WELCOME TO ORION RING & RUN STEM CAMP (SUMMER 2024)

RING & RUN STEM EVENT.

ORION
OPEN ARCHITECTURE RESILIENT IOT FOR OPERATIONAL NETWORKS

Objective: Build a “Smart” Doorbell from Scratch…

- That will…
  - Display streaming video in a Web App when the doorbell button is pressed, or when motion is detected
    - Play a customizable doorbell chime
  - Provide an intercom system to allow remote communication between the App and the person at the door
  - Use Advanced Artificial Intelligence to describe ”who” or “what” is at the door
  - Ensure secure/encrypted communication between the Doorbell and the Web App.
  - Whatever else you can make it do…
- Using…
  - Raspberry Pi 4B
  - Electronic components the make up the doorbell
  - OpenAI Advanced Artificial Intelligence
  - Software that you develop to make it all work

Goals and Outcomes : Learn, Get Inspired, and Get Involved

- ✓Learn about advanced technology…
  - ➢ Raspberry Pi single board computer (SBC) and common uses
    - The basics of the Linux operating system
    - Basic fundamentals of the electronic device prototyping
    - Internet of Things (IoT) Systems
  - ➢ Explore/discover programming concepts
    - Python, JavaScript, CSS (Cascading Style Sheets), and HTML 5
  - ➢ Discover network protocols and methods for ensuring computer/network security
    - TCP/IP, HTTP, HTTPS, MQTT, PKI, TLS, etc.
  - ➢ Exposure to the latest OpenAI Chat GPT (Artificial intelligence) services and models
- ✓Get inspired, discover your talents, perhaps go to university and pursue a high technology career.
- ✓Do Great things!
- ✓ Enjoy what you do!

IoT enabled “Smart” Doorbell Concept

CA cert

Server cert

verify

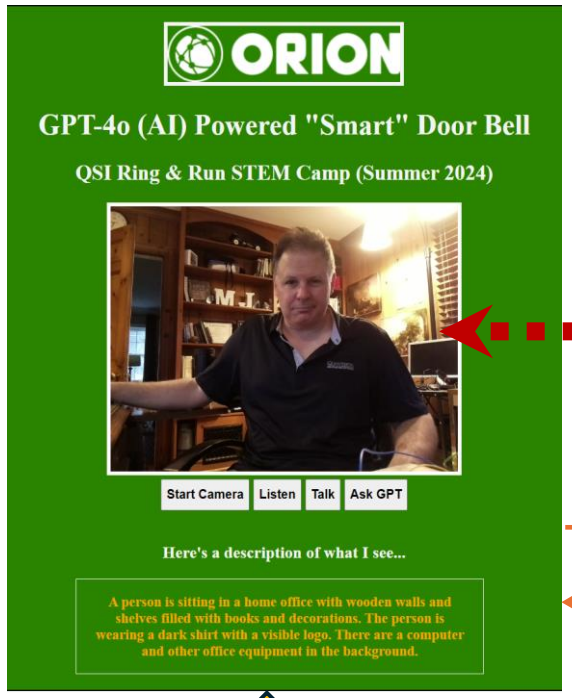
Client App
(Front-End)

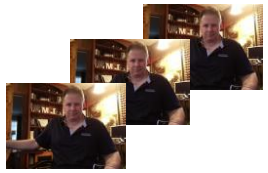
Web app: (based on HTML5 CSS and JavaScript)

Supported modes:
motion and manual (button press)

Port 8001

Python Back-end





HTTPS MJPG video stream

Publish to topic: *ring/#*

device command set

Subscribe to topic: *ring/#*

Publish to topic: *ring/#*

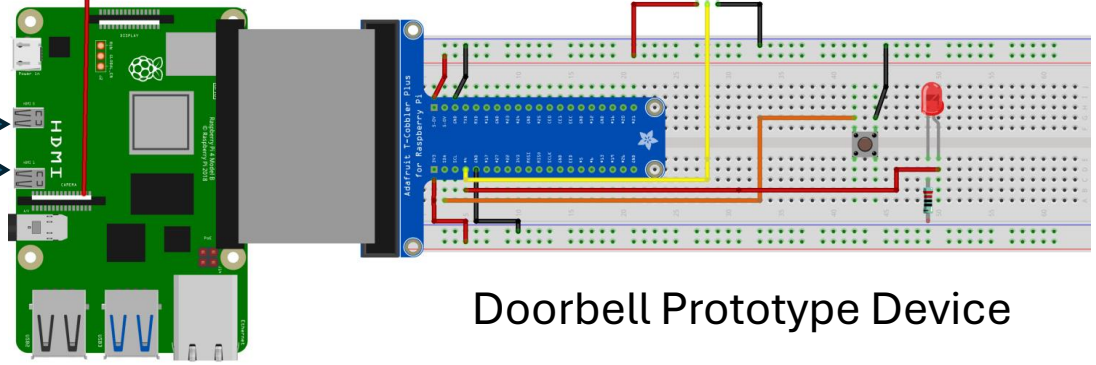
device command set

Subscribe to topic: *ring/#*

Mosquitto (MQTT)Broker

MQTT over Web Socket's

Port 1883, 8883



Raspberry Pi 4B

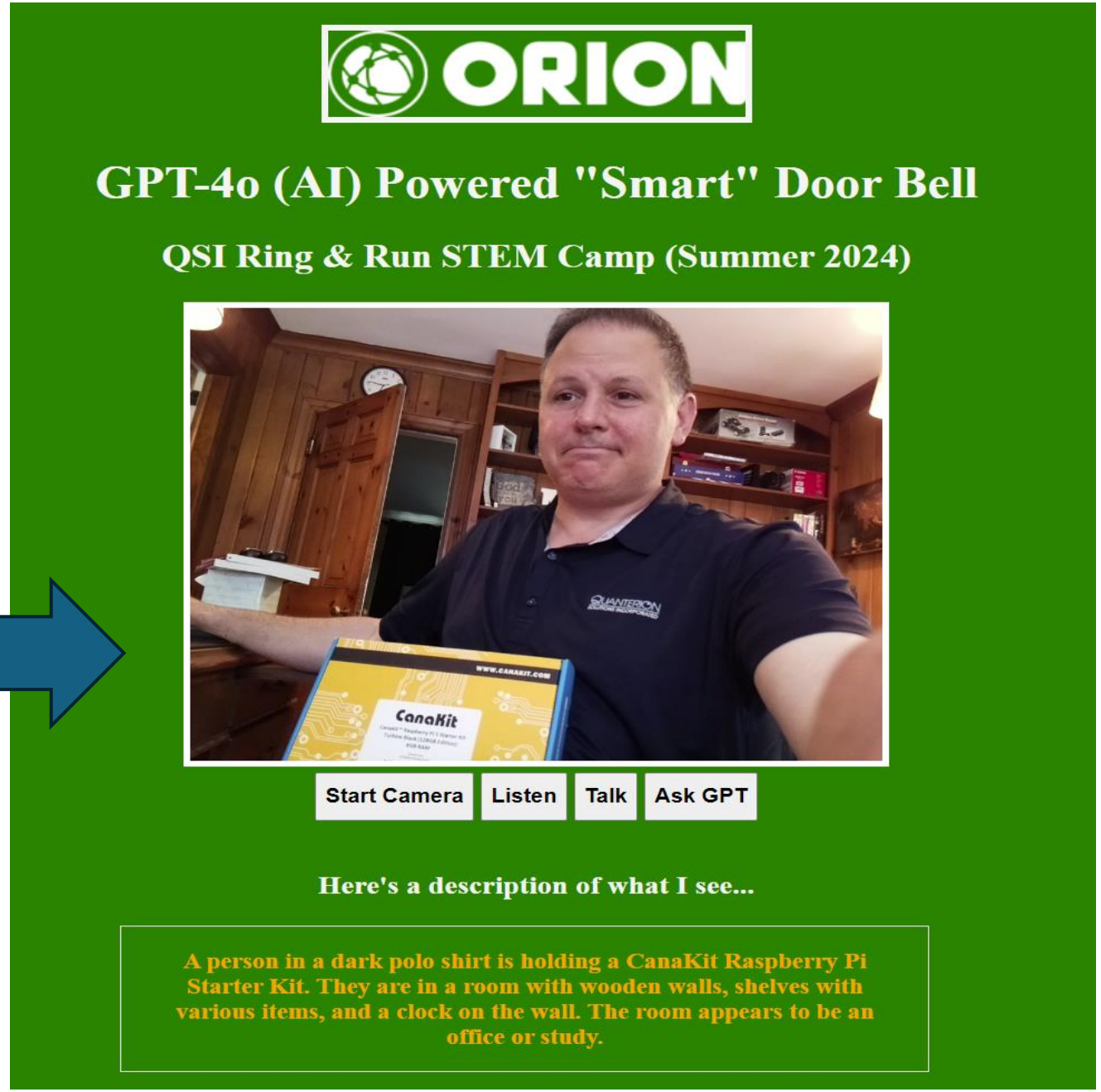
Doorbell Prototype Device

fritzing

OpenAI

https://openai.com/

Completed “Smart” Doorbell with “Smarter” answers!!



Project Resources

- GitHub Project Repo:
  - https://github.com/quanterion-solutions-inc/ORIONSmartDoorBell

Schedule/Project Breakdown

Module 1: (Day 1): Provision The Raspberry Pi for Use

- Write the Raspberry Pi OS (operating system) Image and configure the services
- Spend time getting familiar with the Pi and Linux environment

Module 2: (Day 1): Build the Doorbell device

Module 3: (Day 1 and 2): Implement Doorbell Software

- Implement the Software (front-end web application and server back-end)
- Use the standard IoT communication protocol (MQTT) to interact with and control the Doorbell device

Module 4: (Day 2): Implement PKI SSL/TLS (Transport Layer Security) into the Doorbell Design

- Discuss and implement public key infrastructure.
- Generate x509 standard certificates to establish verified and encrypted between clients and servers

Module 5: (Day 2): Making the Doorbell “Smart”

- Discuss the latest OpenAI Chat GPT (Artificial intelligence) services and models, and how integrate these services into the Doorbell Project

Module 6: (Day 2): Time Permitting) Discuss User Authentication/Authorization

- Consider authentication with the Mosquitto MQTT Broker

Module 7: (Day 2 and 3): Customize the Doorbell in anyway you wish

- Change colors, themes
- Experiment with new/missing features of interest etc.
- Each group 15-20 minute discussed of what they learned, found most interested, and present what you decide to do in Module 6

Drawing and Wrap up (Day 3)

Day 1: Tuesday (7/23)

9:45-12:00 : "Smart" IoT Doorbell Prototype Introduction

- **9:45-10:45** : **Module 1: Provisioning the Raspberry Pi for Use**
- **10:45-11:00** : Break (restrooms, stretch, questions)
- **11:00-12:00** : **Module 2: Construct the Device**

12:00-1:00PM : Lunch and Demo

- **12:30 – 1:00** : Special Demo

1:00-2:00 : Module 3: Implementing the Software (Front-end and Back-end Code)

Day 2: Wednesday (7/24)

9:00-11:00: Module 3 and Module 4

10:45-11:00 : Break (restrooms, ready for tour)

11:00-12:30: Innovare Tour

12:30-1:00PM : Lunch

1:00-2:45 : **Module 4 and 5**

Day 3: Thursday (7/25)

9:00-11:00: Module 5

11-11:15 : Break (restrooms, questions)

11:15 – 12:00 : Module 6

12:00-12:45pm: (Lunch/Demo: VICEROY Team)

12:45-1:45 : Module 7

1:45 – 2:45 (10-minute group presentations, 5 groups)

2:45-3:00pm – drawing and wrap up

RASPBERRY PI DOOR PRIZE!

PI DAY STEM EVENT.

ORION
OPEN ARCHITECTURE RESILIENT IOT
FOR OPERATIONAL NETWORKS

Drawing (Day 3): Thursday 2:30

- All camper's names entered in a drawing for chance to win the latest Raspberry Pi 5

Raspberry Pi 5 Starter Kit PRO - Turbine Black (128GB Edition) (8GB RAM)

ICEBREAKER

RING & RUN STEM EVENT.

ORION

OPEN ARCHITECTURE RESILIENT IOT FOR OPERATIONAL NETWORKS

Icebreaker

- Pick a team name (Air Force Aircraft Theme)
  - F-15 Eagle
  - F-18 Hornet
  - F-22 Raptor
  - F-35 Lightning
  - A-10 Warthog
  - B-21 Raider,
  - B-52 Stratofortress
- Pick a team representative
- Question: Nest vs. Ring doorbell?

Docs: https://github.com/quanterion-solutions-inc/doorbelldocs




HANDS-ON PROJECT

RING & RUN STEM EVENT.

ORION

OPEN ARCHITECTURE RESILIENT IOT FOR OPERATIONAL NETWORKS

IoT enabled “Smart” Doorbell Concept

CA cert

Server cert

Client App
(Front-End)

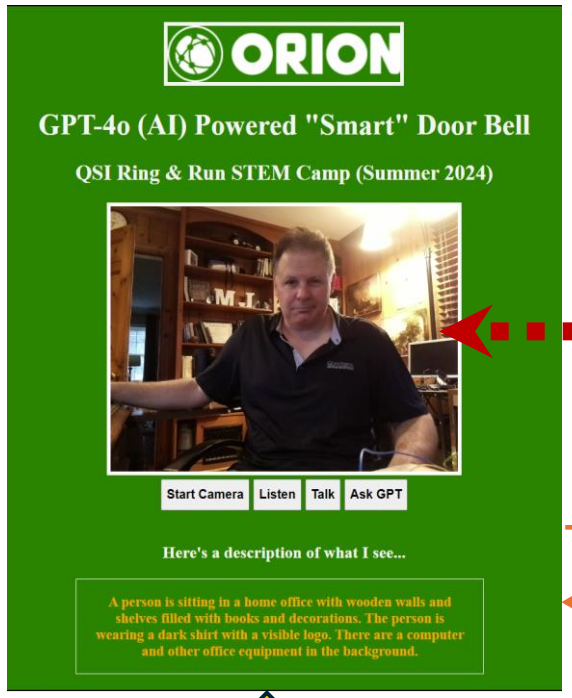
verify

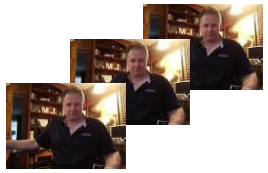
Web app: (based on HTML5 CSS and JavaScript)

Port 8001

Python Back-end

Supported modes:
motion and manual (button press)





HTTPS MJPG video stream

Publish to topic: *ring/#*

device command set

Subscribe to topic: *ring/#*

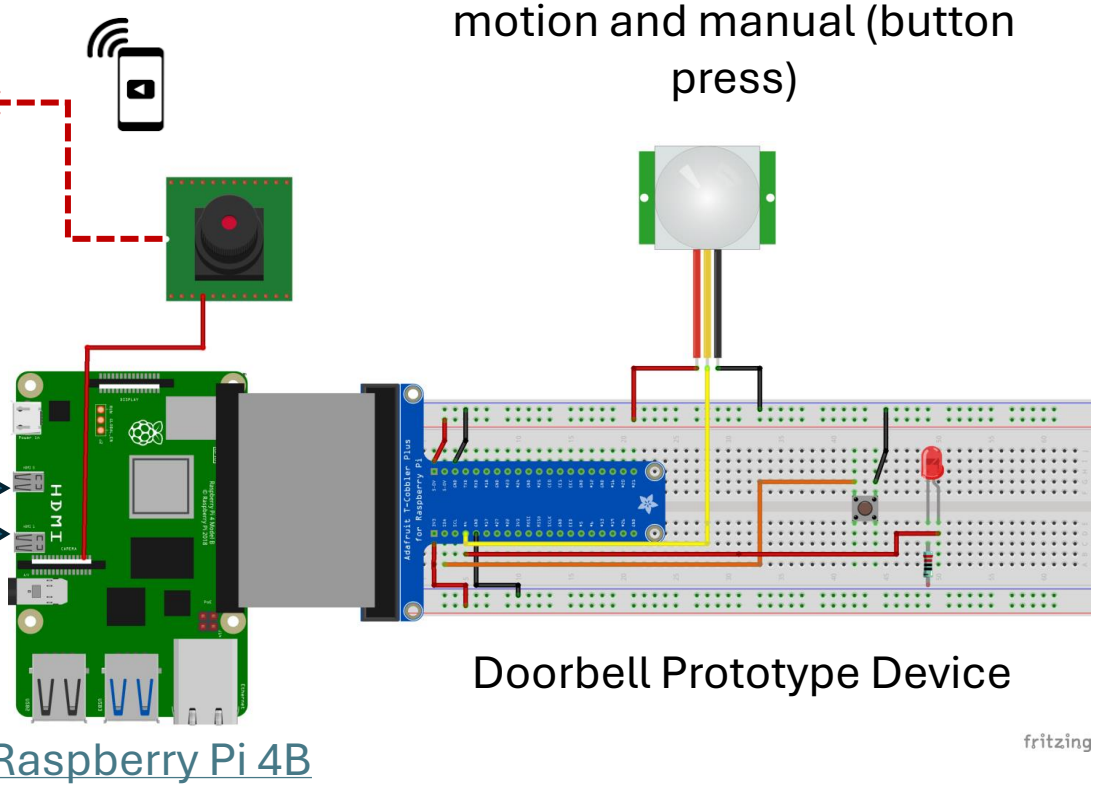
Publish to topic: *ring/#*

device command set

Subscribe to topic: *ring/#*

Mosquitto (MQTT)Broker

MQTT over Web Socket's



Raspberry Pi 4B

Doorbell Prototype Device

fritzing

OpenAI

https://openai.com/

BACKGROUND AND ENABLING TECHNOLOGIES

RING & RUN STEM EVENT.

ORION
OPEN ARCHITECTURE RESILIENT IOT FOR OPERATIONAL NETWORKS

Background and Enabling Technologies

- What is a Raspberry Pi?
  - https://www.youtube.com/watch?v=eZ74x6dVYes
    - Board version comparison:
      - https://socialcompare.com/en/comparison/raspberrypi-models-comparison

      - Common Uses …?
        - Home , “Maker culture”
          - https://www.raspberrypi.com/for-home/
          - https://www.tomshardware.com/features/best-raspberry-pi-projects
        - Education
          - https://www.raspberrypi.org/app/uploads/2018/08/Raspberry-Pi-Computers-in-Schools-2018.pdf
          - https://www.raspberrypi.org/teach
        - Industry
          - https://www.raspberrypi.com/for-industry/
            - https://www.raspberrypi.com/for-industry/space/
            - industrial automation
              - applications for prototyping
              - embedded systems
            - low-cost process controller
        - Air Force: https://www.newsweek.com/artificial-intelligence-raspberry-pi-pilot-ai-475291

https://github.com/quanterion-solutions-inc/doorbelldocs

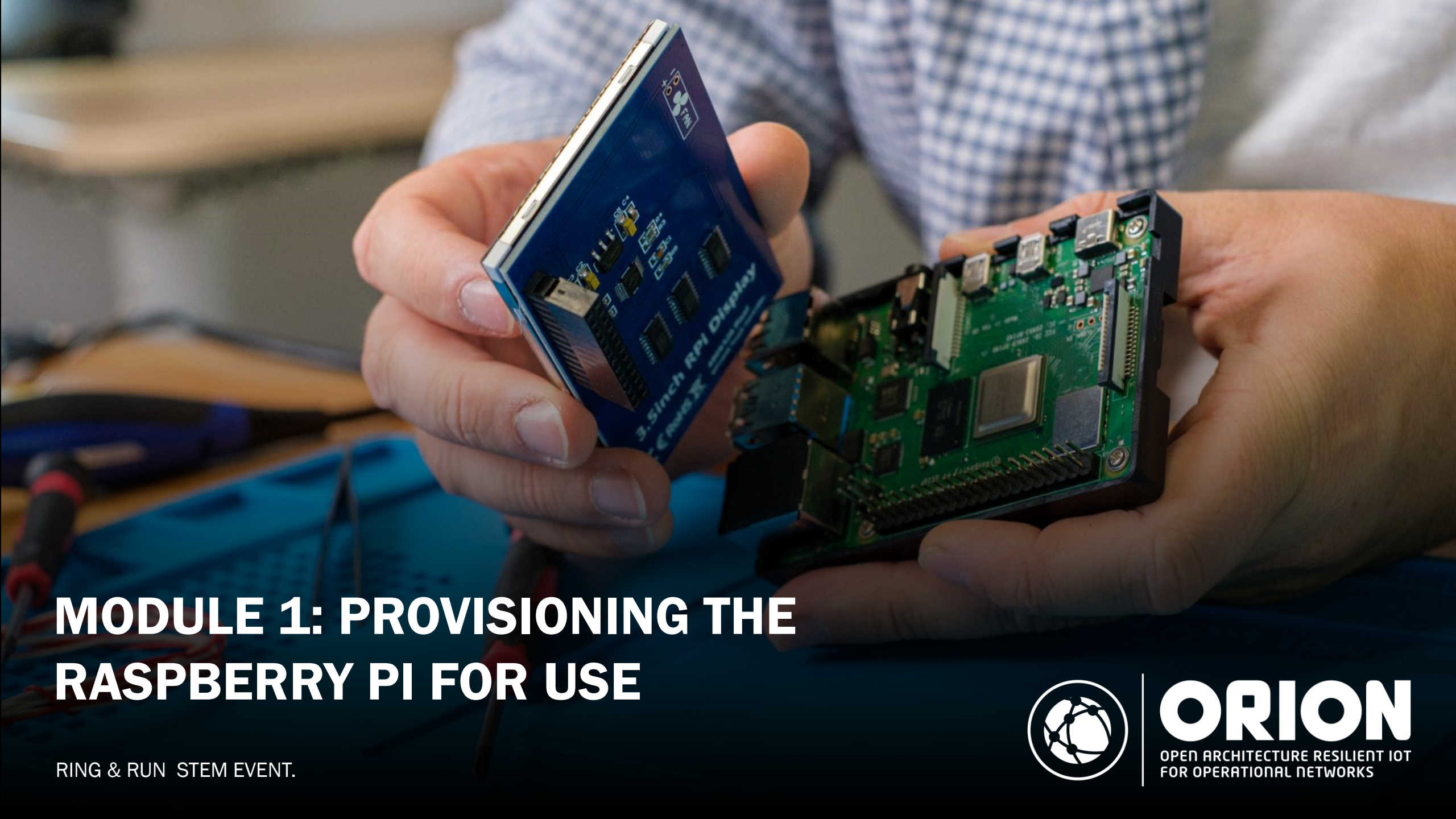
Background and Enabling Technologies

- Raspberry Pi versus Arduino
  - https://www.youtube.com/watch?v=p40OetppIDg
    - https://webbylab.com/blog/arduino-vs-raspberry-pi-key-differences-comparison-table/

- Raspberry Pi 5 (latest version)
    - https://www.raspberrypi.com/news/introducing-raspberry-pi-5/

- What is Internet-of-things (IoT),
  - https://www.youtube.com/watch?v=uEsKZGOxNKw
  - https://www.youtube.com/watch?v=6mBO2vqLv38

- Technology stacks: Python, JavaScript, HTML5,CSS
  - Network communication Protocols: TCP/IP, HTTPS, MQTT, etc.



MODULE 1: PROVISIONING THE RASPBERRY PI FOR USE

RING & RUN STEM EVENT.

ORION
OPEN ARCHITECTURE RESILIENT IOT FOR OPERATIONAL NETWORKS

Module 1: Provisioning the Raspberry Pi for Use

- Write the operating system image: Raspberry Pi OS
  1. Configure services
     - Secure Shell (SSH)
       - raspi-config
         - I2C
         - Virtual Network Computing (VNC)

  2. Linux Primer
     - Short “Linux” command tutorial
       - Take some time to get familiar interacting with Raspberry Pi

Install and Configure the Operating System (OS) and Services

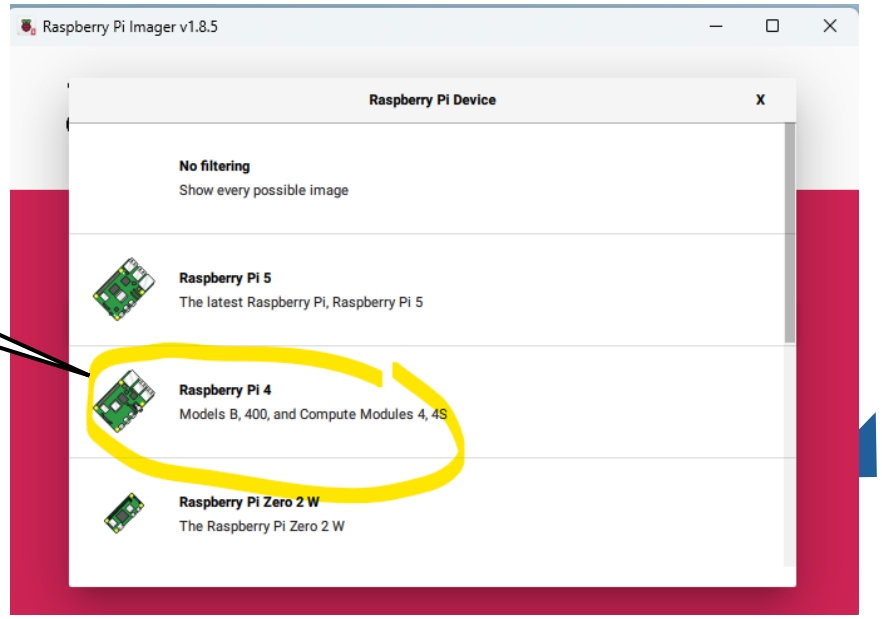
1. **Download and Install Raspberry Pi Imager on Laptop**
   - **Select “Download for Windows”**
     - https://www.raspberrypi.com/software/

2. **Insert micro-SD card into laptop USB card reader and run the imager tool. See screen shots below**

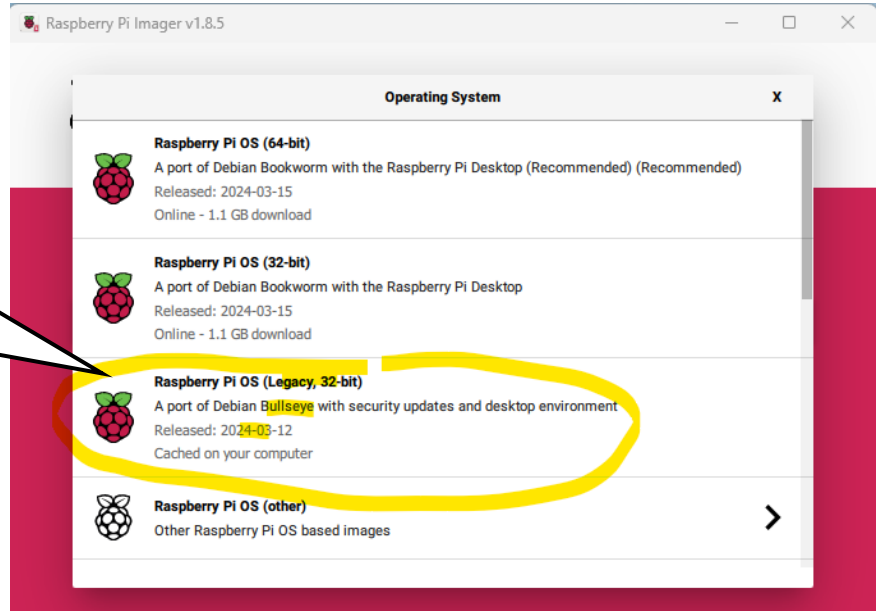
Note: the SD card will only one direction (don’t force it)

Select and Write the Operating System (OS) Image to the SD card

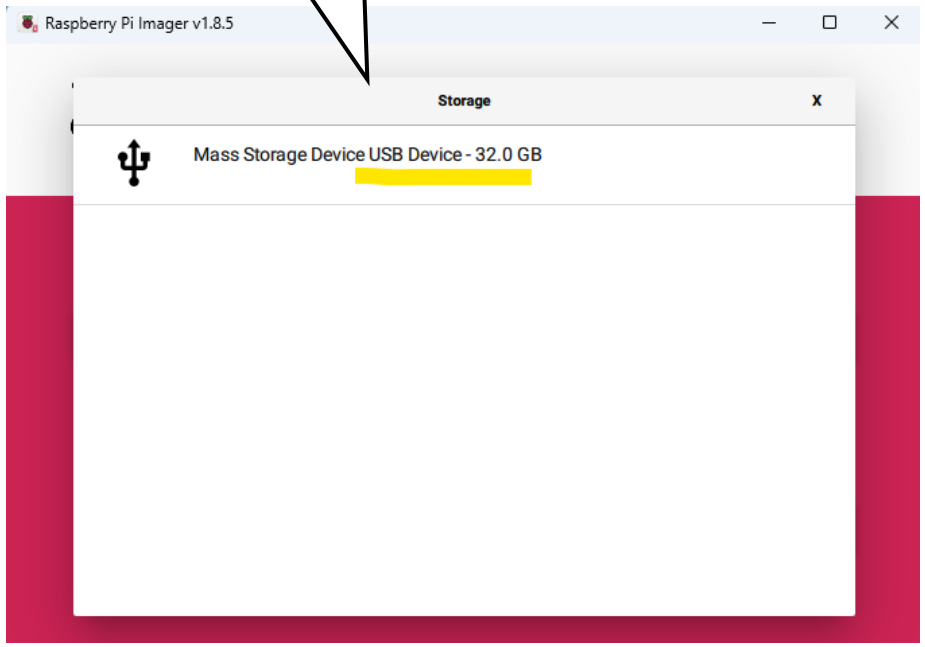
1. **Raspberry Pi device:** select your version **(Pi 4B)**



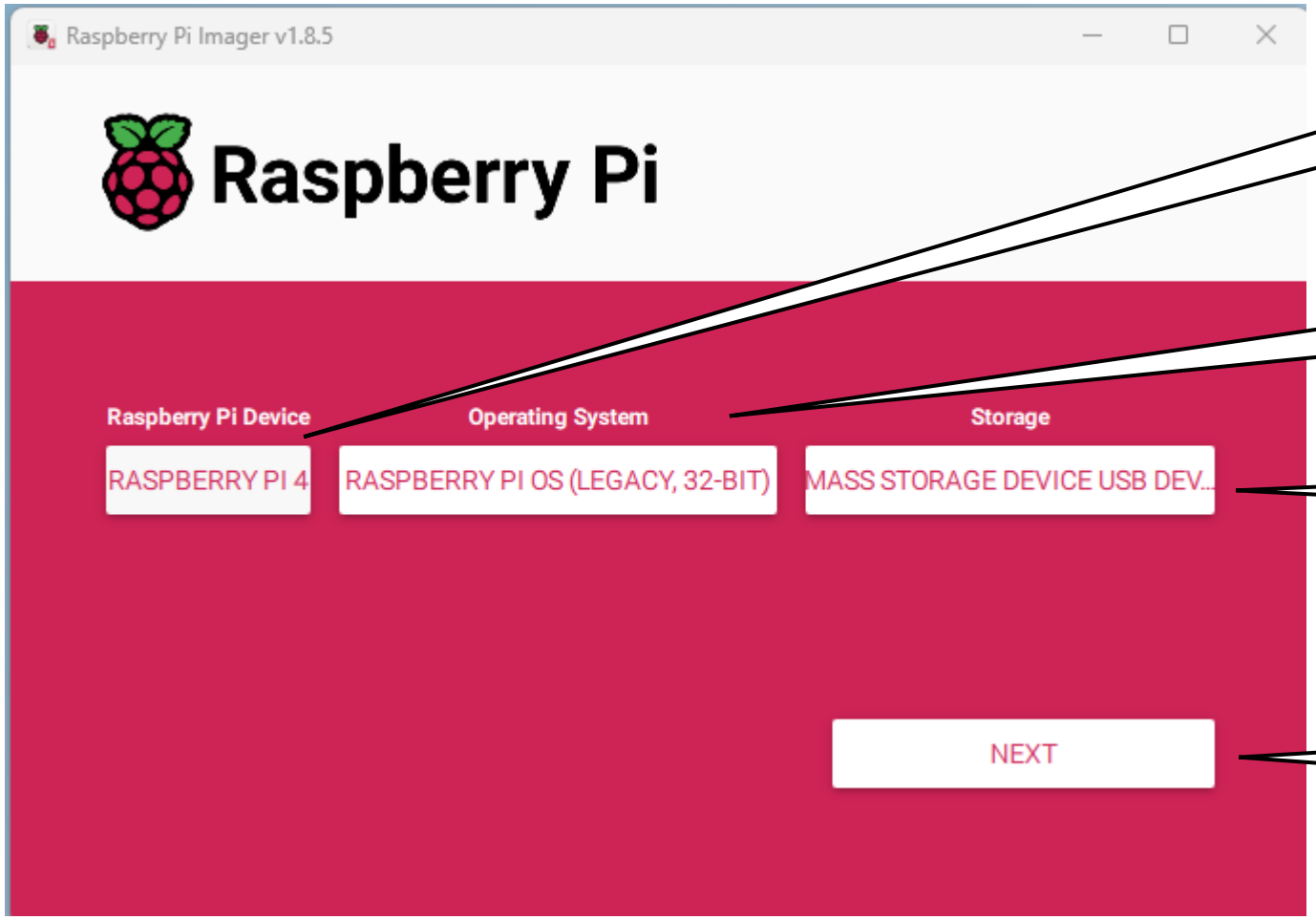
2. **Choose OS**: Select Pi OS (**32 bit (Bullseye)** with desktop env.



3. **Storage location**: Select the micro- SD card from USB reader



Before you proceed, Make sure....



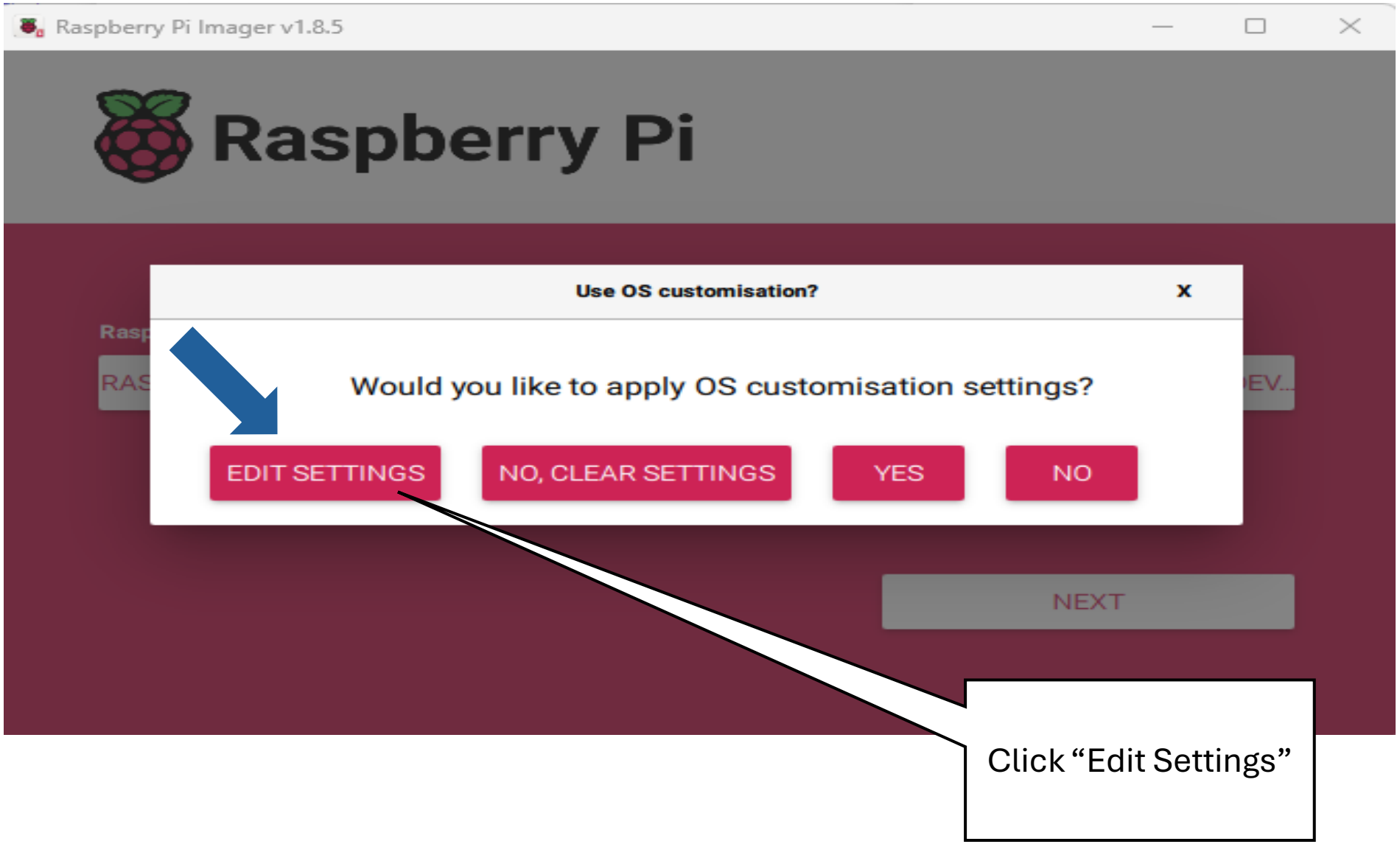
1. Raspberry Pi 4 is selected

2. 32-bit Legacy OS w/ desktop environment

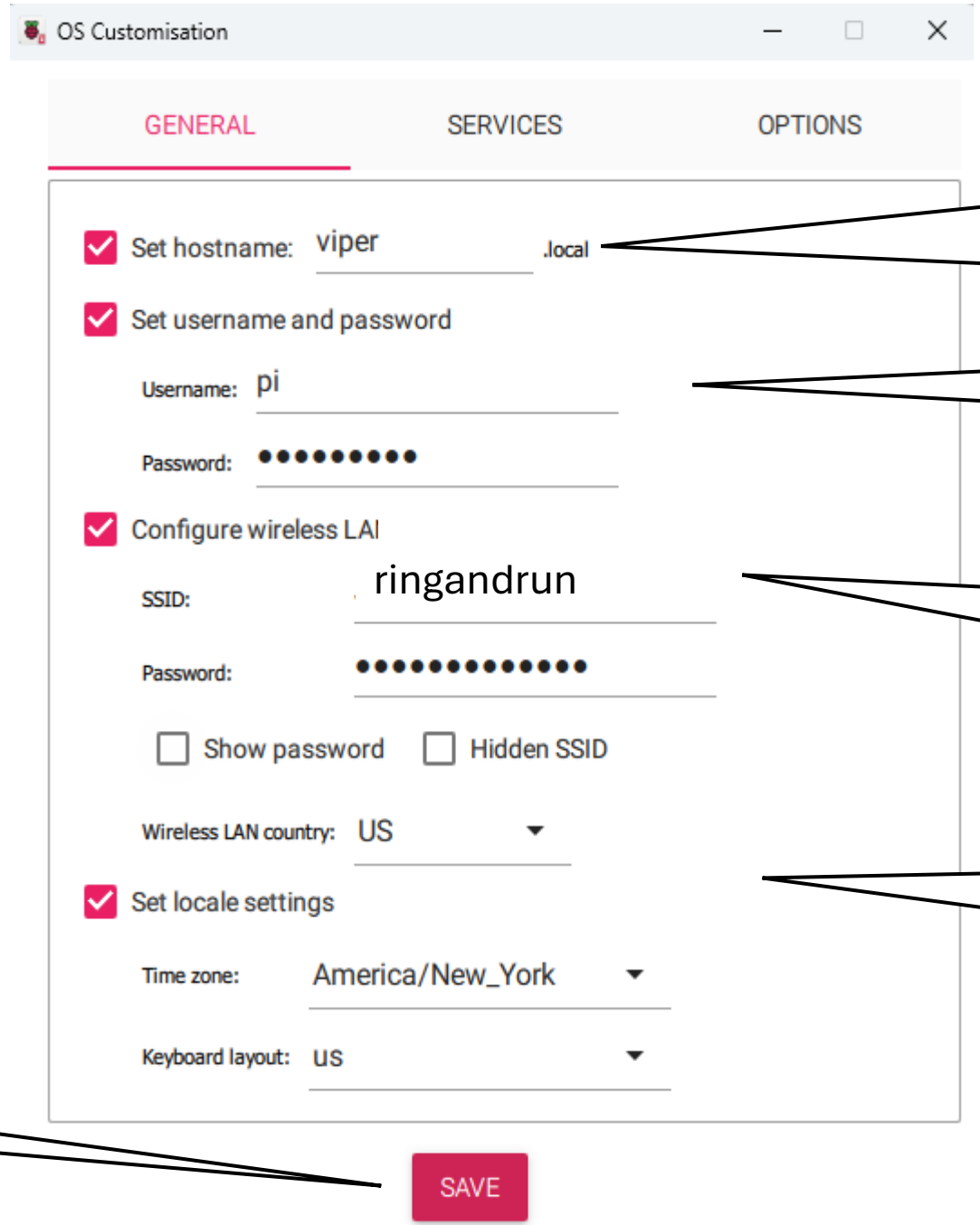
3. Make sure the micro-SD card is selected as the storage

* Only when 1,2, and 3 are verified, click “next”

Customize the Settings:



These settings must be as specified, or your Pi will not be properly configured for the project



Specify the hostname. Use your Team name

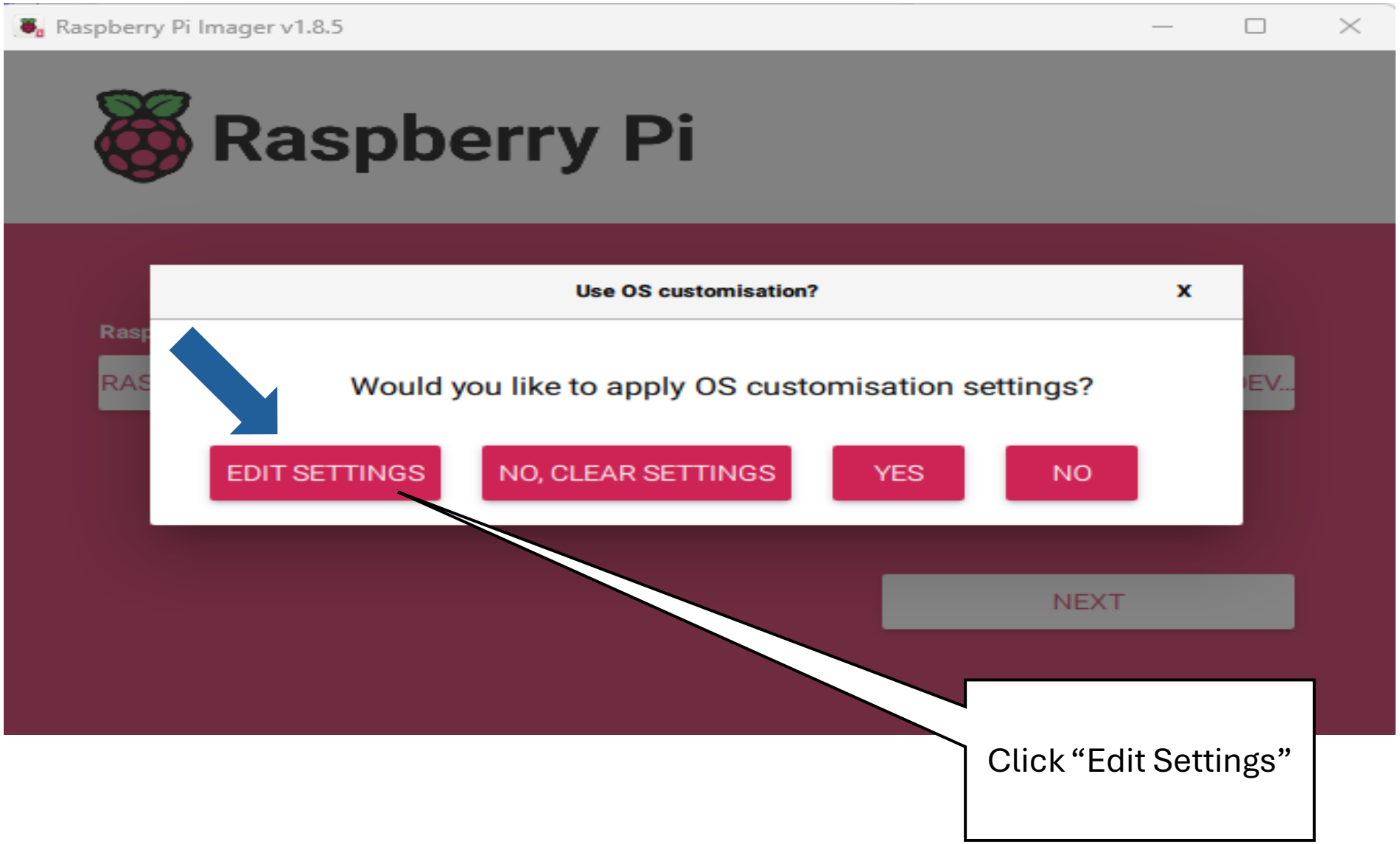
Set the username to “pi” and the Password to “team chooses” (***make it simple to remember***)

Configure the WIFI settings to use the Guest network.

Select the country to ‘US’ and set the locale settings as illustrated

Click “Save”

Customize the Settings:

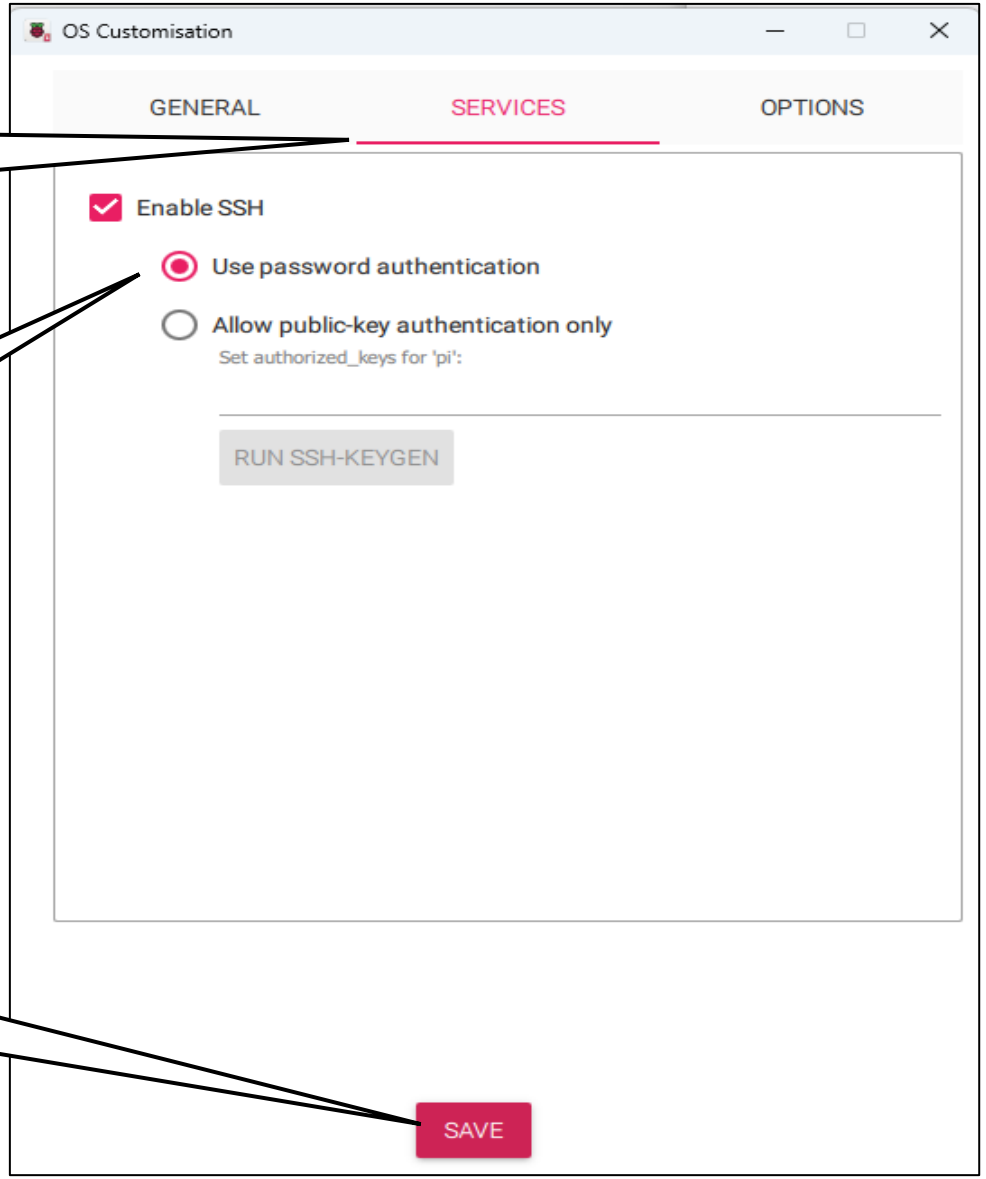


Enable Secure Shell

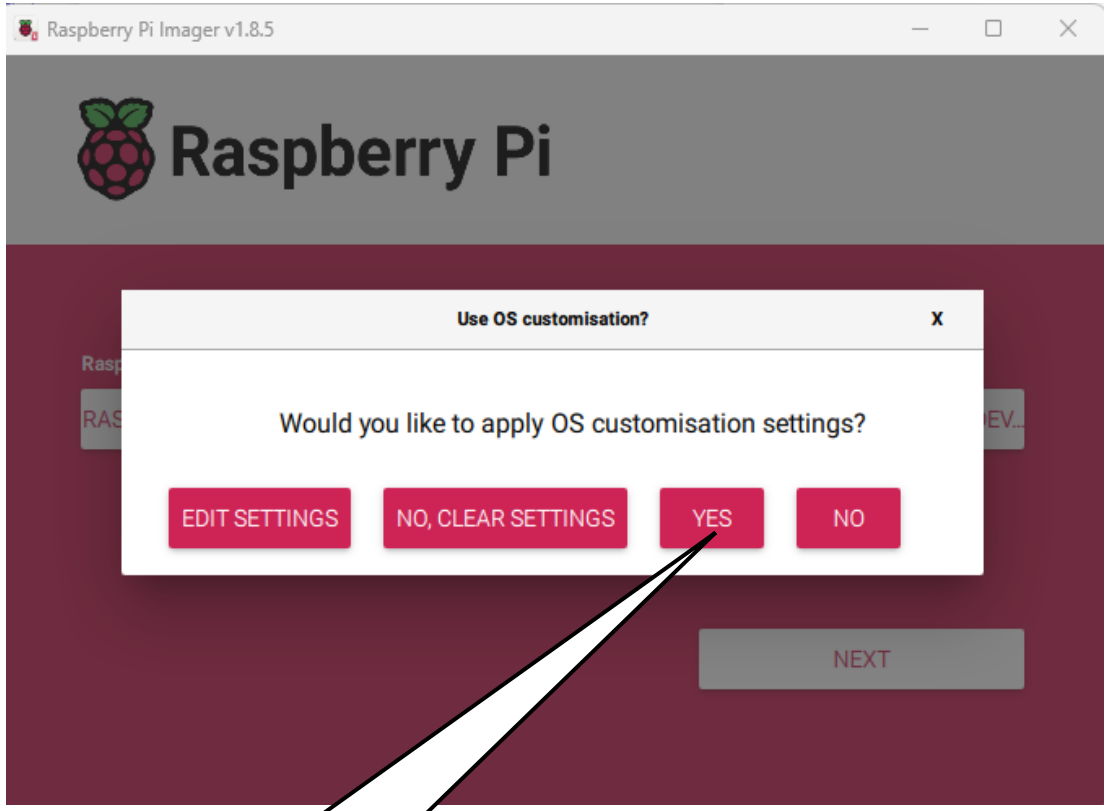
1. Click “Services” Tab

2. Choose password authentication

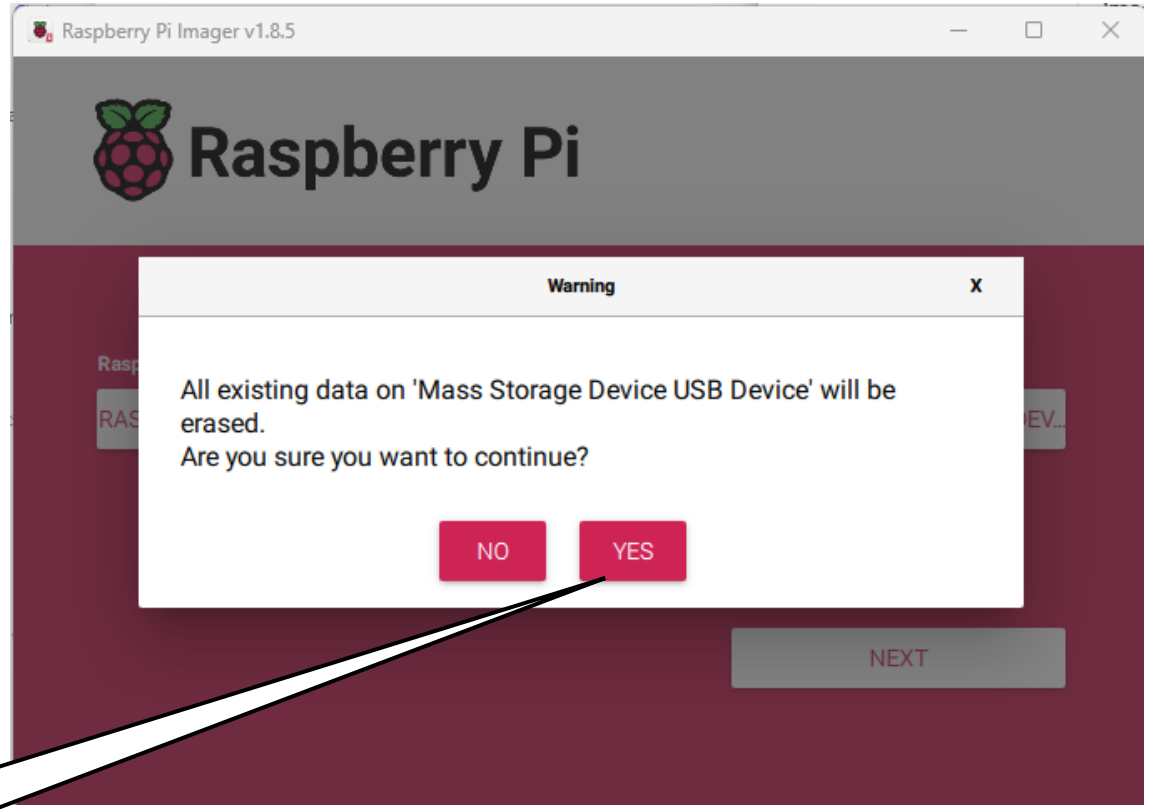
3. Save settings



Write the System Image

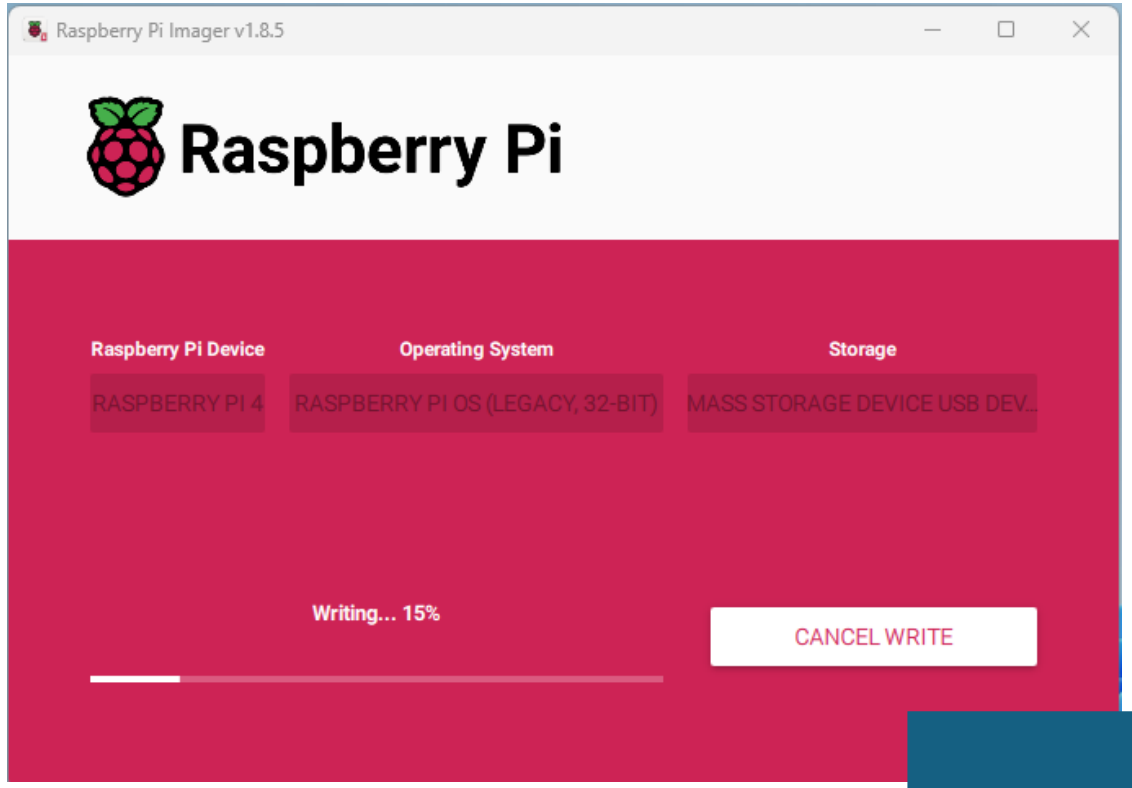


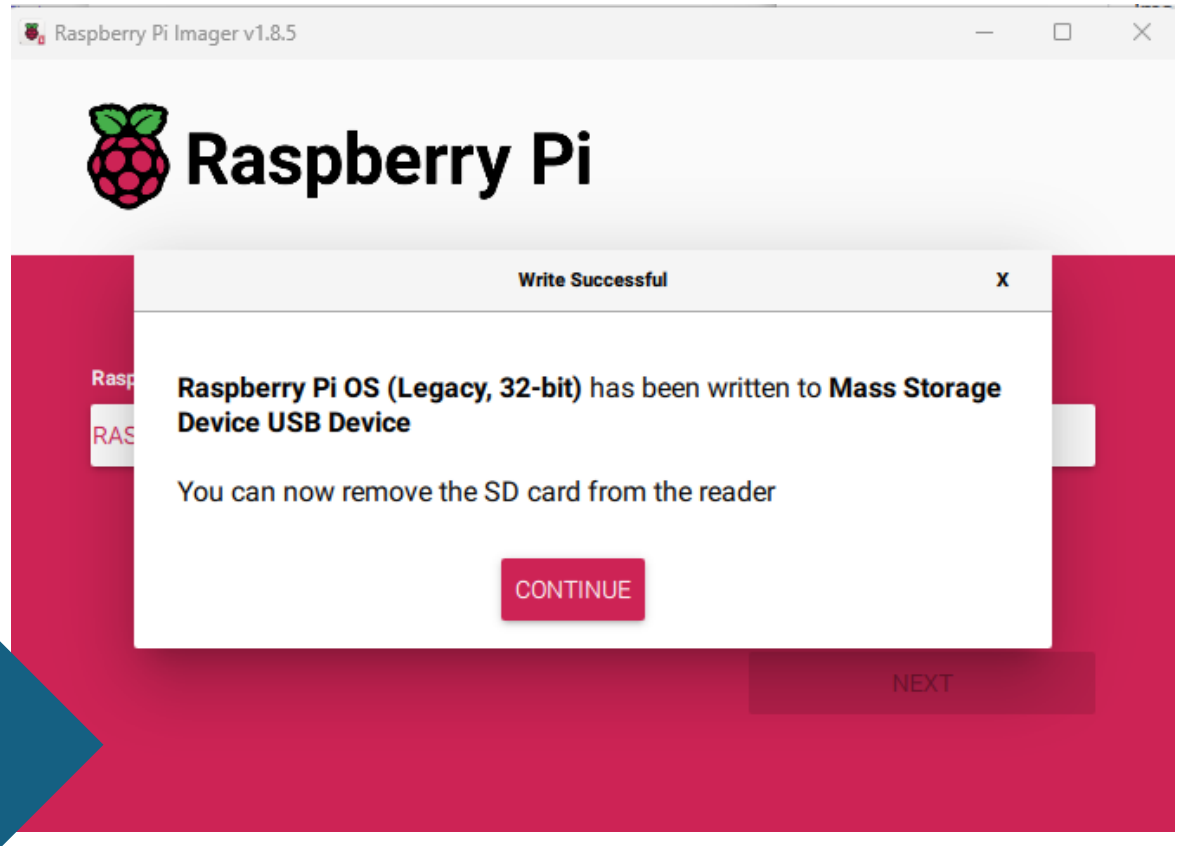
1. Click “Yes”



2. Click “Yes” to write the operating system image

Wait for the OS image to be written to the SD-card





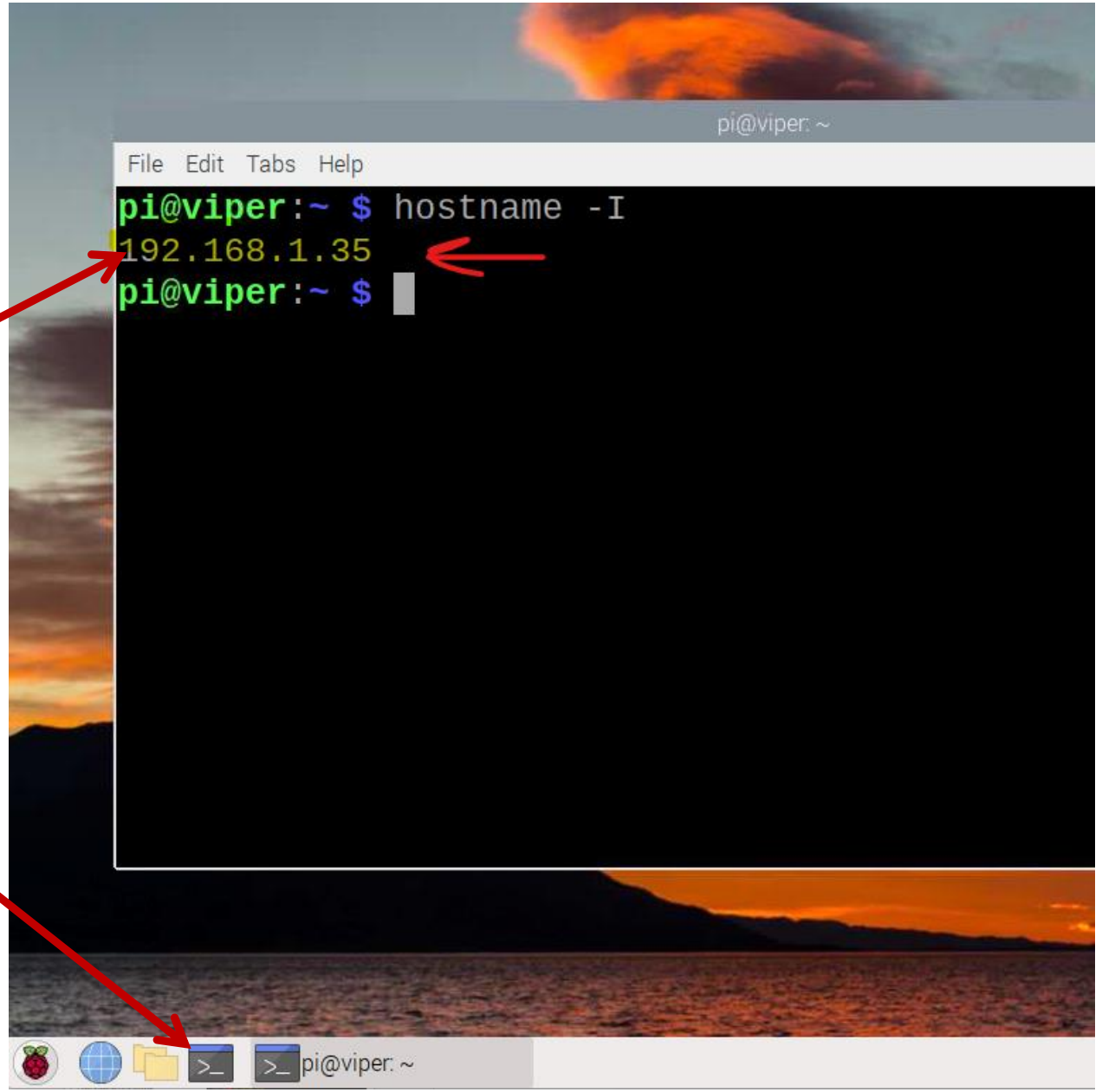
1. Insert SD card into Raspberry Pi
2. Power on the Raspberry Pi

Boot the Raspberry Pi

- With the keyboard and 7” LCD display plugged in
  - Insert the SD-card into the Raspberry Pi (do not force it) and power it on.
    - Open a terminal console and enter the command:

      hostname –I

- Record the IP-Address for use throughout the project

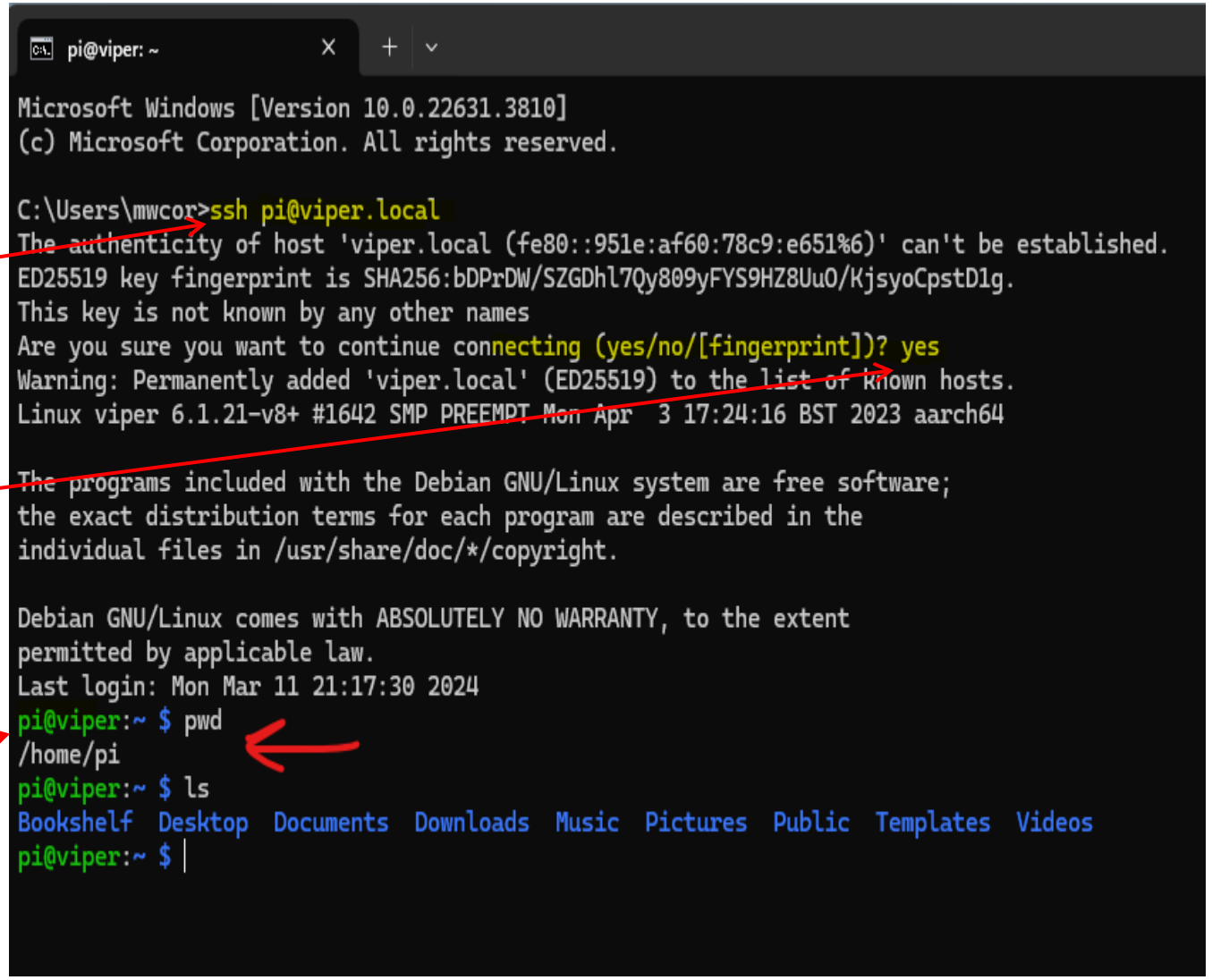


Start an SSH session with the Raspberry Pi

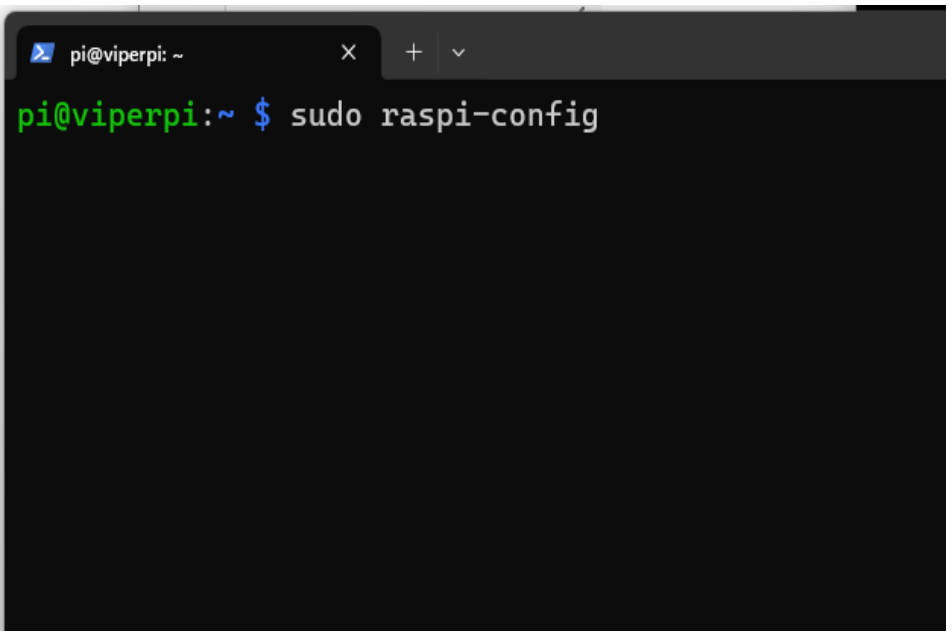
1. Open a new terminal window
2. Type command*: ssh pi@ip-address*

Type command*: yes*

Congratulations. You have a new (remote) connection to the Raspberry Pi

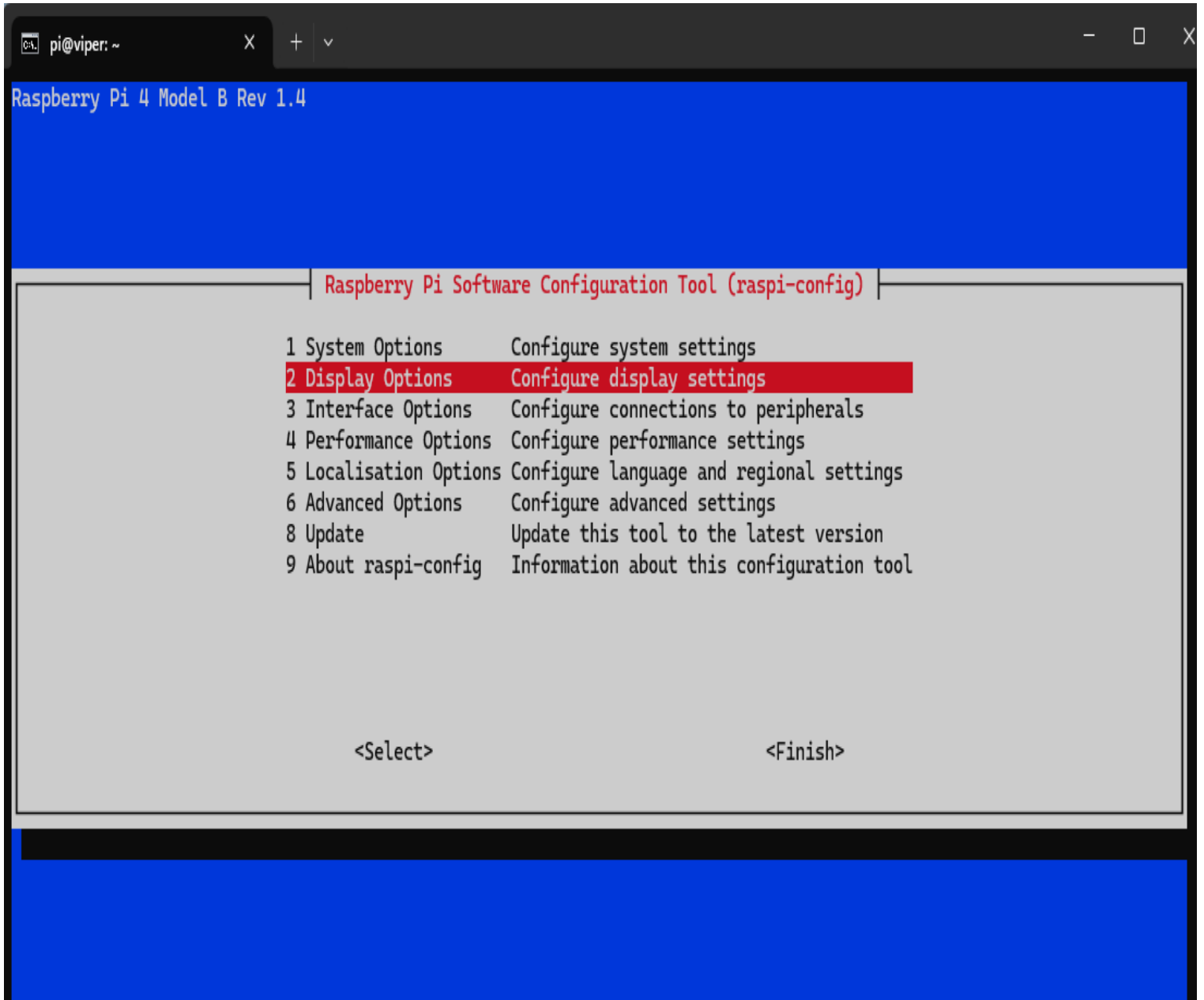


Run the Raspberry Pi configuration tool to configure the services



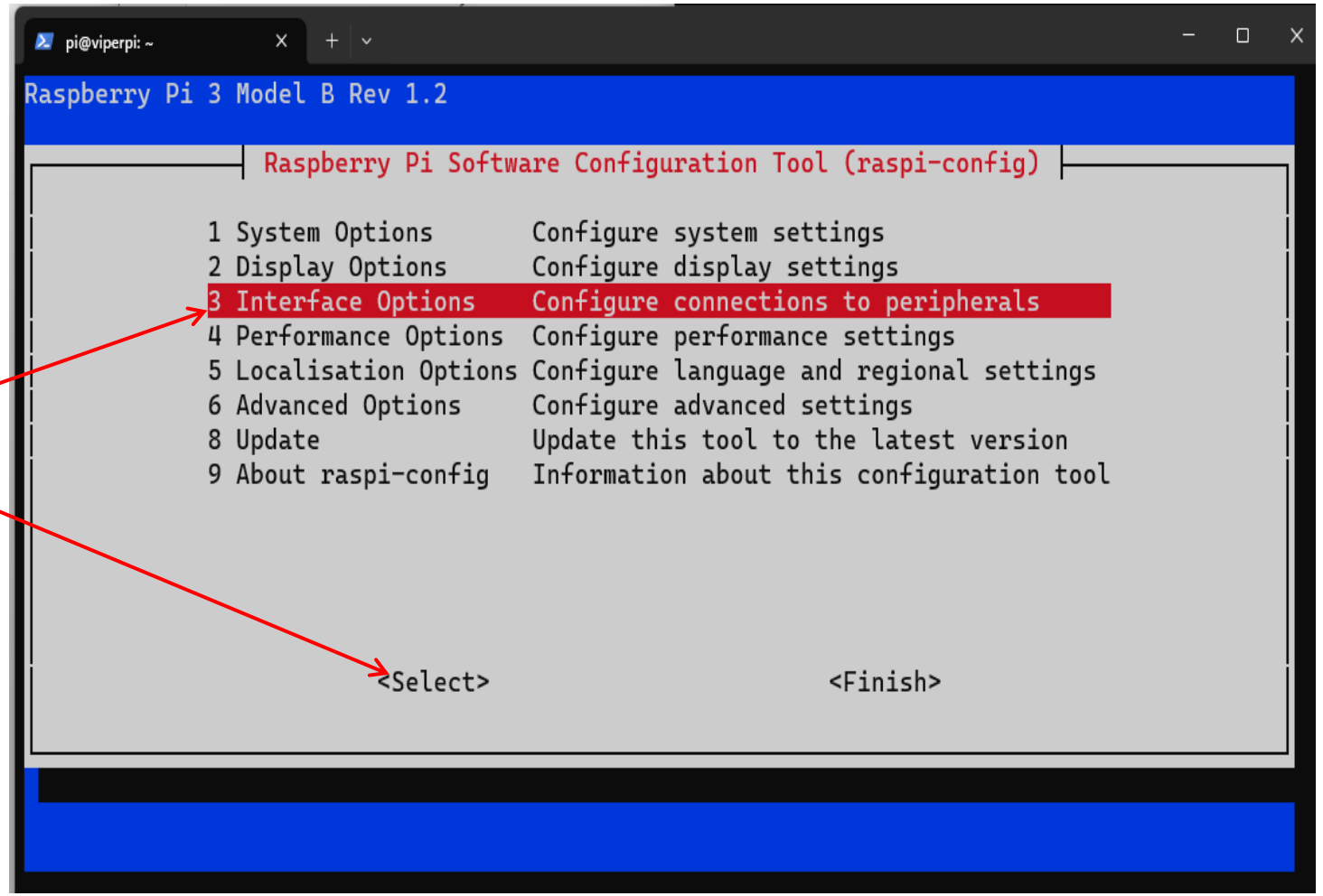
Type command: sudo raspi-config

Choose Display Options



Select “Interface Options”

Select menu item 3:

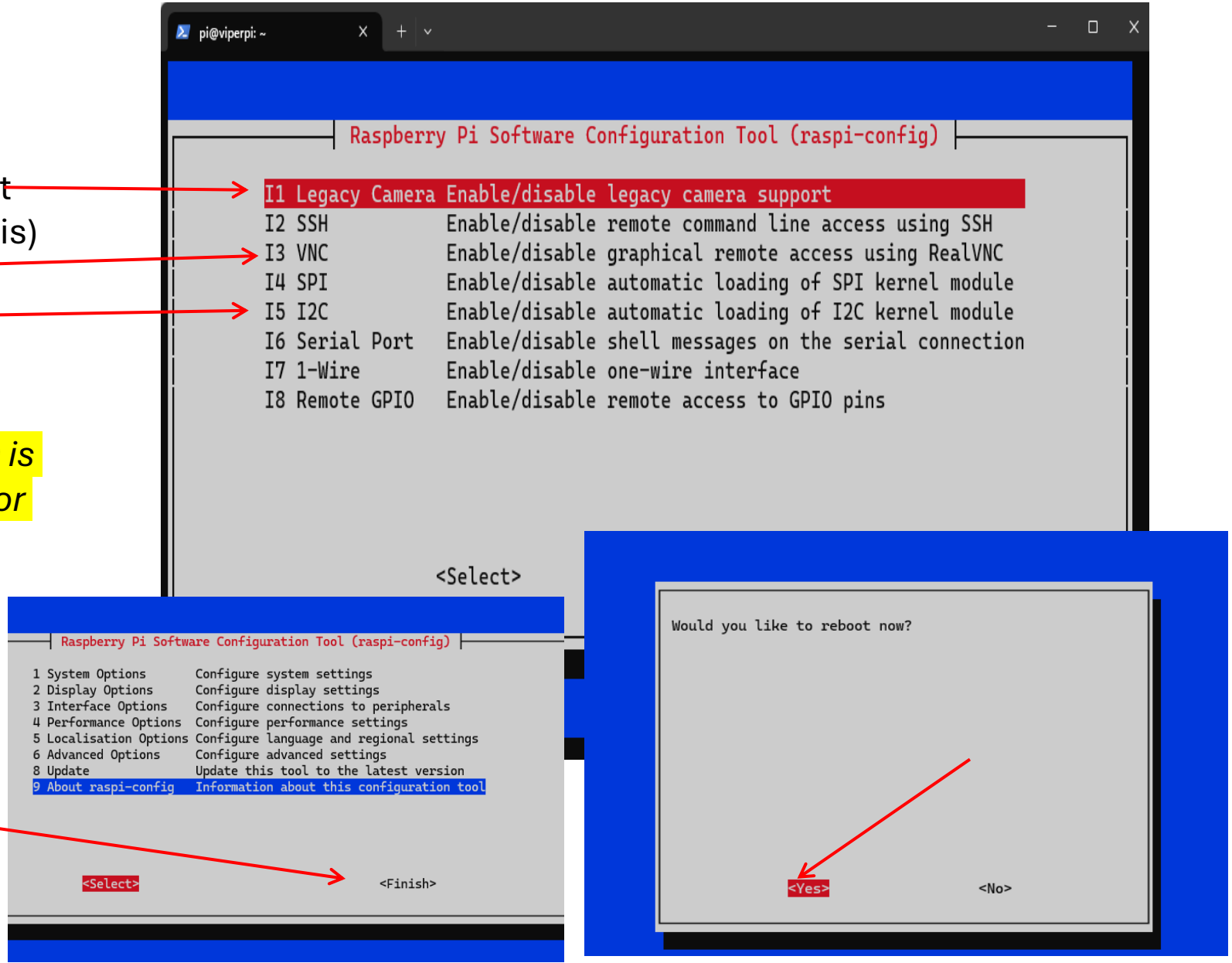


Enable Required Services…

1. Enable Legacy Camera Support
2. SSH is already enabled (skip this)
3. Enable both VNC
4. Enable I2C

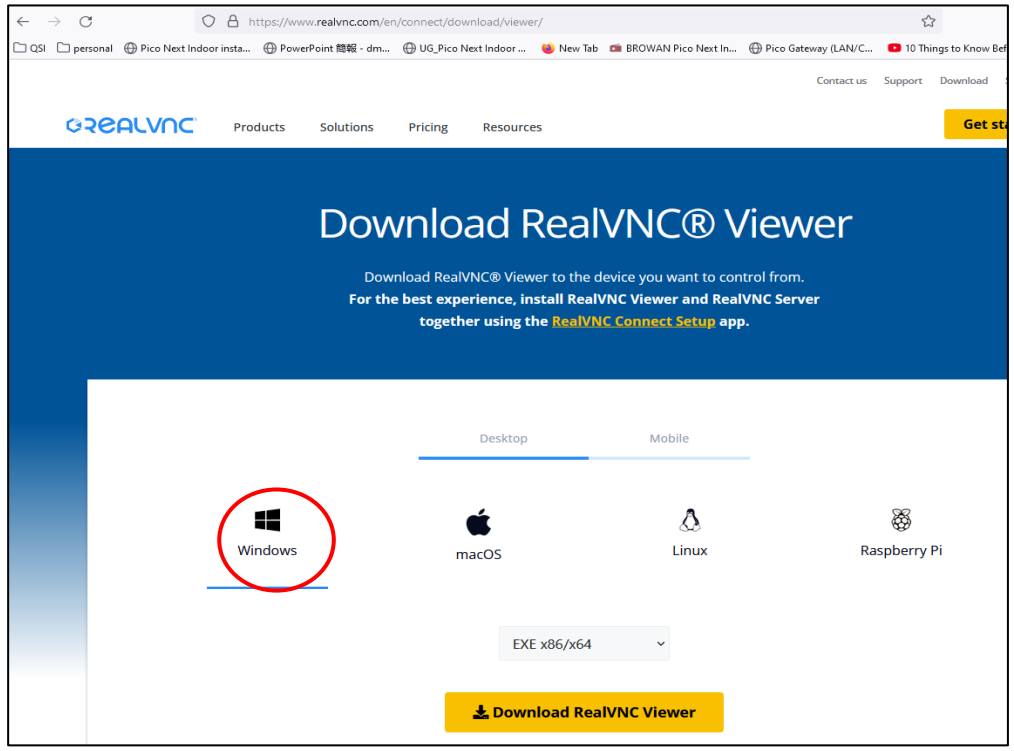
***Note: I2C** (Inter-Integrated Circuit) is widely used a serial bus protocol for establish serial communication integrated circuits (ICs)*

When 1-4 are complete,
Select Finish to exit and Reboot



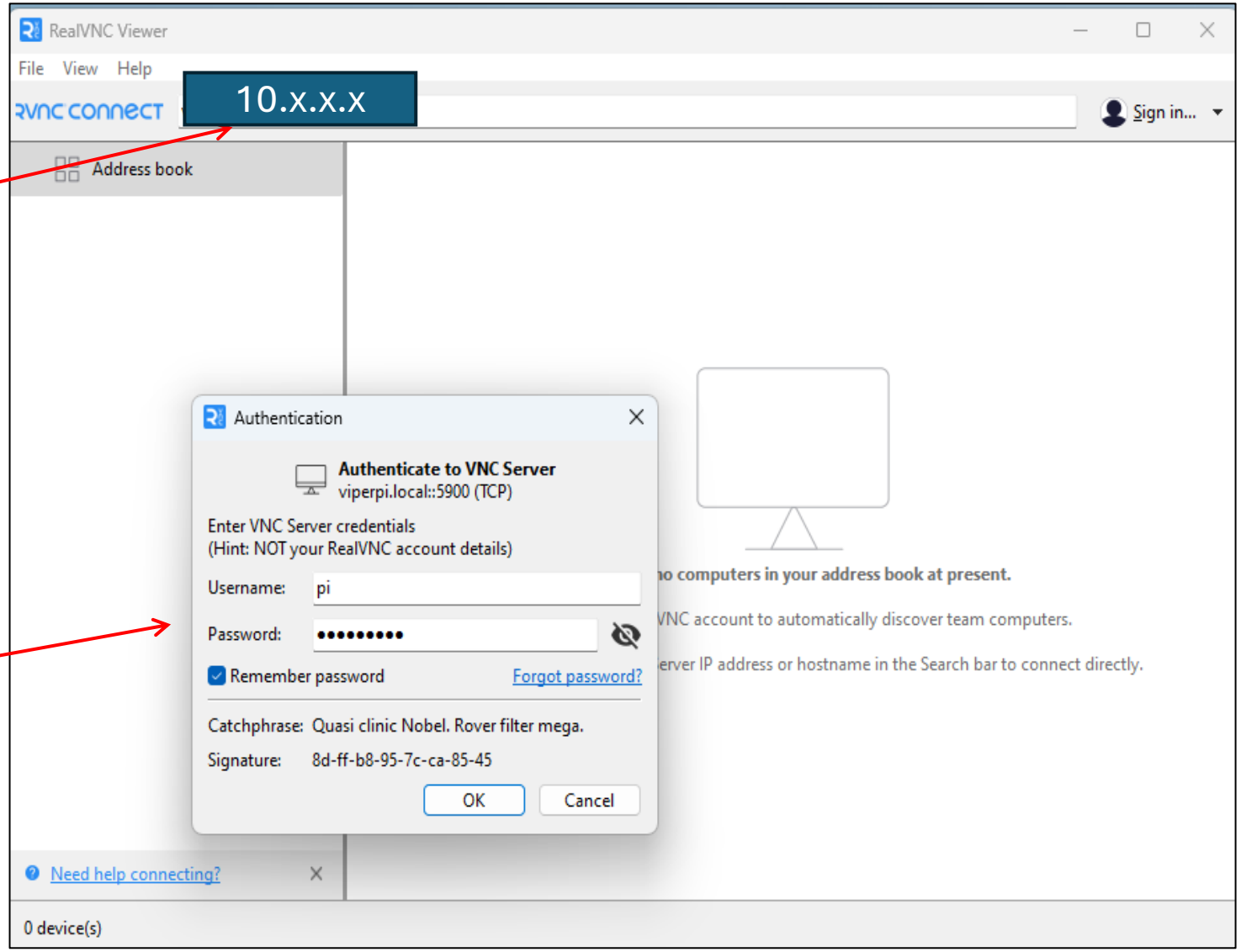
Setup VNC viewer laptop PC

- VNC stands for ***Virtual Network Computing***. It is a cross-platform screen sharing system that was created to **remotely control/view another computer**
  - Download the VNC viewer application to Windows (host) computer:
    https://www.realvnc.com/en/connect/download/viewer/

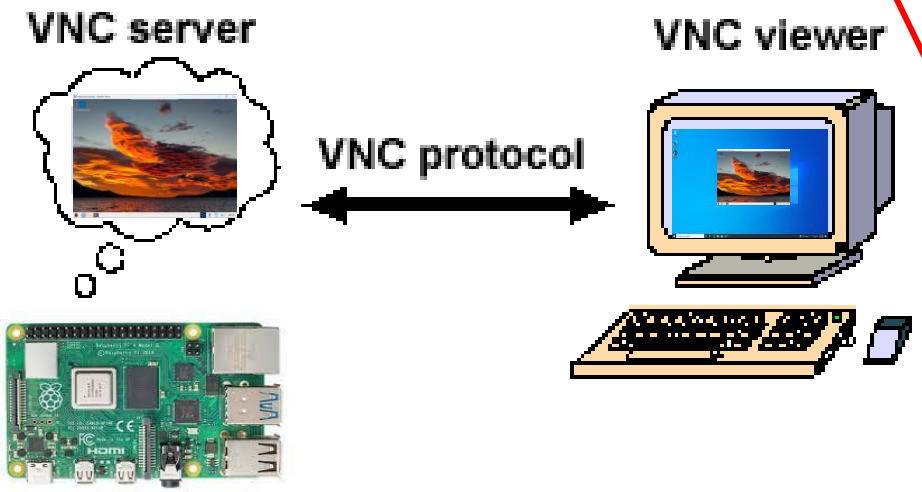


Establish a VNC session with Raspberry PI

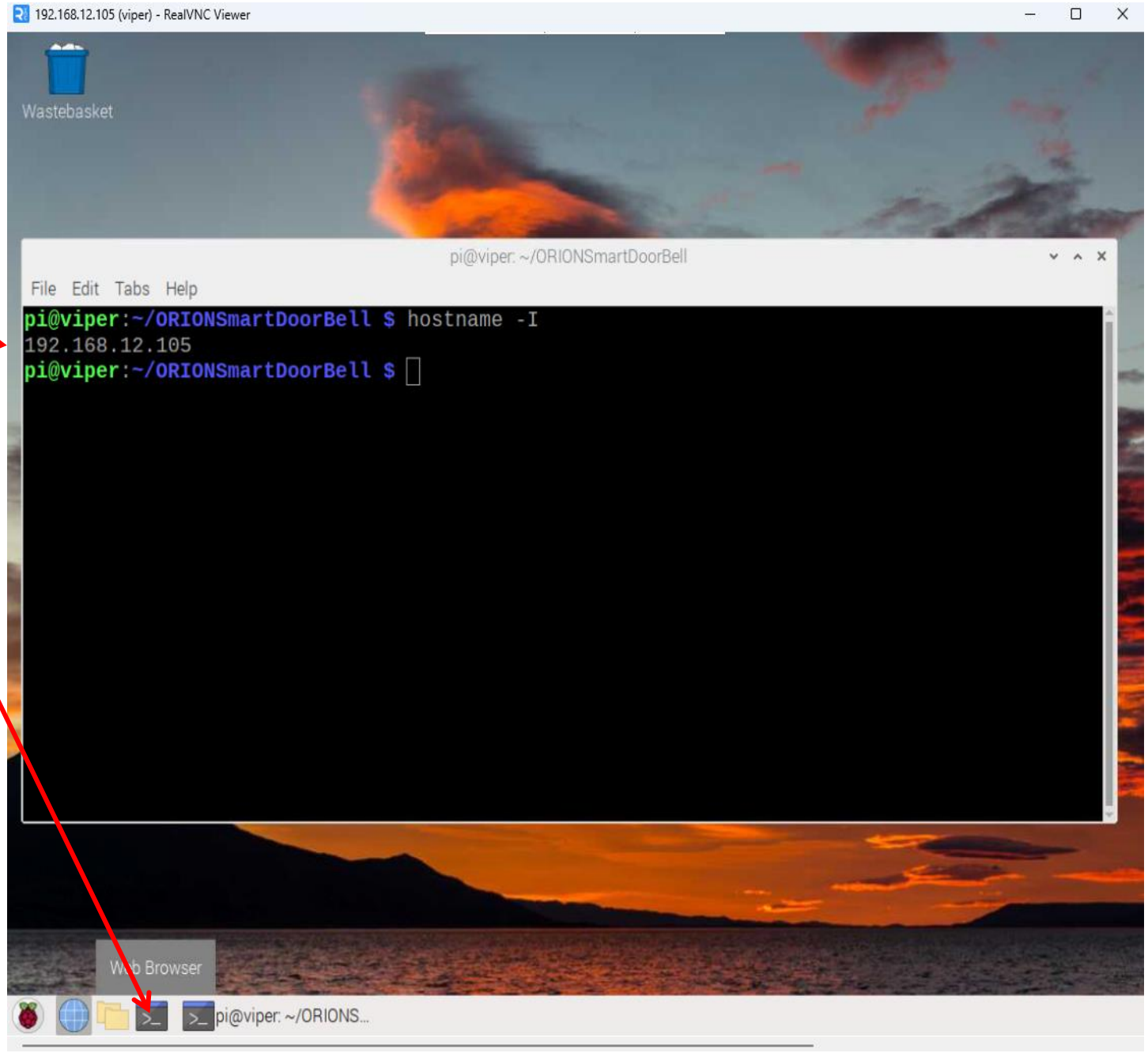
- Open the Real VNC viewer and enter the Raspberry Pi hostname or IP address in the connect box.
    - Enter the IP address (10.x.x.x) for your team's Pi in the Real VNC viewer connect box as shown and press enter
    - Supply the username **"pi",** and the **password your team specified.**



- Remote desktop experience, without the need for a physical monitor and peripherals
  - Open terminal and type command: *hostname –I*
- Record (write down) your IP address



Source: http://web.mit.edu/cdsdev/src/howitworks.html

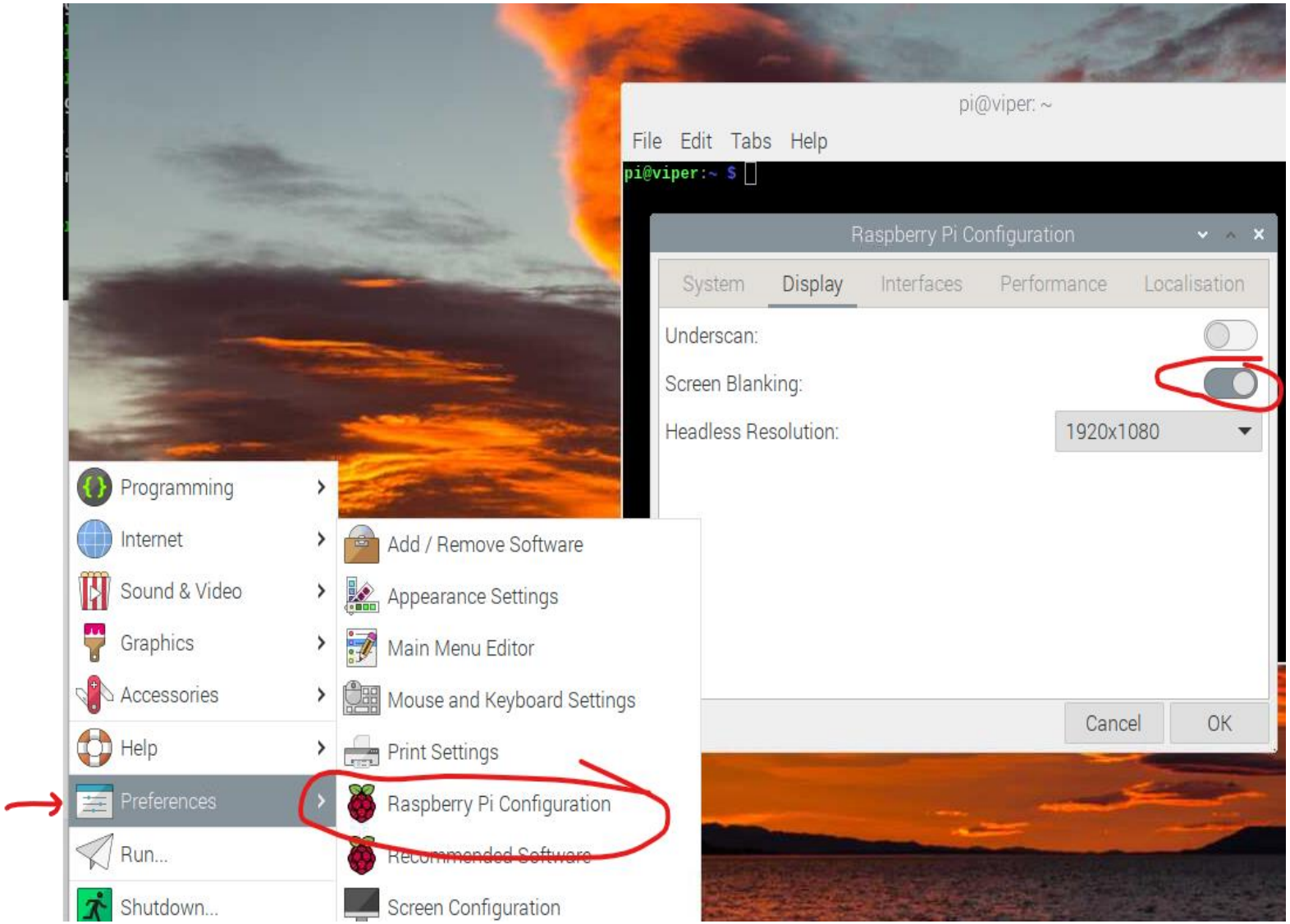


Complete Configuration…

- **Preferences-> Raspberry Pi Configuration**
  - This alternate way (besides raspi-config) to configure your Raspberry Pi

1. Turn off “Screen Blanking”

2. Pair the Bluetooth Speaker



Interlude: Short Linux Primer

- Wait! Isn't this camp about building devices with Raspberry Pi? Why are we pausing to talk about Linux?
  - Ans: Linux is an operating system that runs almost everywhere! In almost all IT technical careers, you will be required to know and use Linux with proficiency.
    - Linux runs in more environments than you may realize:
      - Big enterprise datacenters, Cloud service backends, your Android phone, smart TVs, cameras, wearables (smart watches), etc.
        - And yes… it is the default OS that runs on your Raspberry PI.
          - Called Raspberry Pi OS (based on the Debian GNU/Linux distribution)
            - https://en.wikipedia.org/wiki/Raspberry_Pi_OS

- Reasonable (Funny) Video Intro to Linux
  - https://www.youtube.com/watch?v=zA3vmx0GaO8

What is Linux ?

- https://www.linux.com/what-is-linux/
  - Video (Linux 100s) : https://www.youtube.com/watch?v=rrB13utjYV4
  - https://www.tutorialspoint.com/operating_system/os_linux.htm

  - Distributions: https://www.youtube.com/watch?v=tjx_aSOPjls
    - (For laptops): https://www.youtube.com/watch?v=Z1LKHwY-Pvo
      - For Raspberry PI: https://en.wikipedia.org/wiki/Raspberry_Pi_OS

  - Command cheat sheet:
    - https://www.geeksforgeeks.org/linux-commands-cheat-sheet/#

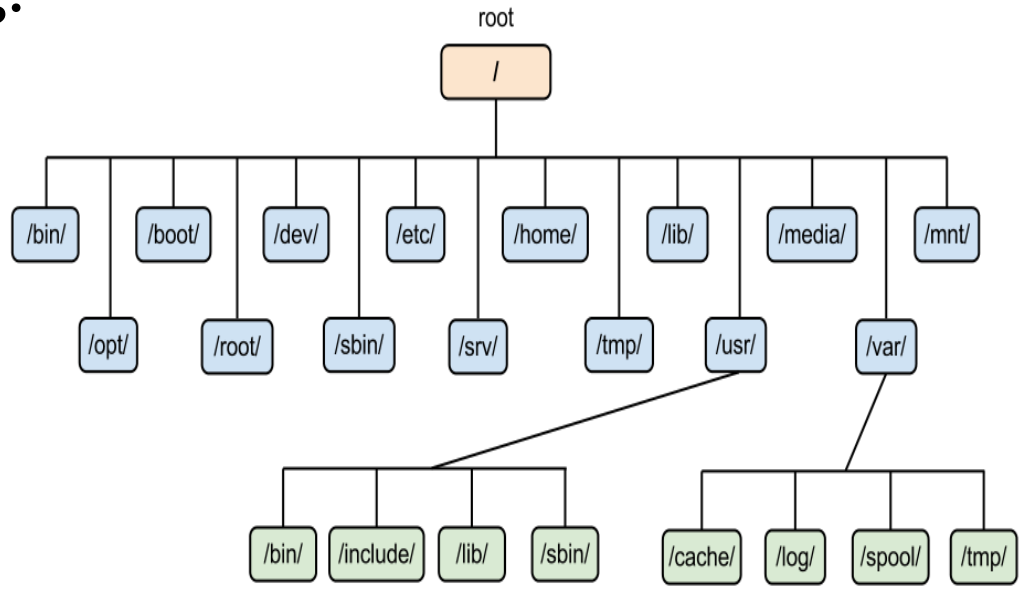
Linux Command Line

- Shell - CLI - primer user interface to operating. It's essentially a program that interprets user commands, and runs other programs

- Understanding the Filesystems
  - Files and folders (called directories) are organized hierarchically (in a tree structure)

- Basic Linux Commands (video)
  - https://www.youtube.com/watch?v=bb_ApuH15YE

- Further study reference: "The Linux Command line" eBook
  - https://linuxcommand.org/tlcl.php



ls – list contents of folder
~ refers to home folder
cd – change to new directory (folder)

Short BASH Video

https://www.youtube.com/watch?v=zsI_nafq_sA

Take 10-15 mins to get familiar with your Raspberry Pi

```
pi@viper:~/ORIONSmartDoorBell/certs $ ls -l
total 28
-rw-r--r-- 1 pi pi 1424 Jul 10 09:47 orion_ca.crt
-rw------- 1 pi pi 1751 Jul 10 09:44 orion_ca.key
-rw-r--r-- 1 pi pi   41 Jul 10 14:14 orion_ca.srl
-rw-r--r-- 1 pi pi 1298 Jul 10 14:14 ring_server.crt
-rw-r--r-- 1 pi pi 1054 Jul 10 10:26 ring_server.csr
-rw------- 1 pi pi 1679 Jul 10 09:57 ring_server.key
-rw-r--r-- 1 pi pi  319 Jul 10 10:24 san.cnf
```

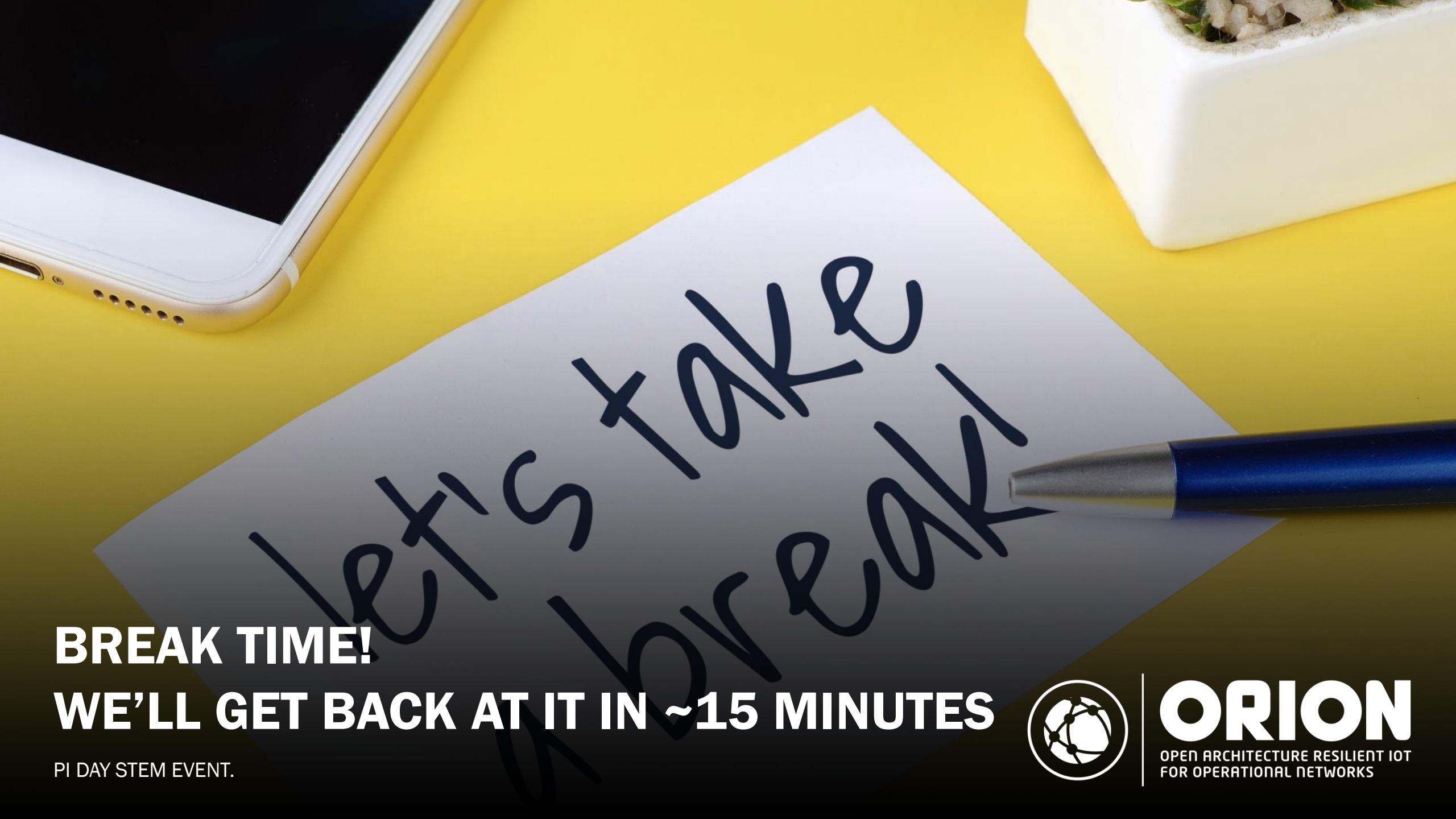
Challenge Questions: Understanding file permissions is one of the most import aspects of using Linux effectively (and securely!!)

1. Review the command out below, describe generally what the output specifies.

2. (Write down) the access permissions for the files: *orion_ca.crt*, and *orion_ca.key*. Specially state the access control "details" for these two files.

3. What is the Unix command for managing file access permissions?

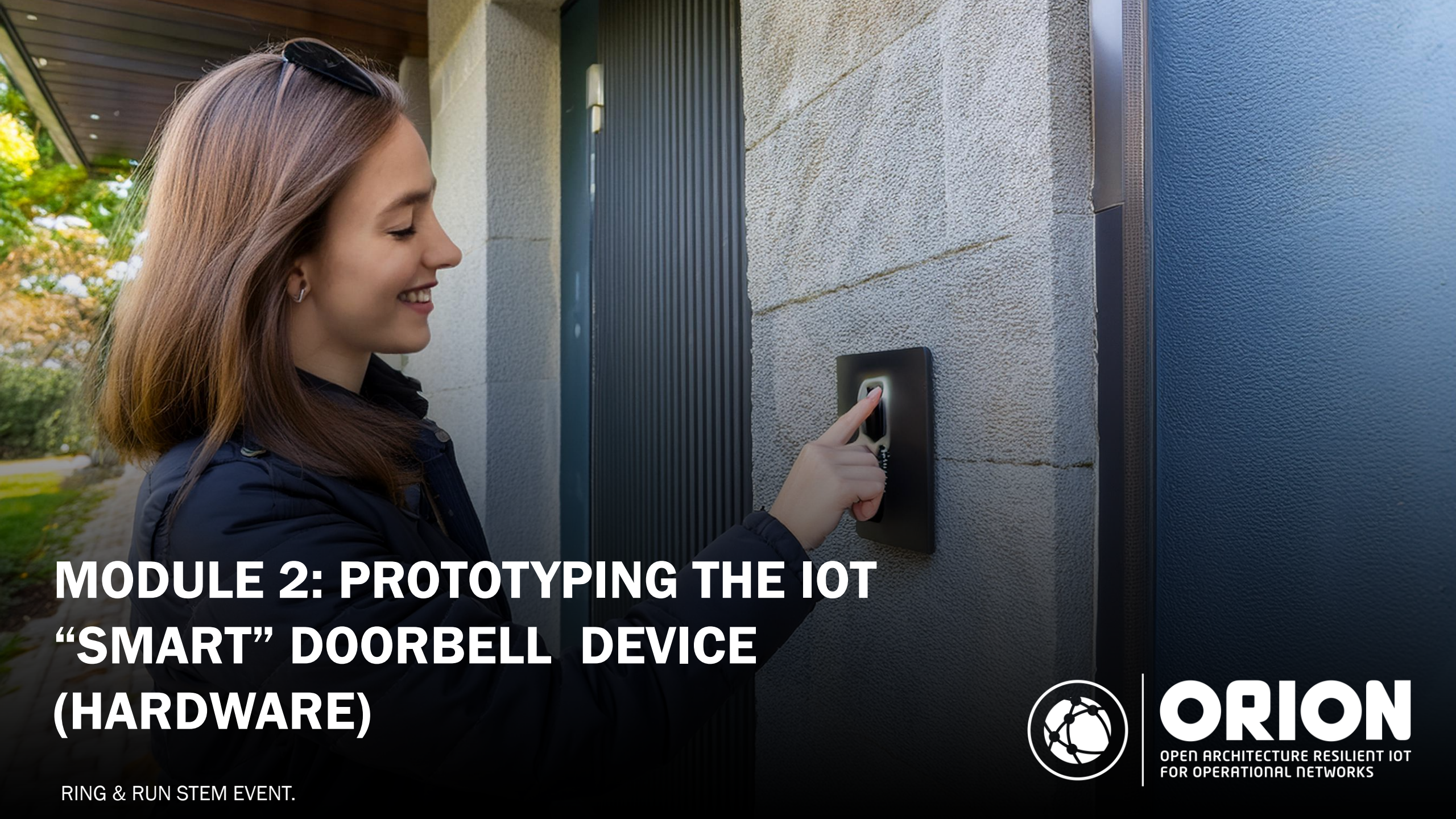
End of Module 1



BREAK TIME!
WE'LL GET BACK AT IT IN ~15 MINUTES

PI DAY STEM EVENT.

ORION
OPEN ARCHITECTURE RESILIENT IOT FOR OPERATIONAL NETWORKS



MODULE 2: PROTOTYPING THE IOT “SMART” DOORBELL DEVICE (HARDWARE)

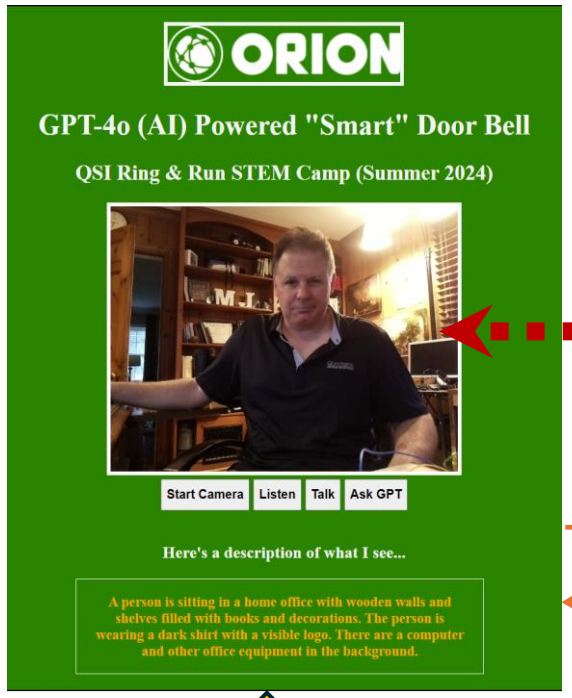
RING & RUN STEM EVENT.

ORION
OPEN ARCHITECTURE RESILIENT IOT FOR OPERATIONAL NETWORKS

IoT enabled “Smart” Doorbell Concept

Client App (Front-End)

Web app: (based on HTML5 CSS and JavaScript)



OpenAI

https://openai.com/

Python Back-end

HTTPS MJPG video stream

Publish to topic: *ring/#*

device command set

Subscribe to topic: *ring/#*

Publish to topic: *ring/#*

device command set

Subscribe to topic: *ring/#*

MQTT over WebSockets

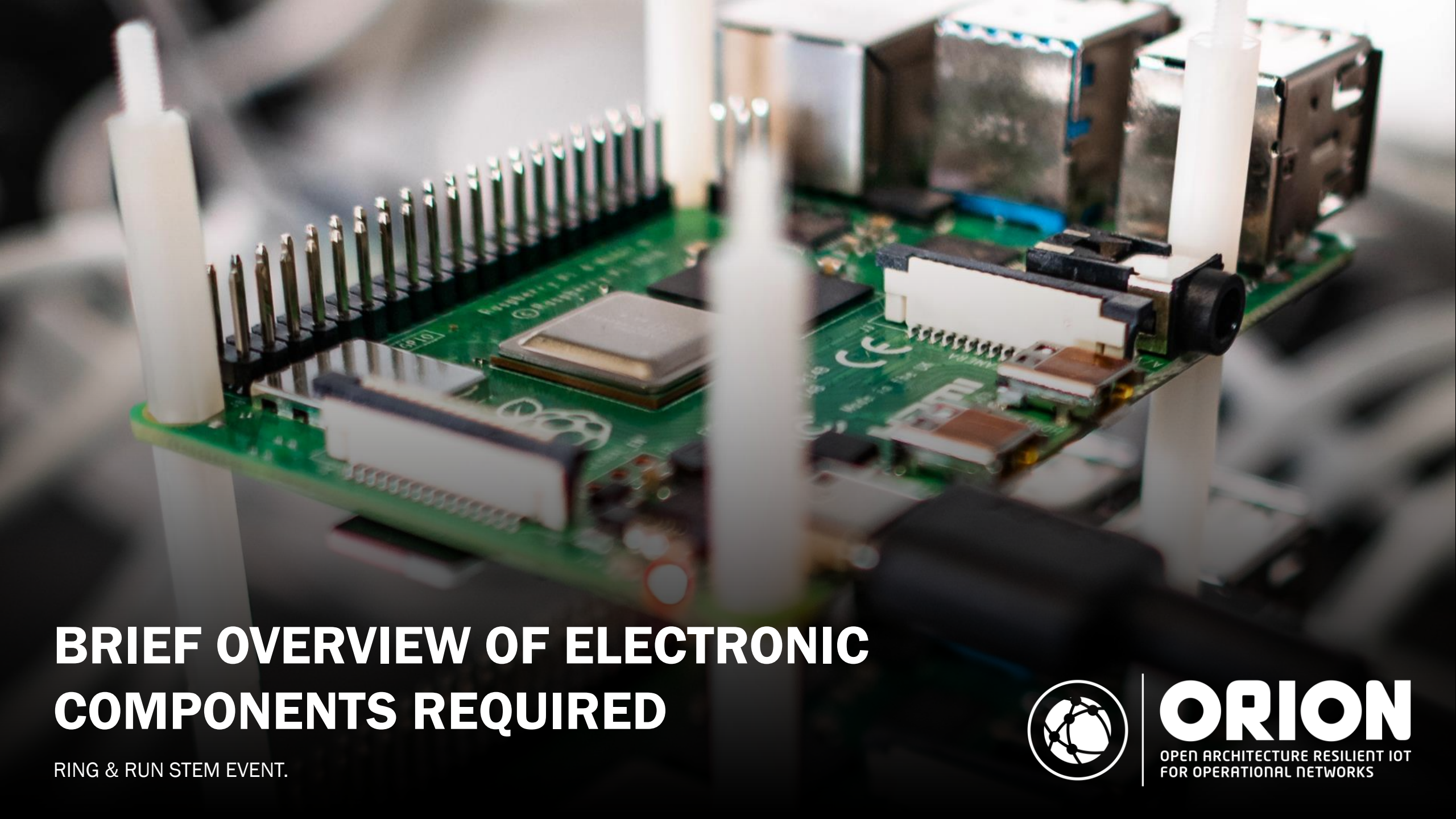
Mosquitto (MQTT)Broker

Raspberry Pi 4B

Supported modes: motion and manual (button press)

Doorbell Prototype Device

fritzing

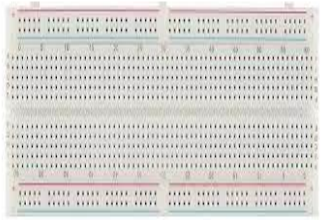


BRIEF OVERVIEW OF ELECTRONIC COMPONENTS REQUIRED

RING & RUN STEM EVENT.

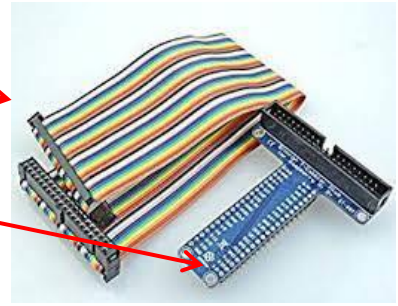
ORION
OPEN ARCHITECTURE RESILIENT IOT FOR OPERATIONAL NETWORKS

- List of components:
  - 1 x Raspberry Pi 4B
  - 1 x half size bread board
  - 1 x 40-pin ribbon cable
  - 1 x T-cobbler (GPIO extender)
  - 1 x 220 Ohm resistor
  - PIR Motion Sensor
  - 1 X 5mm LED
  - Jumper wires
  - Switch button



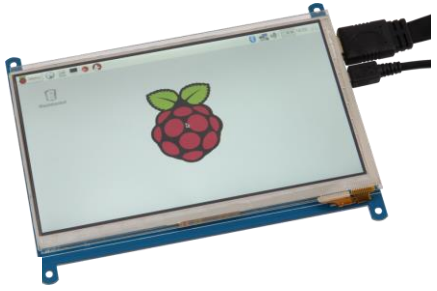
Mini bread board

Camera module



USB microphone

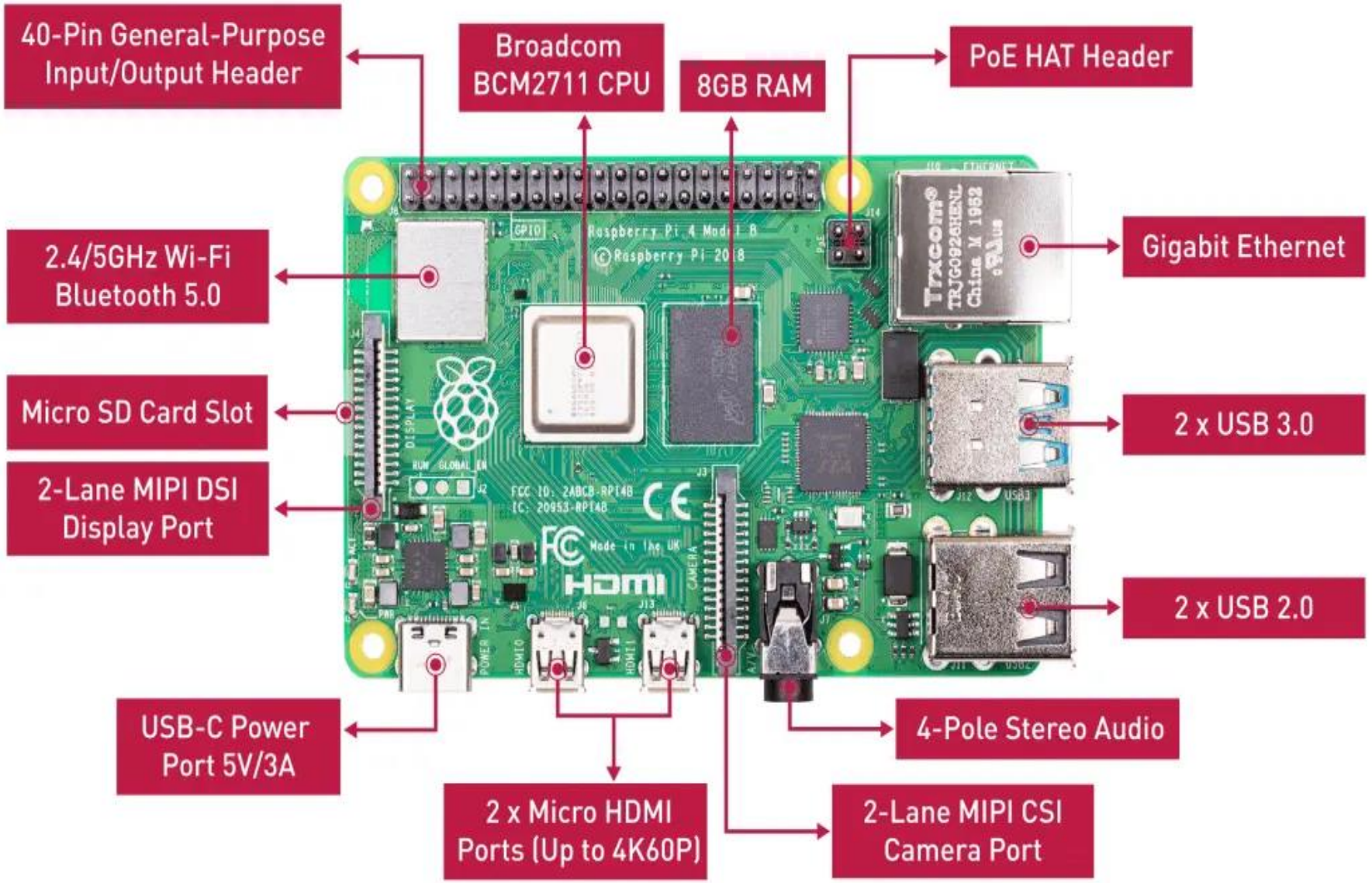
Bluetooth speaker



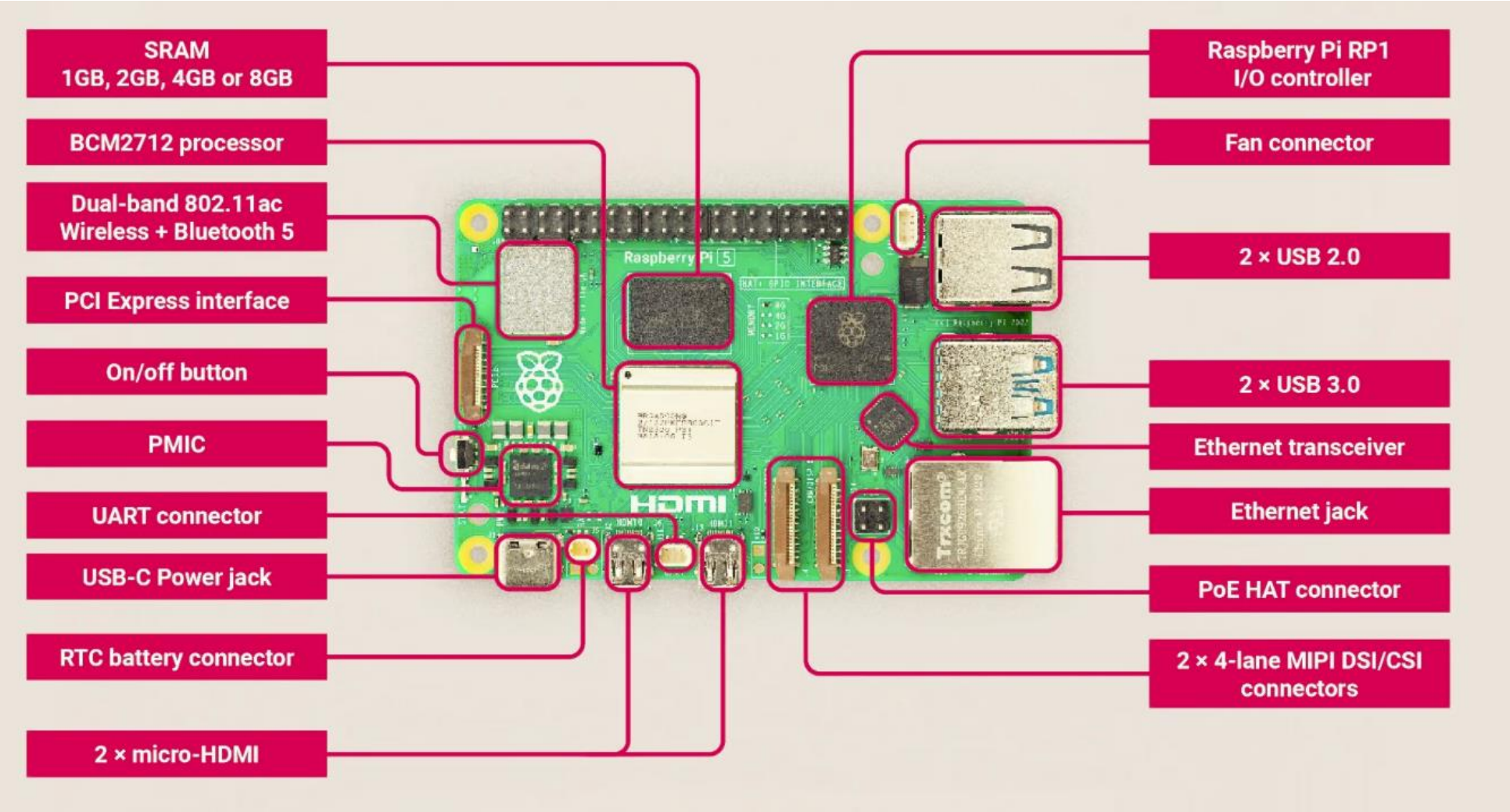
LCD Display

Mini keyboard

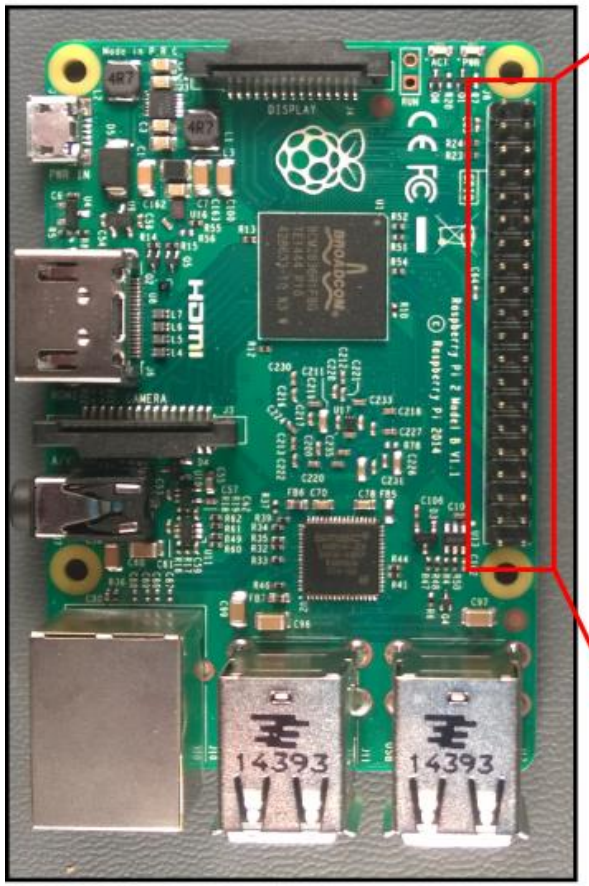
Raspberry Pi 4B



Raspberry 5 Pi



40-pin GPIO (General Purpose Input/Output) Header Overview

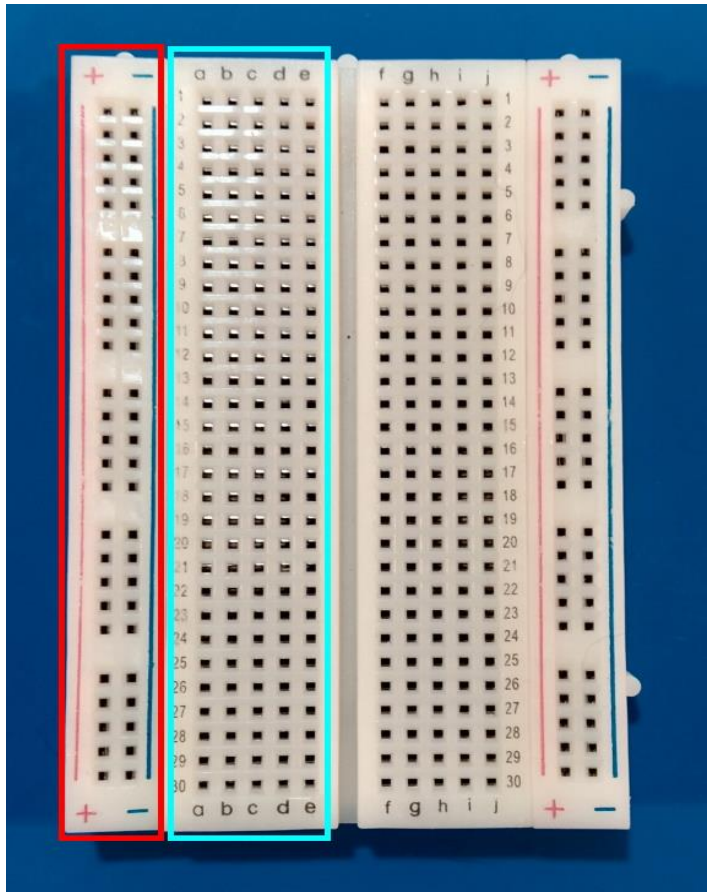


Alternate Function						Alternate Function
	3.3V PWR	1	2	5V PWR		
I2C1 SDA	GPIO 2	3	4	5V PWR		
I2C1 SCL	GPIO 3	5	6	GND		
	GPIO 4	7	8	UART0 TX		
	GND	9	10	UART0 RX		
	GPIO 17	11	12	GPIO 18		
	GPIO 27	13	14	GND		
	GPIO 22	15	16	GPIO 23		
	3.3V PWR	17	18	GPIO 24		
SPI0 MOSI	GPIO 10	19	20	GND		
SPI0 MISO	GPIO 9	21	22	GPIO 25		
SPI0 SCLK	GPIO 11	23	24	GPIO 8		SPI0 CS0
	GND	25	26	GPIO 7		SPI0 CS1
	Reserved	27	28	Reserved		
	GPIO 5	29	30	GND		
	GPIO 6	31	32	GPIO 12		
	GPIO 13	33	34	GND		
SPI1 MISO	GPIO 19	35	36	GPIO 16		SPI1 CS0
	GPIO 26	37	38	GPIO 20		SPI1 MOSI
	GND	39	40	GPIO 21		SPI1 SCLK

- Allows the code running on Raspberry Pi to interact with the outside world…
- Supports digital communication using serial bus protocols including: UART, SPI, i2c
- Pin numbering scheme: Board, BCM/GPIO/WiringPi

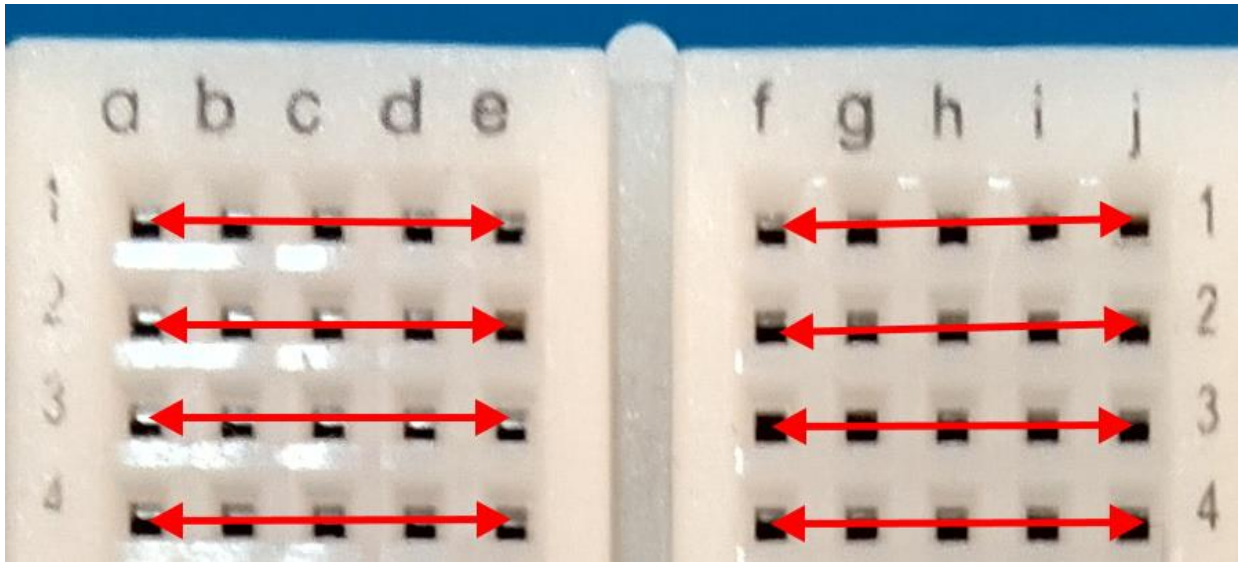
https://learn.sparkfun.com/tutorials/introduction-to-the-raspberry-pi-gpio-and-physical-computing/gpio-pins-overview
https://projects.raspberrypi.org/en/projects/physical-computing/1
https://www.pi4j.com/getting-started/understanding-the-pins/

Breadboard



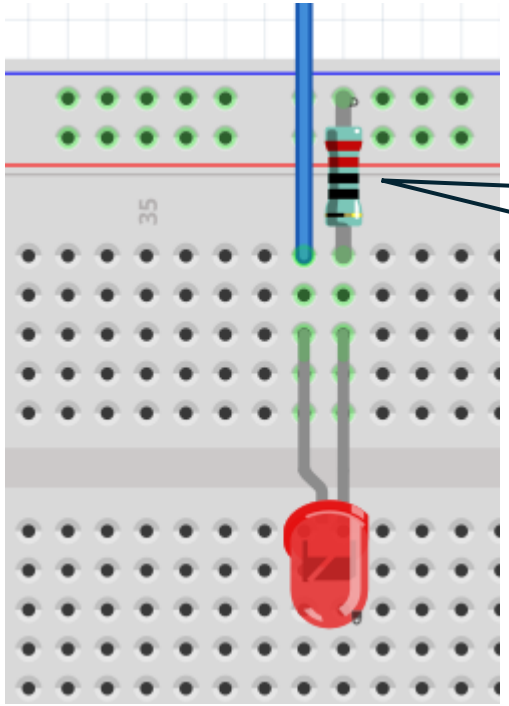
Source: https://learn.microsoft.com/en-us/training/modules/create-iot-device-dotnet/2-construct-iot-hardware - [2]

- The breadboard is organized in rows and columns called socket strips (shown in cyan)
  - *Socket strips are connected vertically*
- The bus strips on the edges (shown in red) are using connecting a power source (3.3v and 5v for the Pi)
  - *Edge strips connect horizontally over the length of the breadboard*
- Socket strips allow components to be connected without soldering or wires. [2]



- Any pin plugged into row 1, column ‘a’ would be connected to any pin plugged into row 1, columns ‘b-e’.
- On the other side of the divider, row 1 columns f-j are similarly connected. [2]

Resistor



Use of 220 Ohm resistor to limit the current through the LED and to prevent excess current that can burn out the LED

A resistor is **a passive two-terminal electrical component that implements electrical resistance as a circuit element**. In electronic circuits, resistors are used to reduce current flow, adjust signal levels, to divide voltages – [3]

[3] - https://en.wikipedia.org/wiki/Resistor

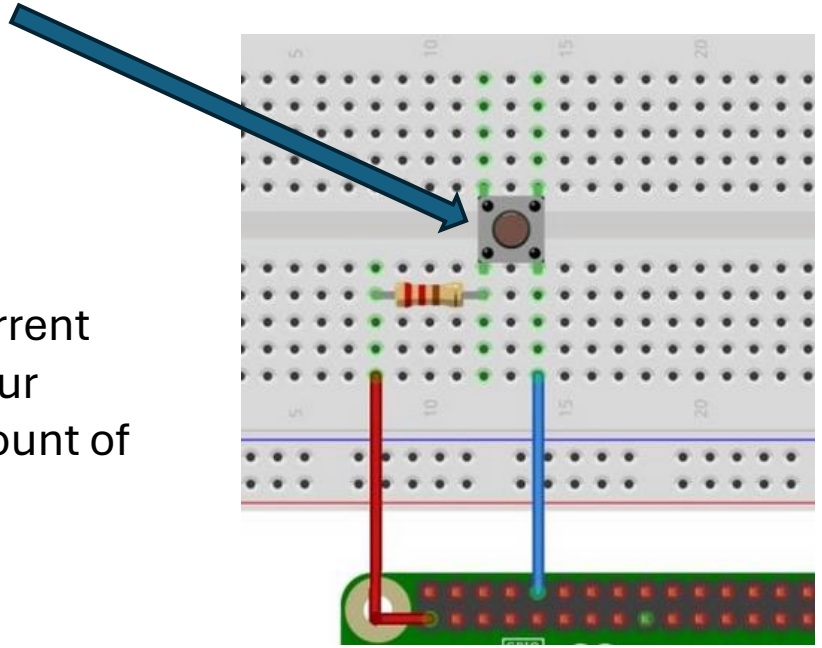
LED and Button Switch

- The longer leg (known as the '**anode**'), is always connected to the positive supply of the circuit. The shorter leg (known as the '**cathode**') is connected to the negative side of the power supply, known as 'ground'.

- **LED** stands for **L**ight **E**mitting **D**iode, and glows when electricity (current) is passed through it. - [2]

- Connect one side of the switch to an input pin on the Raspberry Pi, in this case we use pin 10.

- The other side of the switch we connect to 3.3V on pin 1 using a resistor.
  - The resistor is used as a current limiting resistor to protect our input pin by limiting the amount of current that can flow. [3]

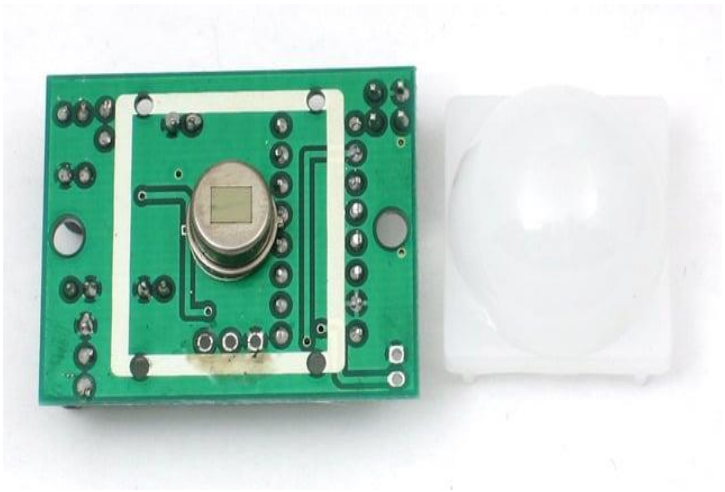


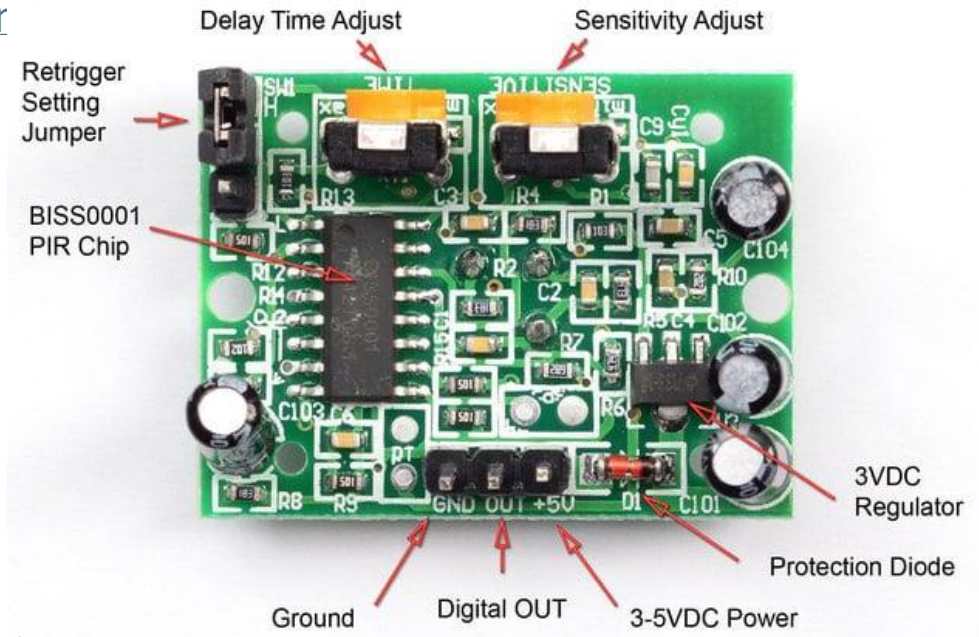
Source: https://thepihut.com/blogs/raspberry-pi-tutorials/27968772-turning-on-an-led-with-your-raspberry-pis-gpio-pins - [2]

Source: https://raspberrypihq.com/use-a-push-button-with-raspberry-pi-gpio/ - [3]

PIR, "Passive Infrared", "Pyroelectric", or "IR motion" sensors

- Sense motion, almost always used to detect whether a human has moved in or out of the sensors range.
- commonly found in appliances and gadgets used in homes or businesses.
- pyroelectric sensor (which you can see below as the round metal can with a rectangular crystal in the center), which can detect levels of infrared radiation.
- Source: https://learn.adafruit.com/pir-passive-infrared-proximity-motion-sensor/overview
  - https://learn.adafruit.com/pir-passive-infrared-proximity-motion-sensor/how-pirs-work



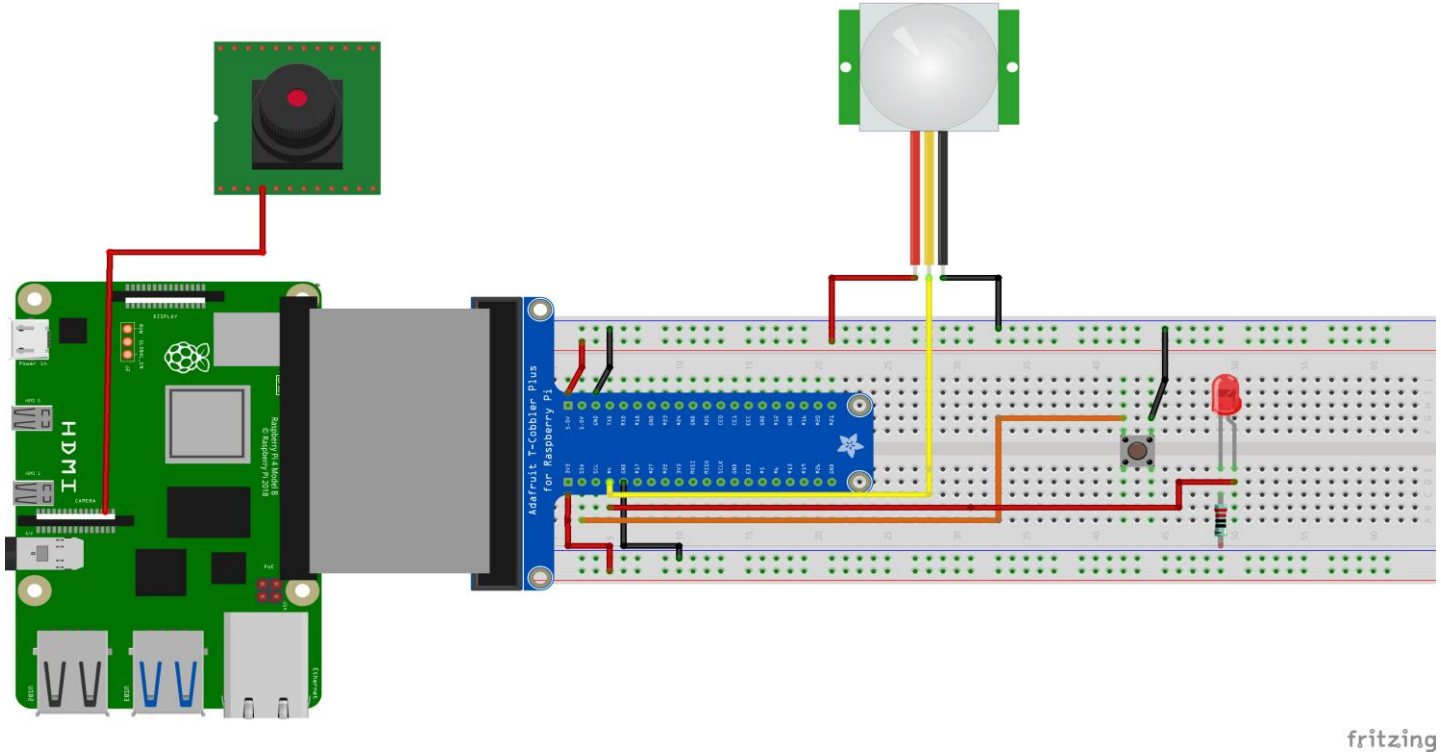


https://learn.adafruit.com/pir-passive-infrared-proximity-motion-sensor/overview

Camera Modules

- https://www.raspberrypi.com/products/camera-module-v2/
- https://www.raspberrypi.com/products/camera-module-3/

“Smart” Doorbell device (Fritzing circuit design) concept

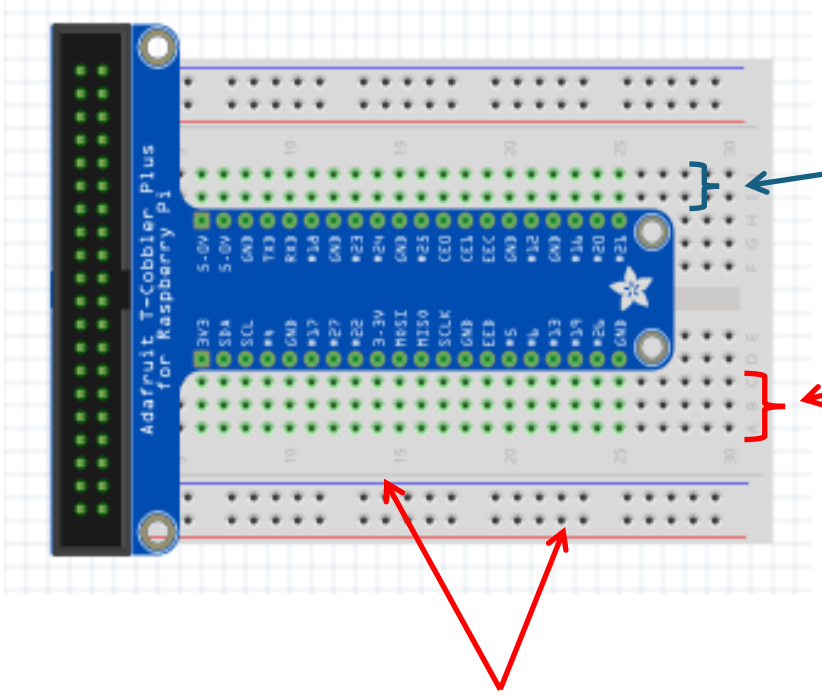


https://fritzing.org/

Fritzing is an open-source hardware initiative that makes electronics accessible as a creative material for anyone

Attach the T-Cobbler to the breadboard

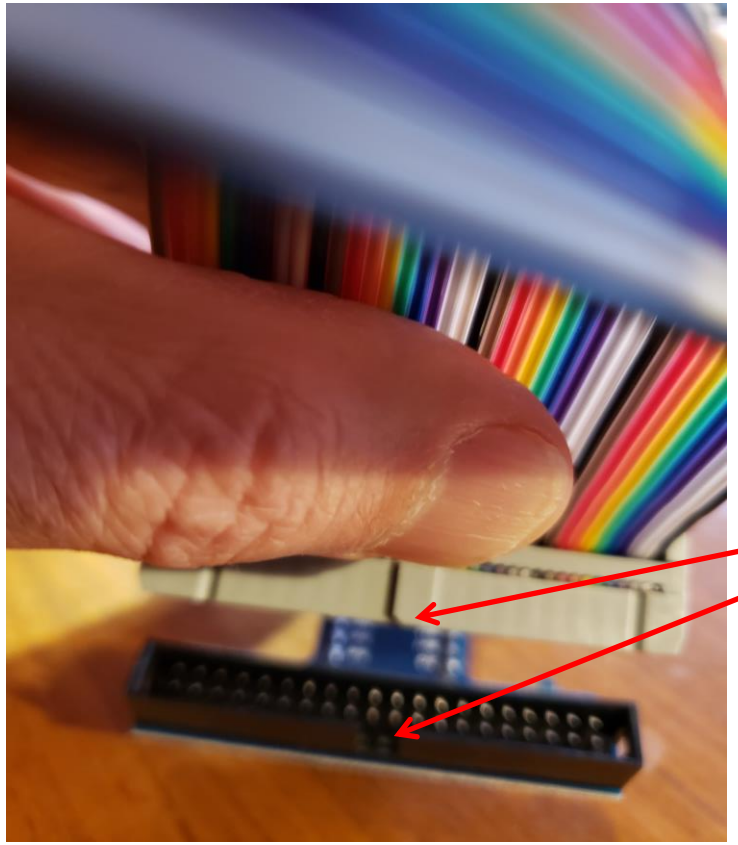
*** Note of the rows on exposed on each side of the t-cobbler (3 on one side, and 2 on the other. Your design should look like this too.**



**** Note the orientation to negative/positive (power) terminal strips**

***** Insert the t-cobbler beginning in the first column of sockets in the breadboard**

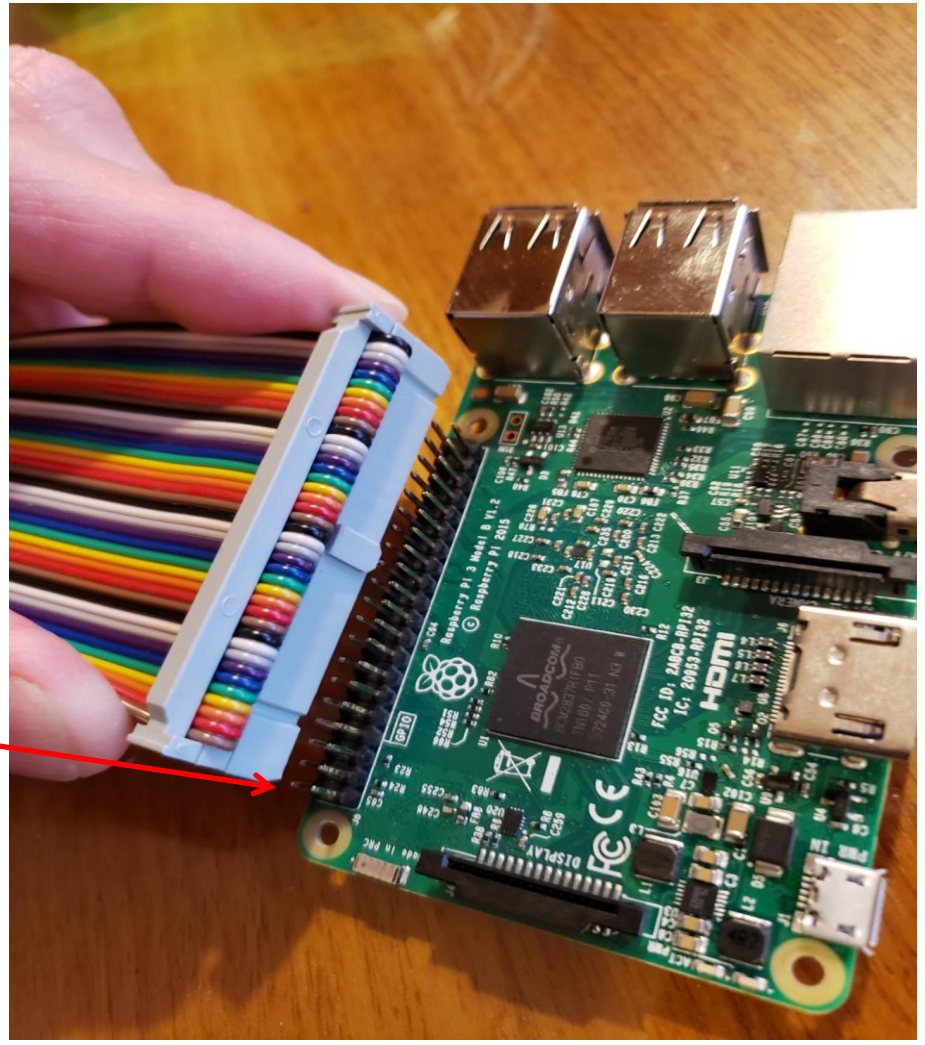
Properly Attaching the Ribbon Cable between the Raspberry Pi and the T-Cobbler



Align the notch on the ribbon cable with the t-cobbler

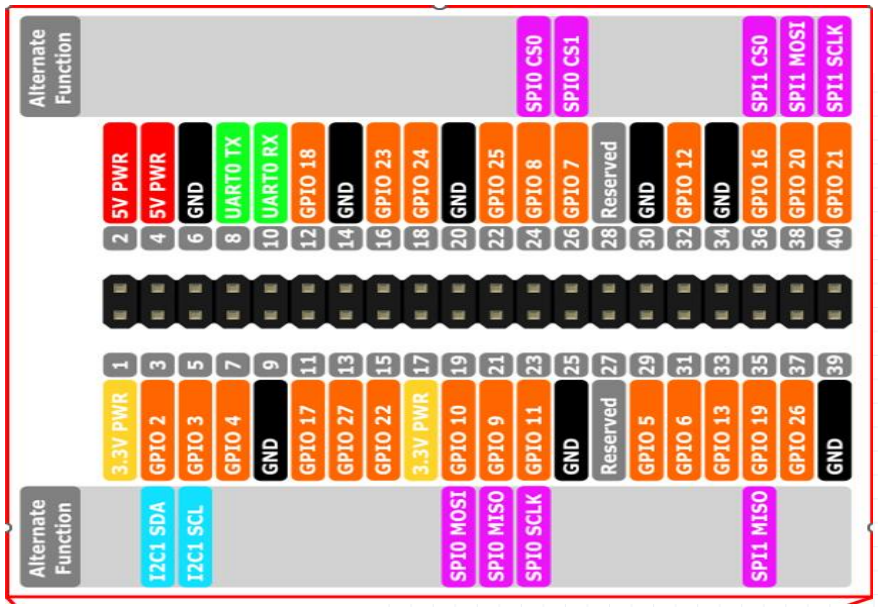
Connect the ribbon cable so it does not cross over the Raspberry Pi (as illustrated)

* Carefully align the ribbon cable to connect all 40 pins of the GPIO header.



Exercise: Construct the Doorbell Device (30 mins)

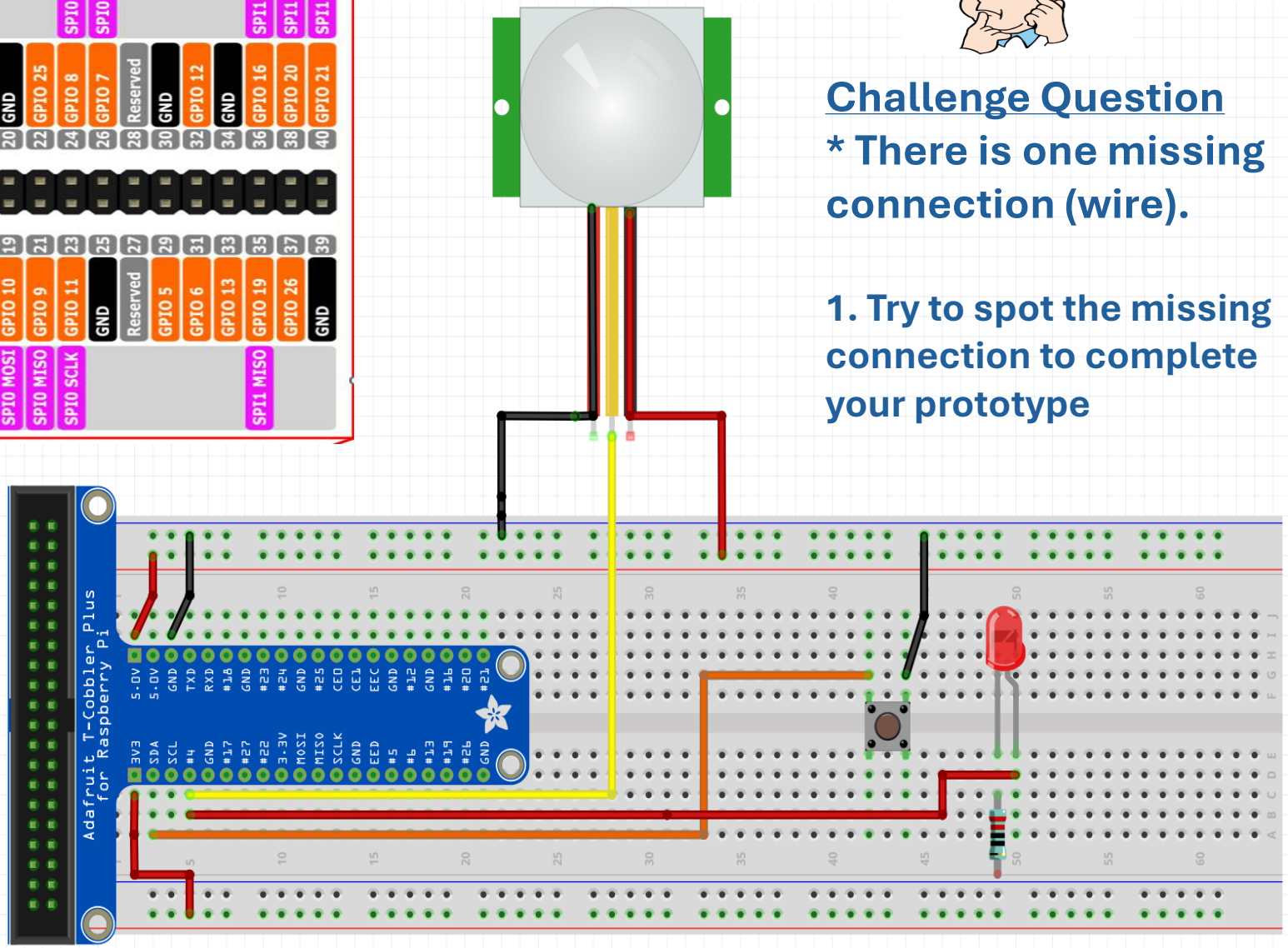
Exercise: Construct the “Smart” Doorbell Device



Challenge Question

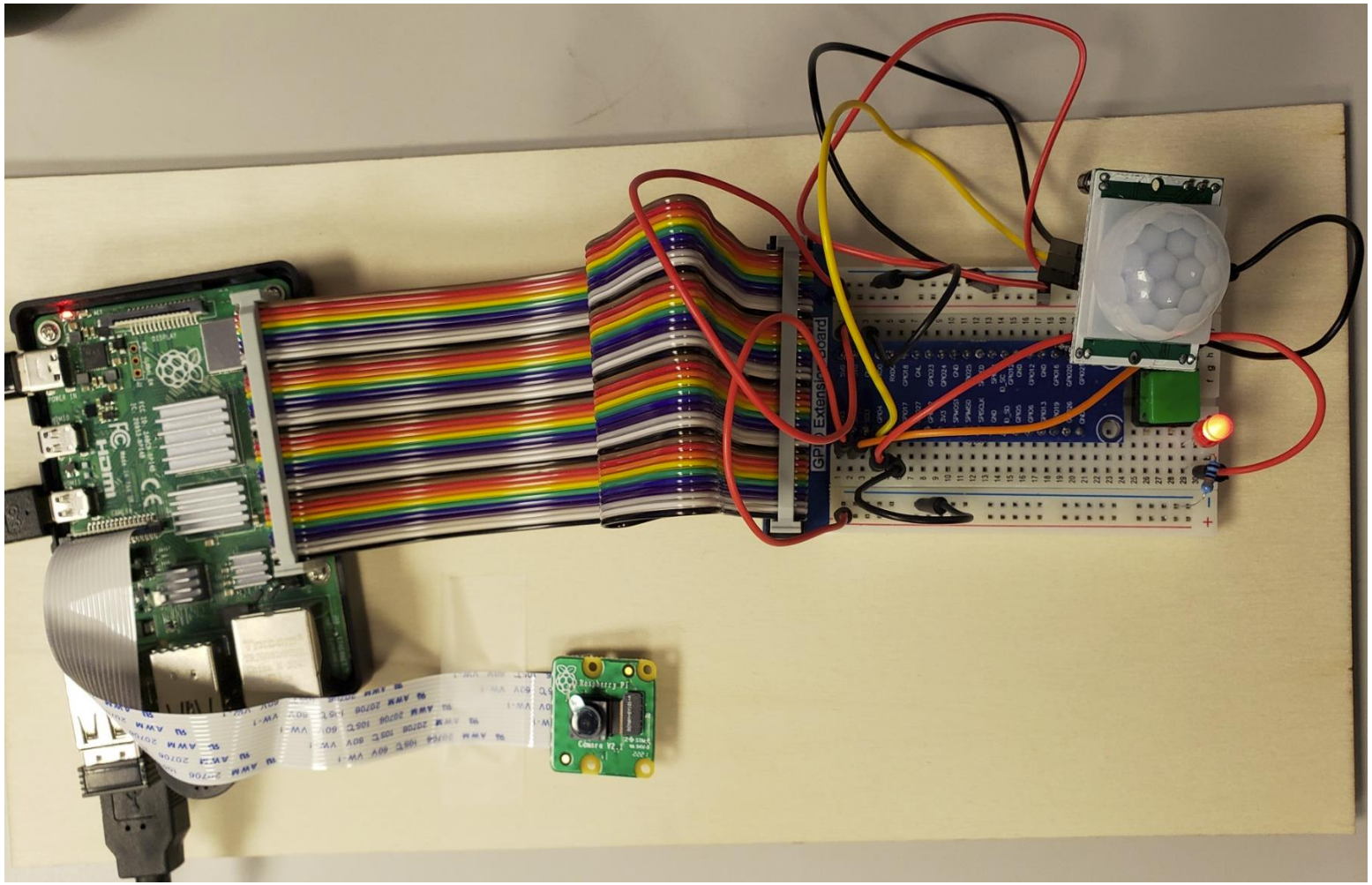
*** There is one missing connection (wire).**

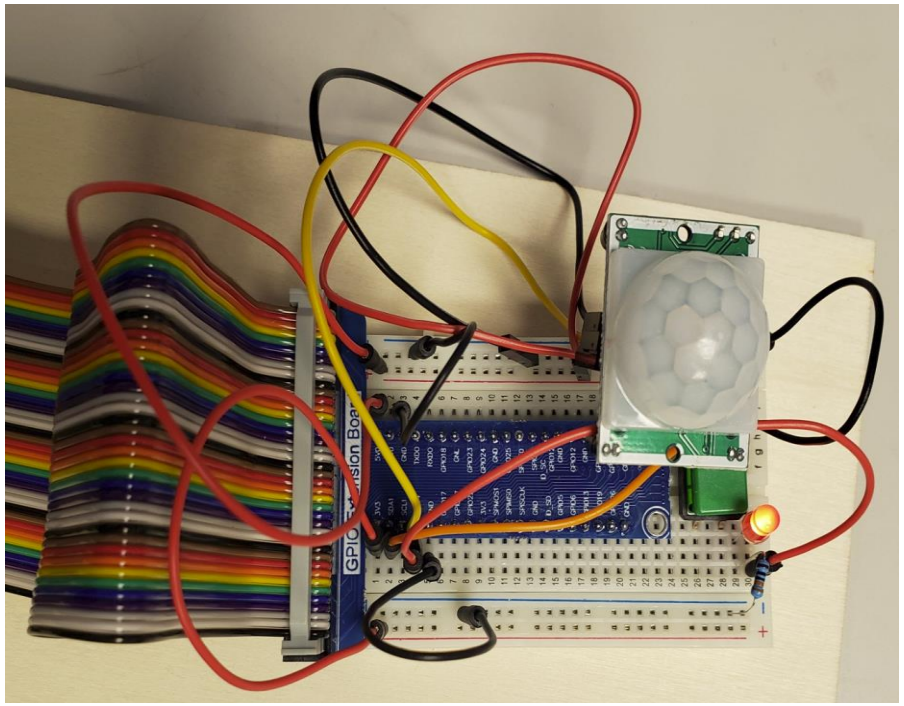
1. Try to spot the missing connection to complete your prototype



* With the power disconnected, attached the camera to raspberry Pi exactly as shown

The Completed “Smart” Doorbell Device





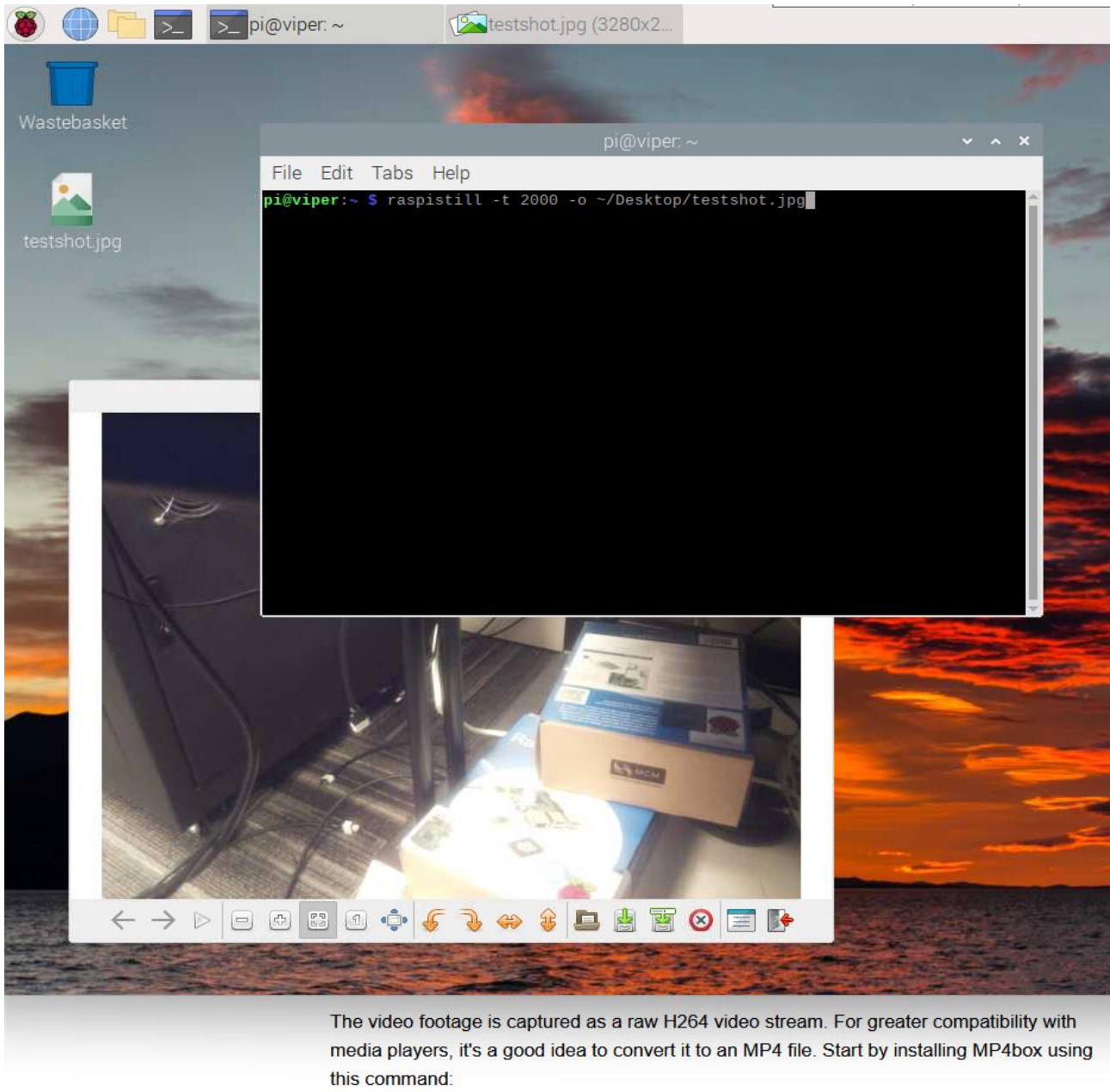
Attach the Device to the Raspberry Pi, and power it on
Note: the LED should illuminate when the PIR sensor detects motion

Test The Camera

- Establish and VNC or SSH session with the Raspberry Pi
- In the terminal, run the command:
  raspistill -t 2000 -o ~/Desktop/testshot.jpg

- If the operation is successful, the command will return and generate a file: testshot.jpg on Desktop (as shown)

- If the command is returns error such as “no camera available”
  1. Make sure the camera is connected properly (with the Raspberry Pi powered off)
  2. Run sudo raspi-config and ensure that Legacy camera is enabled



MODULE 3: IMPLEMENT THE “SMART” DOORBELL SOFTWARE

RING & RUN STEM EVENT.

ORION
OPEN ARCHITECTURE RESILIENT IOT FOR OPERATIONAL NETWORKS

REVIEW:

- 1. Provisioned (wrote and configured the raspberry Pi Image for use)
- 2. Constructed the smart doorbell hardware
- Today develop the software components

IOT ENABLED “SMART” DOORBELL CONCEPT: SETTING DEVICE CONTROL WITH MQTT

CA cert

Server cert

verify

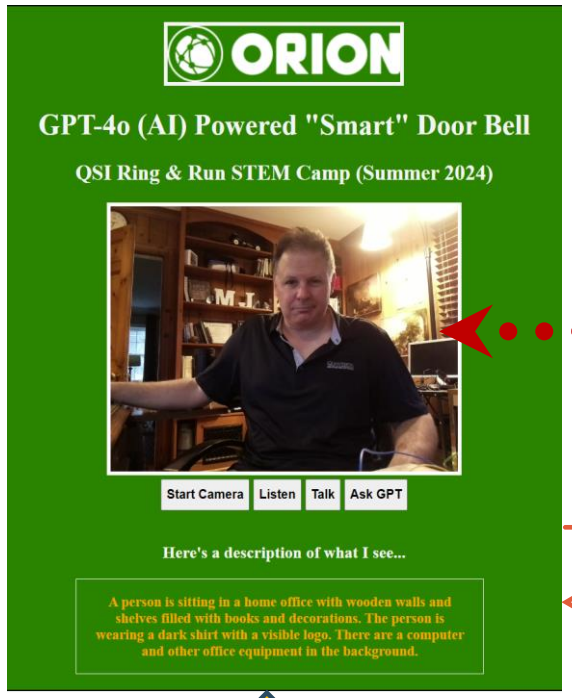
Client App
(Front-End)

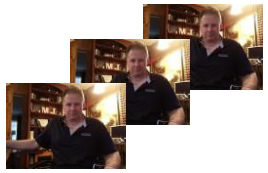
Web app: (based on HTML5 CSS and JavaScript)

Port 8001

Python Back-end

Supported modes:
motion and manual (button press)





HTTPS MJPG video stream

Publish to topic: *ring/#*

device command set

Subscribe to topic: *ring/#*

Publish to topic: *ring/#*

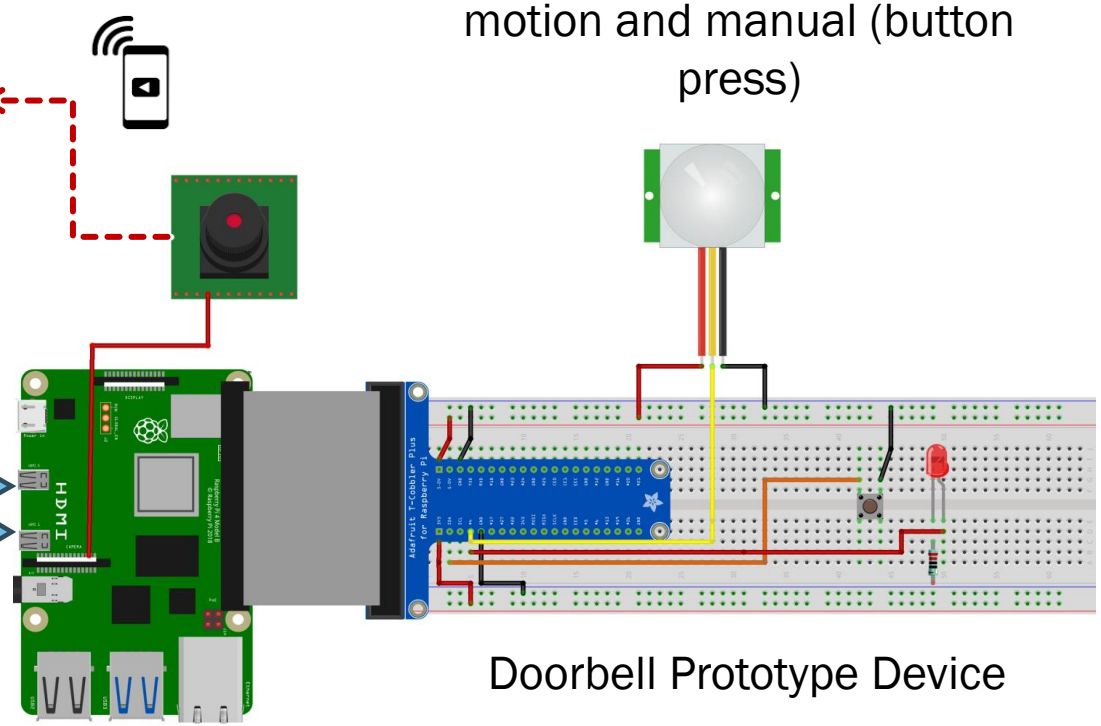
device command set

Subscribe to topic: *ring/#*

Mosquitto (MQTT)Broker

Port 1883, 8883

MQTT over Web Socket's



Raspberry Pi 4B

Doorbell Prototype Device

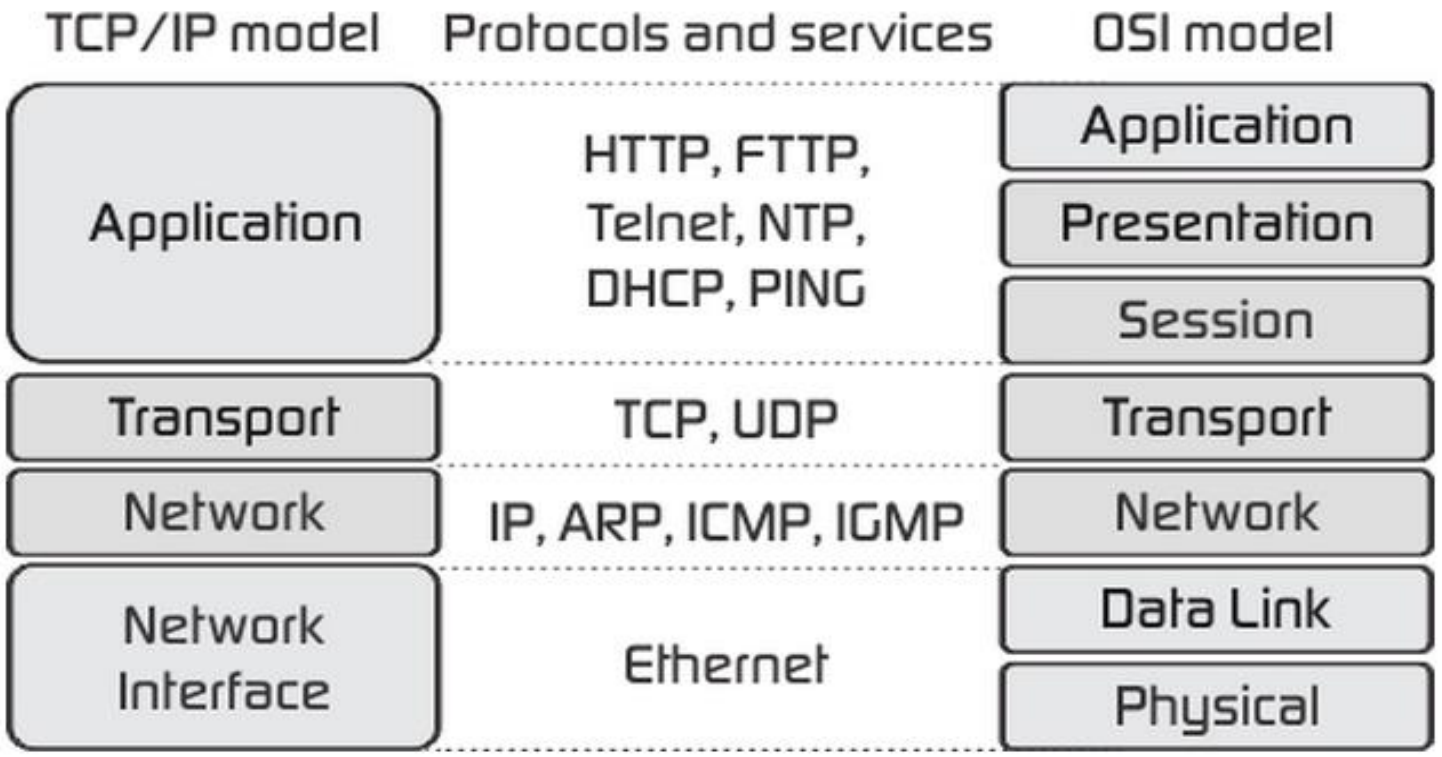
fritzing

OpenAI

https://openai.com/

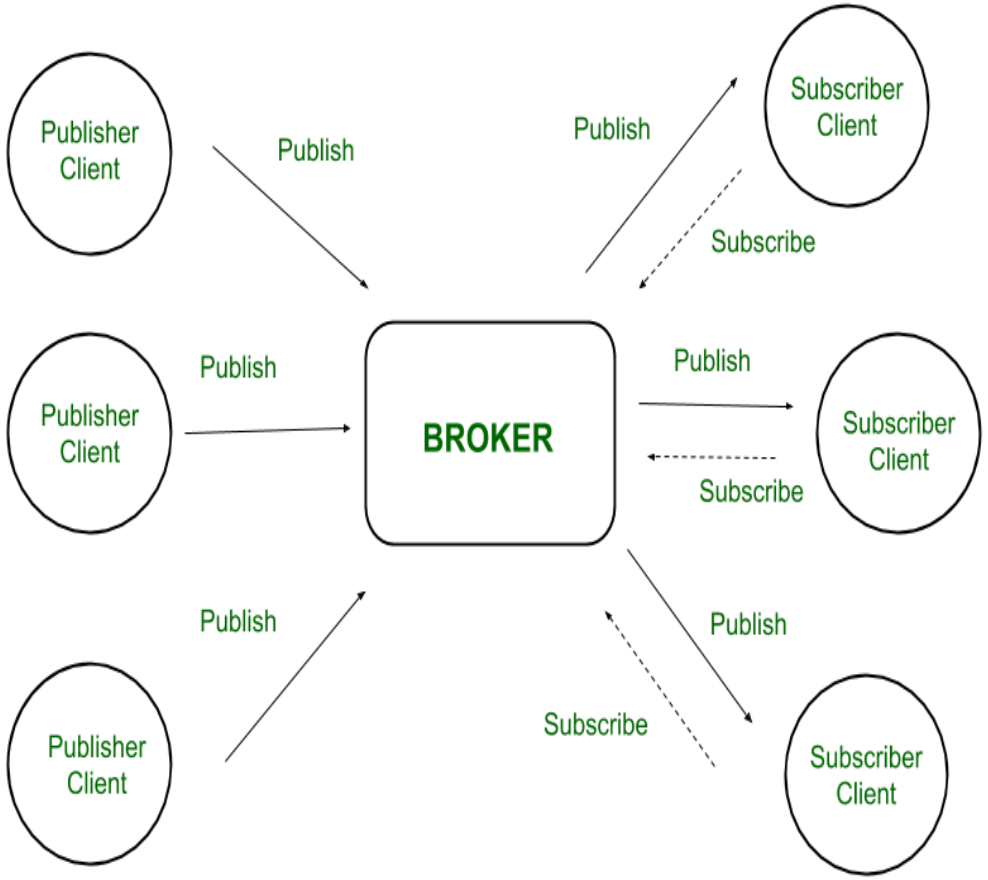
TCP/IP, HTTP, HTTPS (TLS), MQTT

- Computer Networks: https://www.youtube.com/watch?v=kn7ei2ENJbI
- Video: (TCP/IP): https://www.youtube.com/watch?v=vKFLgmSC6do
- Video: https://www.youtube.com/watch?v=PpsEaqJV_A0
- Video: https://www.youtube.com/watch?v=P6SZLcGE4us



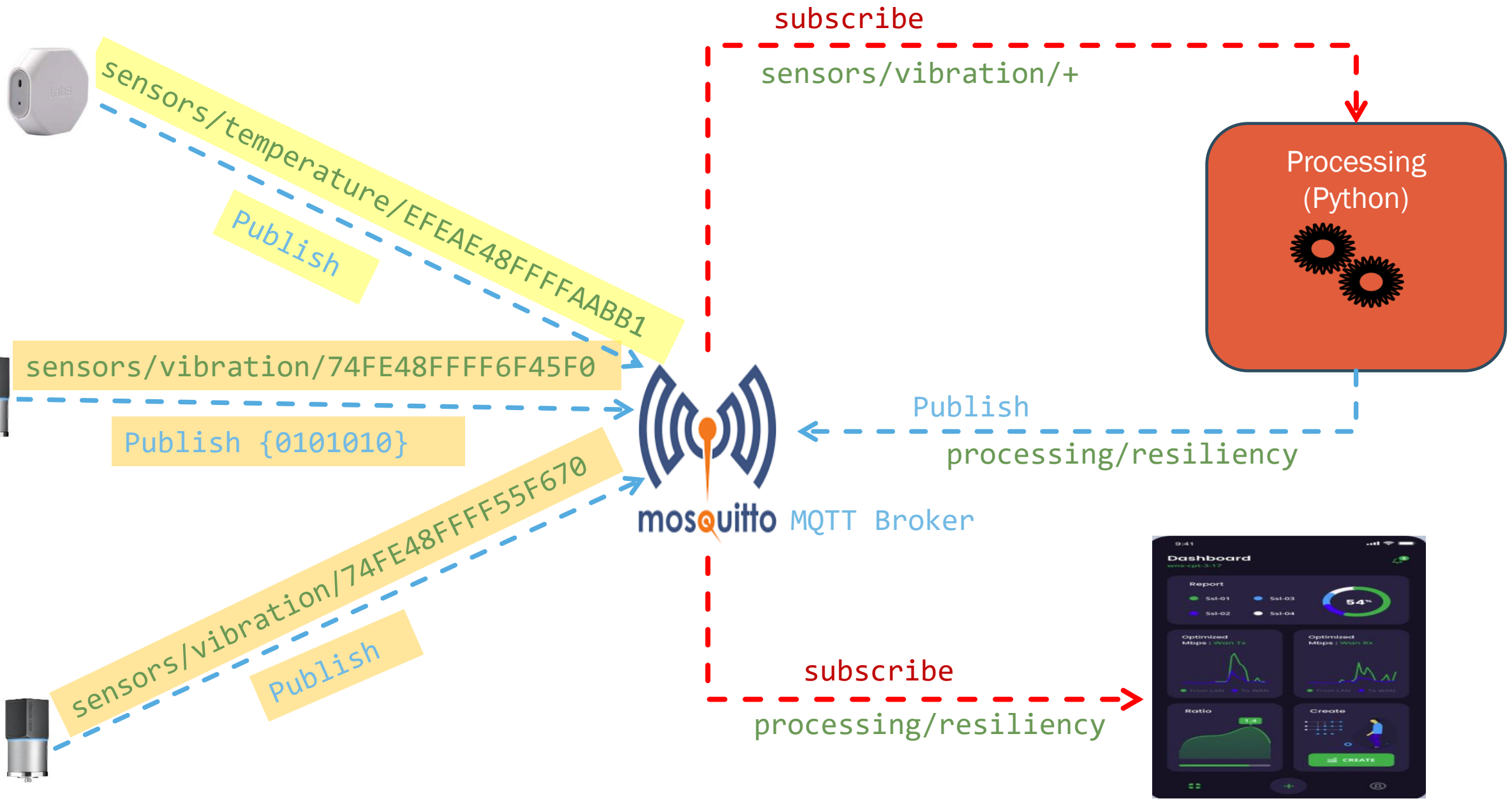
INTEGRATE MQTT SUPPORT

- MQTT stands for **M**essage **Q**ueuing **T**elemetry **T**ransport - it's a simple messaging protocol, designed for constrained devices with low bandwidth. – [1]
  - Perfect for exchanging data between IoT devices.
    - MQTT video: https://www.youtube.com/watch?v=eS4nx6tLSLs
- https://mqtt.org/
- https://learn.sparkfun.com/tutorials/introduction-to-mqtt/all
- https://www.geeksforgeeks.org/fundamental-features-of-mqtt/
- https://randomnerdtutorials.com/how-to-install-mosquitto-broker-on-raspberry-pi/
- MQTT over web sockets (JavaScript) tutorial
  - http://www.steves-internet-guide.com/using-javascript-mqtt-client-websockets/



Source: https://www.geeksforgeeks.org/introduction-of-message-queue-telemetry-transport-protocol-mqtt/ - [1]

MQTT Concept Overview



MQTT Publish/Subscribe Topic Structure

```
match individual (single) sensor subscriptions
# sensors/vibration/74FE48FFFF7A1C6F
# sensors/vibration/74FE48FFFF6F4AD1
# sensors/vibration/74FE48FFFF6F45F0
# sensors/vibration/74FE48FFFF55F670

# sensors/sound/E8E1E1000105034E
# sensors/sound/E8E1E1000105036D

# match all levels of hierarchy after #  (all sensors)
# sensors/#

# match a single level of hierarchy after + (all vibration sensors)
# sensors/vibration/+

# match a single level of hierarchy after + (all sound sensors)
# sensors/sound/+
```

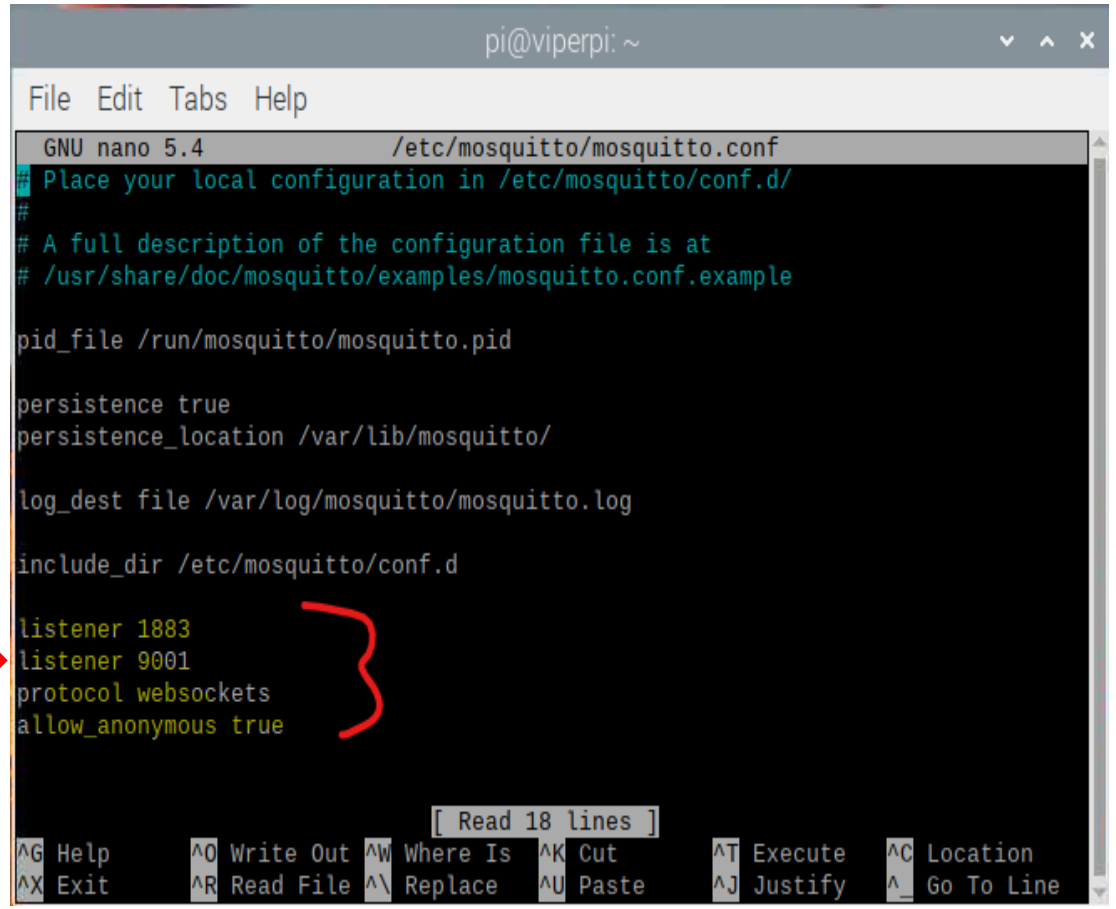
MQTT BROKER CONFIGURATION

- Configure the MQTT Broker to use Web sockets
  - Start (or use existing) SSH session with the Raspberry Pi
  - Install Mosquitto MQTT Broker

    sudo apt install -y mosquitto mosquitto-clients
  - Configure Mosquitto auto start when the Raspberry Pi boots
    - *sudo systemctl enable mosquitto.service*
  - Use *nano* or *vi* to open the configuration file:

    sudo nano /etc/mosquitto/mosquitto.conf
- Append the highlighted configuration to the end of the and press save:

  listener 1883
  listener 9001
  protocol websockets
  allow_anonymous true

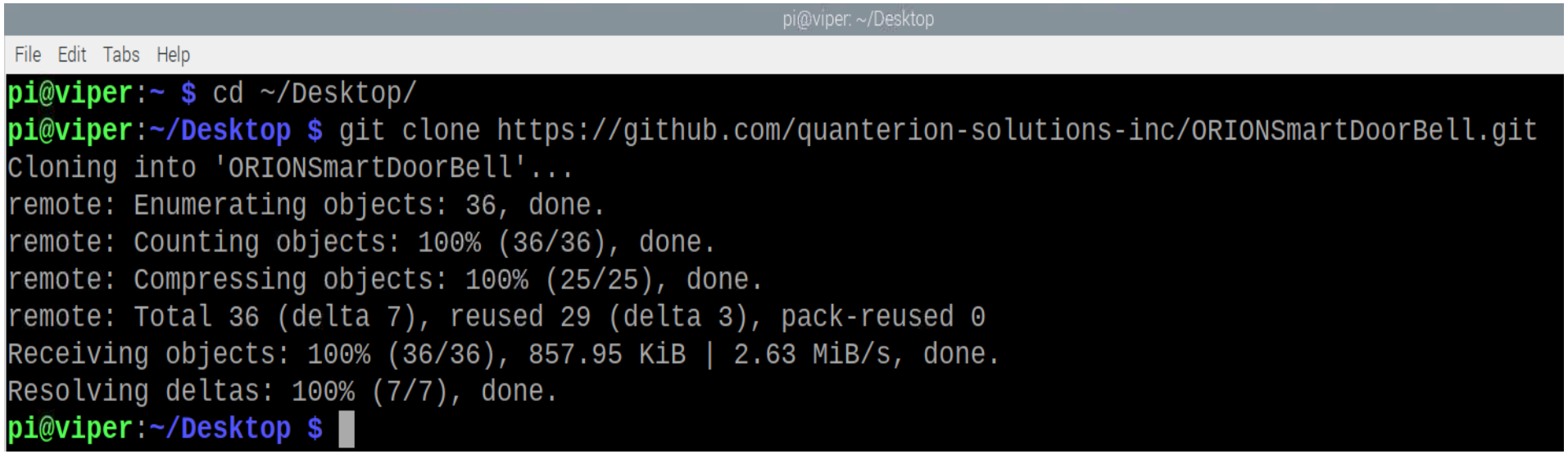


  - For save operation with nano, you need to press Ctrl+o (lowercase “o”) <enter> (which is not intuitive).
- Restart the service: *sudo systemctl restart mosquitto*
  - *Note: if the command returns a message, it indicates a problem with configuration file*

Download the Doorbell Project Source Code

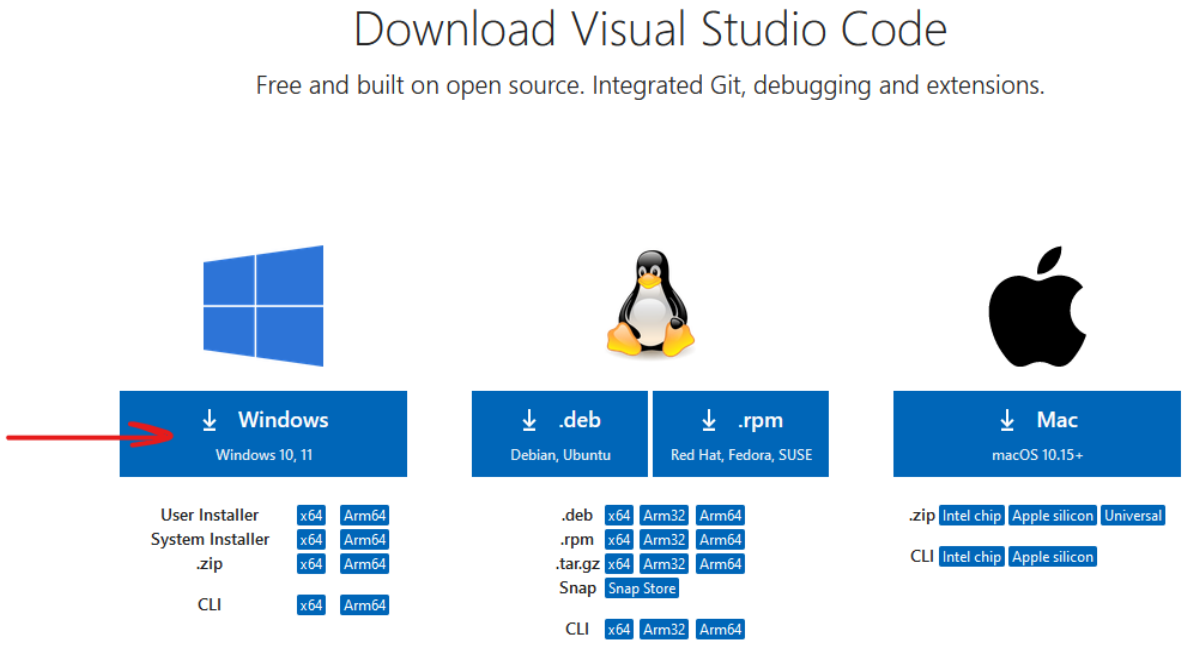
1. Establish a new VNC or SSH to the Raspberry Pi
2. Open a command terminal and navigate to the Desktop
   - clone the GitHub project repository with the following command:

```
cd ~/Desktop
git clone https://github.com/quanterion-solutions-inc/ORIONSmartDoorBell.git
```



Install Visual Studio Code (VSCode) Editor

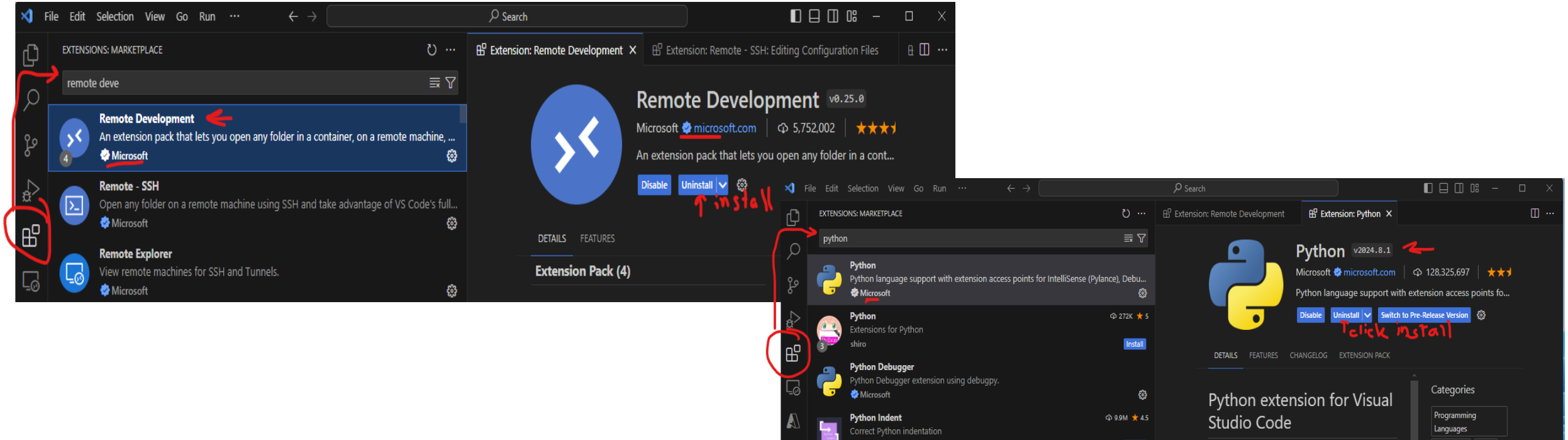
- Download and install Visual studio code for Windows to your development laptop:
  - https://code.visualstudio.com/download
    - VSCode is the most popular software development code editor in use t



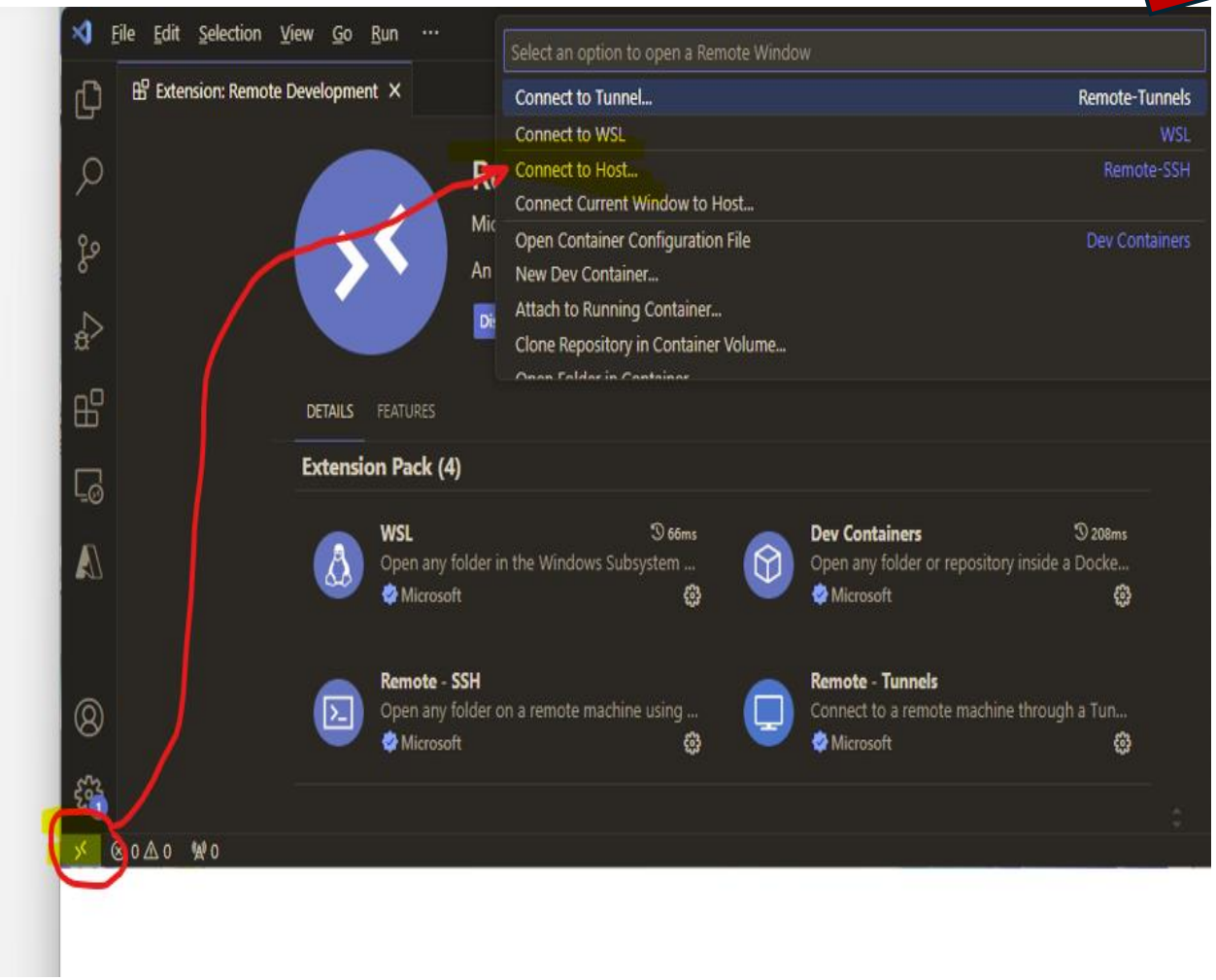
Install VSCode Extensions

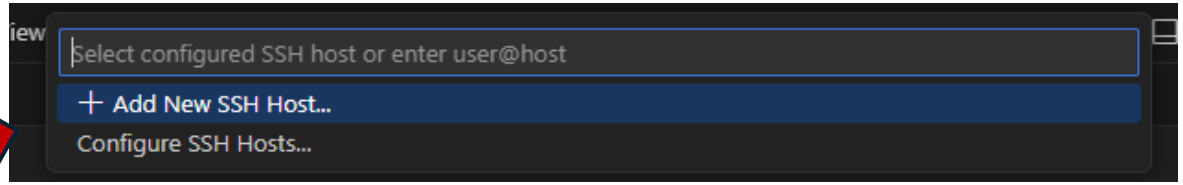
- Click the Extensions and type the extension name is search box to install the following extensions (see illustration)
    1. “Remote Development extension pack” -- provides an SSH managed session for transparent remote develop
    2. “Python” - provides support for IntelliSense (Pylance), debugging (Python Debugger), formatting, linting, code navigation, refactoring, variable explorer, test explorer, and more!

Note: there are many extensions with similar names provide by 3rd parties. Be certain the extensions installed are those provided by Microsoft

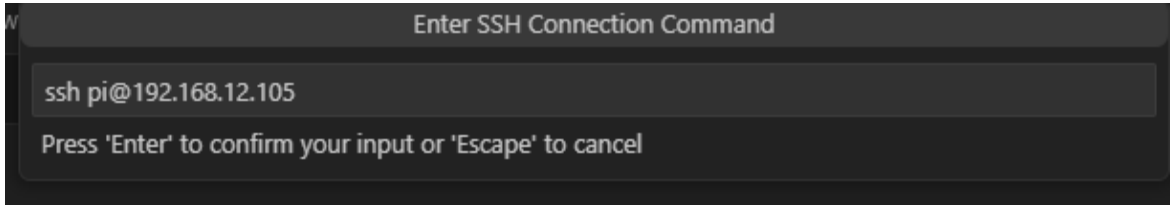


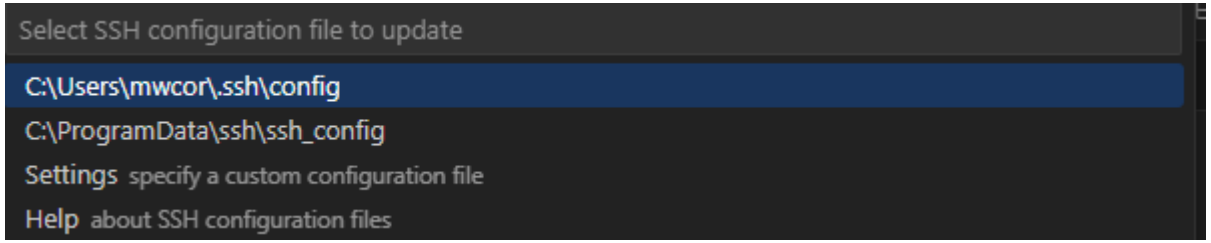
Establish a Remote Connection





Enter familiar “ssh” command: *use the IP address of Raspberry Pi* (as shown)

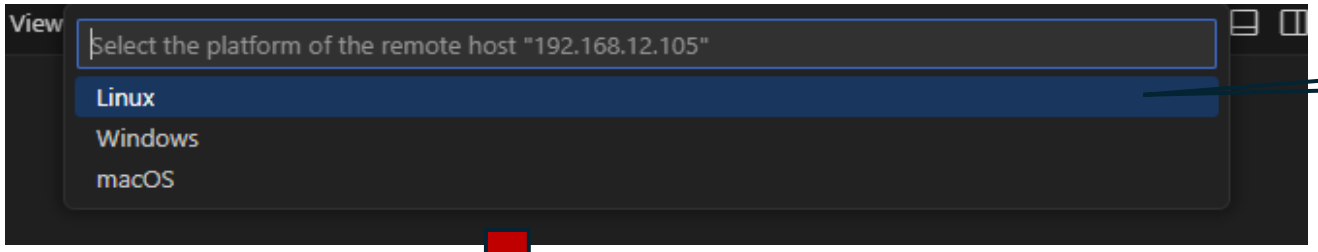




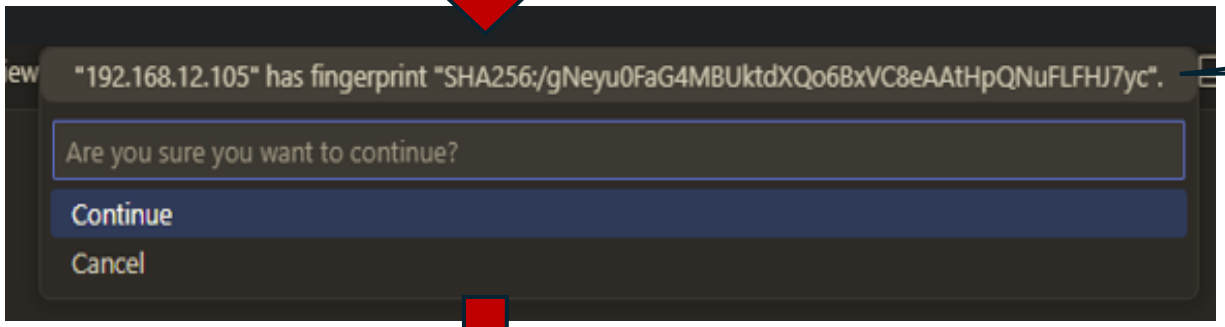
Next slide

Establish Remote Connection (cont.)

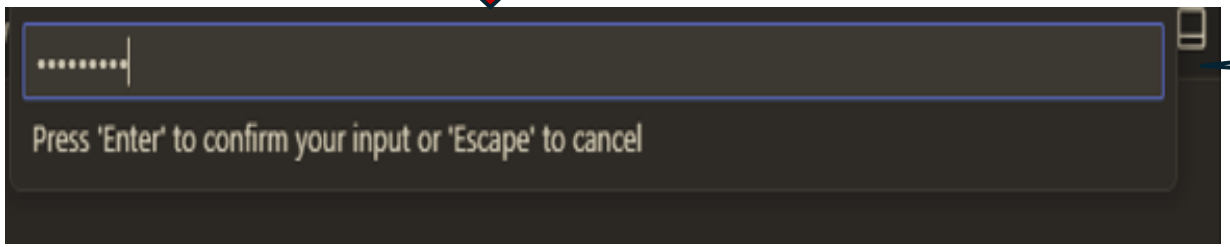
click connect



Select Linux
press Enter

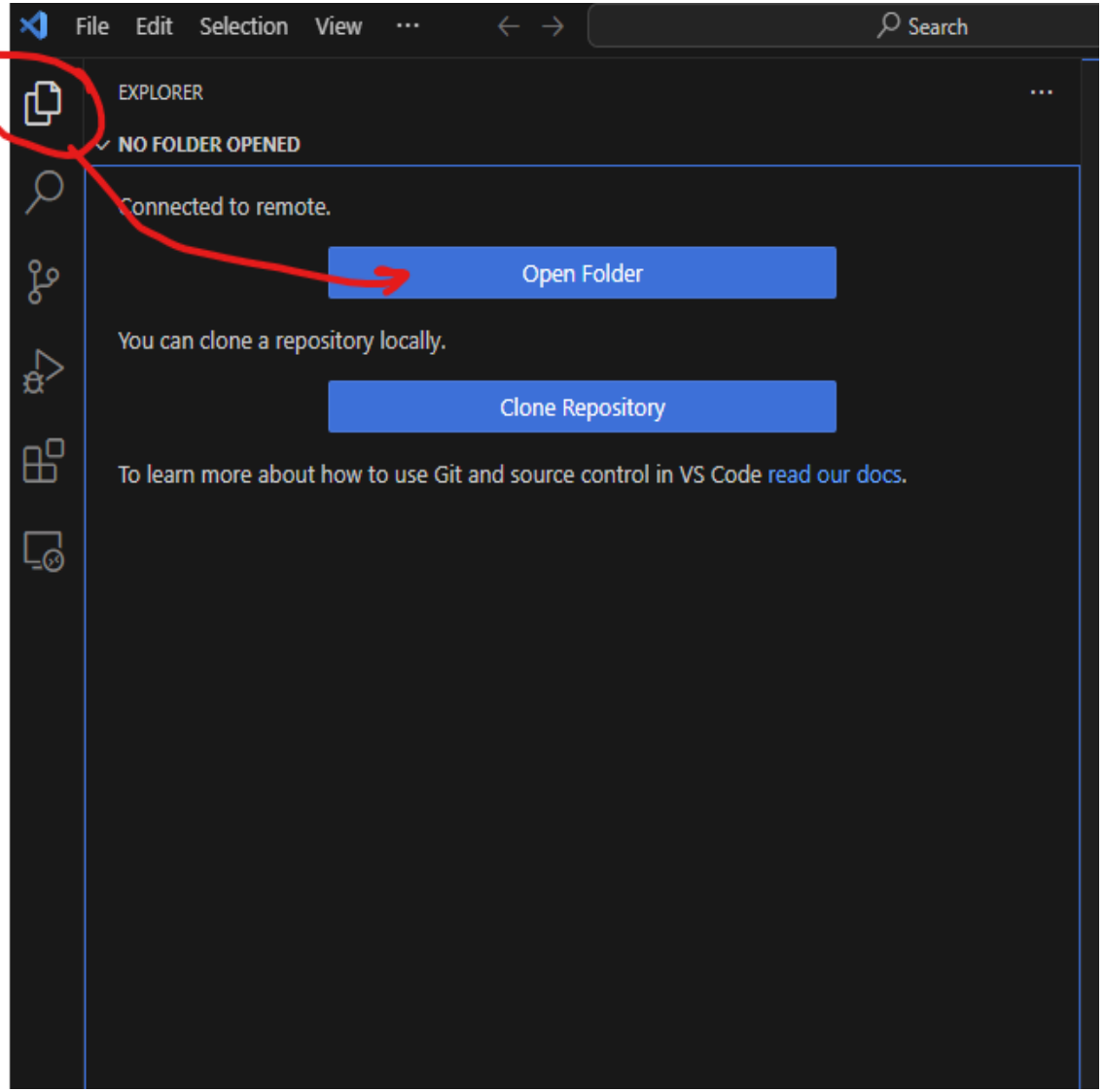


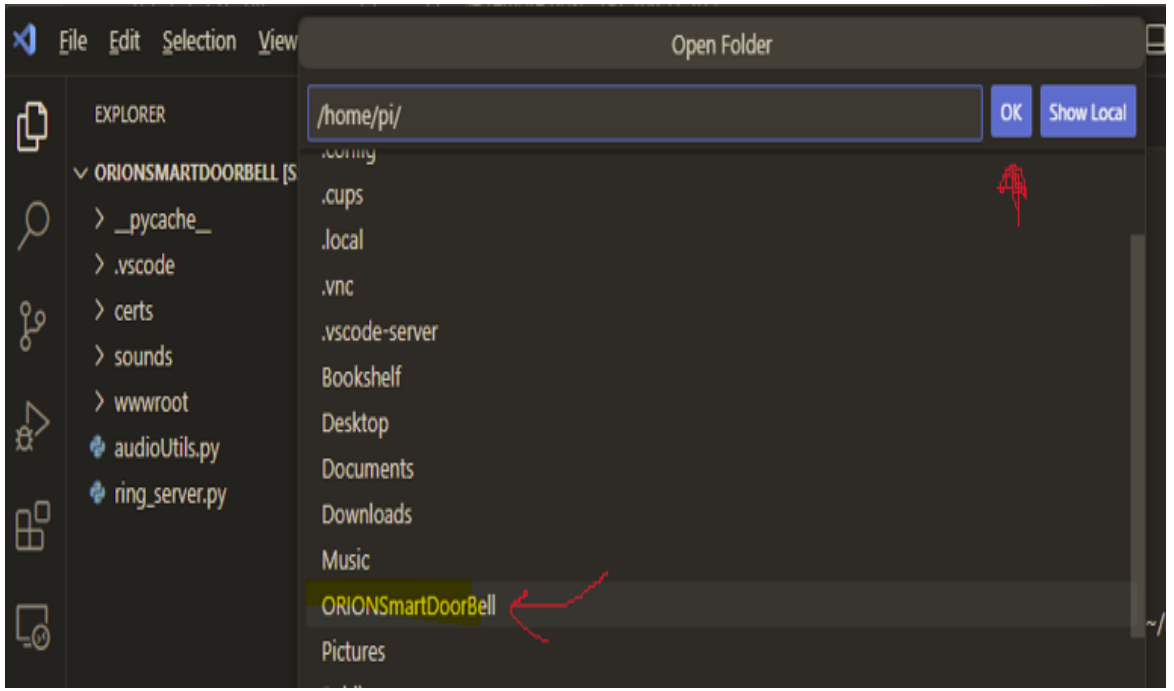
Select
Continue
Press Enter



Enter the
Raspberry PI
password

Open the Doorbell Git Project (source code) Folder



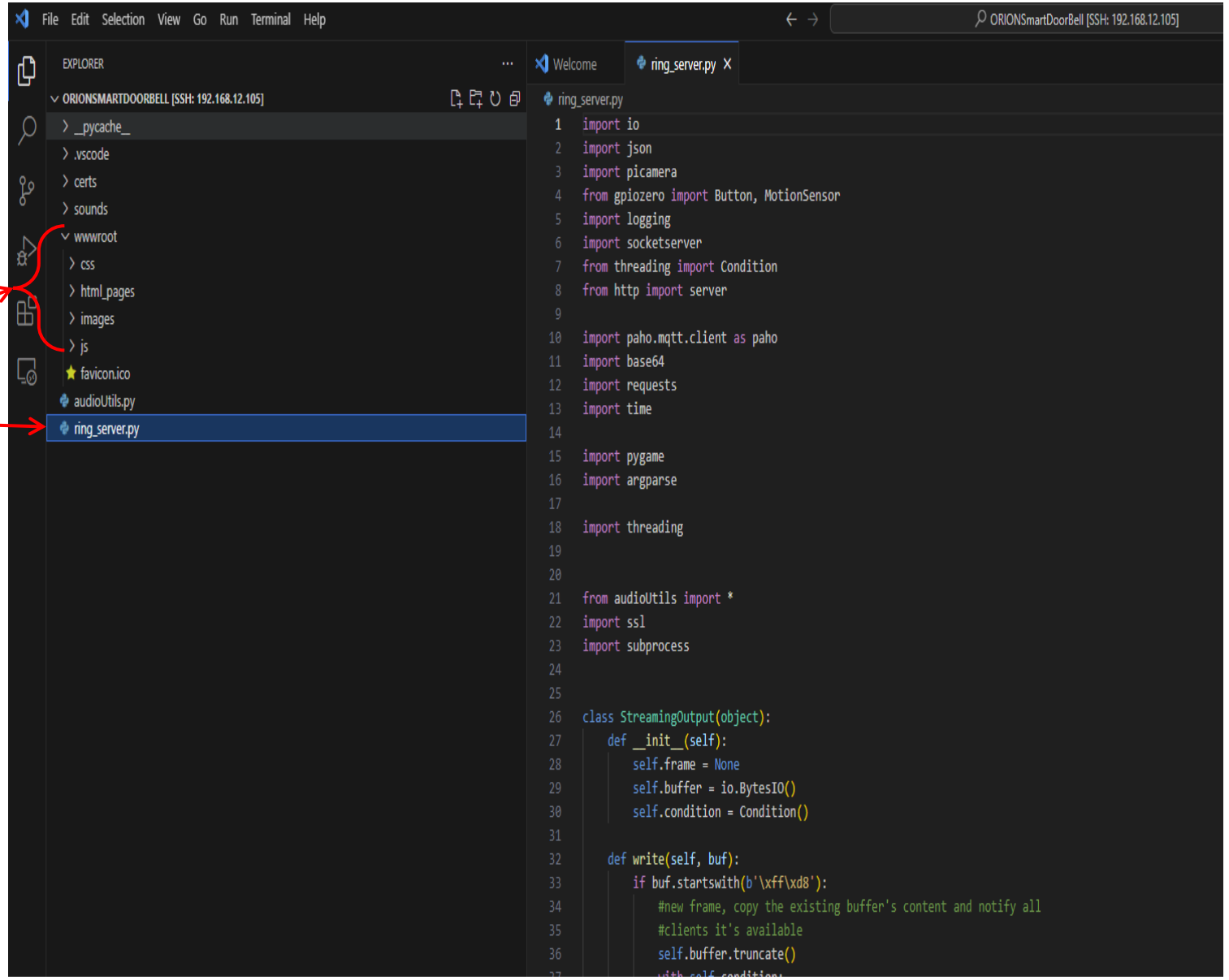


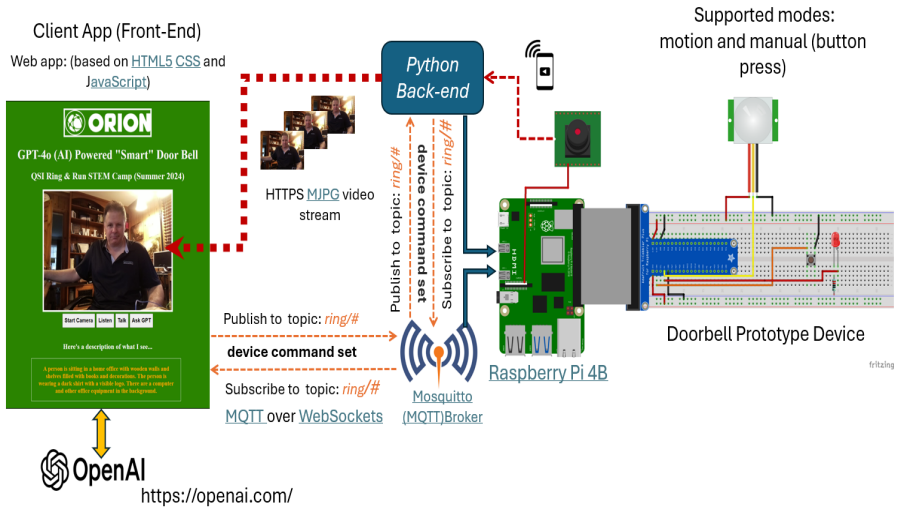
Remote Development Host

* Note: this view being served from the Raspberry Pi to your Remote Dev Host Laptop over a managed SSH session

Doorbell code consists of two parts:

1. Client Application (front-end)
2. Server (back-end)



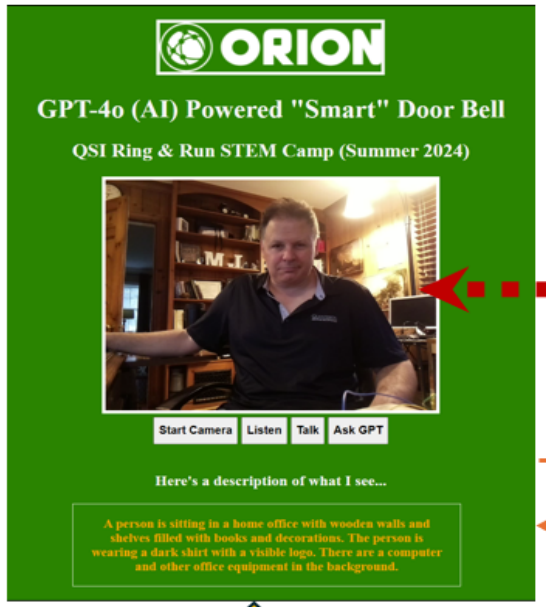


Front-end (web) App

Back-end (Server) Device

Client App (Front-End)

Web app: (based on HTML5 CSS and JavaScript)



OpenAI

https://openai.com/

HTTPS MJPG video stream

Python Back-end

Publish to topic: ring/#

device command set

Subscribe to topic: ring/#

Publish to topic: ring/#

device command set

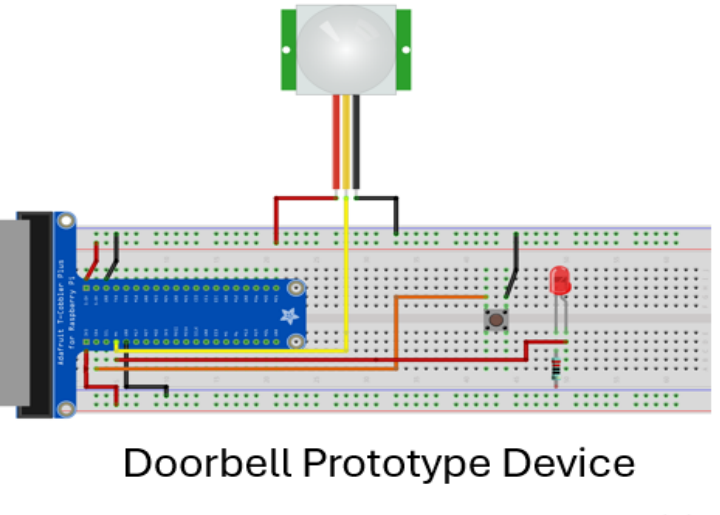
Subscribe to topic: ring/#

MQTT over WebSockets

Mosquitto (MQTT)Broker

Raspberry Pi 4B

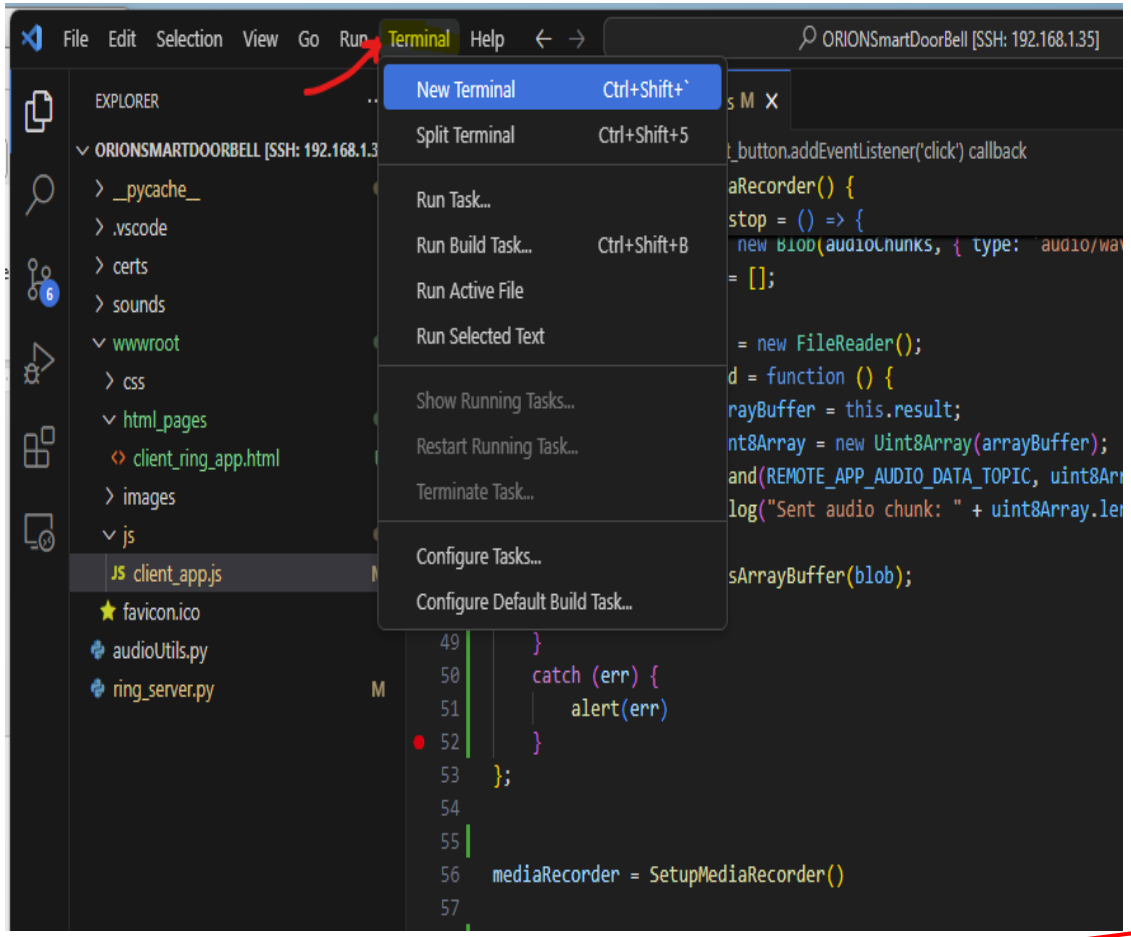
Supported modes: motion and manual (button press)

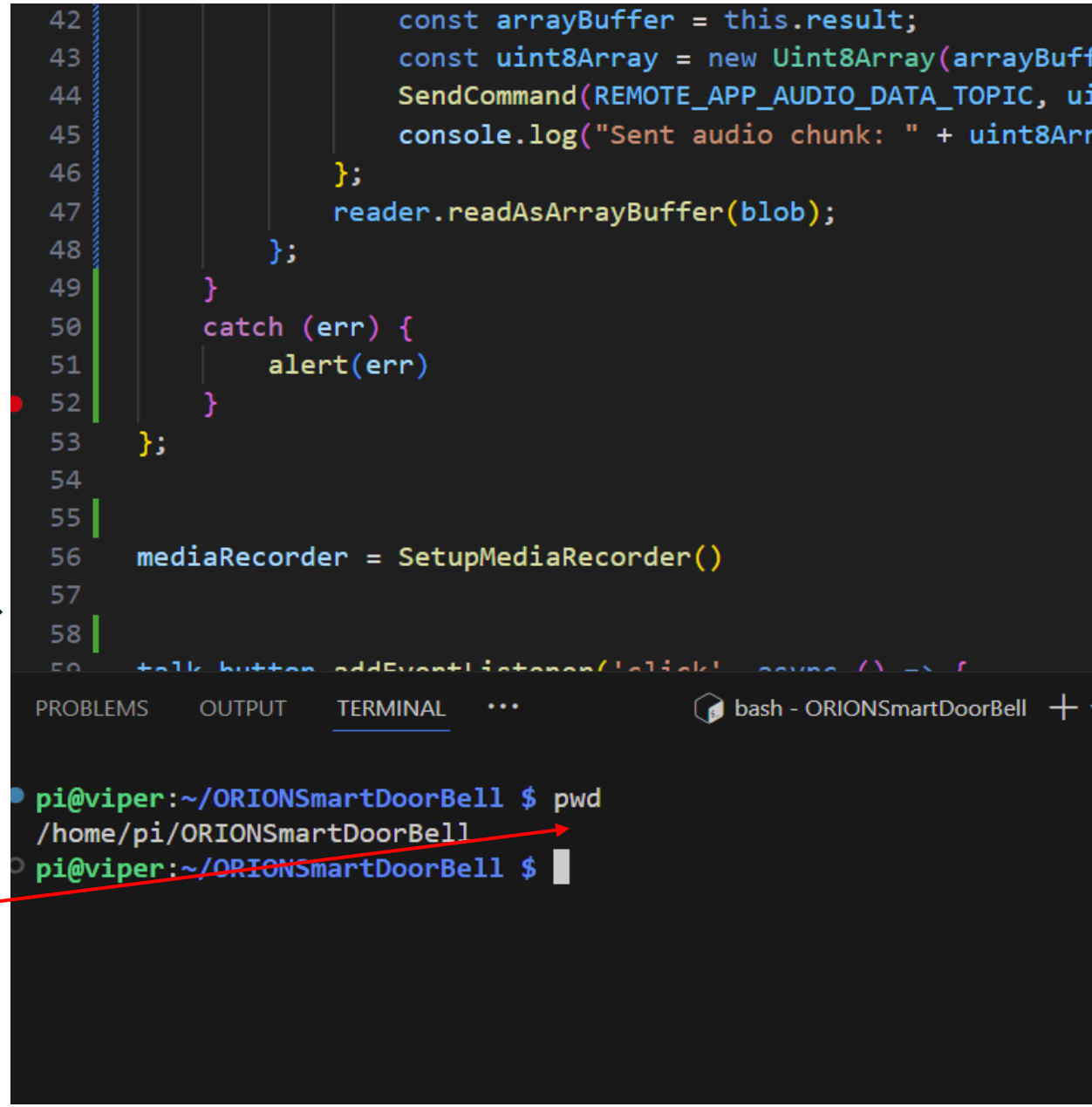


Doorbell Prototype Device

fritzing

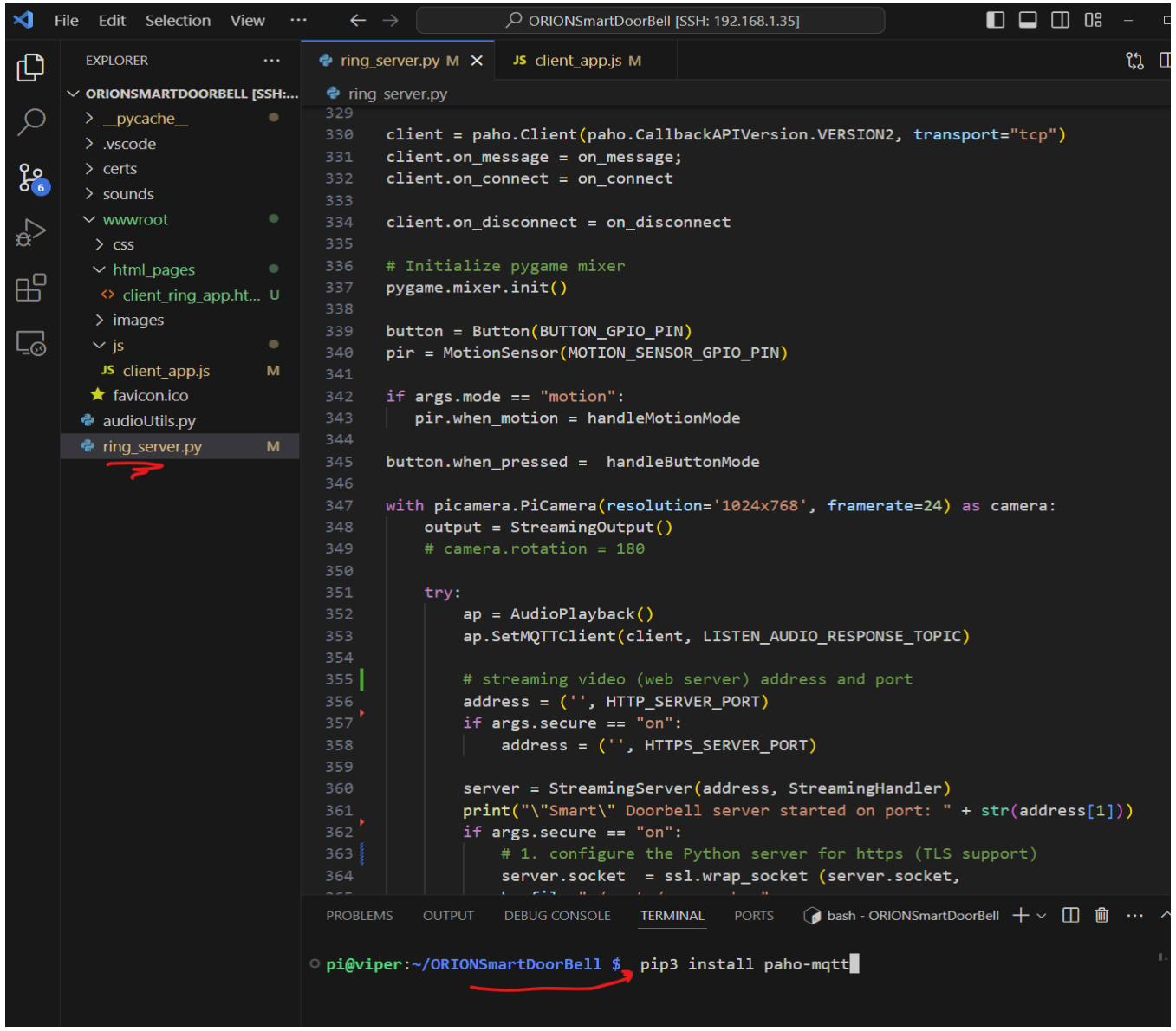
Open New Terminal Window in VSCode





Enter the command *pwd*
to print the current working directory (folder)

The back-end (Python) code in VSCode: *ring_server.py*



Install Prerequisites Python Packages:

1. Controlling the Doorbell using the JavaScript client application (from PC or mobile device) requires the server to implement MQTT functionality. For MQTT support in the server install the: paho MQTT library *package*
   Command: *pip install paho-mqtt*

2. Allowing the person at the door to communicate (speak) via a doorbell intercom requires access to the Raspberry microphone. To read and transmit audio from the Raspberry Pi over MQTT, install audio support for the pyaudio package
   Commands:
   pip install pyaudio
   sudo apt-get install portaudio19-dev python3-pyaudio

- Let's take a few mins to review the code together

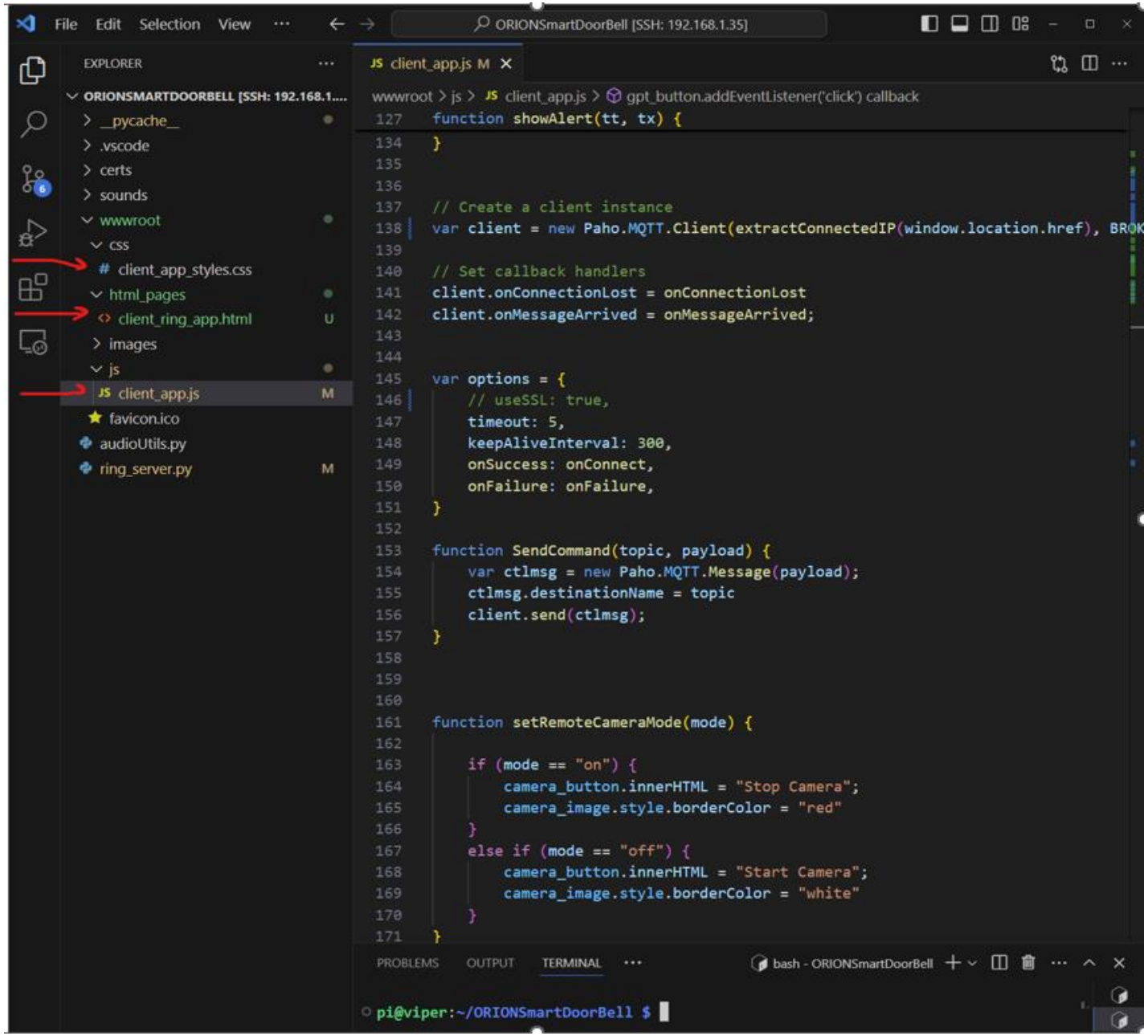
The **Front-end (Web App)** code is implemented using a combination of **JavaScript, HTML5, and CSS**

1. The project folder: *wwwroot* contains a subfolder *html_pages* with single HTML5 page file: *client_ring_app.html*
   - The HTML page defines the "structure and content" for the web page (the client app)

2. The subfolder *js* contains a single JavaScript code file: *client_app.js*
   - The JavaScript code implements the "behavior" for the client app

3. The subfolder *CSS* contains a single cascading style sheet: *client_app_styles.css*
   Cascading style sheets define the "style (visual theme)" for the client app.

Note: the JavaScript code enables the user to control the doorbell. This is implemented using an MQTT publish/subscribe message topic strategy. Because this code runs in the context of the web browser (Chrome, Firefox, Safari etc.) it implements MQTT over Web Sockets.



Supported Doorbell Server (Back-end) Modes:

The back-end server can be started in two modes: “manual” and “motion”. The command below illustrates how to start the server

python ring_server.py --mode manual --secure off 2> /dev/null

Server script name

Optional “mode” command line argument: ***manual*** *(button press) or* ***motion*** *(PIR detection) default = manual*

Optional “security” command line argument: ***on*** *configures for TLS support,* ***off*** *is clear text. Default is off*

Redirect error message from internal libraries

```
pi@viper:~/ORIONSmartDoorBell $ python  ring_server.py  --mode motion  --secure on   2> /dev/null
"Smart" Doorbell server started on port: 8001
connected over web sockets: Success
Connected to MQTT Broker on port 1883 established
Subscribed to topics
play doorbell chime
recording started
```

```
pi@viper:~/ORIONSmartDoorBell $ python  ring_server.py  --mode manual  --secure off   2> /dev/null
"Smart" Doorbell server started on port: 8000
Connected to MQTT Broker on port 1883 established
connected over web sockets: Success
Subscribed to topics
```

```
pi@viper:~/ORIONSmartDoorBell $ python  ring_server.py  --mode manual  --secure on   2> /dev/null
"Smart" Doorbell server started on port: 8001
Connected to MQTT Broker on port 1883 established
connected over web sockets: Success
Subscribed to topics
```

* Notice the port (highlighted) using –secure on versus –secure off

View Doorbell Client Application (in Unsecure mode)

- Download and install the Brave browser on the Windows laptop
  - https://brave.com/download/

- Open The “Brave” browser and copy the following URL into address bar, press enter
  - http://[IP-address]:8000/index.html
    - *Note: replace [IP-address] with IP address of the Raspberry Pi*
      - *Notice the protocol in URL is* ***http***

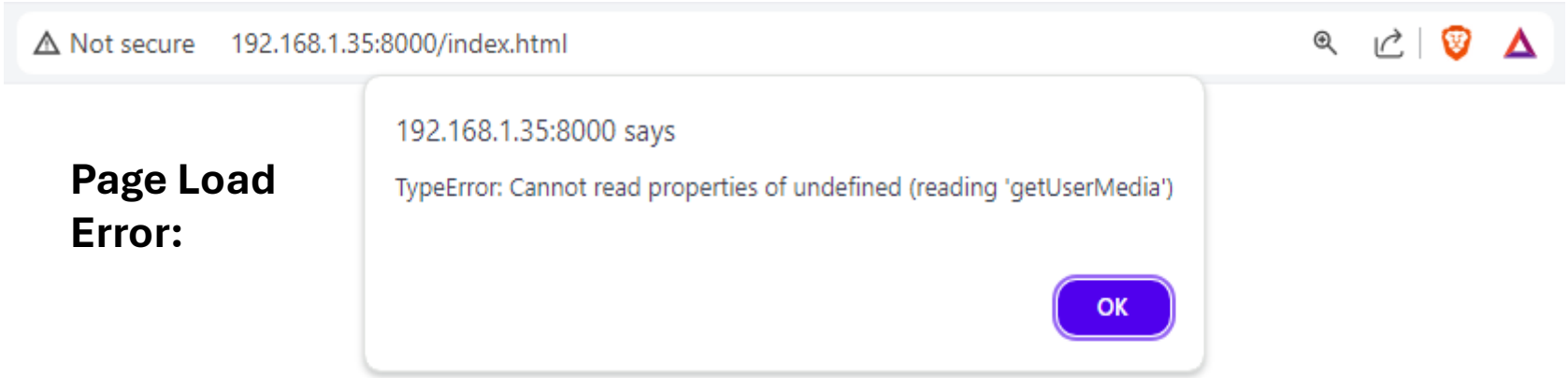
- **Questions:**
  1. Record the initial result of viewing the URL in the browser? What happened?

  2. Now, start the server(command using command below) in “manual” mode with security “off”

```
pi@viper:~/ORIONSmartDoorBell $ python  ring_server.py  --mode manual  --secure off   2> /dev/null
"Smart" Doorbell server started on port: 8000
Connected to MQTT Broker on port 1883 established
connected over web sockets: Success
Subscribed to topics
```

2.1 View The URL again (refresh by pressing Enter with cursor in address bar)
- You should receive a type error message like the following: *** See next slide→**

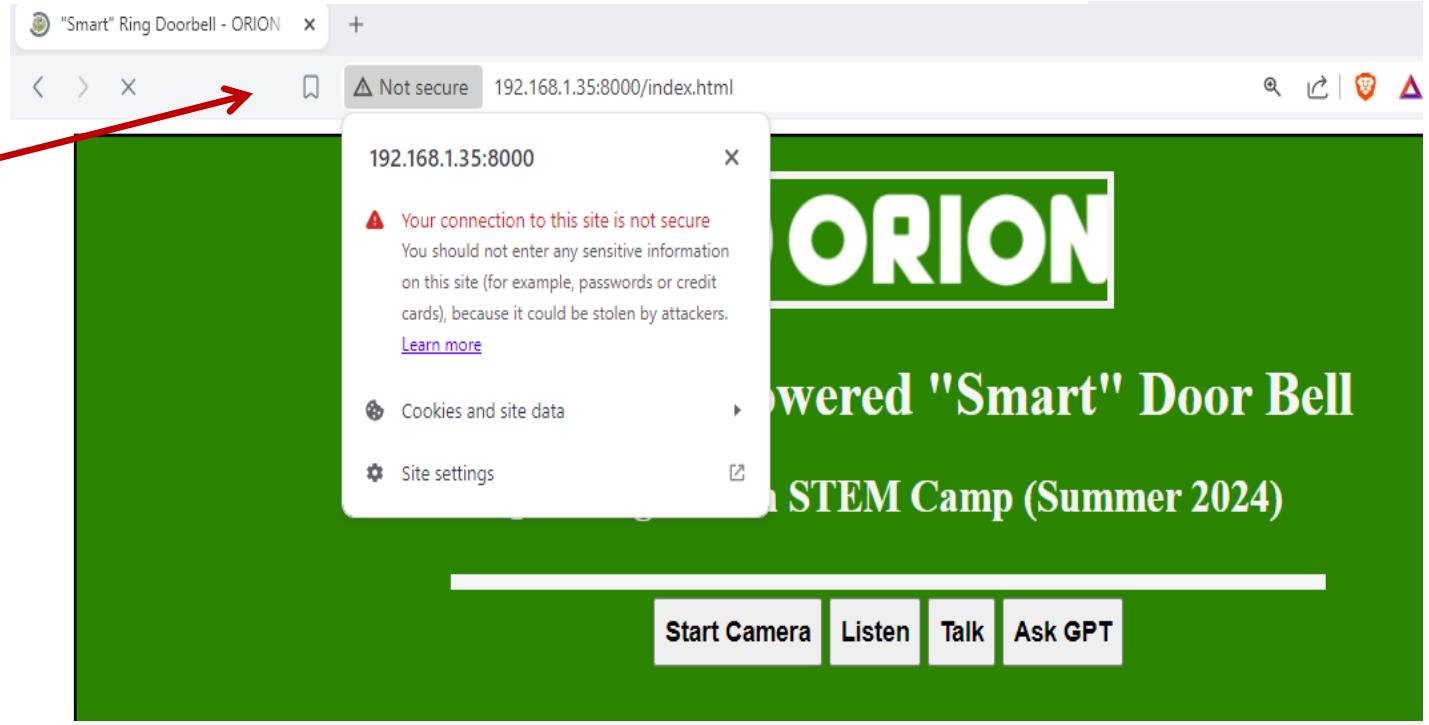
Connect to the Server using the Doorbell Client (Web) Application



Page Load Error:

Press the “Ok” button and the page should Load correctly

Note: the connection is not secure



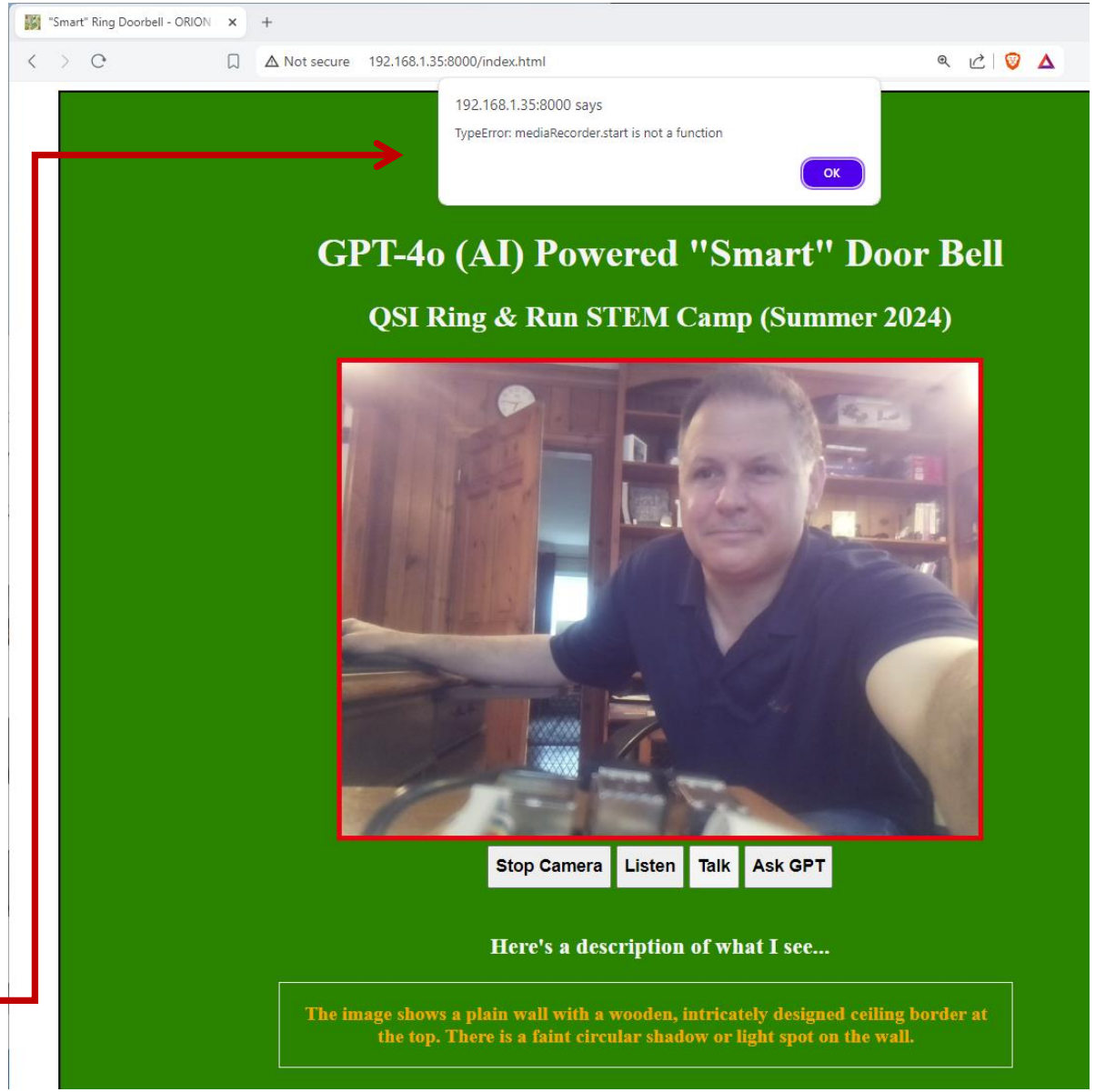
Experiment with the Doorbell Application (in Unsecure mode)

- Notice the server is exposed on TCP port 8000
- Experiment with the buttons, observe and record the behavior
  - *Note: the camera may not behave as expected on the first run. This is an unresolved bug. If this happens, then simply reload page in the browser.*

- **<u>Challenge Questions</u>**

  1. Click **"Ask GPT"** and record the result

  2. When the page first loads, and whenever you click "Talk" a JavaScript error (related to Page Load error illustrated on the previous slide) is displayed
     - Why do you do these errors occur?

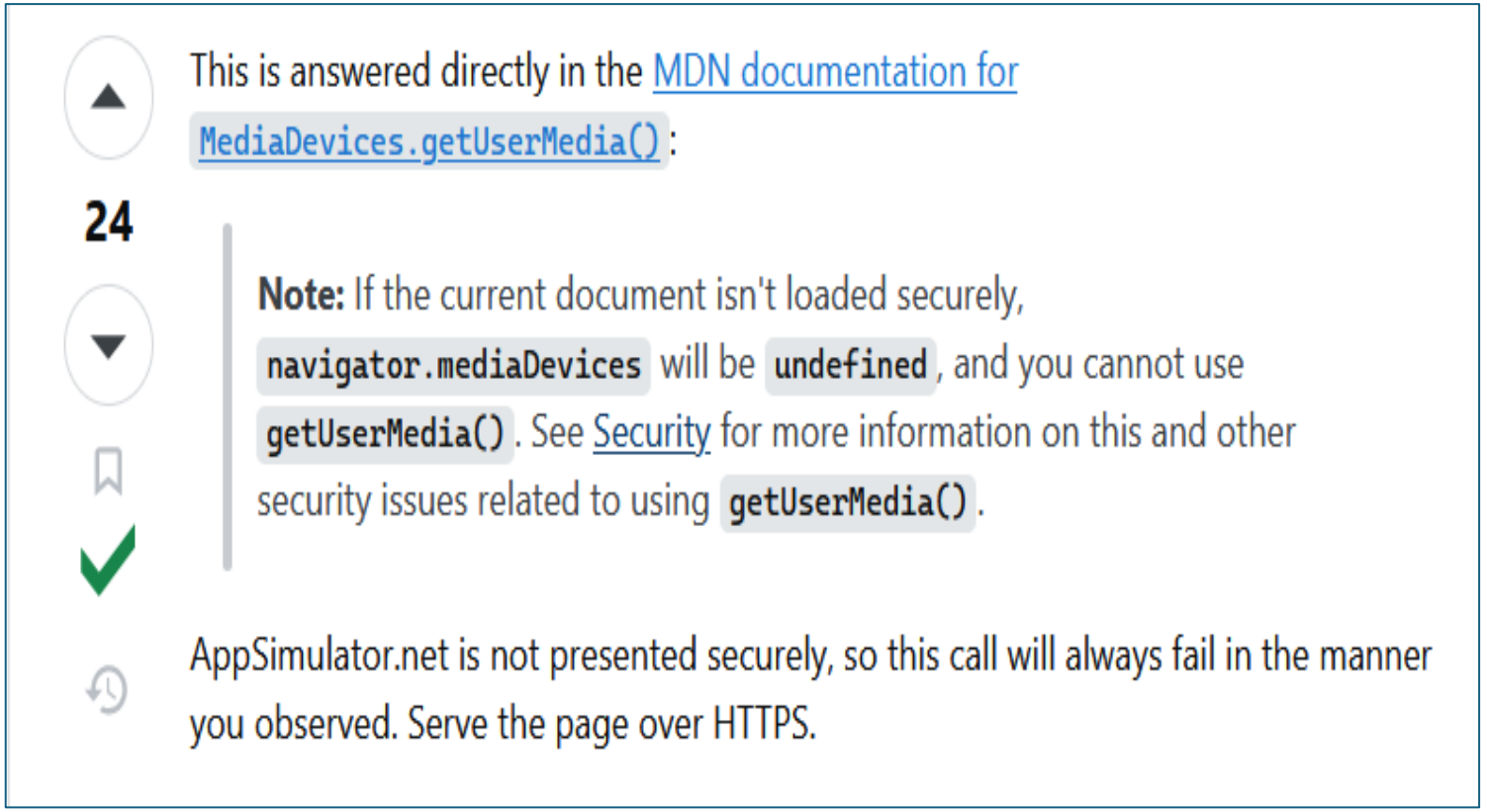


Note: press F12 in the Browser for "Developer Tools", click "Console" to see messages

Answer to Challenge Question (unsecure mode):

- If you said something to related to not using **proper security practices**, then you would be correct!

- Modern Web browser (security) policies will not allow JavaScript code to access certain browser managed media devices (such microphones) over unsecure HTTP connections



Stack Overflow Post

MODULE 4: IMPLEMENT SSL/TLS

RING & RUN STEM EVENT.

ORION
OPEN ARCHITECTURE RESILIENT IOT FOR OPERATIONAL NETWORKS

SHORT INTERLUDE: PKI AND TLS EXPLAINED

- What is PKI (Public key infrastructure) ?
  - https://www.youtube.com/watch?v=uVaUgrxjMe0

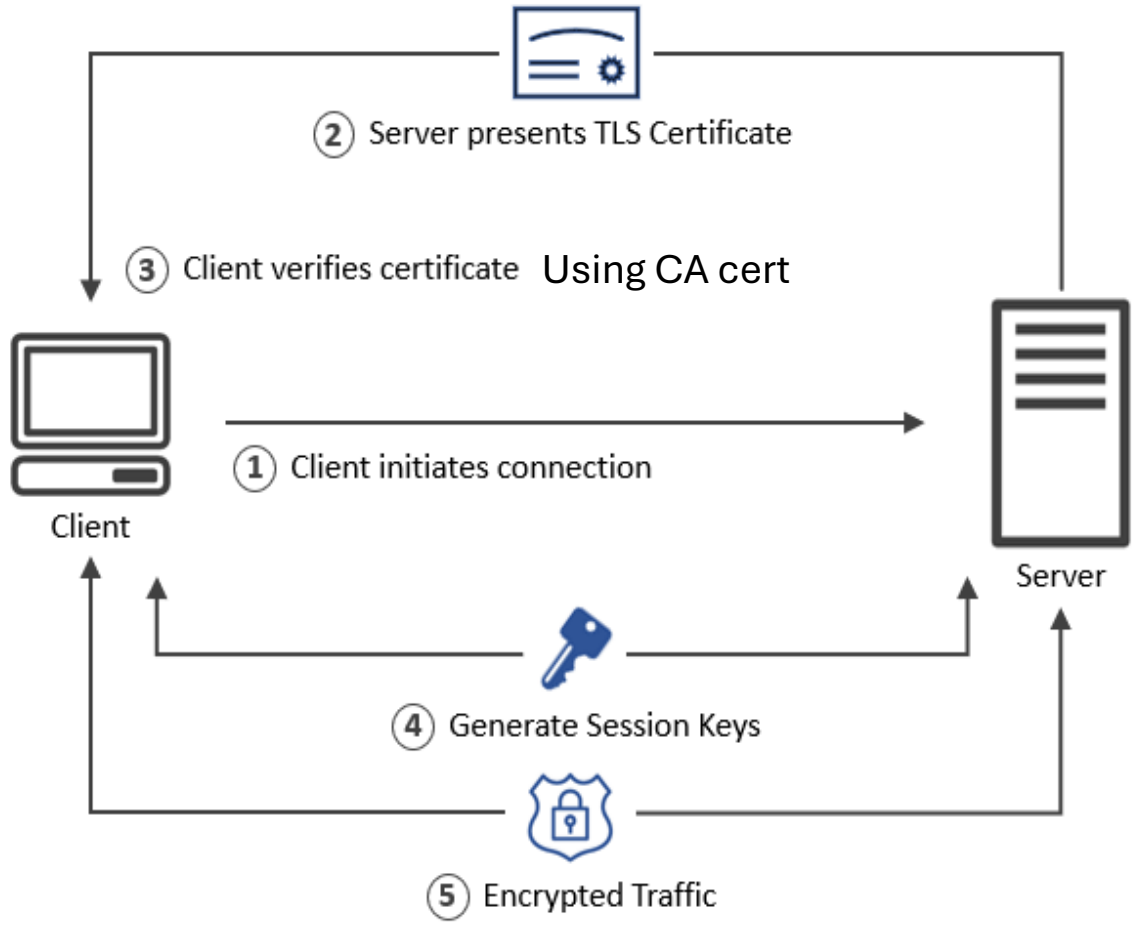
- What is SSL/TLS (HTTPS)?
  - https://www.youtube.com/watch?v=j9QmMEWmcfo

TLS (Concept) Illustrated

Example Usage:

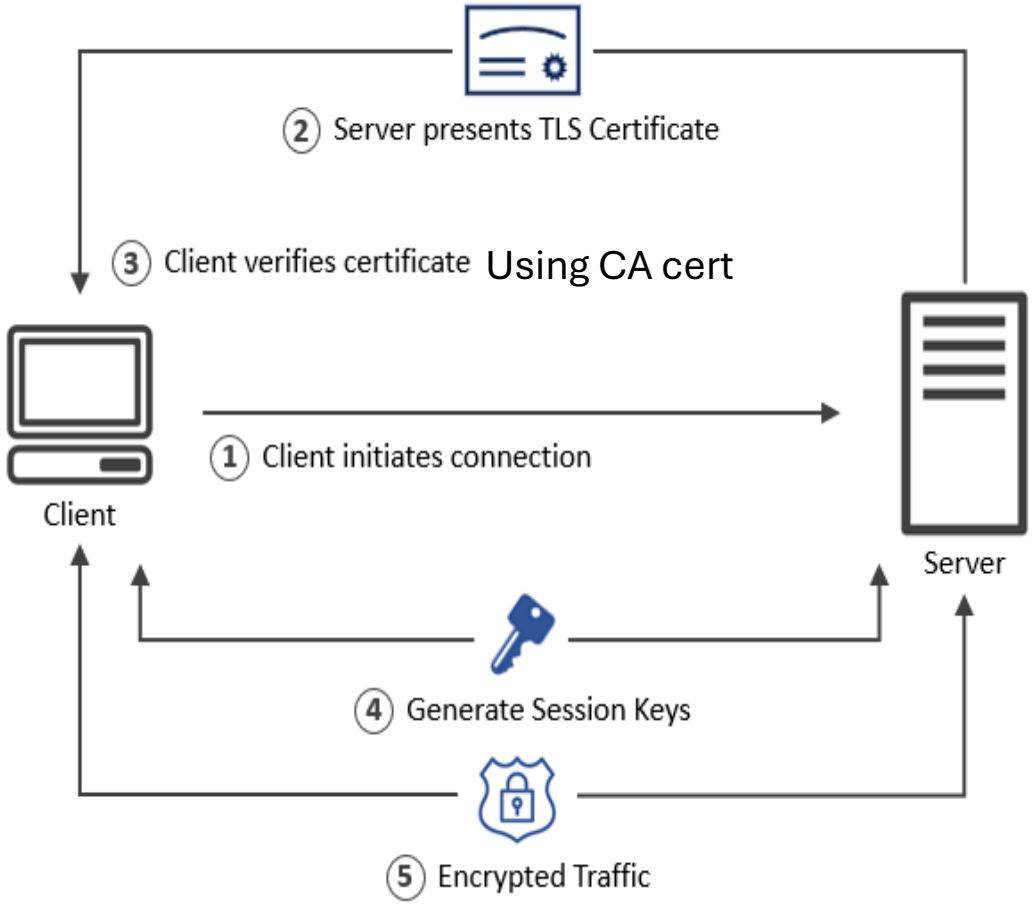
- A typical web browser and web server connection using **SSL/TLS** (**HTTPS**):
  - This connection is used on the Internet to send email in Gmail, and when doing online banking, shopping etc.
    1. Browser connects to server using https.
    2. Server presents certificate to the client (Browser)
    3. Browser verifies the certificate by checking the signature of the CA. To do this the **CA certificate** needs to be in the browser's trusted store (See later)
    4. Browser uses this Public Key to agree a **session key** with the server.
    5. Web Browser and server encrypt data over the connection using the **session key**.

Source: http://www.steves-internet-guide.com/ssl-certificates-explained/

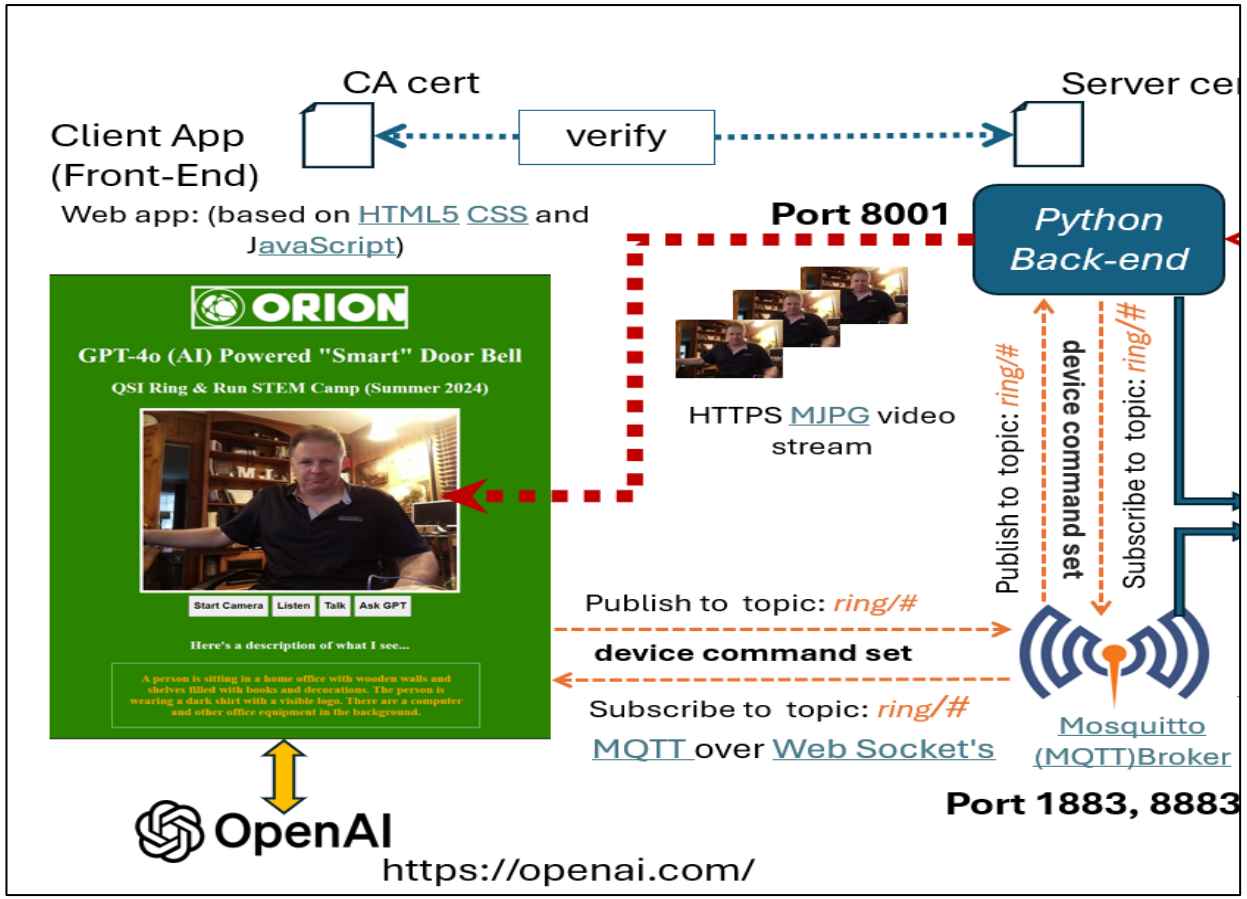


Source: https://www.csa.gov.sg/Tips-Resource/internet-hygiene-portal/information-resources/tls

Implementing TLS into the “Smart” Doorbell Design



Source: https://www.csa.gov.sg/Tips-Resource/internet-hygiene-portal/information-resources/tls



- Steve's guide links:
  - http://www.steves-internet-guide.com/mosquitto-tls/
  - http://www.steves-internet-guide.com/ssl-certificates-explained/

Implementing TLS into “Smart” Doorbell Design (Process Overview)

1. Generate the organization’s certificate request (CSR), and purchase an organization's certificate from a Trusted CA (Verisign, GoDaddy etc.)
   - List of Trusted Certifying Authorities
     - https://developer.visa.com/pages/trusted_certifying_authorities

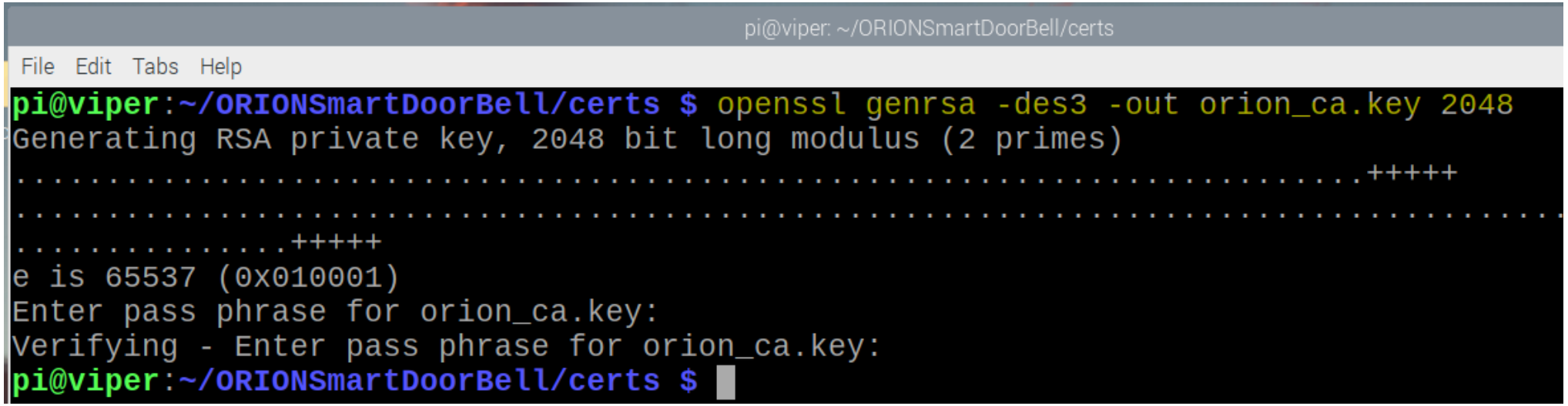
Or

2. Implement all the PKI requirements yourself using the OpenSSL toolset
   1. Generate CA (public/private) key pair and Certificate for signing and verification
   2. Generate the organization’s (public/private) key pair, and CSR
   3. Verify/sign the CSR with the ***organization’s*** CA (private key) to generate a signed server certificate (in this case for the Doorbell application)
   4. Implement TLS in organizations services (.e.g. the Doorbell) and load the signed certificate.
   5. Add the **organization's** CA to Browsers “Trusted Store”

Step 1 of 5: Create the Key Pair For Certificate Authority (CA) Certificate

- VNC to the Raspberry Pi and navigate the project sub folder, named “certs”.

- Create a key pair for your personal ORION CA: *openssl genrsa -des3 -out orion_ca.key 2048*



Note: Use the pass phrase “ORION” (to keep things simple)

Step 2 of 5: Create the CA Certificate

- Create a certificate for the CA using the **orion_ca.key** that we just created:
  openssl req -new -x509 -days 1826 -key orion_ca.key -out orion_ca.crt

```
pi@viper:~/ORIONSmartDoorBell/certs $ openssl req -new -x509 -days 1826 -key orion_ca.key -out orion_ca.crt
Enter pass phrase for orion_ca.key:
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:US
State or Province Name (full name) [Some-State]:New York
Locality Name (eg, city) []:Rome
Organization Name (eg, company) [Internet Widgits Pty Ltd]:ORION
Organizational Unit Name (eg, section) []:IoT Lab
Common Name (e.g. server FQDN or YOUR name) []:ORION.org
Email Address []:admin@orion.org
```

Note: You will be prompted to enter information that will be incorporated into the CA cert. Use the highlighted values in the above example

Step 3 of 5: Create the Key Pair (Back-end) Server

Create a key pair for the server : *openssl genrsa -out ring_server.key 2048*

```
pi@viper:~/ORIONSmartDoorBell/certs $ openssl genrsa -out ring_server.key 2048
Generating RSA private key, 2048 bit long modulus (2 primes)
.....................................+++++
.......................+++++
e is 65537 (0x010001)
pi@viper:~/ORIONSmartDoorBell/certs $
```

Note: This key pair will be used to create the Server Certificate Request (CSR) in the next step

Step 4 of 5: Create Server Certificate Request (CSR)

- The CSR contains an organization's public keys, and domain identity (the common name)
  - The domain identity is often the fully qualified domain name (FQDN) of the requesting organization (for example: www.mydomain.com)

  - The CSR is sent to (a trusted) CA that verifies the CSR (domain ownership etc.)
    - The CA signs CSR with it's private key to generate the certificate which is then sent back to requesting organization.

      - The CA signature (verifiably) binds the requestor's identity (the FQDN / SAN) to it's public key
        - The CA certificate in (the CA chain containing the CA's public key) is publicly available, and used to very the signature of any certificates signed by that CA

      - Trusted CA certificates are preinstalled into the "trusted store" of most web browsers.
        - This is how the browser verifies the identity of the server it is connected to (.e.g. your Bank)

Step 4 of 5: Create Server Certificate Request (CSR) cont.

- In the “certs” subfolder, open ***san.cnf*** with editor (nano or vi), update the file (as shown) with the IP address and hostname of your Raspberry Pi. Save and exit the file
- Create the CSR with the following command:

openssl req -new -out ring_server.csr -key ring_server.key -config san.cnf

```
File  Edit  Tabs  Help
pi@viper:~/ORIONSmartDoorBell/certs $ openssl req -new -out ring_server.csr -key ring_server.key -config san.cnf
pi@viper:~/ORIONSmartDoorBell/certs $
```

- When filling out the CSR form, the ***common name*** is important piece that is verified, and is usually the ***domain name*** of the server/organization
  - Note: for simplicity reasons, we will **use the IP address of the Raspberry Pi as the common name.** Some browsers are configured to require SAN (subject alternative names) so that will be specified using an external configuration file located in the “certs” subfolder

```
File  Edit  Tabs  Help
[ req ]
default_bits        = 2048
distinguished_name  = req_distinguished_name
req_extensions      = req_ext
prompt              = no

[ req_distinguished_name ]
C   = US
ST  = New York
L   = Rome
O   = ORION
OU  = IoT
CN  = 192.168.1.35

[ req_ext ]
subjectAltName = @alt_names

[ alt_names ]
IP.1  = 192.168.1.35
DNS.2 = viper
```

Step 5 of 5: Generate the Server Certificate

- Use our **ORION_CA key** to verify and sign the server certificate. The command below creates the requestioning organization's(the server) certificate: **ring_server.crt**

  - Notice the CSR from Step 4 is using the CA key file to verify and sign *ring_server.crt*

    openssl x509 -req -in ring_server.csr -CA orion_ca.crt -CAkey orion_ca.key -CAcreateserial -out ring_server.crt -days 365 -extensions req_ext -extfile san.cnf

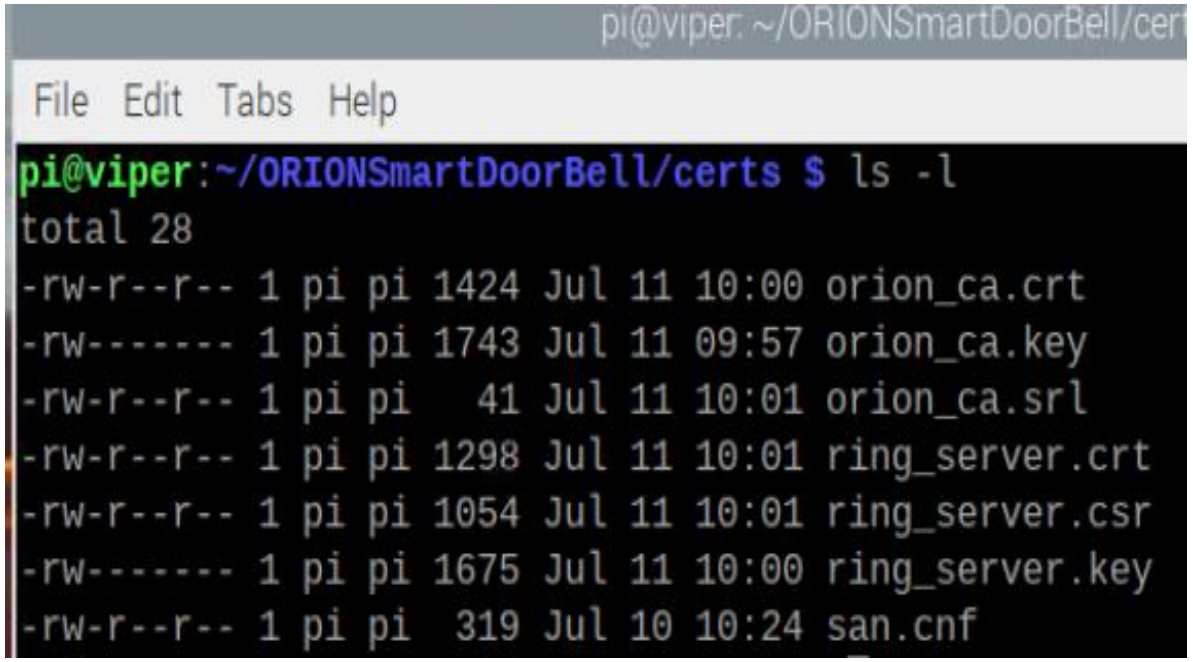
```
pi@viper:~/ORIONSmartDoorBell/certs $ openssl x509 -req -in ring_server.csr -CA orion_ca.crt -
CAkey orion_ca.key -CAcreateserial -out ring_server.crt -days 365 -extensions req_ext -extfile
 san.cnf
Signature ok
subject=C = US, ST = New York, L = Rome, O = ORION, OU = IoT, CN = 192.168.1.35
Getting CA Private Key
Enter pass phrase for orion_ca.key:
pi@viper:~/ORIONSmartDoorBell/certs $
```

  - Note: Our ORION organization is acting as the local CA. This is just as secure as an official (trusted) CA. If ORION maintains a small number of stakeholders, it can just as easily distribute the CA certificate to stakeholders to enable verifiable transactions
    - The issue becomes untenable when broad acceptance is required
      - Anyone can initiate transactions (e.g., ecommerce)
        - In which case, you can't distribute your cert

Congratulations! PKI Requirements Complete

Enter command: ls –l

You output should resemble the following:



- Summary of Activities:
  - Generate CA key pair
  - Generate certificate for the CA using the **CA key**
  - Generate doorbell server (back-end) key pair
  - Create a certificate request **(CSR)**
    - Public key and domain name identity
  - Send CSR to CA Authority for signing and returned certificate
    - Cert binds and identity (domain) to the public key
      - Example: passport binds a picture of person to a name etc. (identity)
  - Client of the organization's services, can verify the identity of the organization using CA cert to verify the organization's certificate (signature verification)

View the Doorbell Application (in Secure mode)

- Start the server (using command below) in “manual” mode with security “on”

```
pi@viper:~/ORIONSmartDoorBell $ python  ring_server.py  --mode manual  --secure on   2> /dev/null
"Smart" Doorbell server started on port: 8001
Connected to MQTT Broker on port 1883 established
connected over web sockets: Success
Subscribed to topics
```

  - *Notice: server port number is 8001 (not 8000 as is the case with insecure mode)*

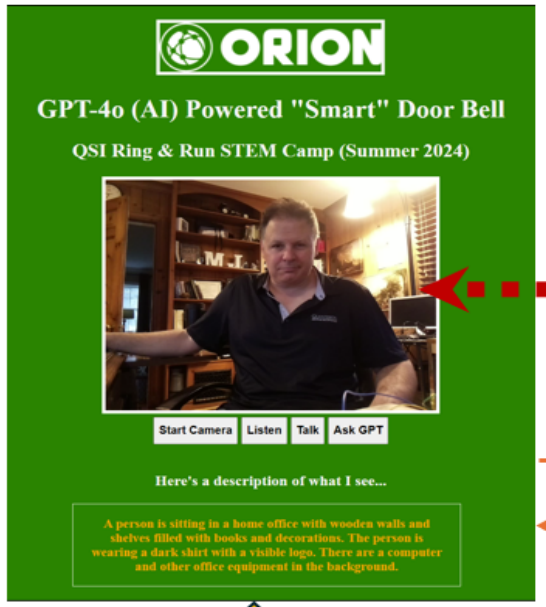
- Open The “Brave” browser and copy the following URL into address bar, press enter
  - https://[IP-address]:8001/index.html
    - *Replace [IP-address] with IP address of the Raspberry Pi*
      - *Notice the protocol in URL has changed to* ***https***

Front-end (web) App

Back-end (Server) Device

Client App (Front-End)

Web app: (based on HTML5 CSS and JavaScript)



OpenAI

https://openai.com/

HTTPS MJPG video stream

Python Back-end

Publish to topic: ring/#

device command set

Subscribe to topic: ring/#

Publish to topic: ring/#

device command set

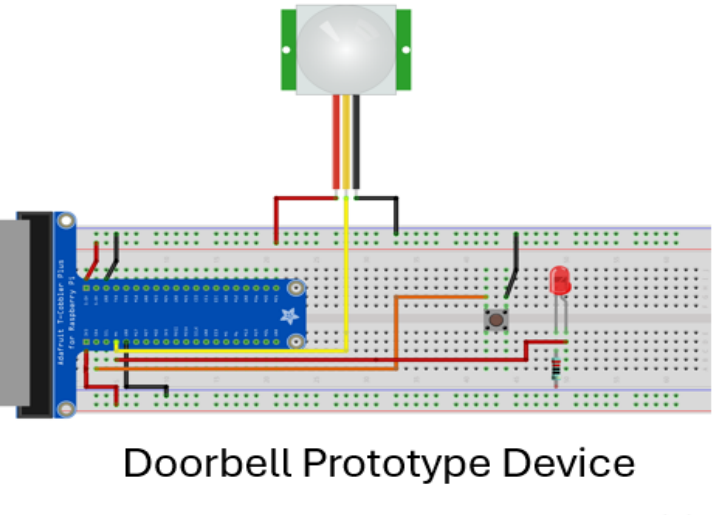
Subscribe to topic: ring/#

MQTT over WebSockets

Mosquitto (MQTT)Broker

Raspberry Pi 4B

Supported modes: motion and manual (button press)



Doorbell Prototype Device

fritzing

IoT enabled “Smart” Doorbell Concept with TLS

CA cert

Server cert

Client App
(Front-End)

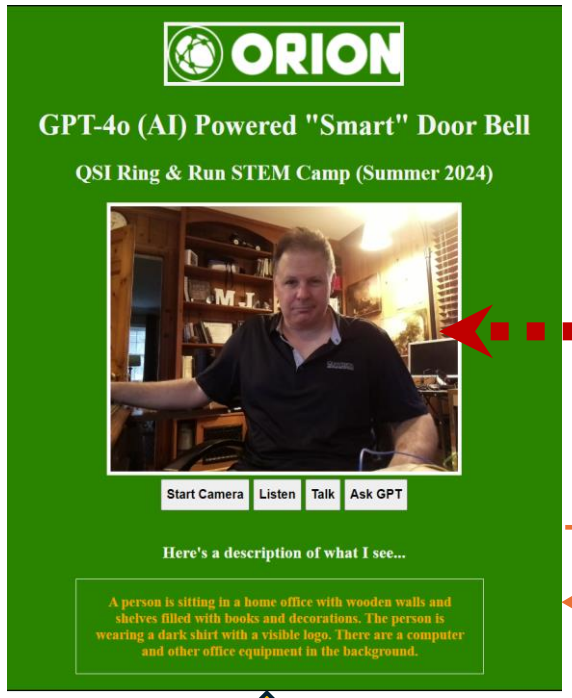
verify

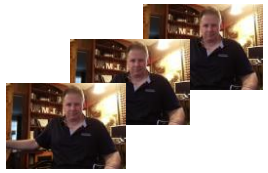
Web app: (based on HTML5 CSS and JavaScript)

Supported modes:
motion and manual (button press)

Port 8001

Python Back-end





HTTPS MJPG video stream

Publish to topic: *ring/#*

device command set

Subscribe to topic: *ring/#*

Publish to topic: *ring/#*

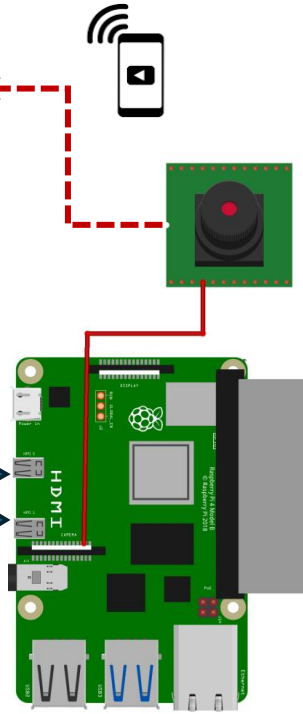
device command set

Subscribe to topic: *ring/#*

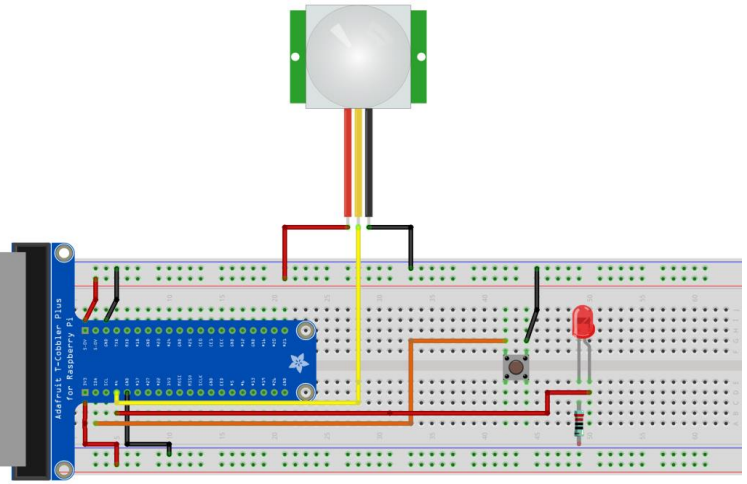
Mosquitto (MQTT)Broker

MQTT over Web Socket's

Port 1883, 8883



Raspberry Pi 4B



Doorbell Prototype Device

fritzing

OpenAI

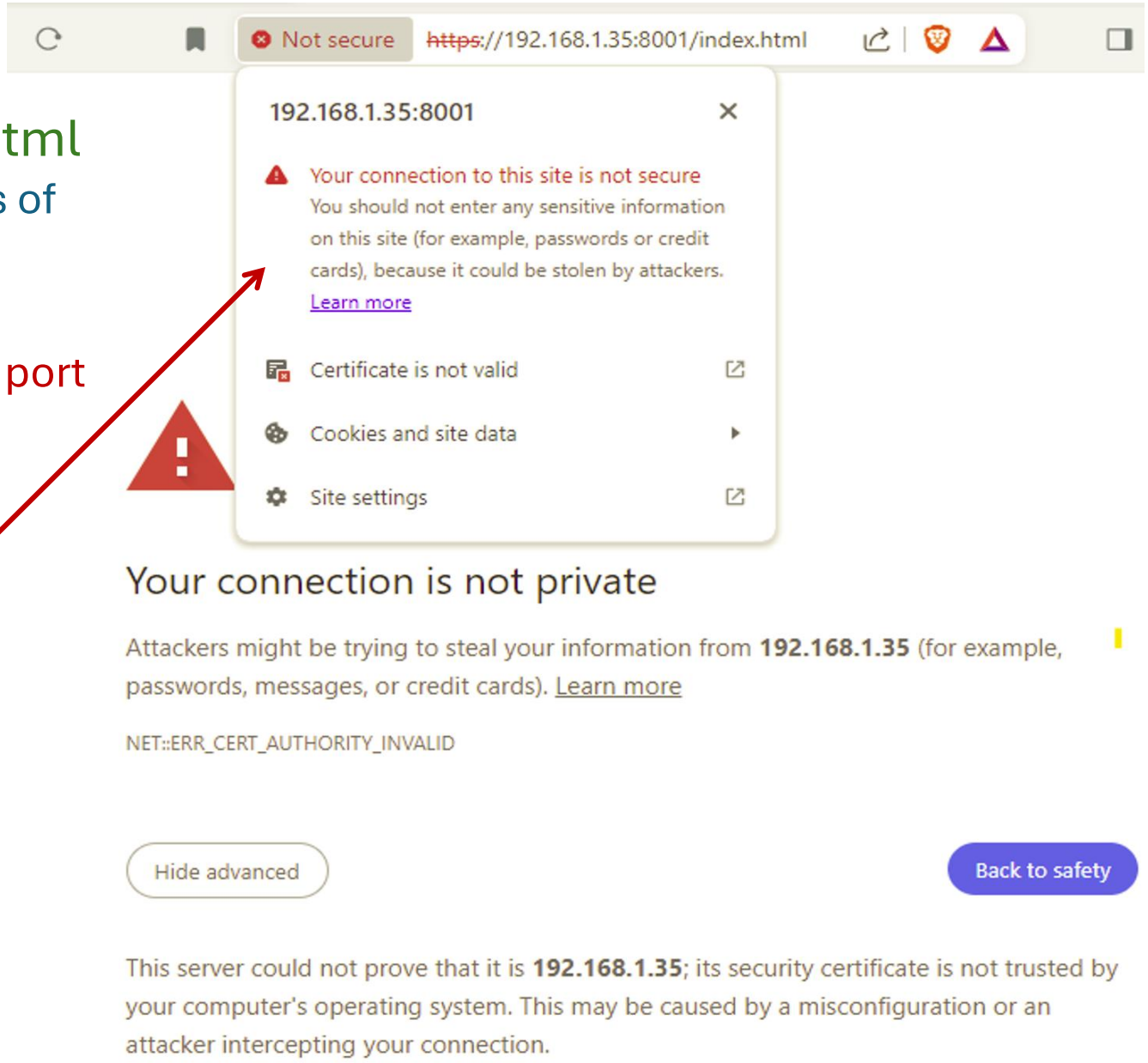
https://openai.com/

Viewing the Doorbell Application (Secure mode)

- URL:
  - https://[IP-address]:8001/index.html
    - Replace [IP-address] with IP address of the Raspberry Pi
      - Notice the protocol in URL has changed to **https** and exposed port 8001

Challenge Questions:

1. What is the browser indicating?
2. How would we rectify this error?



Answer to Challenge Question (secure mode):

- …Certificate Error!
  - missing (valid) PKI certificates to verify the identity of server and establish an encrypted session between the client and the server
    - How do we rectify these issues?
      - ✓Add the CA cert to Brower's "Trusted Store"

Adding the ORION CA Certificate to the Browser Trusted Store

- The local (ORION CA) signed the ring server certificate with it's private key
  - When a user connects to the ring server using a web browser, the browser can verify the identity of the server by verifying the signature in the server's certificate

- The CA certificate contains the CA's public key, (and thanks to PKI asymmetric encryption), the browser can use the CA public key to very signature generated with the CA's private key.

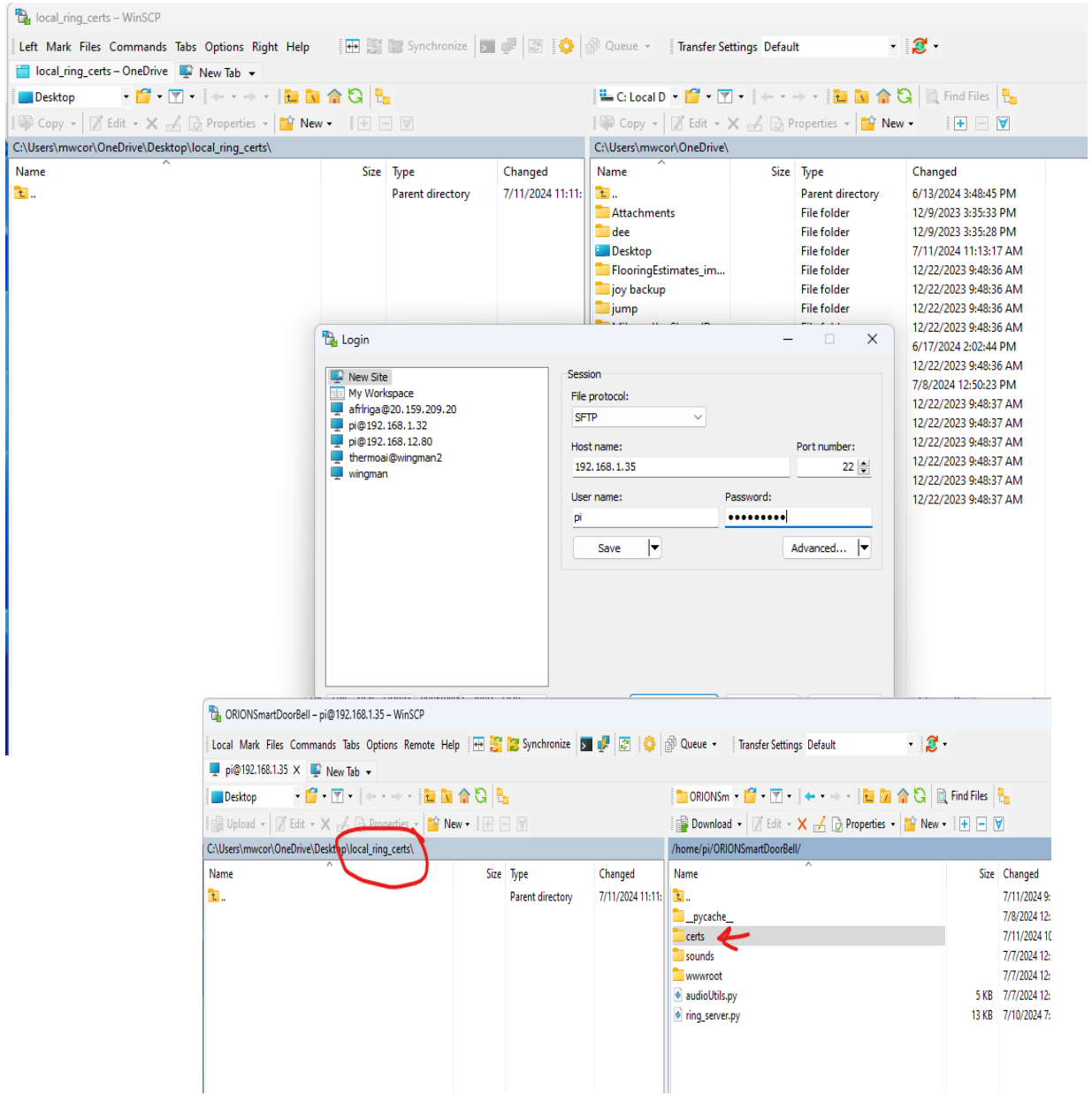
- To accomplish this, we begin by adding the ORION CA certificate to the browser's "Trusted Store" - a folder on the device where certificates are stored/managed
  - We will accomplish this with "Brave "browser that we previously installed on the laptop.

Use WinSCP to copy the Certs Folder

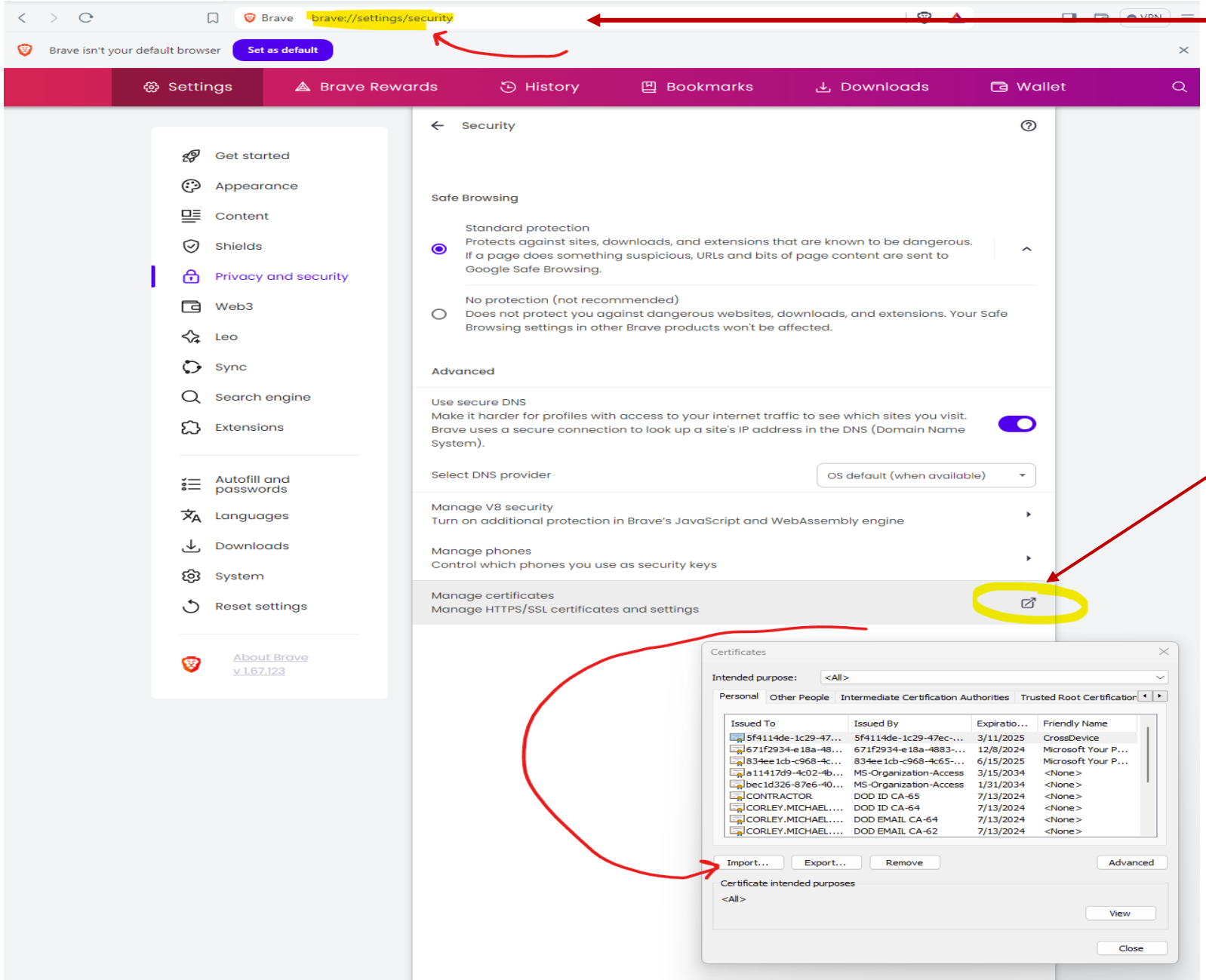
- We need the certificates (currently stored on the Raspberry Pi) copied to the Windows laptop so we can import the ORION CA cert into the Brave browser's "Trusted Store" location:
  - For this we will use WinSCP
    - Download WinSCP here: https://winscp.net/eng/download.php

1. Create a folder on the Windows 11 Desktop, named: "local_ring_certs"

2. Open WinSCP, connect to the Raspberry Pi, and copy the project subfolder "certs" to the "local_ring_certs" folder on the Windows Desktop



Adding the ORION CA Certificate to the Brave Trusted Store (cont.)

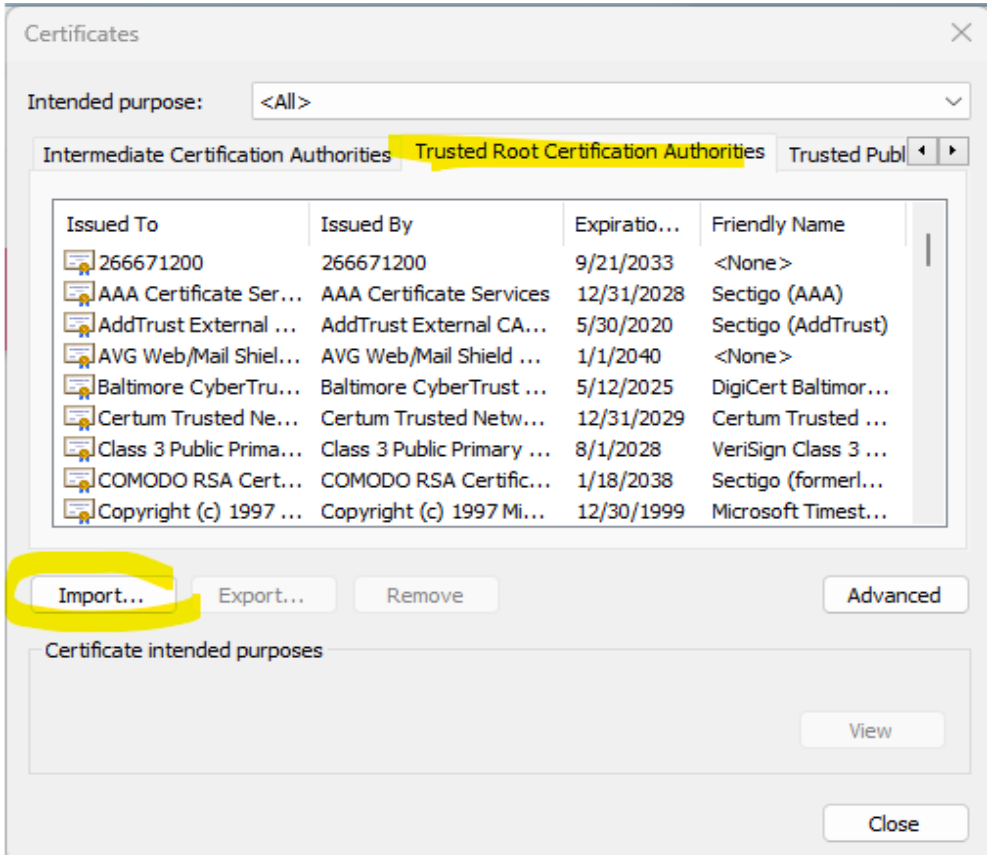


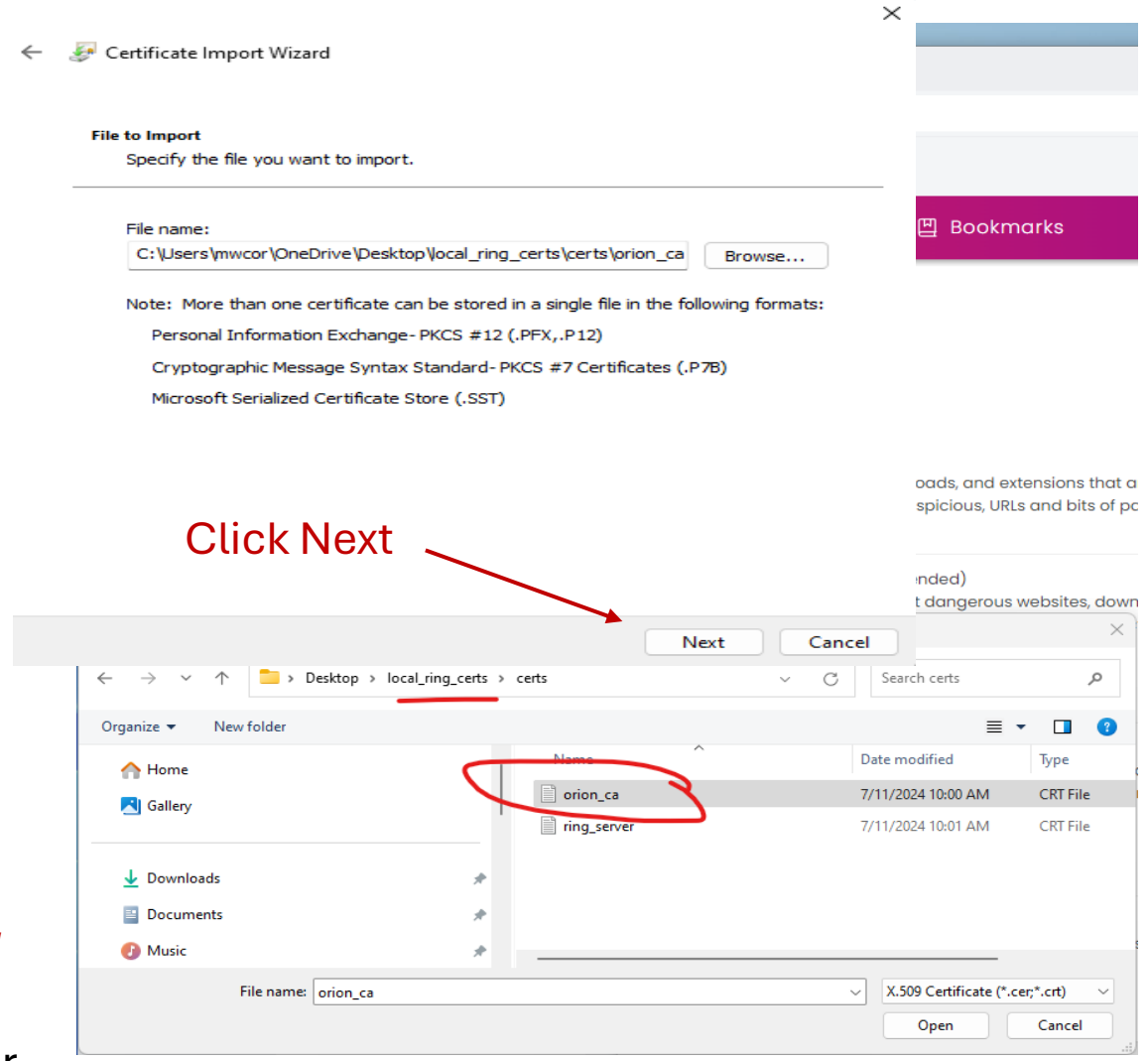
1. Open Brave browser and enter the following in the address bar: *brave://settings/security*

2. *Click "Manage Certificates"*

3. See next slide

Adding the ORION CA Certificate to the Brave Trusted Store (cont.)

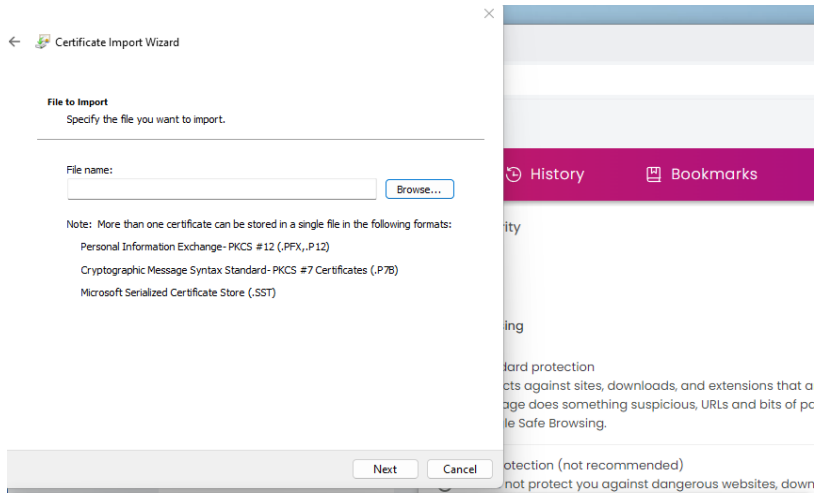


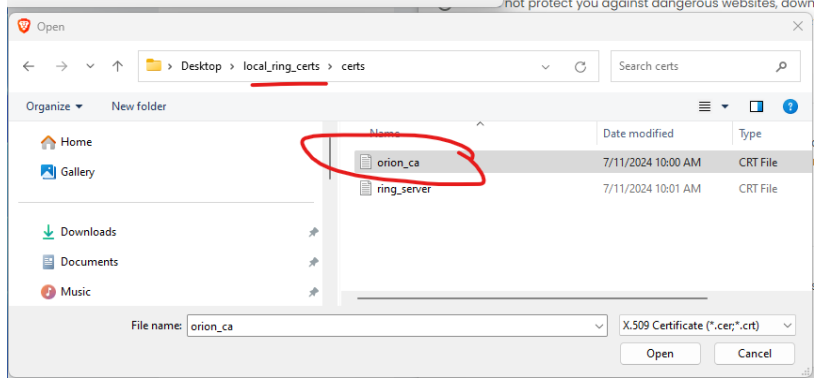


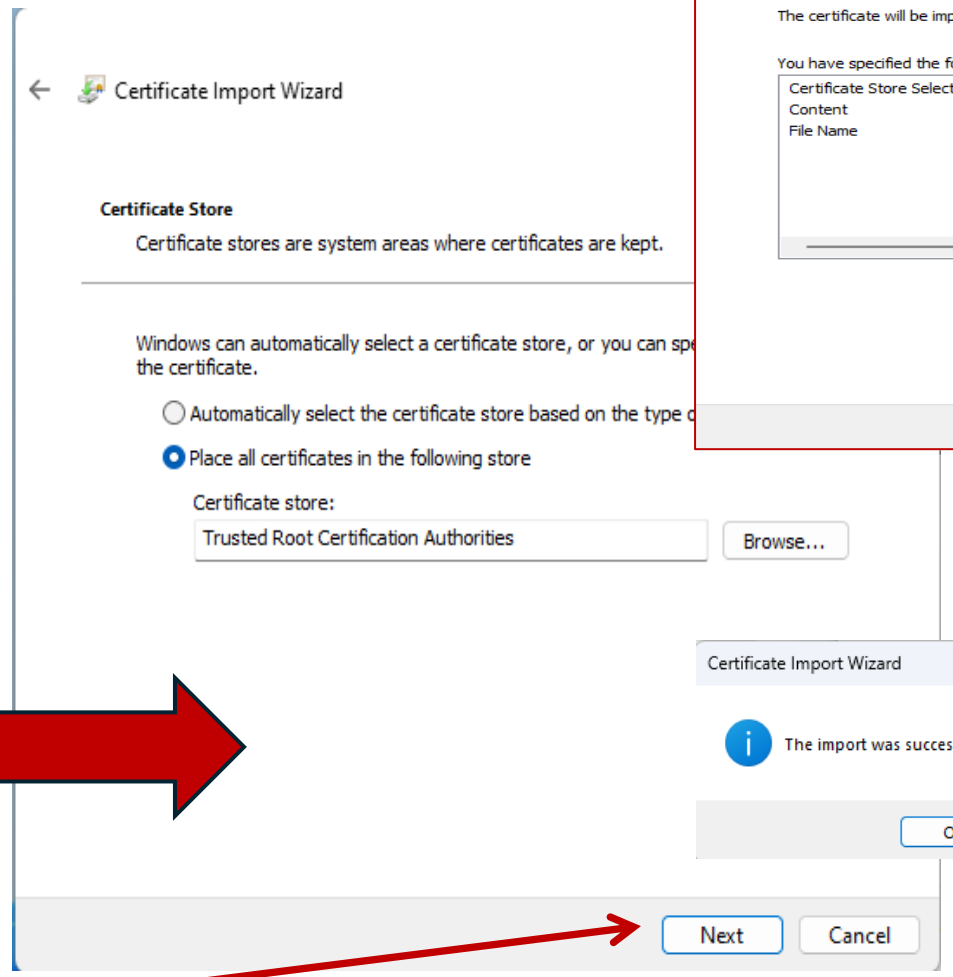
1. *Click "Trusted Root Certifications" Tab*
   - *Notice the official list of Trusted CA certs displayed*

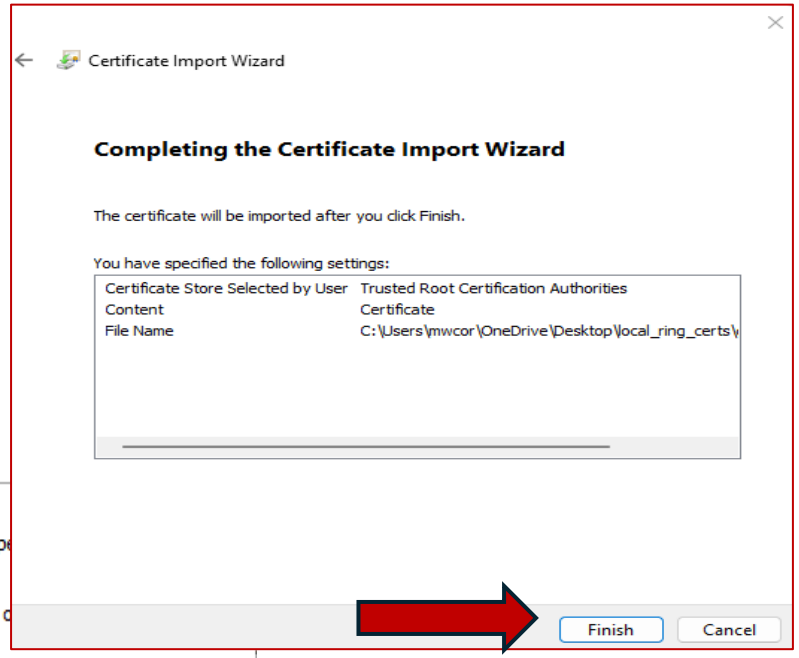
2. Click "Import" and navigate to "local_ring_certs" folder on the Windows Desktop, select "orion_ca.crt" and click "Open"

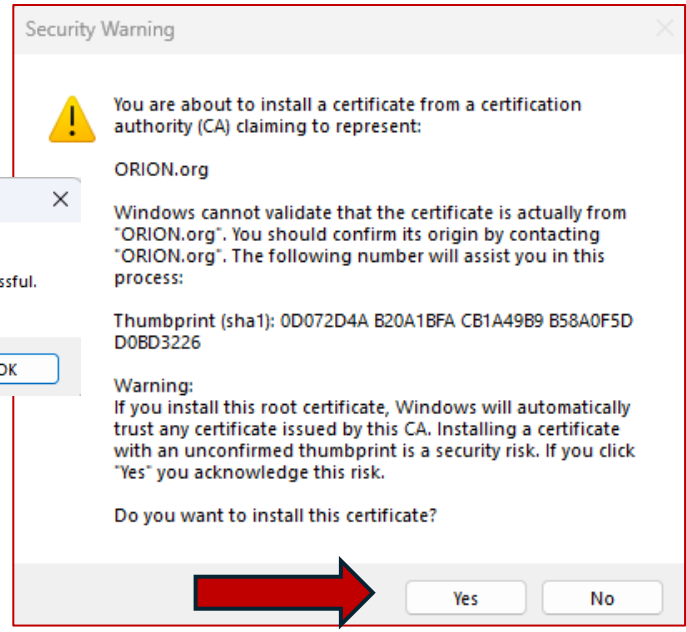
Adding the ORION CA Certificate to the Brave Trusted Store (cont.)





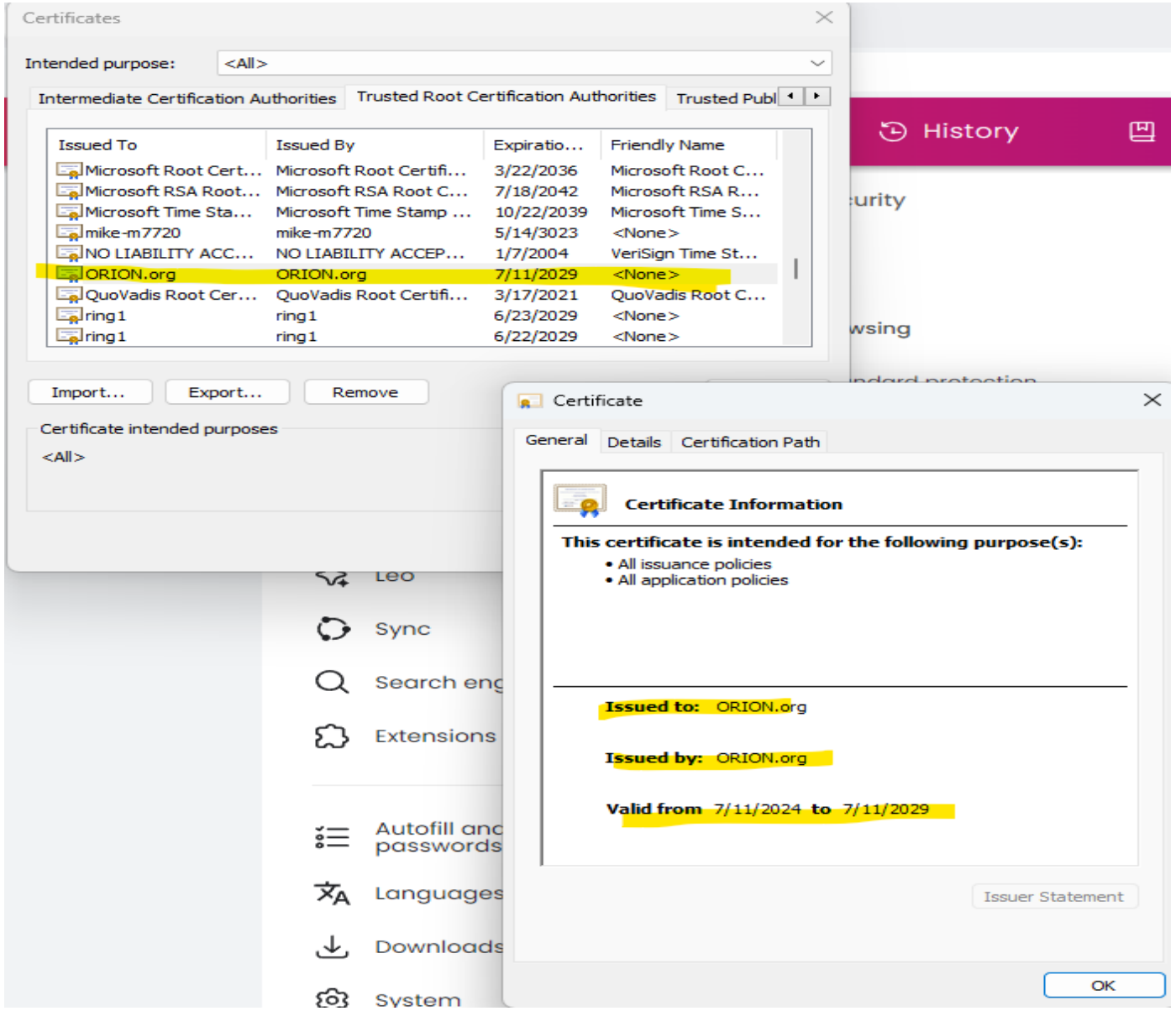






Finally, Click “Next” to add the Orion CA to the Trusted Store

Congratulations, You've Added the ORION Trusted CA Certificate



View the Doorbell Application (Secure mode)

- Start the server(using command below) in “manual” mode with security “on”

```
pi@viper:~/ORIONSmartDoorBell $ python  ring_server.py  --mode manual  --secure on   2> /dev/null
"Smart" Doorbell server started on port: 8001
Connected to MQTT Broker on port 1883 established
connected over web sockets: Success
Subscribed to topics
```

  - *Notice: server TCP port number is 8001 (not 8000 as is the case with insecure mode)*

- Open The “Brave” browser and copy the following URL into address bar, press enter
  - https://[IP-address]:8001/index.html
    - *Replace [IP-address] with IP address of the Raspberry Pi*
      - *Notice the protocol in URL has changed to* ***https***
      - *Notice the TCP port in request has changed to 8001*

View the Doorbell Application(Secure mode): With PKI Certs. installed

Open Brave and enter following URL into the address bar: https://[IP-address]:8001/index.html
Replace [IP-address], with IP address of your Raspberry Pi

The certificate error observed in slide 101, has been successfully corrected, however, viewing the **console** output in **developer tools (F12)** we have a different error.

Challenge Question: What does this error message indicate ?

- Hint: View the Mosquitto Broker configuration file: ***nano /etc/mosquito/mosquitto.cnf***

Viewing the Doorbell (Secure mode): With PKI Certs. Installed (cont.)

Answer: If you said the Mosquitto MQTT Broker is not currently configured for HTTPS/TLS support, then you would be correct!

Let's update Mosquitto MQTT Broker configuration!

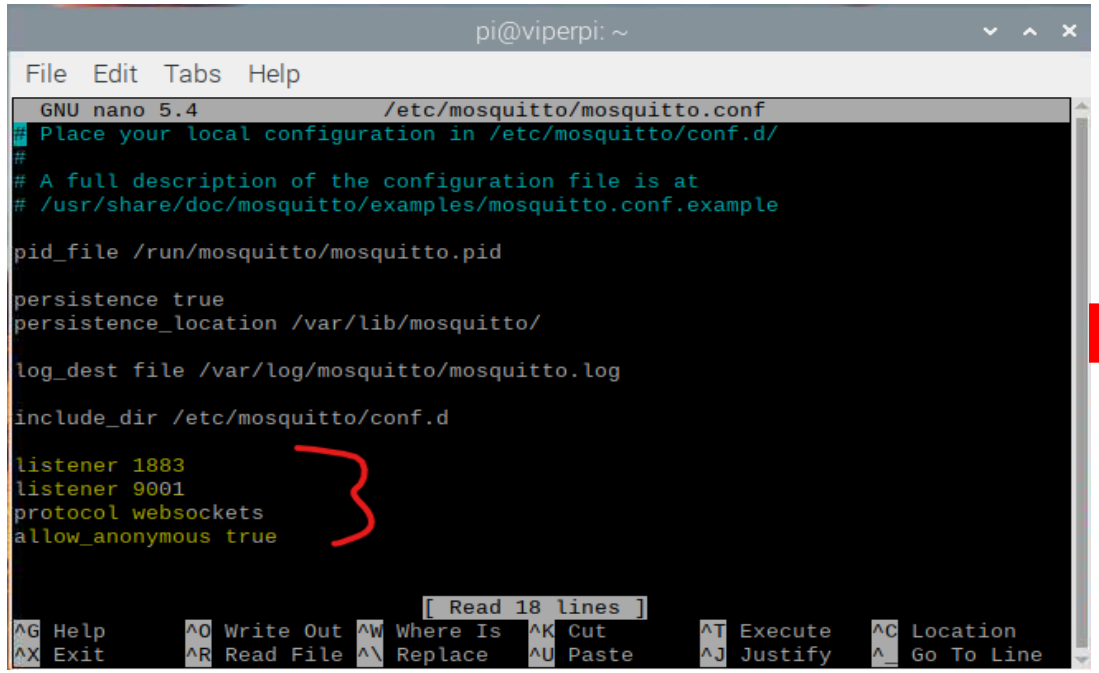
Update the MQTT Broker Configuration (see Slide 69 for the current configuration)

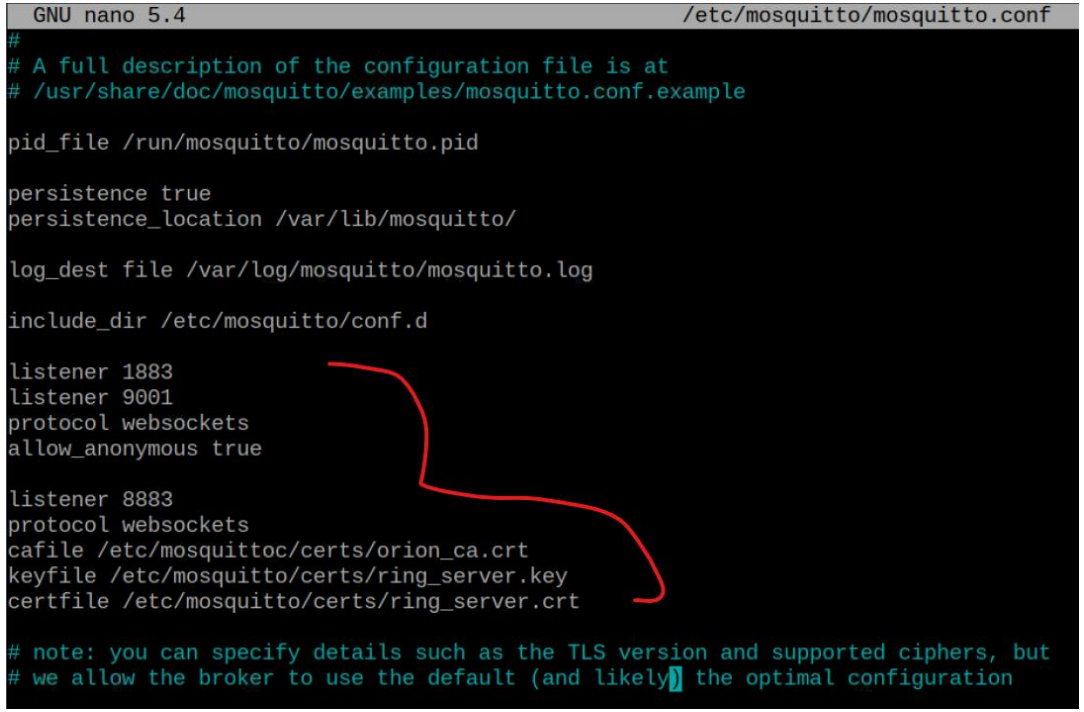
- From VNC session, or from the VSCode terminal, and open the mosquito broker configuration file: *sudo nano /etc/mosquitto/mosquitto.conf*
  - The curent configuration file should resemble the screen shot below.
- To update the Broker configuration for compliance with TLS, we need to add another "listener" endpoint that uses the certificates we created previously.

- Append following lines to the end of the mosquitto.conf file:

```
listener 8883
protocol websockets
cafile /etc/mosquitto/certs/orion_ca.crt
keyfile /etc/mosquitto/certs/ring_server.key
certfile /etc/mosquitto/certs/ring_server.crt
```

- The updated mosquitto.conf should resemble the screen shot below:





Explanation of the Updated Broker Configuration

- listener – configures the Broker (server) to "listen" TCP/MQTT connections on TCP port 8883

- websockets – the ideal choice to enable the web browser (JavaScript client) to participate in lightweight, real-time (bidirectional) MQTT communication with an external MQTT Broker/Server

- The presence of the certificate files: cafile, keyfile, and certfile tells the Broker to support TLS on *listener* exposed on port 8883

- * **We need to copy the certificate files from the doorbell project "certs" subfolder to "certs' subfolder of the MQTT Broker: *<u>/etc/mosquitto/certs</u>***

```
  GNU nano 5.4                                    /etc/mosquitto/mosquitto.conf
#
# A full description of the configuration file is at
# /usr/share/doc/mosquitto/examples/mosquitto.conf.example

pid_file /run/mosquitto/mosquitto.pid

persistence true
persistence_location /var/lib/mosquitto/

log_dest file /var/log/mosquitto/mosquitto.log

include_dir /etc/mosquitto/conf.d

listener 1883
listener 9001
protocol websockets
allow_anonymous true

listener 8883
protocol websockets
cafile /etc/mosquittoc/certs/orion_ca.crt
keyfile /etc/mosquitto/certs/ring_server.key
certfile /etc/mosquitto/certs/ring_server.crt

# note: you can specify details such as the TLS version and supported ciphers, but
# we allow the broker to use the default (and likely) the optimal configuration
```

Copy the Certificate files to the Broker “certs” subfolder

- From VNC session terminal (or from the VSCode terminal), navigate to the doorbell project “certs” subfolder and copy certificate files to Broker configuration folder with the following command:
  sudo cp orion_ca.crt ring_server.key ring_server.crt /etc/mosquitto/certs/

Your output should resemble that in the following screenshot:

```
pi@viper:~/ORIONSmartDoorBell/certs $ sudo cp orion_ca.crt ring_server.key ring_server.crt /etc/mosquitto/certs/
pi@viper:~/ORIONSmartDoorBell/certs $ ls /etc/mosquitto/certs/
orion_ca.crt  README  ring_server.crt  ring_server.key
pi@viper:~/ORIONSmartDoorBell/certs $ ls -l /etc/mosquitto/certs/
total 16
-rw-r--r-- 1 root root 1424 Jul 11 14:34 orion_ca.crt
-rw-r--r-- 1 root root  130 Sep 30  2023 README
-rw-r--r-- 1 root root 1298 Jul 11 14:34 ring_server.crt
-rw------- 1 root root 1675 Jul 11 14:34 ring_server.key
```

<u>Challenge Question</u>: are the above file permissions adequate for securing for certificates and key files ? Describe Why or why not? (Group answer to follow)

Restart the Broker to apply the update service: *sudo systemctl restart mosquitto*

Uh-Oh…The restart command returned an error?
Why?

Hint: consider the file ownership as shown in `ls` command output above

Restart the Broker service with the Updated Configuration:

- The Broker restart command produced an error message similar to the following:

```
pi@viper:~/ORIONSmartDoorBell/certs $ sudo systemctl restart mosquitto
Job for mosquitto.service failed because the control process exited with error code.
See "systemctl status mosquitto.service" and "journalctl -xe" for details.
```

- Notice from the output of the ls command below, that the files are all owned by the "root" user account
  - We copied them using sudo which assumes "root" user privileges
    - mosquitto process (executed as the "mosquitto" user) is attempting to the "read" these files.
      - The *server key file* which rightly set so only the owner and read/write it.
        - Because it's owned by root, the mosquitto process cannot read it.

```
pi@viper:~/ORIONSmartDoorBell/certs $ sudo cp orion_ca.crt ring_server.key ring_server.crt /etc/mosquitto/certs/
pi@viper:~/ORIONSmartDoorBell/certs $ ls /etc/mosquitto/certs/
orion_ca.crt  README  ring_server.crt  ring_server.key
pi@viper:~/ORIONSmartDoorBell/certs $ ls -l /etc/mosquitto/certs/
total 16
-rw-r--r-- 1 root root 1424 Jul 11 14:34 orion_ca.crt
-rw-r--r-- 1 root root  130 Sep 30  2023 README
-rw-r--r-- 1 root root 1298 Jul 11 14:34 ring_server.crt
-rw------- 1 root root 1675 Jul 11 14:34 ring_server.key
```

- The solution is to change the ownership of *ring_server.key* to mosquito user account with following command: *sudo chown mosquitto:mosquitto ring_server.key*
  - *Note: Be certain to ensure that ONLY the mosquitto user has read/write privileges. No other user should have access. Otherwise, your TLS configuration is comprised because the private key leaked*

Broker service Restarted Successfully…

Proper file permissions:

- private key is secret (must be unreadable by all except the owner user/process) “mosquitto” user, and super user: “root”
- The certificate files are public
  - CA cert loaded in browser trusted store
  - Server cert is handed over to the client app (browser), to prove the server’s identity?

```
pi@viper:/etc/mosquitto/certs $ ls -l
total 16
-rw-r--r-- 1 root      root      1424 Jul 11 14:34 orion_ca.crt
-rw-r--r-- 1 root      root       130 Sep 30  2023 README
-rw-r--r-- 1 root      root      1298 Jul 11 14:34 ring_server.crt
-rw------- 1 mosquitto mosquitto 1675 Jul 11 14:34 ring_server.key
pi@viper:/etc/mosquitto/certs $ sudo systemctl restart mosquitto
pi@viper:/etc/mosquitto/certs $
```

If your certificate is stolen, and your common name (www.domain.com) spoofed, would your identity be comprised? (group discussion to follow)

View the Doorbell App (Secure mode)
With PKI Certs. Installed, and Broker Configuration Updated

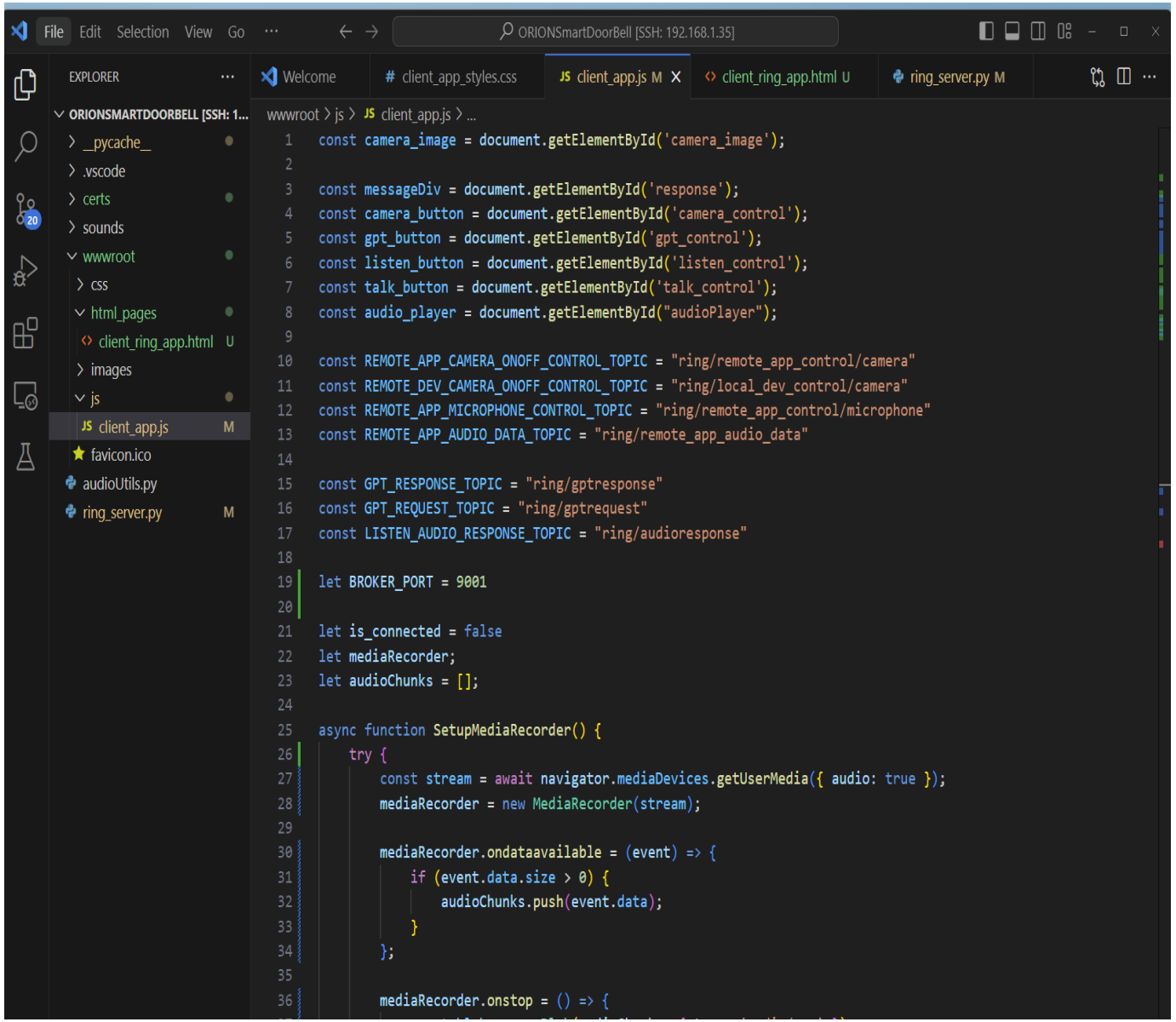
Open Brave and enter following URL into the address bar: https://[IP-address]:8001/index.html
Replace [IP-address], with IP address of your Raspberry Pi

- We updated the Broker configuration, BUT… the same error is displayed as before
  - Notice the error message about an insecure WebSocket using endpoint: 'ws://[ip-address]:9001/mqtt'
  - Refer to updated Broker configuration file, and see the next slide for a hint

Update the Client Application (JavaScript) Code to Use TLS

- In the VSCode editor, open the JavaScript file located in: *wwwroot/js/client_app.js*
  - Recall that the SSL/TLS enabled "listener" we specified for the MQTT broker configuration uses web sockets exposed on TCP/TLS port 8883
    - We need to update the *client_app.js* code to use this SSL/TLS configuration

  - **Challenge Question**: Review the *client_app.js* code and identity the changes required to make code compliant with above mentioned TLS configuration. (group discussion to follow)
    - Hint: there are two different lines of code that need to be updated



Review (back-end) Server for TLS

- In the VSCode editor, open the server code file: *ring_server.py*
- **<u>Challenge Questions</u>**:
  1. Study the code on lines 330 - 350 and describe what these accomplish?
  2. Notice line 301: "*host="127.0.0.1*"
     - Describe what "127.0.0.1" is, and how it is being used in the Doorbell server ?

ring_server.py M | client_app.js M

ring_server.py > ...

```
305  host="127.0.0.1"
306  doorbell_sound_file_path = "./sounds/bell1.mp3"
307
308  BUTTON_GPIO_PIN=2
309  MOTION_SENSOR_GPIO_PIN=4
310
311  client = paho.Client(paho.CallbackAPIVersion.VERSION2, transport="tcp")
312  client.on_message = on_message;
313  client.on_connect = on_connect
314
315  client.on_disconnect = on_disconnect
316
317  # Initialize pygame mixer
318  pygame.mixer.init()
319
320  button = Button(BUTTON_GPIO_PIN)
321  pir = MotionSensor(MOTION_SENSOR_GPIO_PIN)
322
323  if args.mode == "motion":
324      pir.when_motion = handleMotionMode
325
326  button.when_pressed =  handleButtonMode
327
328  with picamera.PiCamera(resolution='1024x768', framerate=24) as camera:
329      output = StreamingOutput()
330      # camera.rotation = 180
331
332      try:
333          ap = AudioPlayback()
334          ap.SetMQTTClient(client, LISTEN_AUDIO_RESPONSE_TOPIC)
335
336          # streaming video (web server) address and port
337          address = ('', HTTP_SERVER_PORT)
338          if args.secure == "on":
339              address = ('', HTTPS_SERVER_PORT)
340
341          server = StreamingServer(address, StreamingHandler)
342
343          print("\"Smart\" Doorbell server started on port: " + str(address[1]))
344          if args.secure == "on":
345              # 1. configure the Python HTTP server for HTTPS (TLS support)
346              # wrap the TCP socket with an SSL support, then load the certs.
347              server.socket  = ssl.wrap_socket (server.socket,
348                  keyfile="./certs/ring_server.key",
349                  certfile="./certs/ring_server.crt",
```

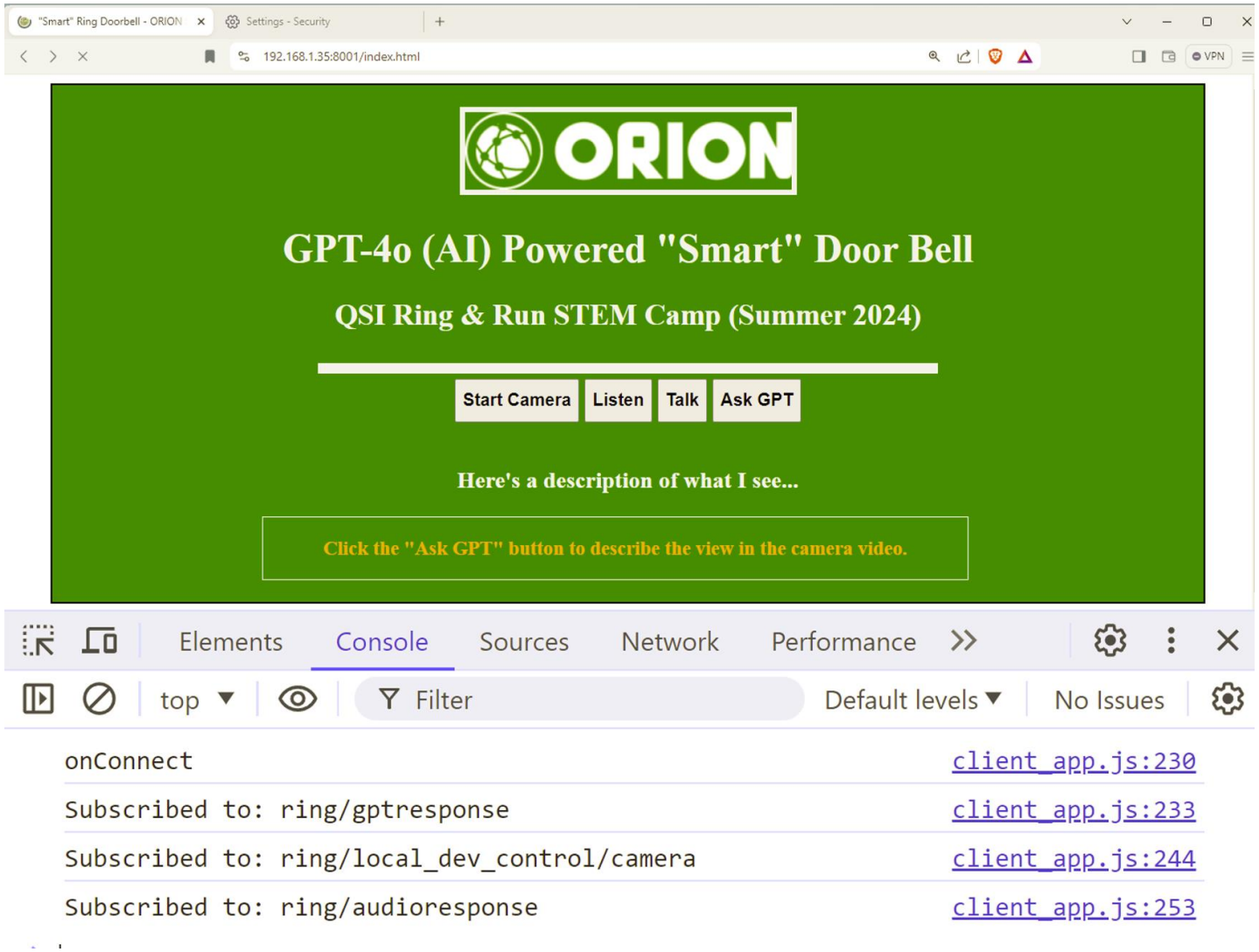
Finally, View the App in Secure Mode Again.. It Works!! Congratulations!

You have successfully implemented TLS into the Doorbell Design

Spend a couple of minutes interacting the with app

Notice “Ask GPT” doesn’t function as expected

Exercise: you might try on your mobile device. Email the CA certificate (only) to yourself, and from your mobile device, open and click on certificate to install it.



MODULE 5: MAKING DOORBELL "SMART" WITH ARTIFICIAL INTELLIGENCE

RING & RUN STEM EVENT.

ORION
OPEN ARCHITECTURE RESILIENT IOT FOR OPERATIONAL NETWORKS

Making the Doorbell "Smart"…

- How?
  - Using Advanced Artificial Intelligence (AI) Models
    - GPT (Generative pre-trained Transformer) and LLM (Large Language Models) are two advanced and related models for natural language processing, created by OpenAI.
    - Videos:
      - What is an LLM?: https://www.youtube.com/watch?v=zKndCikg3R0
      - Inspirational : https://www.youtube.com/watch?v=Zq710AKC1gg
      - How do they work? : https://www.youtube.com/watch?v=5sLYAQS9sWQ
      - https://www.youtube.com/watch?v=VwN49pC2h_I
  - GPT-4o ("o" for "omni") is a step towards much more natural human-computer interaction—it accepts as input any combination of text, audio, image, and video and generates any combination of text, audio, and image outputs.
    - https://openai.com/index/hello-gpt-4o/
  - You can test the older ChatGPT3.5 (without logging in for free), but visiting this link
    - https://chatgpt.com/

LLM versus GPT …

- A large language model and a general pre-trained transformer both refer to advanced machine learning models based on the transformer architecture. However, they have some differences in their focus and application.
  - **Large Language Model: A large language model, like OpenAI's GPT (Generative Pre-trained Transformer) series, is specifically designed and trained for natural language processing tasks**. These models are trained on vast amounts of text data and are capable of generating human-like text, understanding context, and answering questions. They can be fine-tuned for specific tasks like translation, summarization, or sentiment analysis. Examples of large language models include GPT-3, GPT-4, BERT, and RoBERTa.

  - **General Pre-trained Transformer: A general pre-trained transformer is a more broad term for models based on the transformer architecture. While these models can also be used for natural language processing tasks, they can be applied to a wider range of problems, including computer vision, speech recognition, and reinforcement learning**. These models are pre-trained on large datasets and can be fine-tuned for specific tasks. Examples of general pre-trained transformers include ViT (Vision Transformer) for computer vision tasks and Conformer models for speech recognition tasks. [1]

Source: https://www.brinwilson.com/whats-the-difference-between-a-large-language-model-llm-and-a-general-pre-trained-transformer-gpt/ [1]

OpenAI Overview

- Introduction: https://openai.com/index/chatgpt/
- Common Uses:
  - https://openai.com/
  - https://openai.com/index/introducing-chatgpt-edu/

  - The way we do life in future will be different:
    - https://copilot.microsoft.com/
      - https://www.cnet.com/tech/services-and-software/microsoft-copilot-embraces-the-power-of-openais-new-gpt-4-o/

Challenge Question:

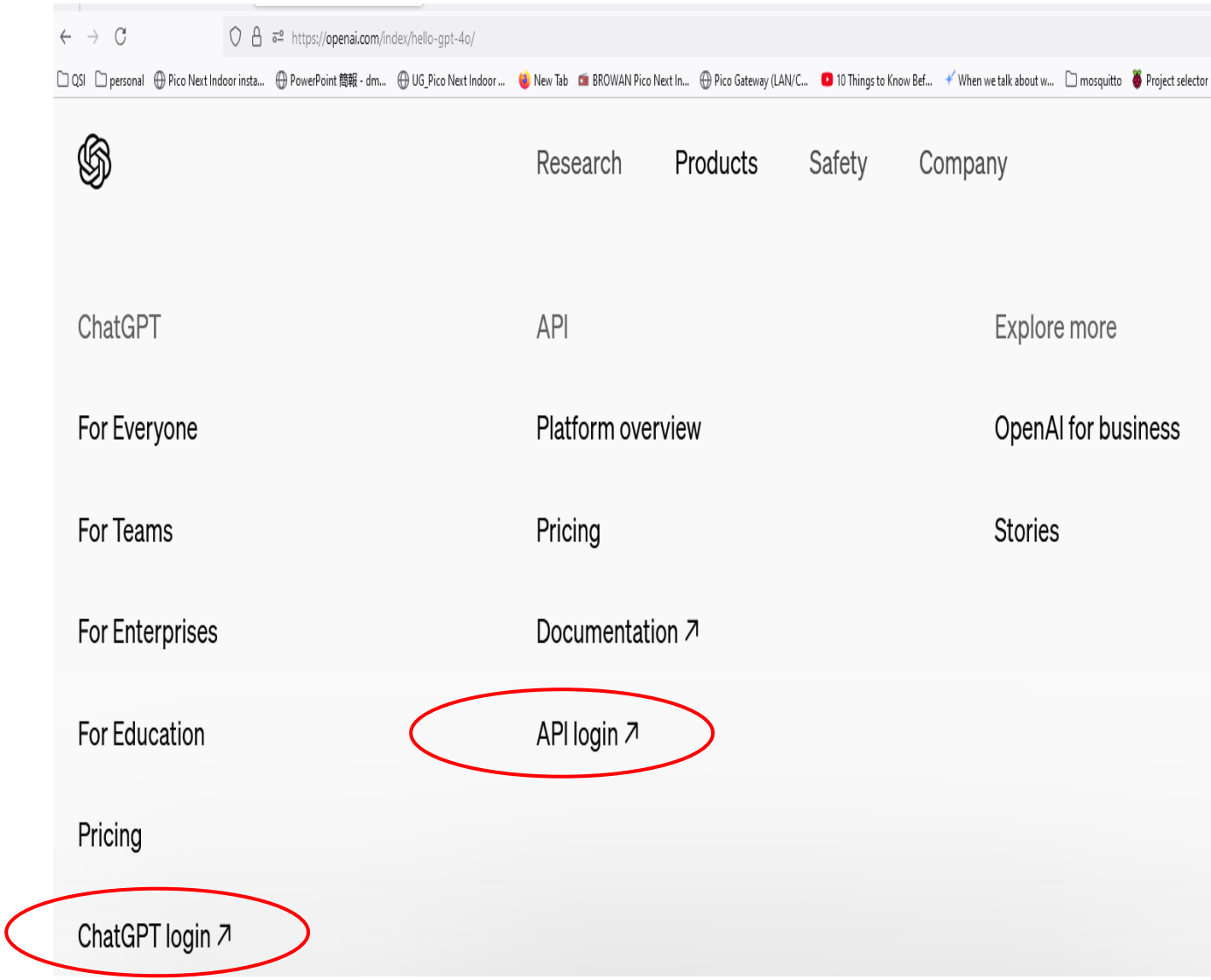
- Can anyone think ways that AI will impact our lives?
  - Several of the image slides in this presentation are AI generated. Can you identify them?

OpenAI Overview cont/

- Platform Overview
  - https://openai.com/api/
- OpenAI Models availability:
  - https://platform.openai.com/docs/models
- It’s not free, but is cost effective
  - https://openai.com/api/pricing/
- Fine-tuning models
  - https://www.spaceo.ai/blog/how-to-train-openai-gpt-models/
- Custom models
  - https://openai.com/form/custom-models/

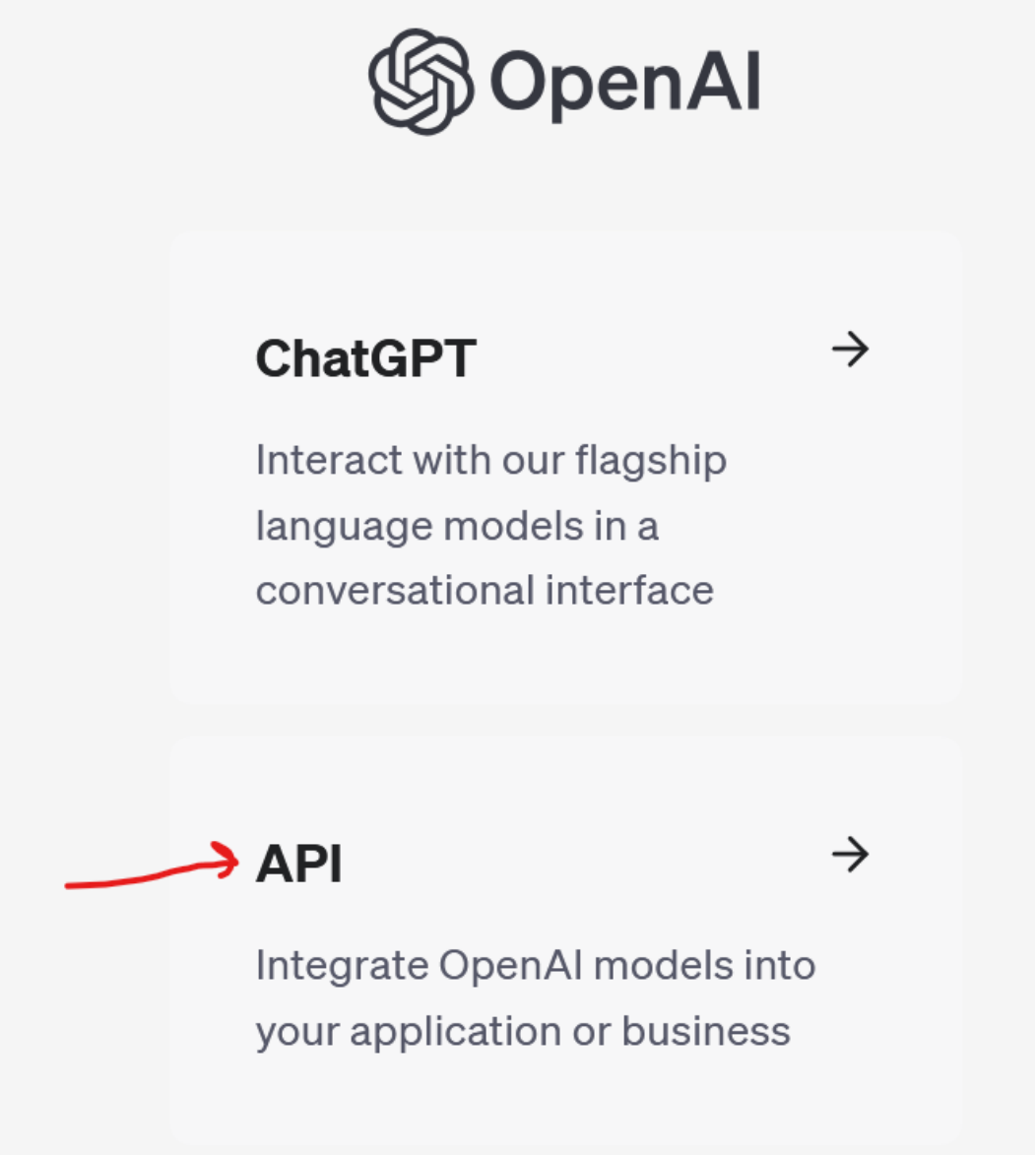
Interacting with ChatGPT?

- Two primary products:
  - Both use separate logins:
    1. ChatGPT Login
    2. API Login
  - **Chatbot (Virtual assistant)** experience
  - Programmatically, through what is referred to as **an "application" programming interface (API)**
    - In our case the *"application"* is the Doorbell project



How did you say I integrate OpenAI models into the Doorbell Design?

- Answer:
  - Really easily, using the API approach!
  - Step 1: Click the “API Login”, create and account and sign in
  - Step 2: Once you’re signed in you can create an organization, and new projects and add users to projects within different roles



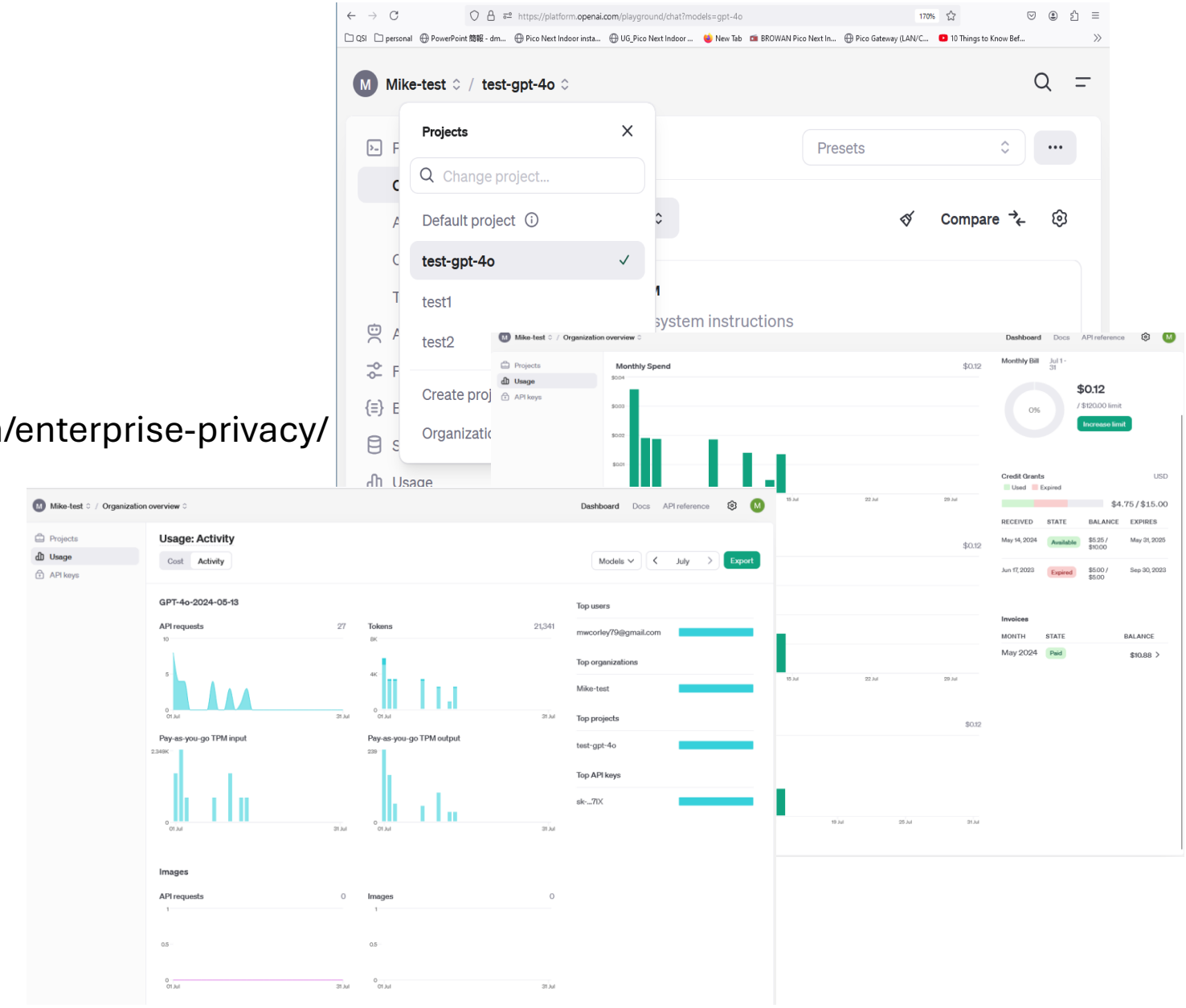
How did you say I integrate OpenAI models into the Doorbell Design?

- Projects enable you to organize, select, and fine-tune models for consumption, and create API keys for secure programmatic access to services
  - *Note: this how we will integrate the "Omni" model for the Doorbell*

https://openai.com/enterprise-privacy/

- At the organization level
  - Monitor/control model usage (API requests (for inputs audio, images, text),
  - Define/add projects, assign users etc.

- Manage Costs and billing
  - API uses Token based approach for mapping resource consumption to billing



Risks? What?

- Privacy
  - Your data…
    - https://openai.com/enterprise-privacy/
    - https://openai.com/policies/usage-policies
  - Zero Data Retention
    - https://community.openai.com/t/zero-data-retention-information/702540
- Getting Wrong Answers…
  - https://www.linkedin.com/pulse/does-chatgpt-give-correct-answers-explained-jon-rishworth-q8qze
- Safety and Ethics
  - https://www.kaspersky.com/resource-center/preemptive-safety/is-chatgpt-safe
  - https://medium.com/@gargg/guiding-the-future-of-ai-an-inside-look-at-openais-ethical-policies-3503f17b6b0b

Integrating OpenAI models into the Doorbell Project: Synopsis

- The (back-end) server uses the “picamera” package to stream camera imagery (defaulted 24 frames per second) to the font-end (web application).
  - Note: you can view *StreamingServer* Code in *ring_server.py (*lines 28-142)

- When someone presses the doorbell button (when set in “manual” mode), and when motion is detected (in “motion” mode), the camera will start recording and streaming captured camera frames to the front-end app.
  - The user of the client (web) app can stop the streaming by pressing “Stop camera” and resume steaming by pressing “Start camera” to manually initiate streaming.
    - Note: this works because the client app connects and dispatches corresponding MQTT control message/topics to the Broker, which in turn forwards message (using pub/sub semantics) back-end (subscribed) server for processing

- When the user clicks the “Ask GPT” button (with the camera streaming), an MQTT message is dispatched to signal the server, which subsequently receives the request, captures image frame and dispatches the image to OpenAI for to be processed by the model specified in the API invocation.

- The OpenAI model (GPT-4o) will process the input (image frames) and return intelligent (human like) description of what it sees in the image, thus making the Doorbell “Smart”

IoT enabled “Smart” Doorbell Concept

CA cert

Server cert

verify

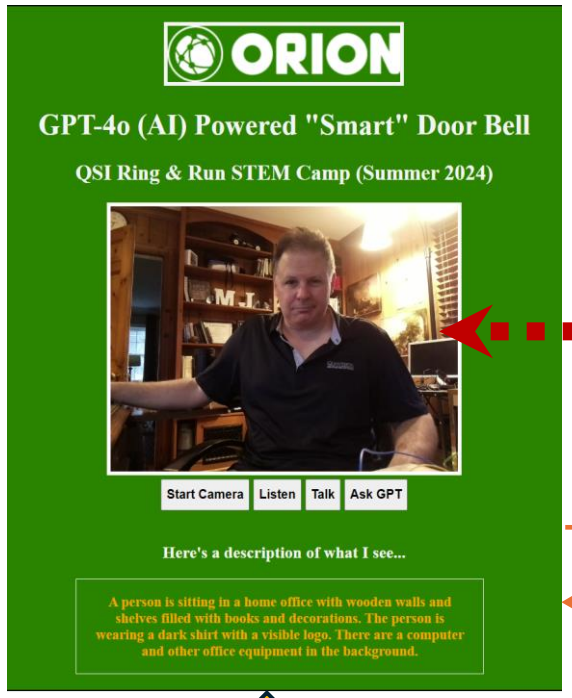
Client App
(Front-End)

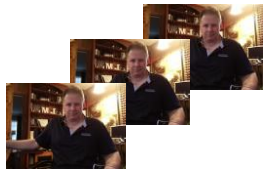
Web app: (based on HTML5 CSS and JavaScript)

Supported modes:
motion and manual (button press)

Port 8001

Python Back-end





HTTPS MJPG video stream

Publish to topic: *ring/#*

device command set

Subscribe to topic: *ring/#*

Publish to topic: *ring/#*

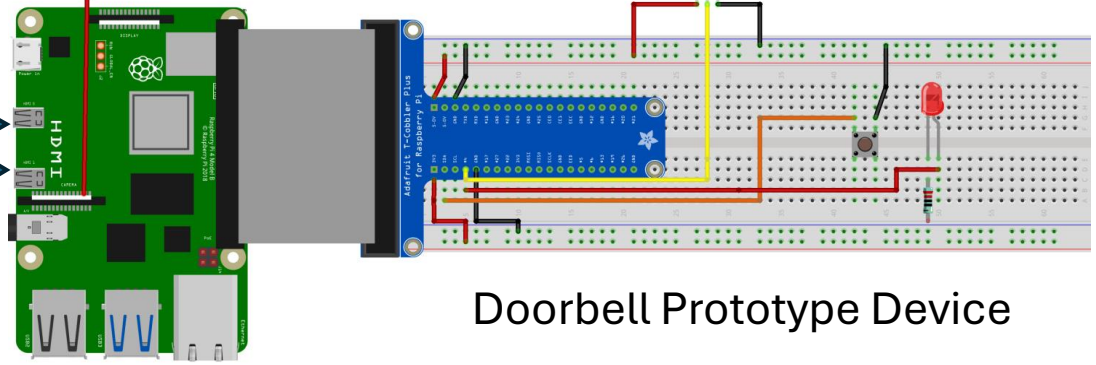
device command set

Subscribe to topic: *ring/#*

Mosquitto (MQTT)Broker

MQTT over Web Socket's

Port 1883, 8883



Raspberry Pi 4B

Doorbell Prototype Device

fritzing

OpenAI

https://openai.com/

OpenAI Integration Concept

Note: for Module 6, you need to complete the code required to upload image frames to he OpenAI service
* See Next slide for details

GPT-4o Model

OpenAI API

```
handleGPTRequest(model):
```

Python Back-end Server

Image frame

Text description

OpenAI

Subscribe to topic:
ring/GPT_REQUEST_TOPIC

Publish to topic:
ring/GPT_RESPONSE_TOPIC

Subscribe to topic:
ring/GPT_RESPONSE_TOPIC

Publish to topic:
ring/GPT_REQUEST_TOPIC

Mosquitto MQTT Broker

(Text description)
ring/GPT_RESPONSE_TOPIC

Reference: https://www.uscloud.com/blog/the-differences-between-openai-and-microsoft-azure-openai/

Integrate OpenAI (flagship) model - GPT-4o ("omni") into the Doorbell Project

1. Open a terminal in the Remote VSCode session
   - Install the officiate OpenAI python package: *pip install openai*

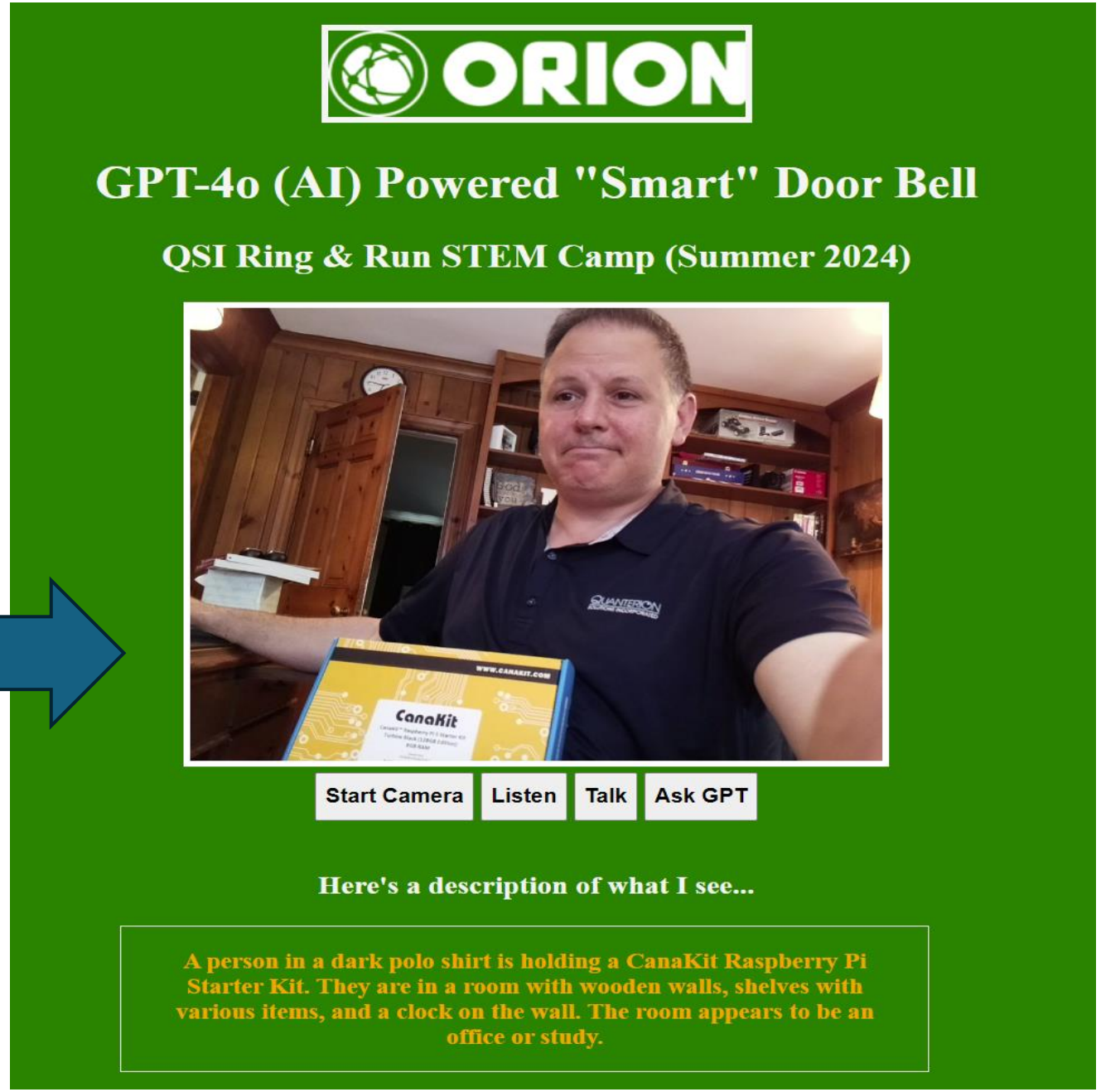
2. **Challenge Exercise**:
   - Study the *ring_server.py* server code function: "***handleGPTRequest()"*** beginning on line 225
   - Study the function "***Open_AI_Tell_Me_Who_Is_There(base64_image)"*** beginning on line 141
   - Study the "**Getting Started, vision"** guide given in the references below: "***Uploading base64 encoded images"***
     - Study these resources very carefully, and try to complete the function: "***Open_AI_Tell_Me_Who_Is_There(base64_image)"***

- ***Note:*** **completing this function correctly is all that is required to successfully integrate the OpenAI service and thus , make the Doorbell "Smart"**
  - ***You will need an API KEY to access the GPT-4o service (this will be provided). You do not need to create an account***

   - You **will need to export the API KEY environment variable in VSCode shell terminal** using the following command: *export DOORBELL_KEY="[api-key]"*
     - **On line 144 uncomment API key variable:** `api_key = os.getenv('DOORBELL_KEY')`

- References:
  - **Getting started guide, vision (study this for the Doorbell project): https://platform.openai.com/docs/guides/vision**
    - API reference page: https://platform.openai.com/docs/api-reference/introduction

Completed “Smart” Doorbell with “Smarter” answers!!



MODULE 6: CUSTOMIZE THE DOORBELL

RING & RUN STEM EVENT.

ORION
OPEN ARCHITECTURE RESILIENT IOT FOR OPERATIONAL NETWORKS

3D Printed Cases

- Working with Ken Kowalski to add house your prototype in a 3d printed case
  - Discussion
    - Printer types
    - Background and skills/tools required
      - Solid Works
      - Tinker CAD

Ideas for Customizing the Doorbell …

- Investigate your own creative interests:
  - If you have ideas you want to test/try to get doorbell to do something we haven't discussed, then now is your chance:
    - Here are some suggestions (only suggestions) for ways to customize your Doorbell
      1. **Create a custom "Theme"**: Modify the CSS styles and/or the HTML code of Client (Web) Application
         - Change the colors (the background, text, page layout, etc.) etc. to be consistent with your team's preference
      2. Implement a new command argument in ring_server.py to enable **custom doorbell ring tones** whenever the button is pressed, or motion is detected.

         Note: You should be able find and download mp3 ring tones from the internet.

Ideas for Customizing the Doorbell …

3. The doorbell should have a way to **store video clips to the Cloud** so they can be viewed at later time.

- When the doorbell detect motion and/or when the button is pressed the doorbell should record video for a specified period of time and transmit recorded video to Cloud storage.

    - This would allow video to be shared and reviewed remotely, at a later date by authorized individuals.
        - Work with your team members to develop a concept for how you might extend the doorbell design to support storing and viewing video in the Cloud.
            - Research Cloud Computing storage services (there are many):
                - Examples include AWS S3 Buckets and Azure BLOB storage

Ideas for Customizing the Doorbell …

4. Experiment with AI services. Do some reading in the OpenAI documentation, search online, or “ask” GPT for ways to fine tune vison model for best results.

- Hint: read the community forums, consider how the system role can influence the systems behavior in terms of the results it provides: https://community.openai.com/t/the-system-role-how-it-influences-the-chat-behavior/87353
  - Nice Tutorial: https://www.datacamp.com/tutorial/gpt4o-api-openai-tutorial

- Search the (alternative) as well as open source and free services (similar) to OpenAI) to use in its place
  - LLMs with Vision: https://roboflow.com/model-feature/llms-with-vision-capabilities
  - Vison LLM: https://github.com/OpenGVLab/VisionLLM
  - Google’s Gemini: https://ai.google.dev/gemini-api/docs/vision?lang=python

5. We did not discuss, nor implement authentication. This has significant security implications.
   - See the next slide for discussion about authentication versus authorization

User Authentication? Really?

- We used PKI / TLS (certificates) to “authenticate” that a server is the server it proclaims to be (i.e., the identity of the server)

- We can also use certificates to authenticate clients, and users alike.
  - Personal certificates (browser)
  - smart cards

- A common way to authenticate people is using “username” and “password” approach

- Note: Authentication is not the same as authorization.
  - The later deals with managing the level of access a user has within a system
    - Often with a “role-based” approach

- Note: We did not implement “user” authentication for the “Smart” doorbell

  …meaning someone else can likely connect to, and control your Doorbell

Setting Mosquitto MQTT Broker Authentication

- Follow the link an GPT-4o chat for setting up authentication for
  - https://chatgpt.com/share/df72014c-90d3-444e-9b35-eda1711dcd94

TEAM PRESENTATIONS

ORION
OPEN ARCHITECTURE RESILIENT IOT
FOR OPERATIONAL NETWORKS

Team Presentations

- Each team take to time build a 10-minute presentation
  - discussion, or demo whatever you would like

- Describe in some way what you team identified and perhaps decided to try for a customization to the Doorbell

- Describe what inspired you most about your time here are the Ring and Run camp
  - What did you take away?
  - What did you find most interesting and helpful?

Where do we go from here?

- Other STEM camps, competitions, etc.
  - https://www.griffissinstitute.org/who-we-work-with/afrl/stem
- Online based learning resources
  - https://randomnerdtutorials.com/getting-started-with-raspberry-pi/
  - https://projects.raspberrypi.org/en
  - Let's Learn .NET - IoT (Internet of Things) (youtube.com)
- College Preparation course work
  - Computer Science/Programming/Cybersecurity offered
    - Math courses are a critical part of the rigor for Computer Scientists and Engineers of all sorts
  - College careers
    - Computer Science, Computer/Electrical Engineering
      - Speak to your teachers/guidance counselors
      - Reach out to us, anytime!

THANK YOU FOR JOINING US!

PLEASE COMPLETE THE ONLINE SURVEY!

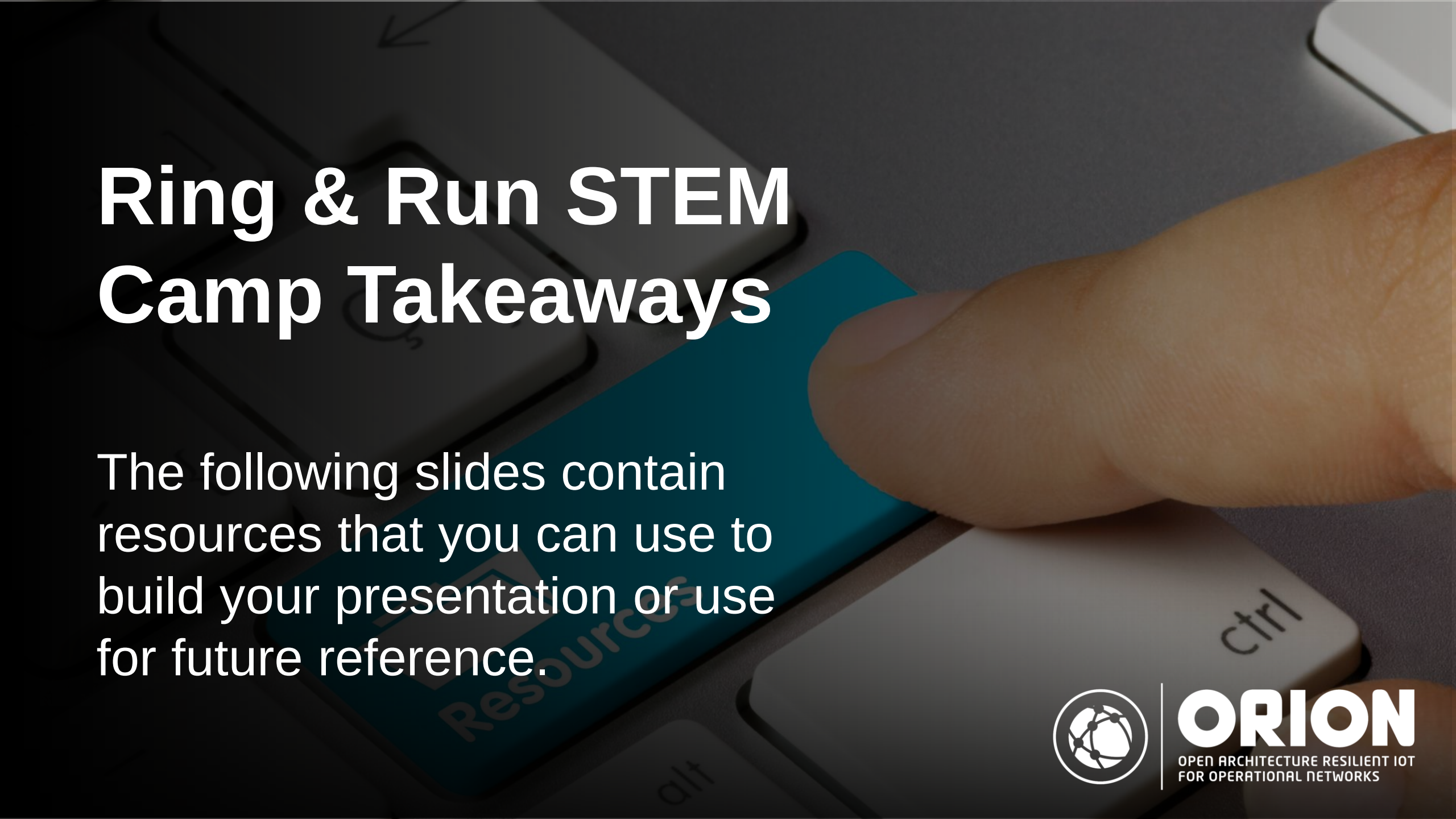
RING AND RUN STEM EVENT SPONSORED BY AIR FORCE RESEARCH LABORATORY, HOSTED BY GRIFFISS INSTITUTE.

CURRICULUM AND INSTRUCTION BY QUANTERION SOLUTIONS INCORPORATED.

ORION
OPEN ARCHITECTURE RESILIENT IOT FOR OPERATIONAL NETWORKS

Ring & Run STEM Camp Takeaways

The following slides contain resources that you can use to build your presentation or use for future reference.



ORION
OPEN ARCHITECTURE RESILIENT IOT FOR OPERATIONAL NETWORKS

IoT Resources

The proliferation of broadband Internet connectivity, increasing processing power of small devices, and the exponential growth of cloud computing capabilities have enabled an IoT revolution.

An IoT Overview is HERE.

ORION
OPEN ARCHITECTURE RESILIENT IOT FOR OPERATIONAL NETWORKS

Message Protocol Resources

The IoT realm involves many disparate devices, services, and user applications. One messaging protocol that has emerged as a leader for IoT applications is the Message Queueing Telemetry Transport (MQTT).

ORION
OPEN ARCHITECTURE RESILIENT IOT FOR OPERATIONAL NETWORKS

Raspberry Pi Resources

This single board computer is an excellent resource for learning and experimenting. It provides a wealth of capability in a small form factor at a reasonable price.

A Raspberry Pi Overview HERE.

ORION

OPEN ARCHITECTURE RESILIENT IOT FOR OPERATIONAL NETWORKS

Electronic Components

IoT solutions typically interface with electronic components to sense some environmental condition and then perform some sort of action in response.

For your continued exploration:
Physically Interfacing with a Raspberry Pi IoT Prototypes Using a Breadboard

ORION
OPEN ARCHITECTURE RESILIENT IOT FOR OPERATIONAL NETWORKS