LABORATORY 2: Raspberry I/O

Pre-requisite: Raspbian OS

Part 1: Digital output in Raspberry

Theory Concepts:

The raspberry Pi is a SoC that can be used to generate a lot of systems. For that reason, is important to understand its working and learn how to handle de Inputs and outputs of this system.

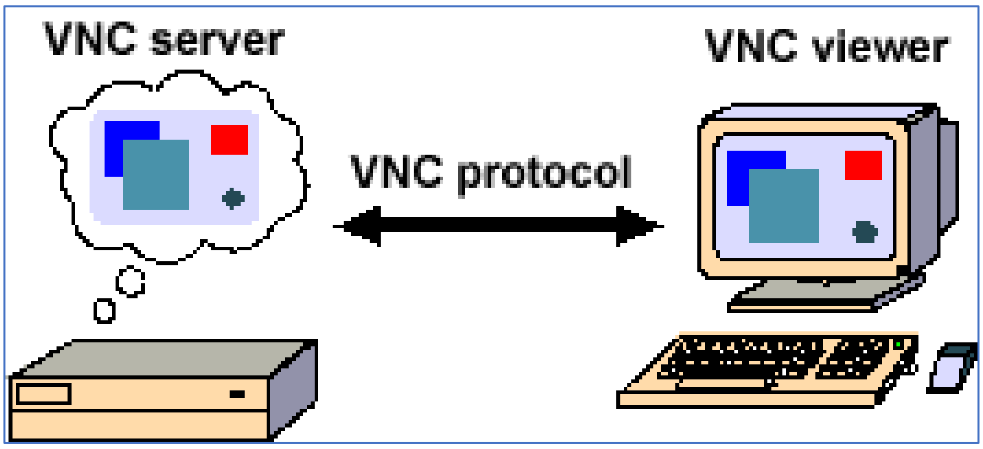
Raspberry pi will require a screen to visualize all the work done. In this laboratory we will learn how to use our laptops as raspberry monitor and work using VNC communication protocol.

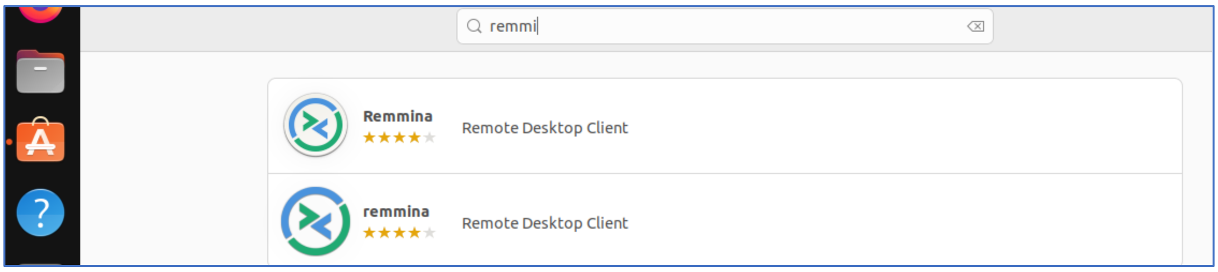
VNC Installation:

What is VNC?

Virtual Network Computing is a graphical desktop-sharing system used to control remotely computers. This system will allow users to visualize the graphic interface of a specific system from another source.

- To use VNC viewer we Will need to know the user, password and ip from our raspberry pi.
- We can change the password using the passwd command in terminal or in raspberry pi preferences.





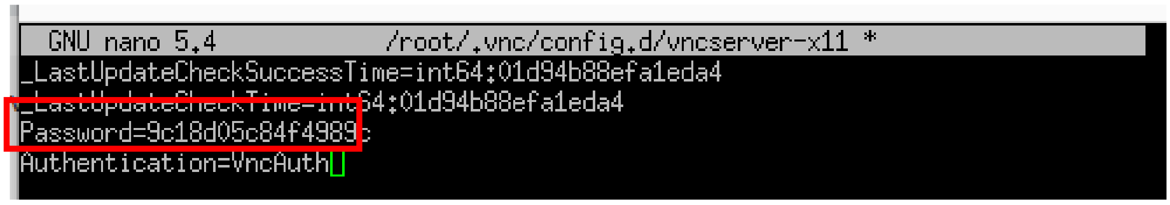
Install Remmina client in Ubuntu laptop.

In the raspberry:

- Enable VNC
- Activate VNC password with the command sudo vncpasswd -service
- Access using nano to the vnc server config:

  sudo nano /root/.vnc/config.d/vncserver-x11
- Add the following info at the end of the file:

  Authentication=VncAuth

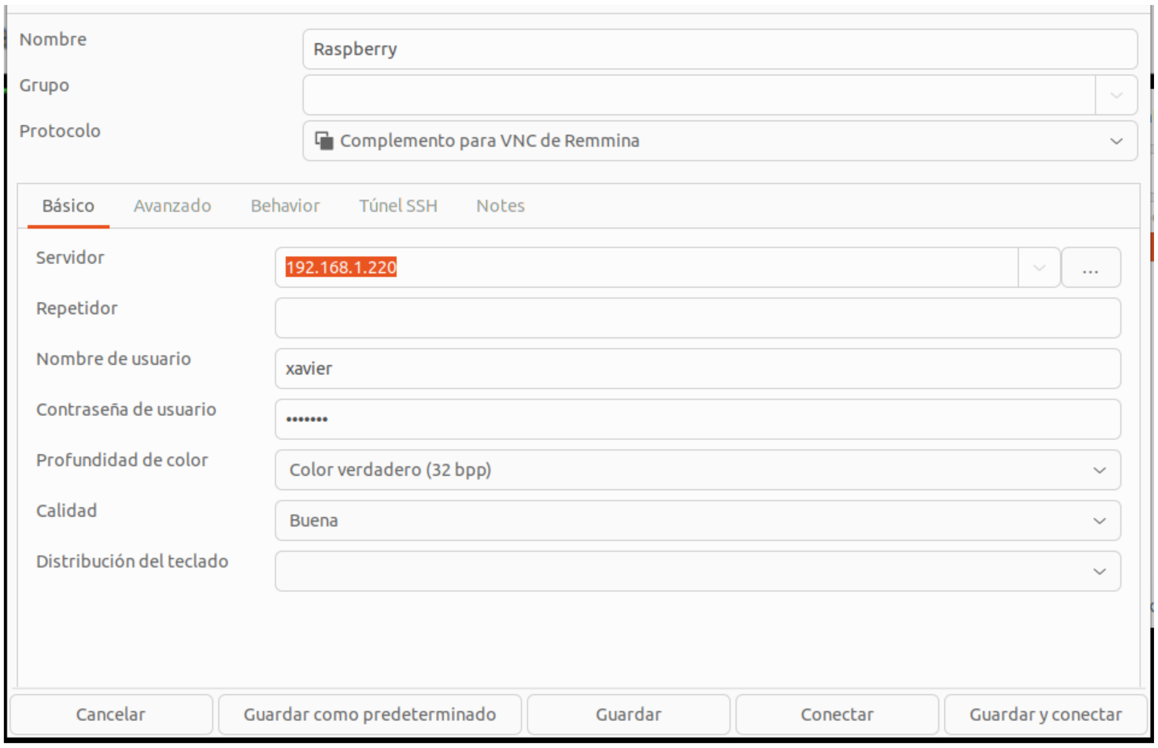


```
  GNU nano 5.4              /root/.vnc/config.d/vncserver-x11 *
_LastUpdateCheckSuccessTime=int64:01d94b88efa1eda4
_LastUpdateCheckTime=int64:01d94b88efa1eda4
Password=9c18d05c84f4989c
Authentication=VncAuth
```

- Restart VNC using:

  sudo systemctl restart vncserver-x11-serviced

Finally, we must open and configure Remmina:



Nombre	Raspberry
Grupo	
Protocolo	Complemento para VNC de Remmina

Básico · Avanzado · Behavior · Túnel SSH · Notes

Servidor	192.168.1.220
Repetidor	
Nombre de usuario	xavier
Contraseña de usuario	•••••••
Profundidad de color	Color verdadero (32 bpp)
Calidad	Buena
Distribución del teclado	

We will select VNC complement, put the ip from the raspberry, the user, and the password.

As an alternative, **VNC Viewer** can be used in Ubuntu. It is an easy use software that will provide similar features to Remmina without too much configuration.

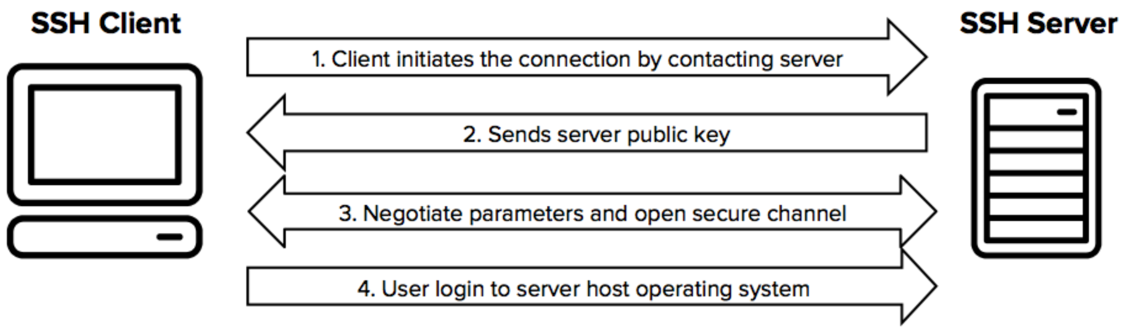
To install VNC Viewer you can follow the next link:

https://www.realvnc.com/en/connect/download/viewer/linux/?lai_sr=10-14&lai_sl=l

Or alternatively you can download it in your Ubuntu OS using the Ubuntu store.

SSH Communication:

Ssh needs a hostname/ip, port, user credential and password.


SSH Client
SSH Server
1. Client initiates the connection by contacting server
2. Sends server public key
3. Negotiate parameters and open secure channel
4. User login to server host operating system

To use SSH we can follow the next commands.

- hostname –I
- ssh [username]@[ip address]

¿Which port is used by default?

Raspberry will use the GPIO module to generate outputs and inputs depending on the code we generate. Some important concepts to consider are:

- GPIO.setwarnings(False) will disable the GPIO warnings, allowing us to work without initial inconvenient.
- GPIO.setmode() will configure the gpio numbering. How many modes it has and what are their names?
- GPIO.setup(pin, GPIO.type) This method will defy a pin number or definition as output or input. How can we use this method?
- How can we make our code work infinitely? What differences exist between for and while?
- GPIO.output(pin, GPIO.value) Generates an output value, this can be HIGH or LOW.
- GPIO.input(pin_to_read) Reads the value of a gpio input.

Let’s try a practice example:

```
# Pin definitions
led_pin = 12

# Suppress warnings
GPIO.setwarnings(False)

# Use "GPIO" pin numbering
GPIO.setmode(GPIO.BCM)

# Set LED pin as output
GPIO.setup(led_pin, GPIO.OUT)

# Blink forever
while True:
    GPIO.output(led_pin, GPIO.HIGH) # Turn LED on
    time.sleep(1)                   # Delay for 1 second
    GPIO.output(led_pin, GPIO.LOW)  # Turn LED off
    time.sleep(1)                   # Delay for 1 second
```

How can we implement a button to the previous code?

Exercise 1. Using a button and 2 LEDs develop a system that can modify the LED status. The system should:

- State 1: Blink both LEDs alternating the sequence with 1 second intervals (when LED 1 is turned on LED 2 is turned off).
- State 2: Both LEDs have to blink simultaneously with 2 seconds intervals.
- State 3: In this state both LEDS have to be turned ON indefinitely.
- State 4: This state turns off both LEDs.

The system should return to state 1 after state 4.

Modify the code developed to work using functional programming.

Exercise 2. Using 4 LEDs and 2 buttons make a binary counter with the values from table 1. The counter should have the following features:

- When button 1 is pushed, the binary counter increases 1 value.
- When button 2 is pressed, the binary counter decreases 1 value.
- Each led represent the binary values.
- When the counter reaches the value 15 it should return to the first value
- When the counter is 0 it shouldn´t keep reducing the value (should stay in 0 after the button press)

Decimal	Binary	Hex
0	0000	0
1	0001	1

2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

Table 1. Decimal, hex and binary convention values.

Finally print in console the value in 3 representations (hex, binary, decimal).

Modify the code developed to work using OOP programming.

Exercise 3. We have a greenhouse that needs to cultivate different types of vegetables. After some studies the team determined that the optimal temperature for a good crop goes from 12 to 20 °C. To improve the greenhouse the team decided to develop a temperature control system with the following features:

- Turn on the heater if the temperature register is below 12 °C.
- Turn on the fan when the temperature goes above 20 °C.
- Turn off both devices if the temperature is optimal.

For this system we will use a red LED to simulate the heater and a small fan. Initially the temperature must be generated by a random number. After that replace the random number generator for a DHT22 sensor for temperature measurement.

Exercise 4. Generate a program that will turn on 1 of 4 possible LEDs. To achieve this the system will have 2 buttons, the first one will select the LED to turn on and the second one will increase 1 second to the time the LED is turned on. When the first button is pressed the LED to activate will change to the next one and the time to turn on will reset to 1 second.

Implement functional programming.