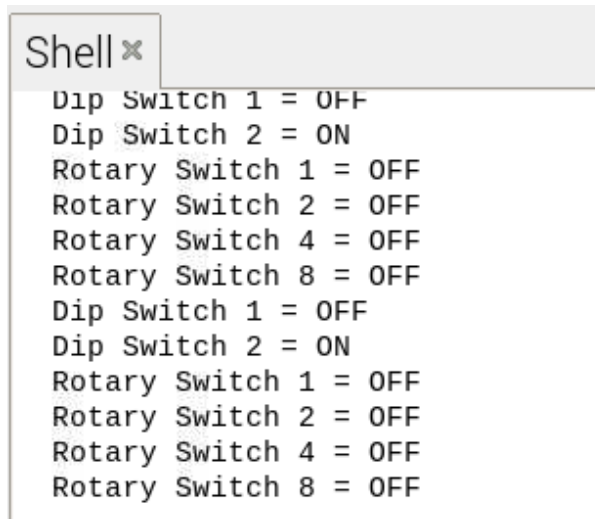
```
Shell x
L-HIGH-CLOSED
L-HIGH-CLOSED
-----
-- ** --
-----
-- ** --
-----
-- ** --
-----
L-HIGH-CLOSED
-----
```

Figure 24: Result

---

## 13: Dip Rotary Switches

- This Python script continuously checks the position of dip rotary switches every 0.5 seconds and prints their status as either ON or OFF. It runs in a loop, updating the status regularly.

A terminal window titled "Shell" with a close button (X) in the top-left corner. The window displays the output of a Python script that checks the status of dip and rotary switches. The output is as follows:

```
Dip Switch 1 = OFF
Dip Switch 2 = ON
Rotary Switch 1 = OFF
Rotary Switch 2 = OFF
Rotary Switch 4 = OFF
Rotary Switch 8 = OFF
Dip Switch 1 = OFF
Dip Switch 2 = ON
Rotary Switch 1 = OFF
Rotary Switch 2 = OFF
Rotary Switch 4 = OFF
Rotary Switch 8 = OFF
```

Figure 25: Result

---

**d) Review the Tinkercad Simulation of the H-Bridge that you did and find the correct output pins on the ET1 to use in order to get a similar behaviour.**

- Motor1 PWM = 18
- Motor1 IN1 = 17
- Motor1 IN2 = 22
- Motor2 PWM = 19
- Motor2 IN1 = 24
- Motor2 IN2 = 4