Automated Watering system with IoT:

Implementation of precision irrigation with microcontroller

group 6

Overview

Objective

Develop an automated IoT-based small-scale watering system that monitors soil humidity and controls irrigation.

Inspiration

Inspired by modern smart agriculture trends and the microcomputer lecture labs.

Applications

Smart farming and gardening systems and automated greenhouse operations.

Outline

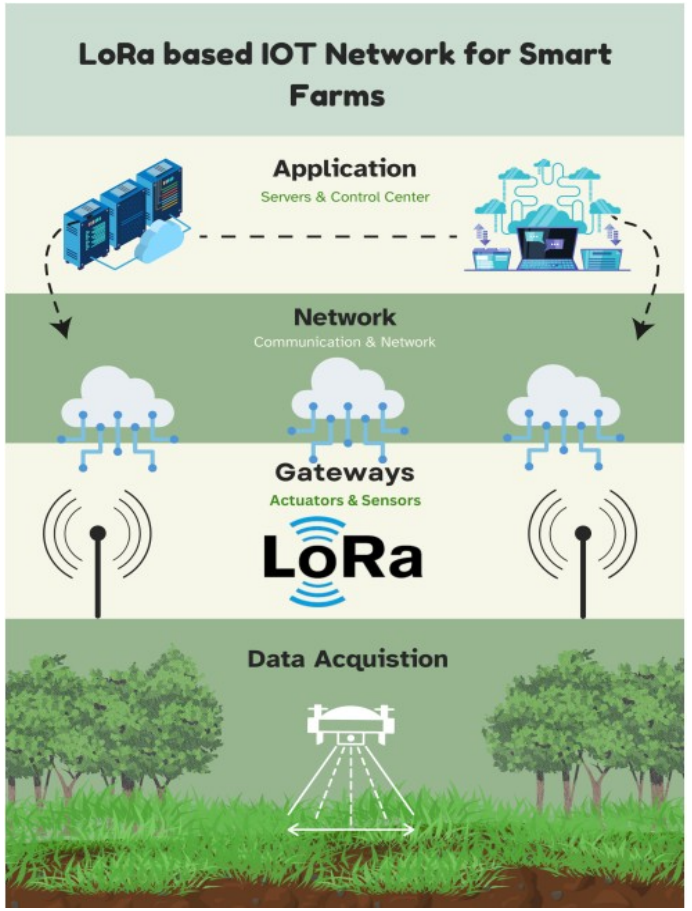
- Background research and Literature review
- System architecture
- Methodology
- Expected challenges and solution
- Relevance to course

Background Research and literature review

Research - arXiv:2409.11200

Title: LoRa Communication for Agriculture 4.0: Opportunities, Challenges, and Future Directions (2024)

- Long Range (LoRa) technologies is key enabler for agriculture IoT system.
  - Established a smart agriculture network with UAV、 sensor and monitor、 data server
- Challenge and future direction for LoRa-based IoT in smart agriculture.
  - Localization and Positioning Techniques.
  - Data compression and optimization.
  - Sustainable energy solution and energy efficiency.

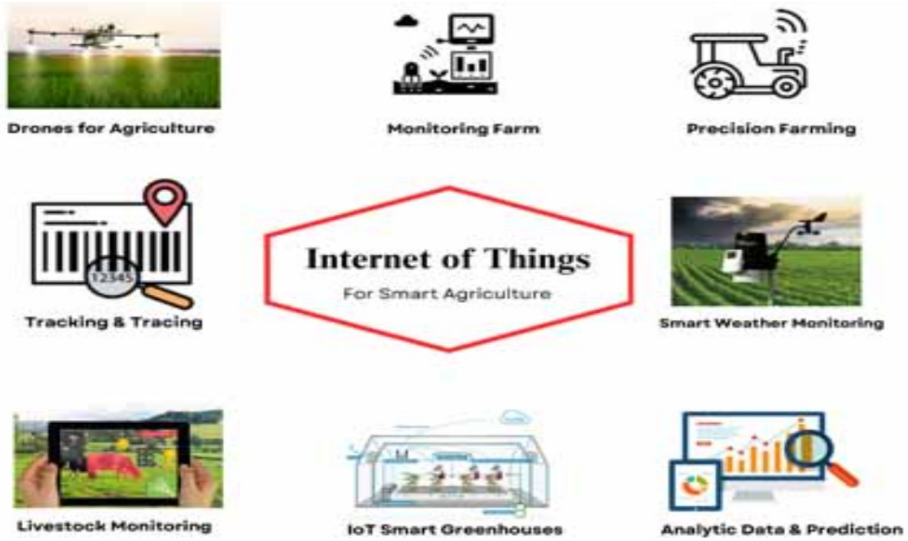


https://arxiv.org/abs/2409.11200

Research – IEEE 2169-3536

Title: Artificial Intelligence and Internet of Things for Sustainable Farming and Smart Agriculture (2023)

- AI and IoT have revolutionized agriculture and assist company in positive way.
  - More productive.
  - Reducing the amount of waste.
  - Satisfying the need for food.
- Smart farming has become an important component of sustainable agriculture.
  - Harvesting and recycling.
  - Management of nutrient in crop.
  - Control of pests and diseases.



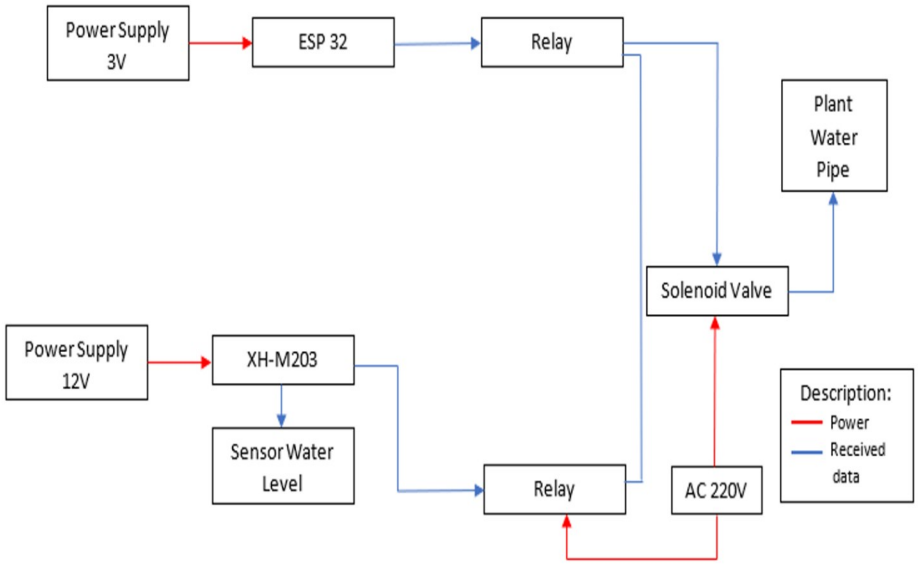


https://ieeexplore.ieee.org/document/10190626

Research – BIO Web of Conferences 137, 01014

Title: Implementation of an IoT-Based Automated Watering System for Melon Cultivation (2024)

- Solving agriculture task for some water-shortage region
  - Developing precision irrigation system.
  - Derive smart agriculture and resource management.
- Promoting the widespread adoption of IoT-based smart agriculture.
  - Make research cost efficiently .
  - More feasibility IoT agriculture .



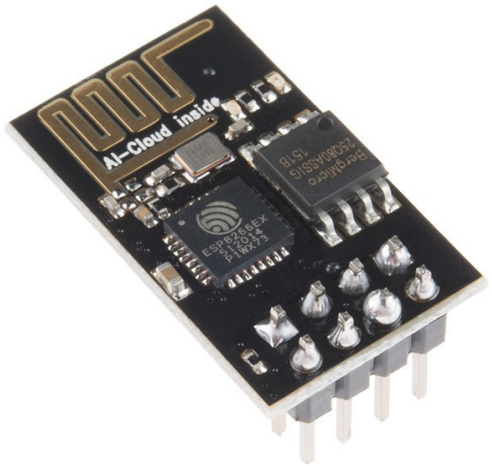
https://www.bio-conferences.org/articles/bioconf/abs/2024/56/bioconf_icbcd2024_01014/bioconf_icbcd2024_01014.html

System Architecture

System architecture - experimental equipment



Arduino



ESP8266

- Read the data from moisture sensor
- control the operation of motor

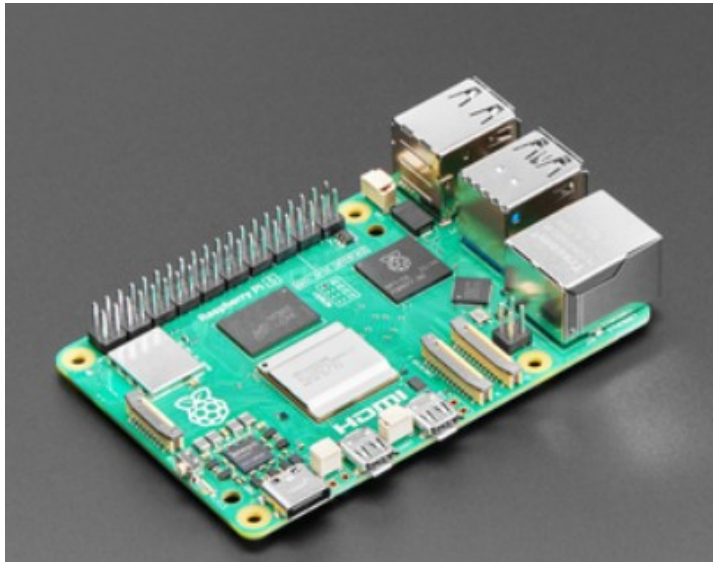
- Provides Wi -Fi connectivity for Arduino

System architecture - experimental equipment



pump

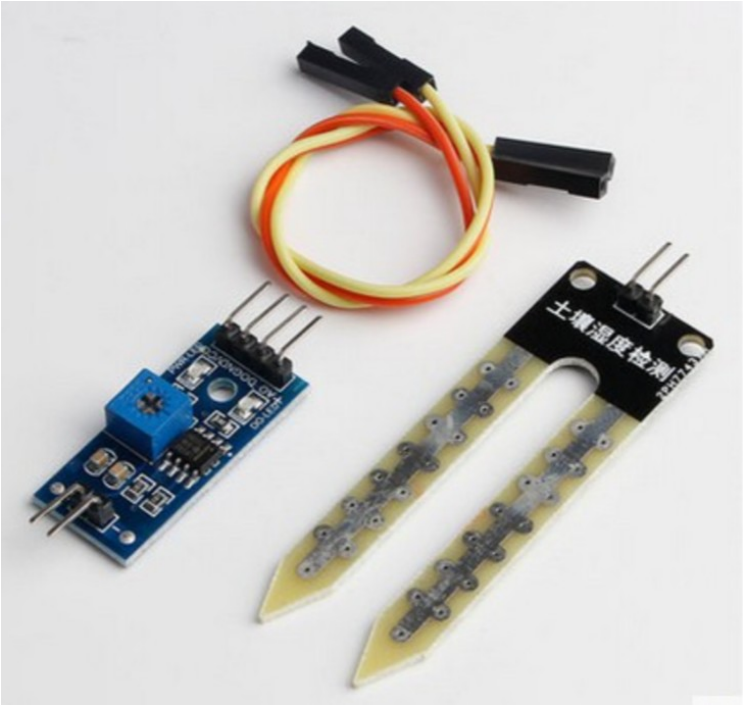
- Control by Arduino.
- Performing watering operation



Raspberry Pi

- Serves as a server of system
- Record data
- Provides web interface

System architecture - experimental equipment



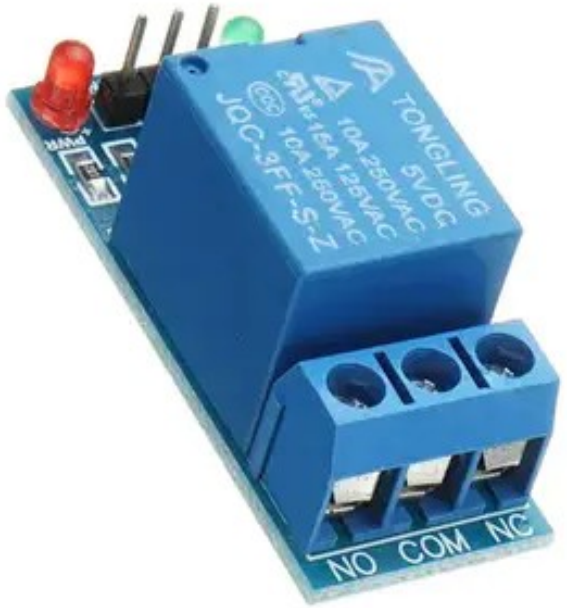
soil moisture sensor

- Measure moisture content in the soil

Input voltage : 3.3 ~ 3.5 (V)

output voltage : 0 ~ 2.3 (V)

System architecture - experimental equipment

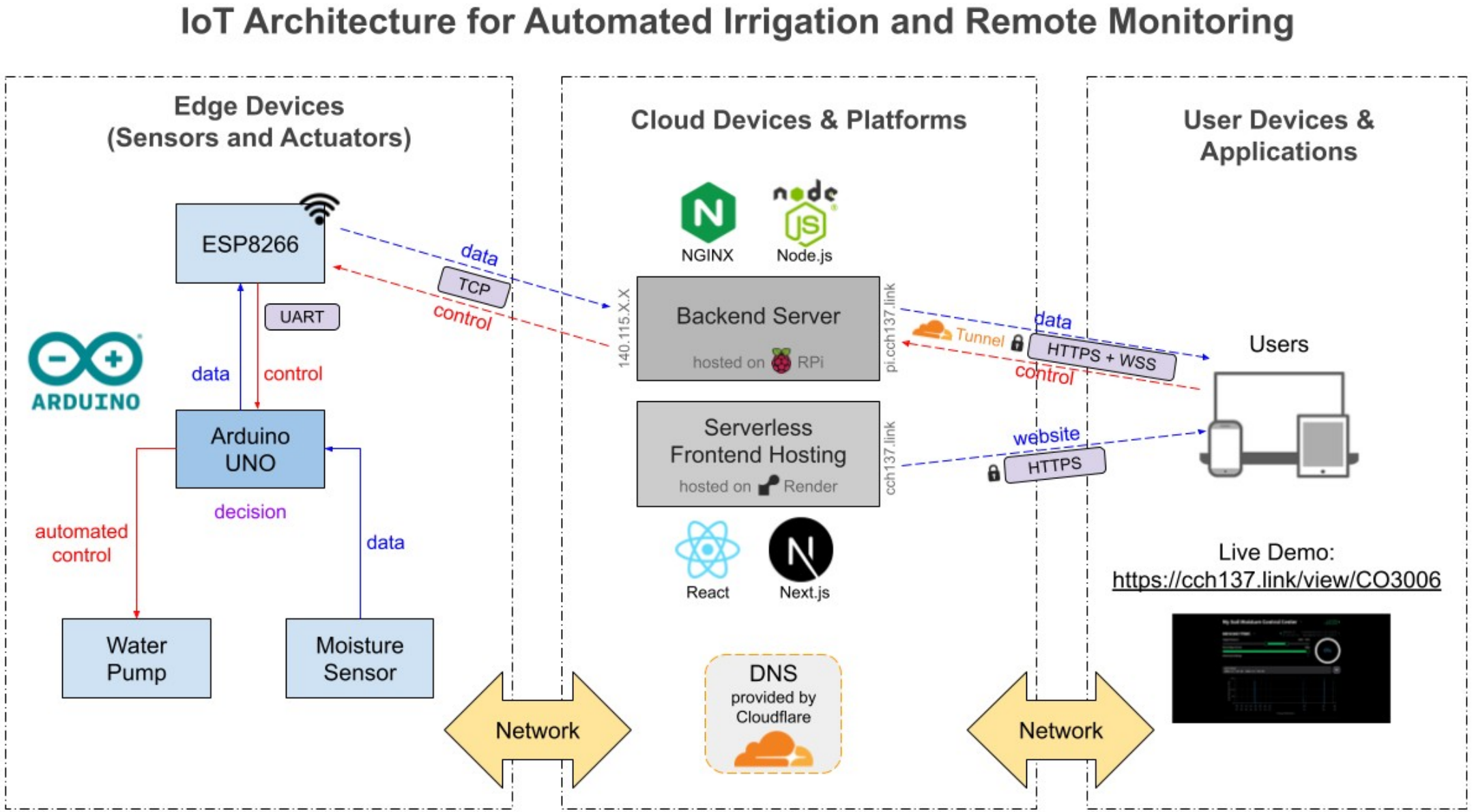


Relay
(High level trigger)

-Because Arduino can’t provide enough voltage to the pump, so we need to install batteries and use a relay to control the pump’s operation.

-When receiving the signal “1”, it will turn on the pump.

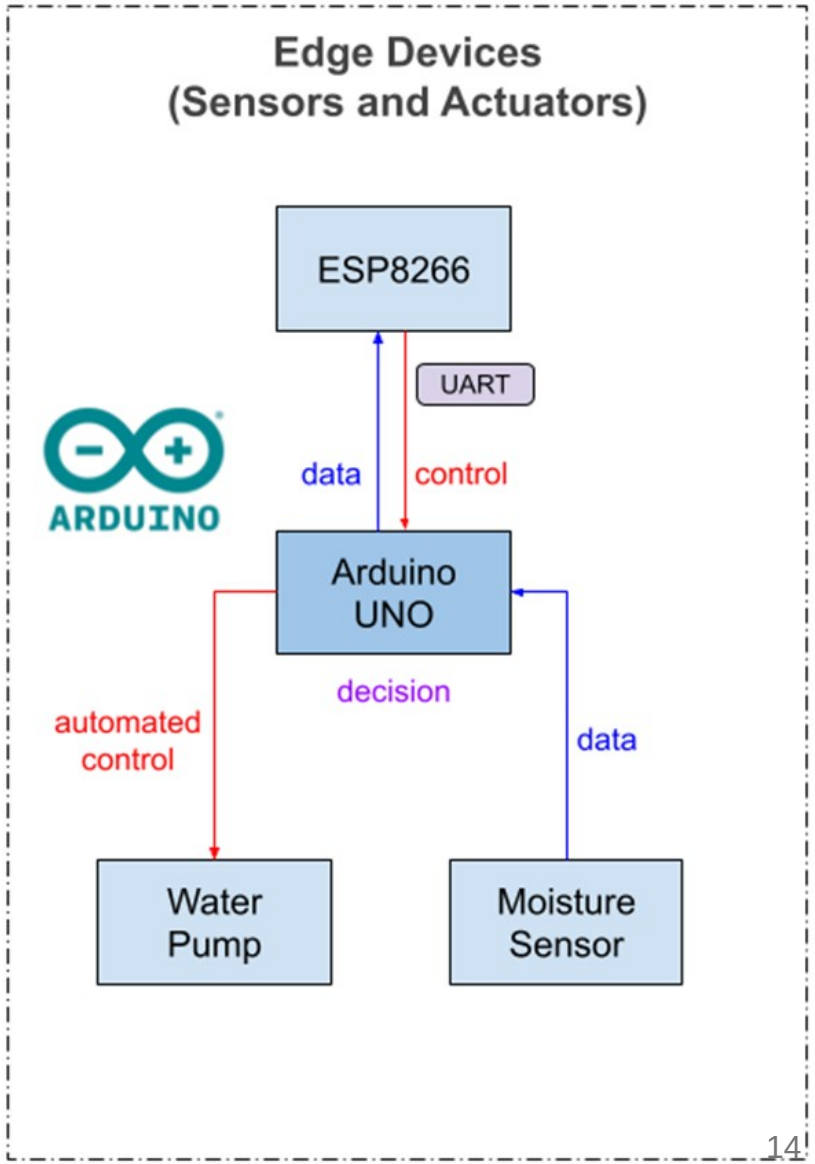
System architecture - system diagram



System architecture - system diagram

System function – Edge Devices

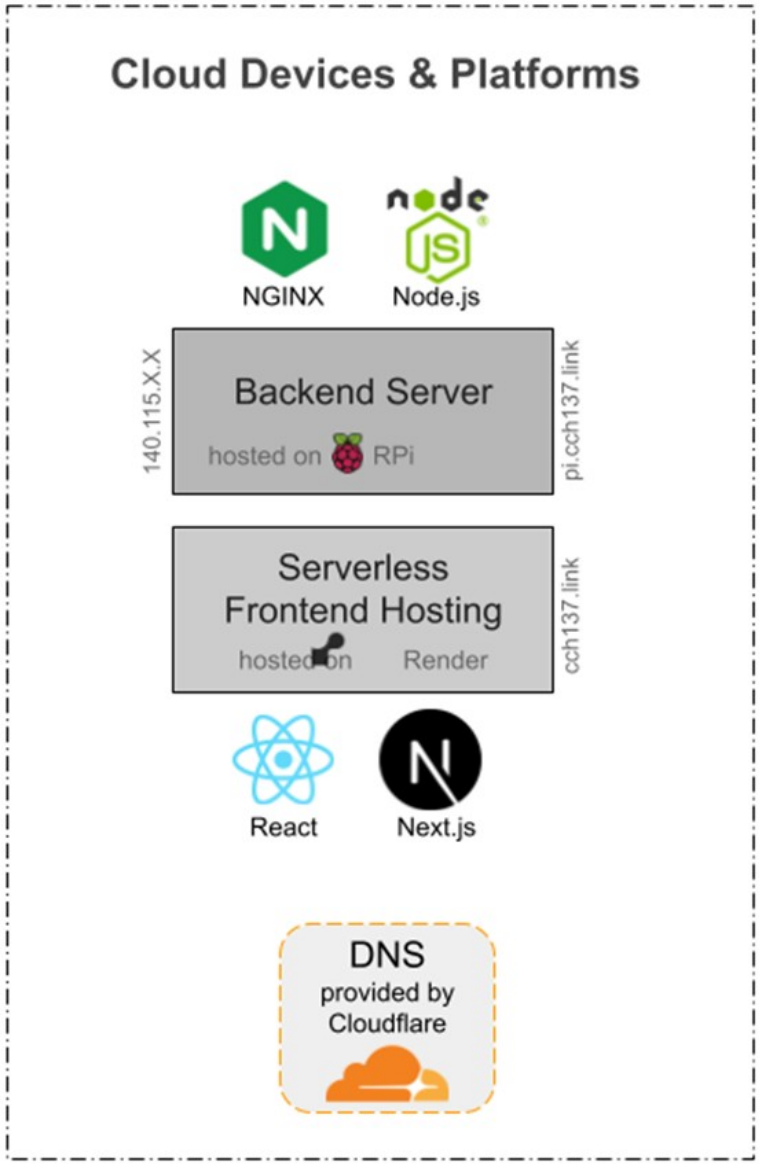
- Detect soil humidity and upload it to server.
- Make watering decisions based on detected data and settings.
- Connect to the network, automatically reconnect after disconnection.



System architecture - system diagram

System function – Cloud Platforms

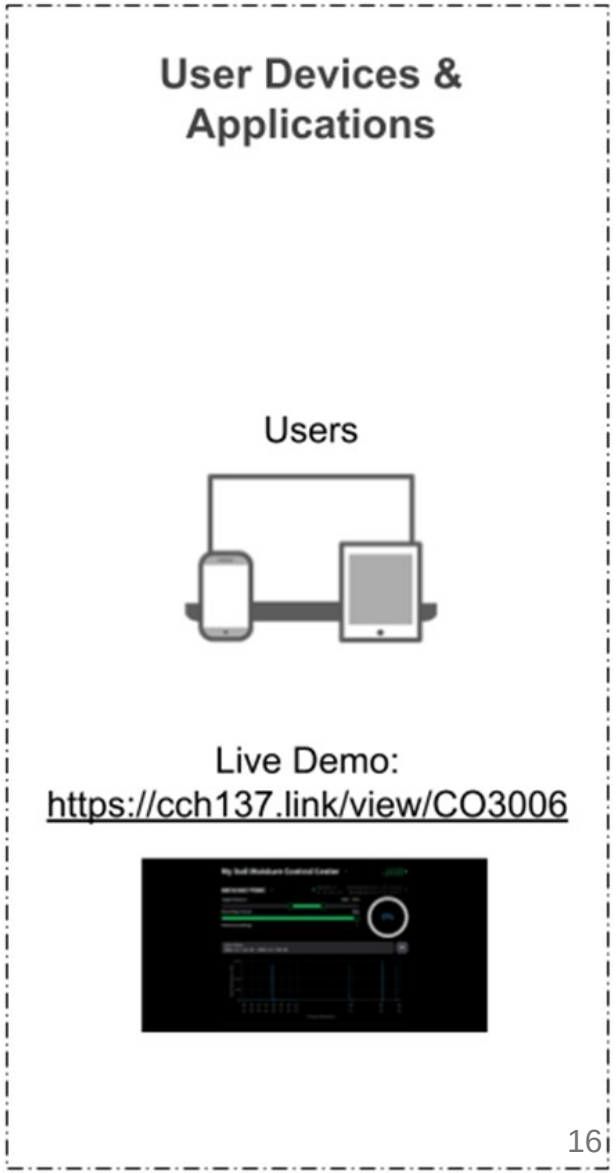
- Build connection with edge devices.
- Save detected data.
- Forward and process data as a Gateway.



System architecture - system diagram

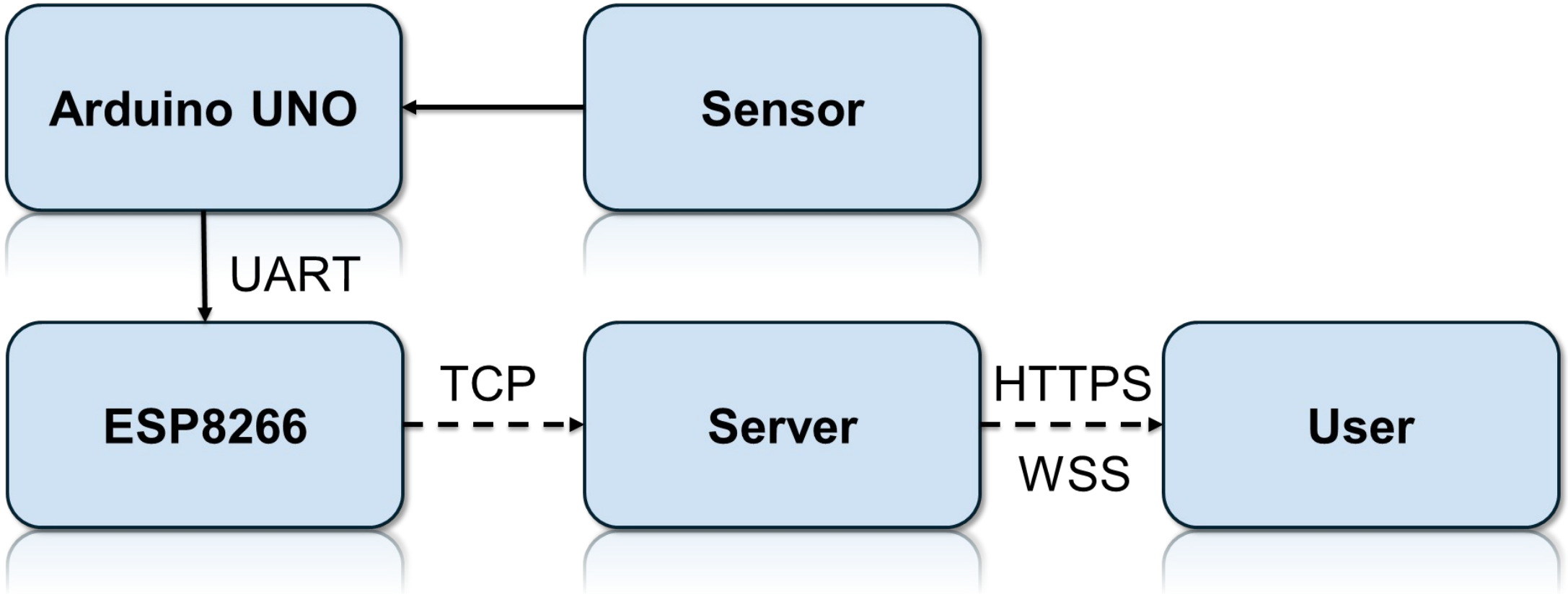
System function – Application

- Enable user to review the historical and real-time data.
- Allow users to manipulate edge devices settings according to their own demand.



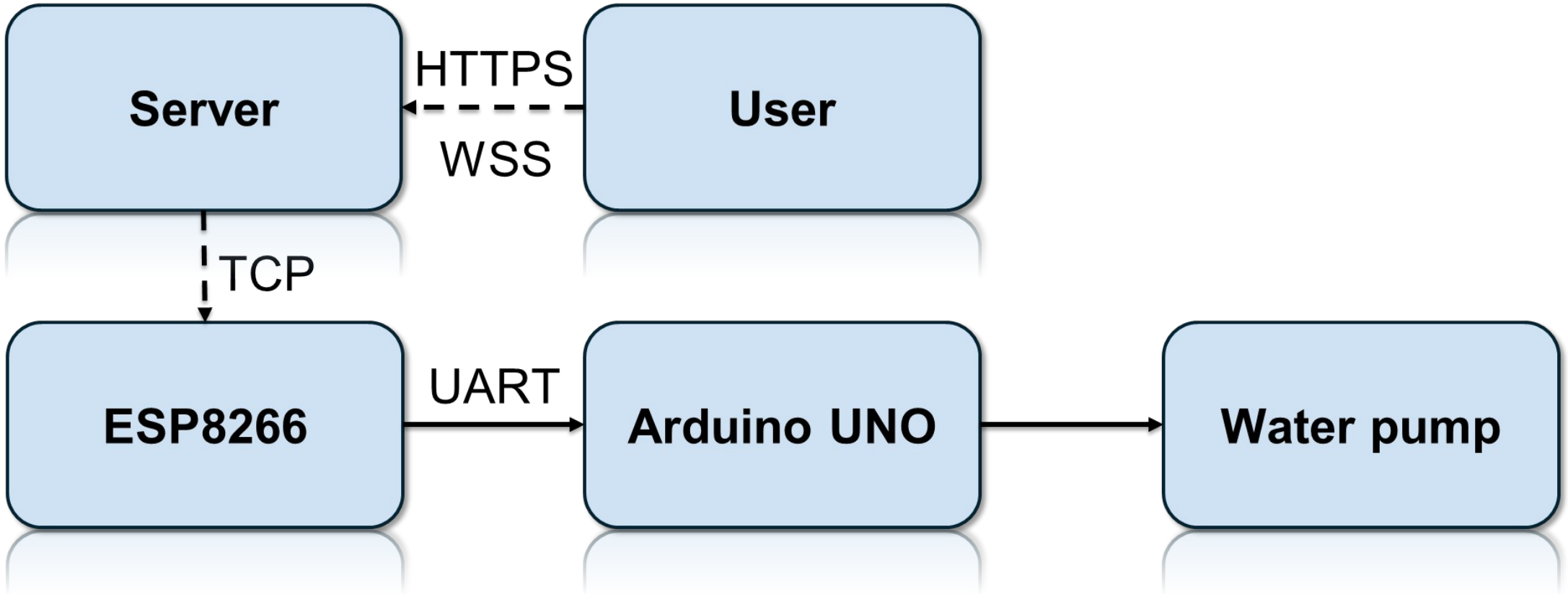
System architecture - system diagram

Data trend – humidity percentage



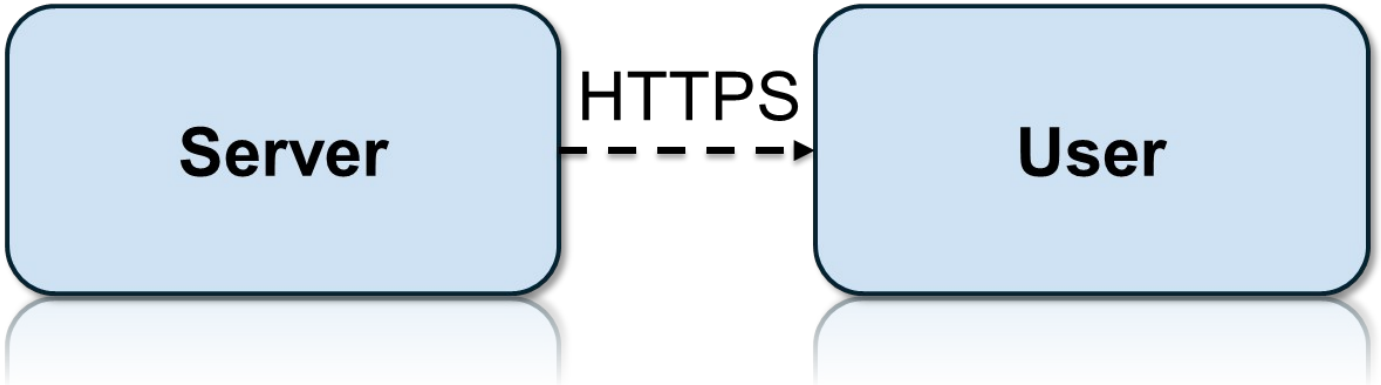
System architecture - system diagram

Data trend – Decision parameter setting



System architecture - system diagram

Data trend – Website & Historical data



Methodology

Methodology-Edge devices

Definition & formula:

$$M = \left(1 - \frac{\max(V_{raw} - V_{offset},\ 0)}{1023 - V_{offset}}\right) \times 100$$

M: Soil moisture percentage (0~100%)

V_{raw}: Return value of the sensor (0~1023)

V_{offset}: Offset of V, we use it to correct the value.

The default value is 350 (0~1022)

Methodology-Edge devices

M: Soil moisture percentage (0~100%)
L: The lower limit of soil moisture percentage (0~100%)
U: The upper limit of soil moisture percentage (0~100%)
I: The time interval of transmitting data. The unit is milliseconds.

Operation logic:

1. When $M<L$, start watering.
2. When $M \geq U$ and watering, stop watering.
3. Return M to the server per I milliseconds.

V_{offset}, L, U, I can be changed by user.

Methodology-Equipment wiring

ESP8266	Program	Operation
TX	CP2102 RX	Arduino A4(simulated RX)
RX	CP2102 TX	Arduino A3(simulated TX)
EN		Arduino 13
3.3V	Breadboard 3.3V	Breadboard 3.3V
GND	Breadboard GND	Breadboard GND
RST	Triggered before programing	Triggered before operation
GPIO0	Breadboard GND	
GPIO2		

Arduino	Program	Operation
3.3V		Breadboard 3.3V
5V		Breadboard 5V
GND	No wiring is allowed during programing	Breadboard GND
RST		Arduino 7
5		Relay IN
7		Arduino RST
13		ESP8266 EN
A3		ESP8266 RX
A4	No wiring is allowed during programing	ESP8266 TX
A5		Sensor AO

Methodology-Equipment wiring

Relay	
IN	Arduino 5
VCC	Breadboard 5V
GND	Breadboard GND
Others	Motor

Sensor	
AO	Arduino A5
VCC	Breadboard 5V
GND	Breadboard GND

Breadboard	
Breadboard GND	Arduino GND ESP8266 GND Sensor GND Relay GND
Breadboard 3.3V	Arduino 3.3V ESP8266 3.3V
Breadboard 5V	Arduino 5V Relay VCC Sensor VCC

Methodology-Custom packet format

1. Reduced Bandwidth Usage: Custom packets are more efficient than JSON and XML, making them particularly suitable for IoT scenarios.

2. 1-Byte Header: Each header is 1 byte, with a value range of 0 to 255.

3. Opcode: The header value serves as an opcode, and devices execute actions based on the opcode.

4. Flexible Payload: Each packet can have a different payload design depending on the requirements.

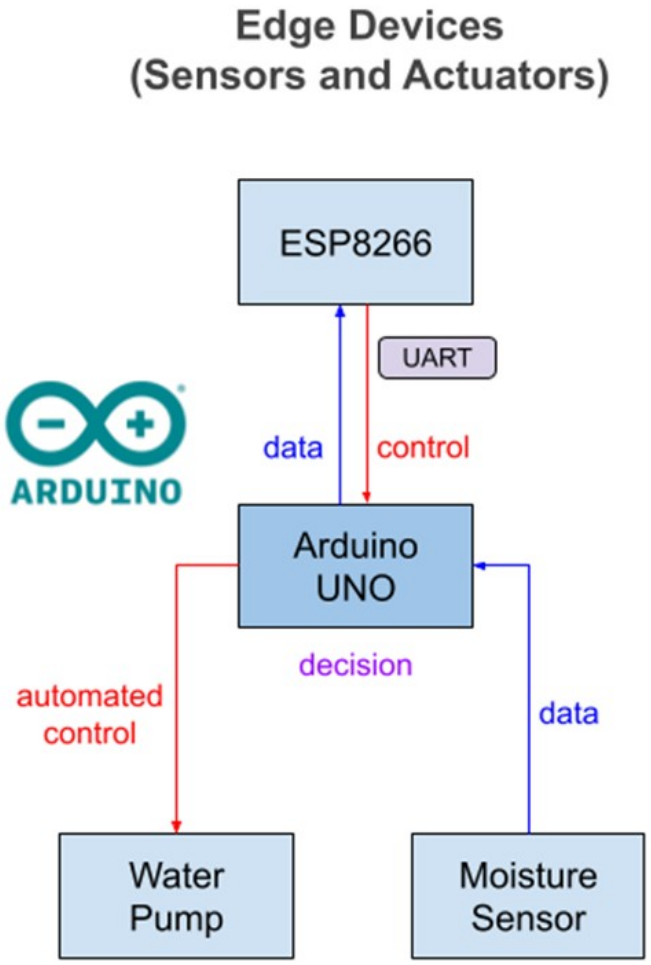
Methodology-Custom packet format

Opcode (uint8)	Description	Packet length (byte)
General		
0	Invalid opcode, initial value of packet header	0
Server → ESP8266 (TCP)		
101	PING	1
102	PONG	1
112	Set variables (restore the last setting at startup or triggered by the user updating the value) Payload: [uint32 × 4] V_offset, L, U, I	1+4+4+4+4
113	Request Arduino variable configuration	1
ESP8266 → Server (TCP)		
101	PING	1
102	PONG	1
110	Submit soil moisture percentage (forwarded from Payload: [uint8]H	1+1
111	Submit the current edge configuration (forwarded Payload: [uint32 × 4] V_offset, L, U, I	1+4+4+4+4
114	Request the variable configuration of the server (f	1

Methodology-Custom packet format

ESP8266 → Arduino (UART)		
112	Set variables (forwarded from server) Payload: [uint32 × 4] V_offset, L, U, I	1+4+4+4+4+1
113	Request the variables of the edge device (forwarc	1+1
120	Transfer text messages to Arduino's Serial Payload: [string] s message string (ending with \n)	1+len(s)+1
Arduino → ESP8266 (UART)		
110	Submit soil moisture percentage Payload: [uint8]H	1+1+1
111	Submit the current edge variable configuration Payload: [uint32 × 4] V_offset, L, U, I	1+4+4+4+4+1
114	Request the variable configuration of the server	1+1

Methodology-UART



UART (Universal Asynchronous Receiver-Transmitter)

-We use UART to build bidirectional communication between Arduino and ESP8266.

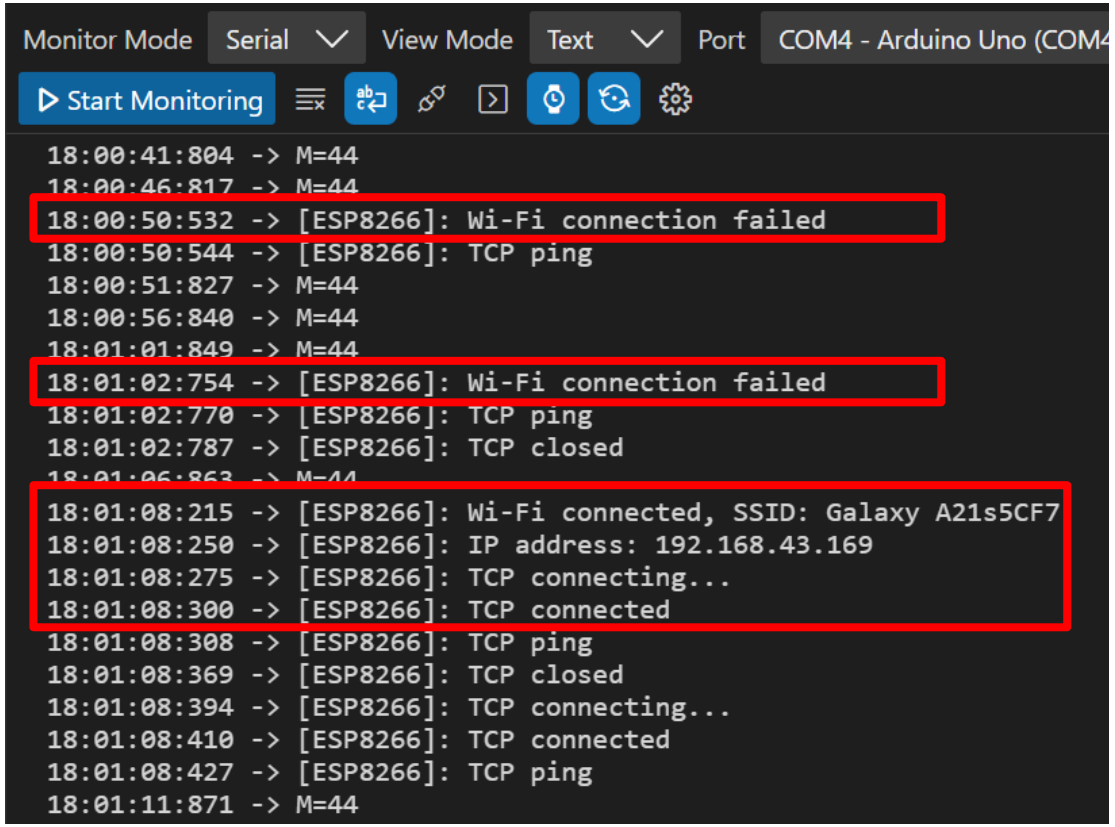
-On hardware, we connect the TX and RX of Arduino to the RX and TX of ESP8266 respectively.

-On software, we use serial to send and receive data, choosing 9600 Baud rate.

Methodology – Auto-restart mechanism (AR)

Sometimes, we may encounter unexpected connection failures ……

* We intentionally turned off the Wi-Fi hotspot to test its reconnection mechanism.



Possible reasons of failures

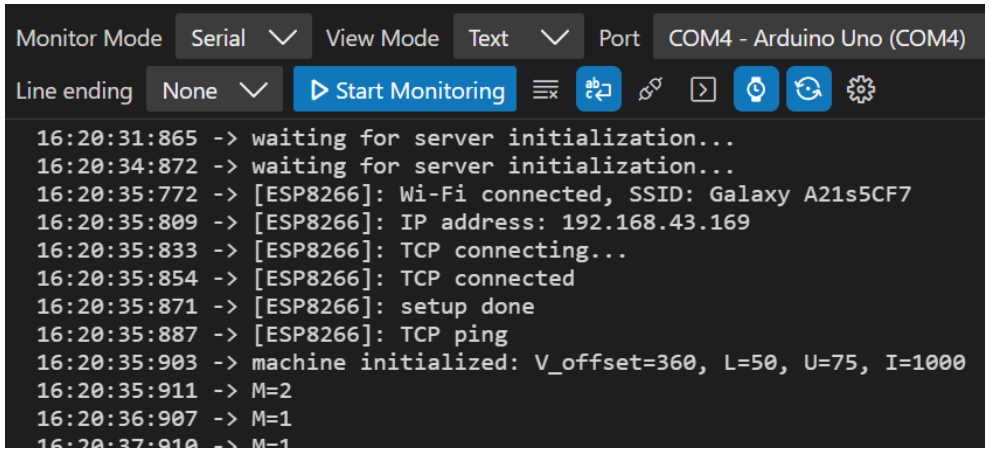
- Connection interruptions
- Insufficient memory
- Unstable signals
- Data packet errors (parsing failure)

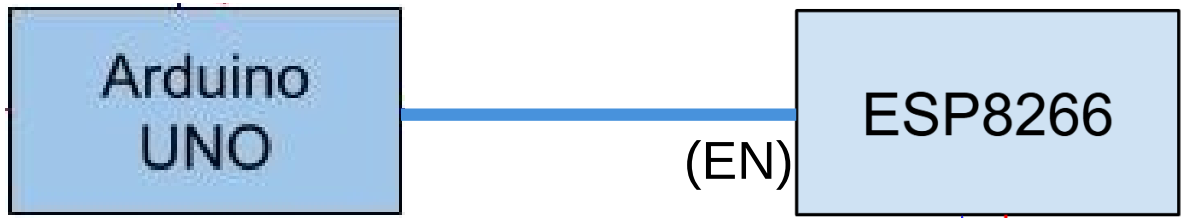
Methodology – Auto-restart mechanism (AR)

Therefore, we designed an auto-restart mechanism in the code.

* Success in connected





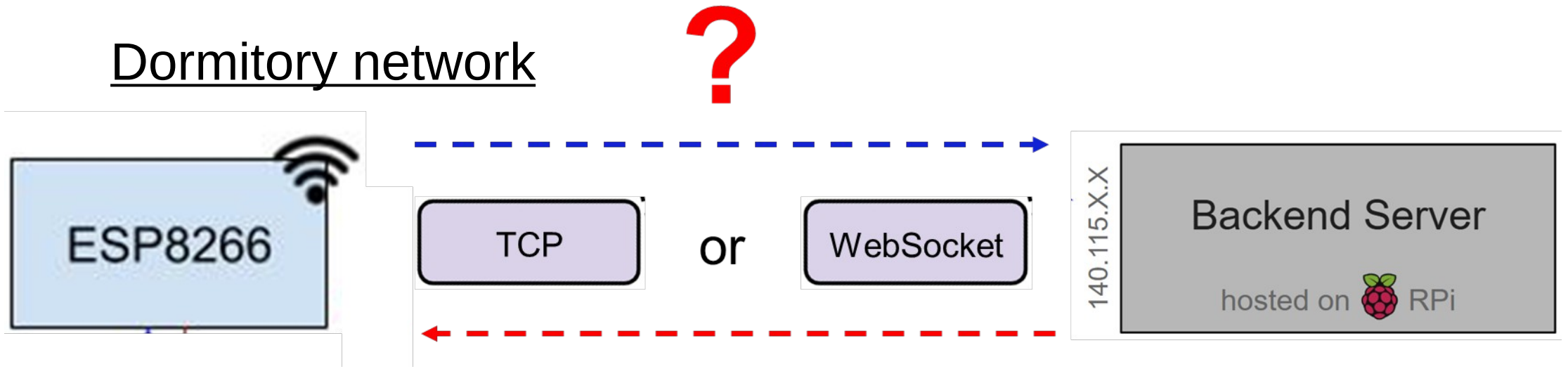


- Restart the system when an anomaly is detected.
- Control the EN pin of the ESP8266 to trigger a reset.

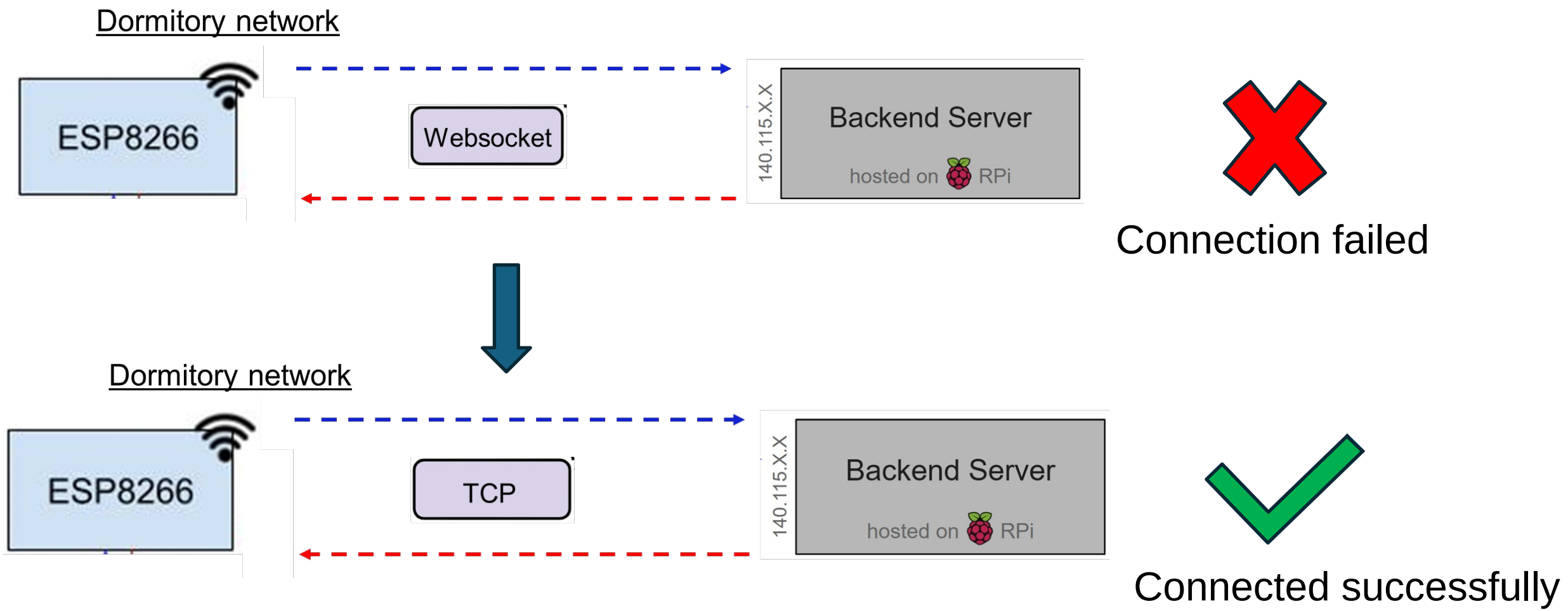


- If the Wi-Fi or WebSocket disconnected……
- System will attempt to restore the connection to ensure stable functionality

Methodology – TCP or WEBSOCKET

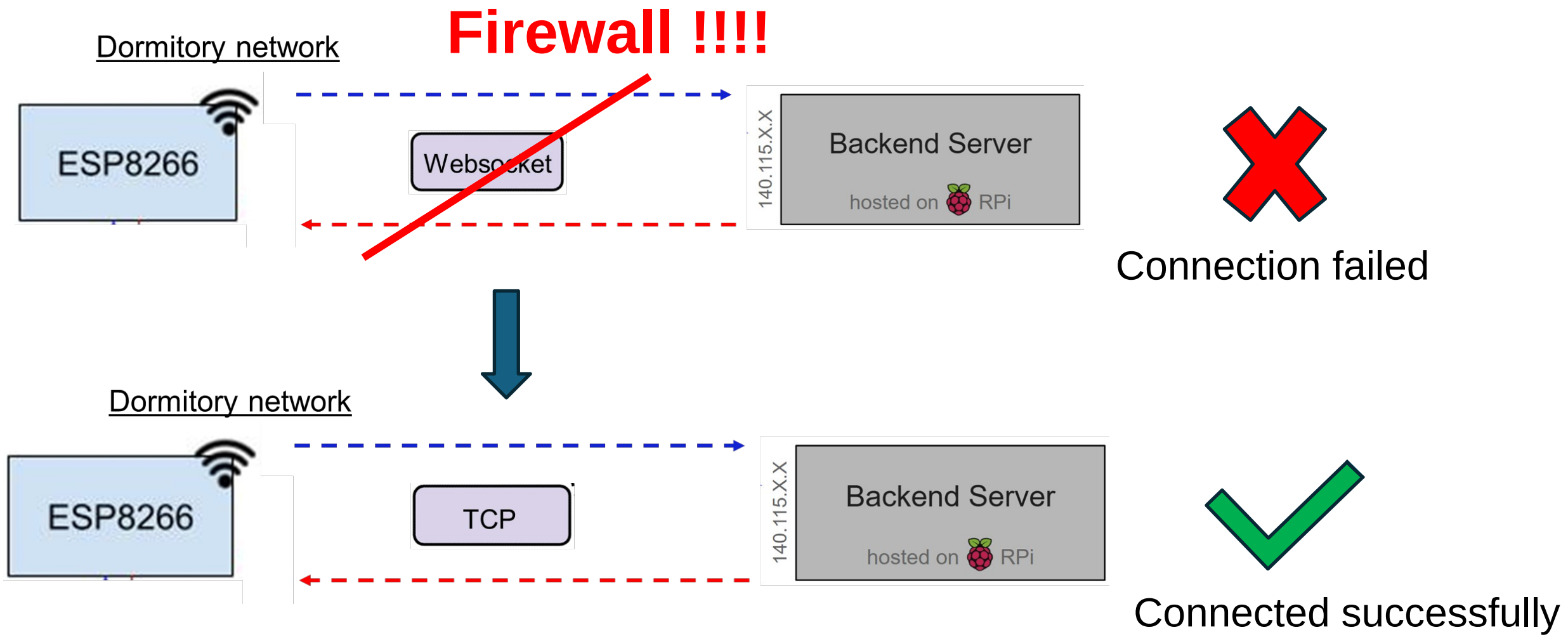


Methodology – TCP v.s WEBSOCKET(assumption)



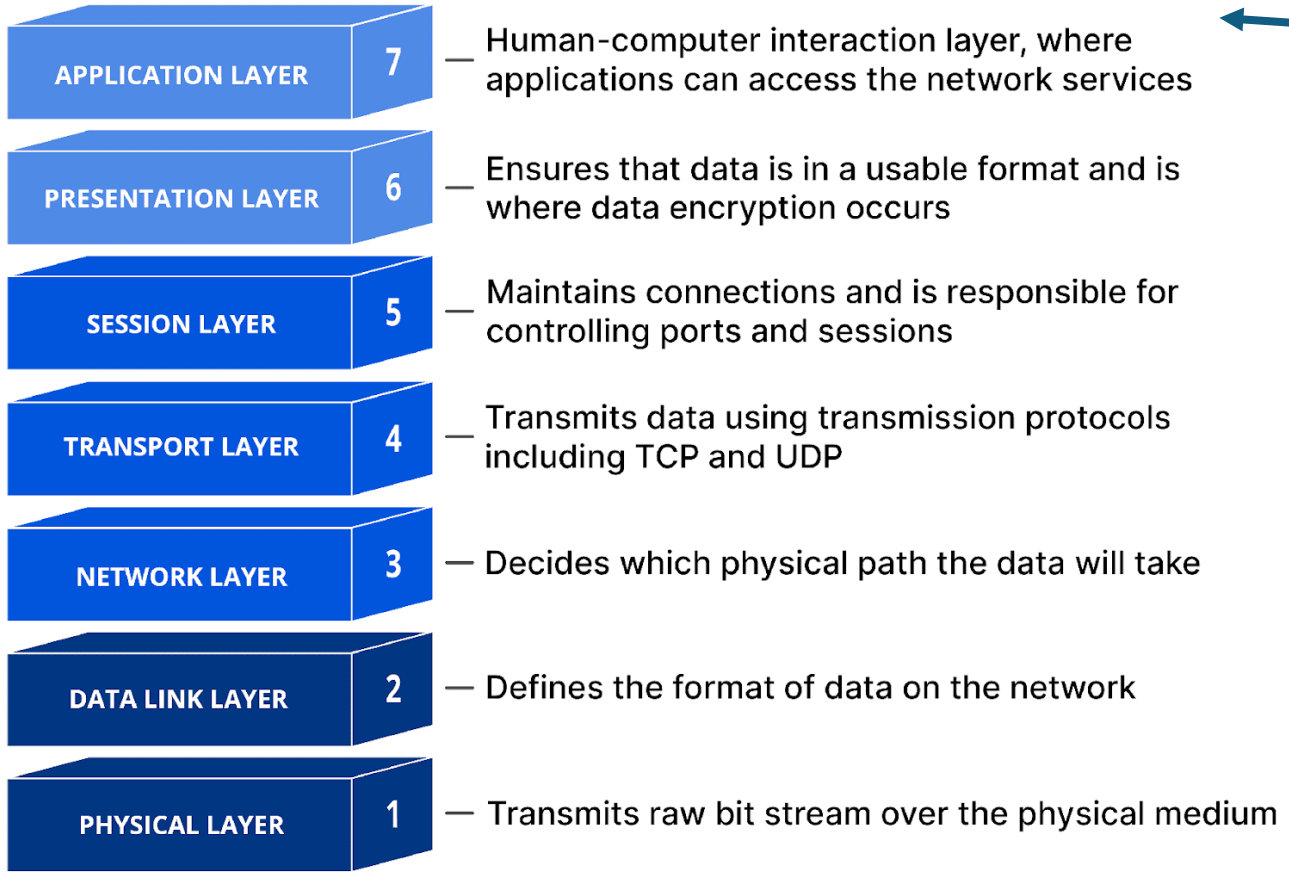
Success in connected when we change Websocket into TCP

Methodology – TCP v.s WEBSOCKET(assumption)



Success in connected when we change Websocket into TCP

Methodology – TCP v.s WEBSOCKET(assumption)



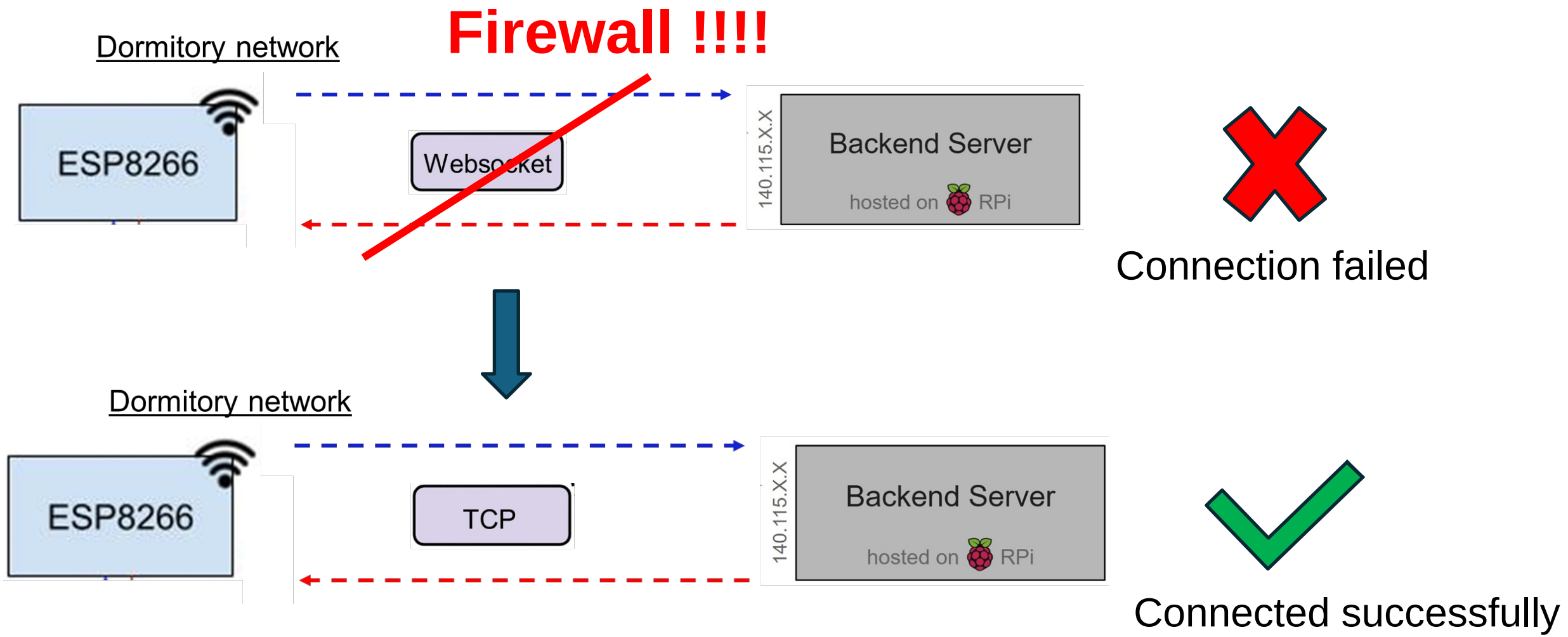
OSI Model

HTTP 、 Websocket
(7_ Application Layer)

TCP 、 UCP
(4_ Transport Layer)

- Firewalls are often configured with rules to filter specific application-layer.

Methodology – TCP v.s WEBSOCKET(assumption)



Success in connected when we change Websocket into TCP

Challenges and solutions

Challenges & Solutions

How to design the watering device?

water bottle in a high place

using a pumping motor

Challenges & Solutions

Why not plants?

A: We are mainly designing a precision irrigation tool.

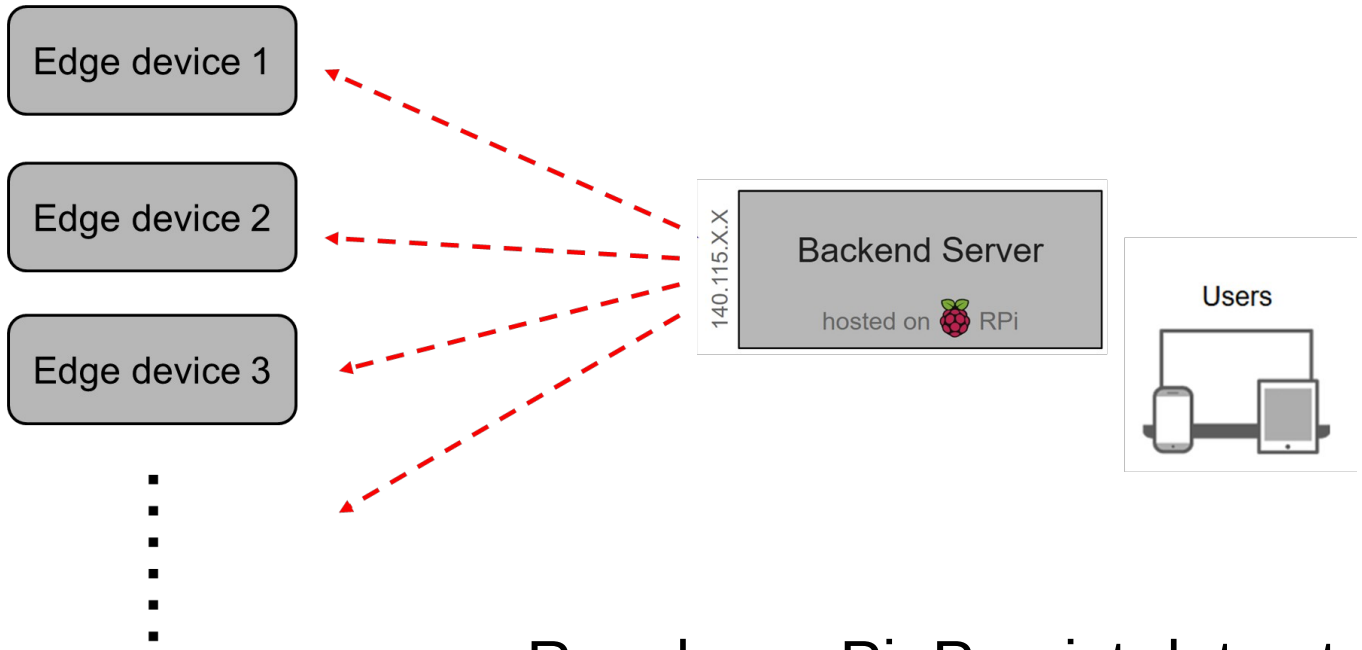
Automated Watering system with IoT:

Implementation of precision irrigation with microcontroller

Precision irrigation tool combined with automatic sensors and microcontroller implementation

Challenges & Solutions

Cost and commodification?

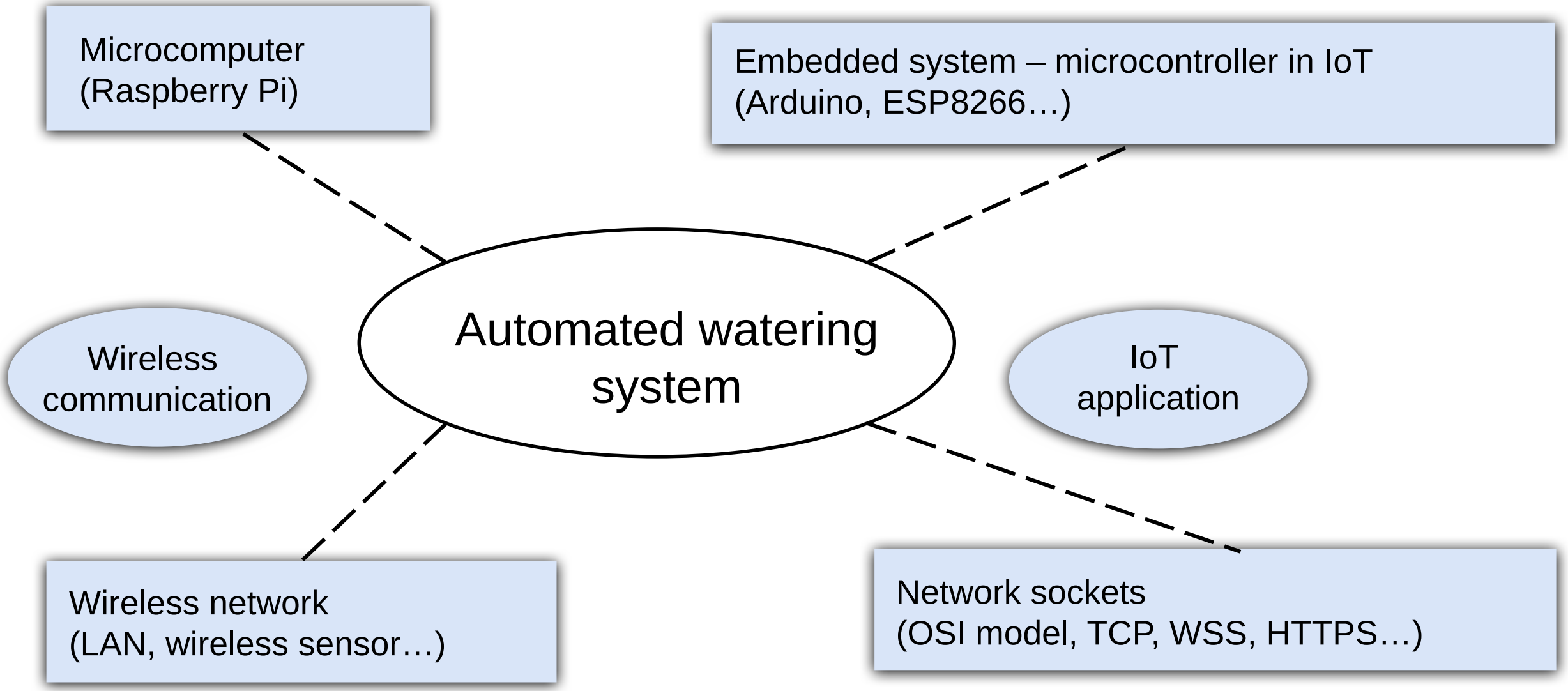


- Feasibility IoT agriculture
- Automated watering system
- Cost – efficiency

Raspberry Pi :Persist data storage and act as a server

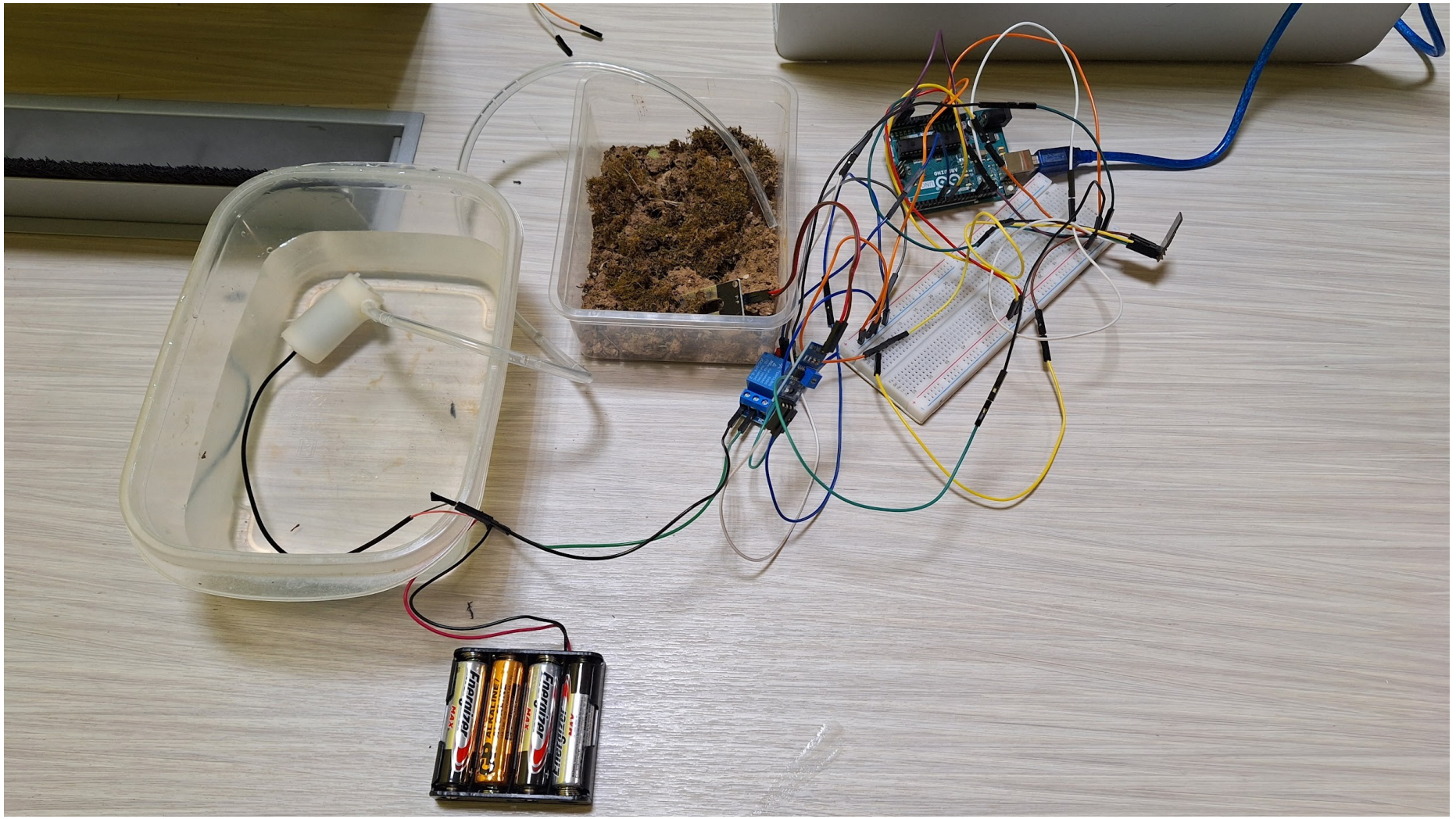
Relevance to course

Project relevance to the course

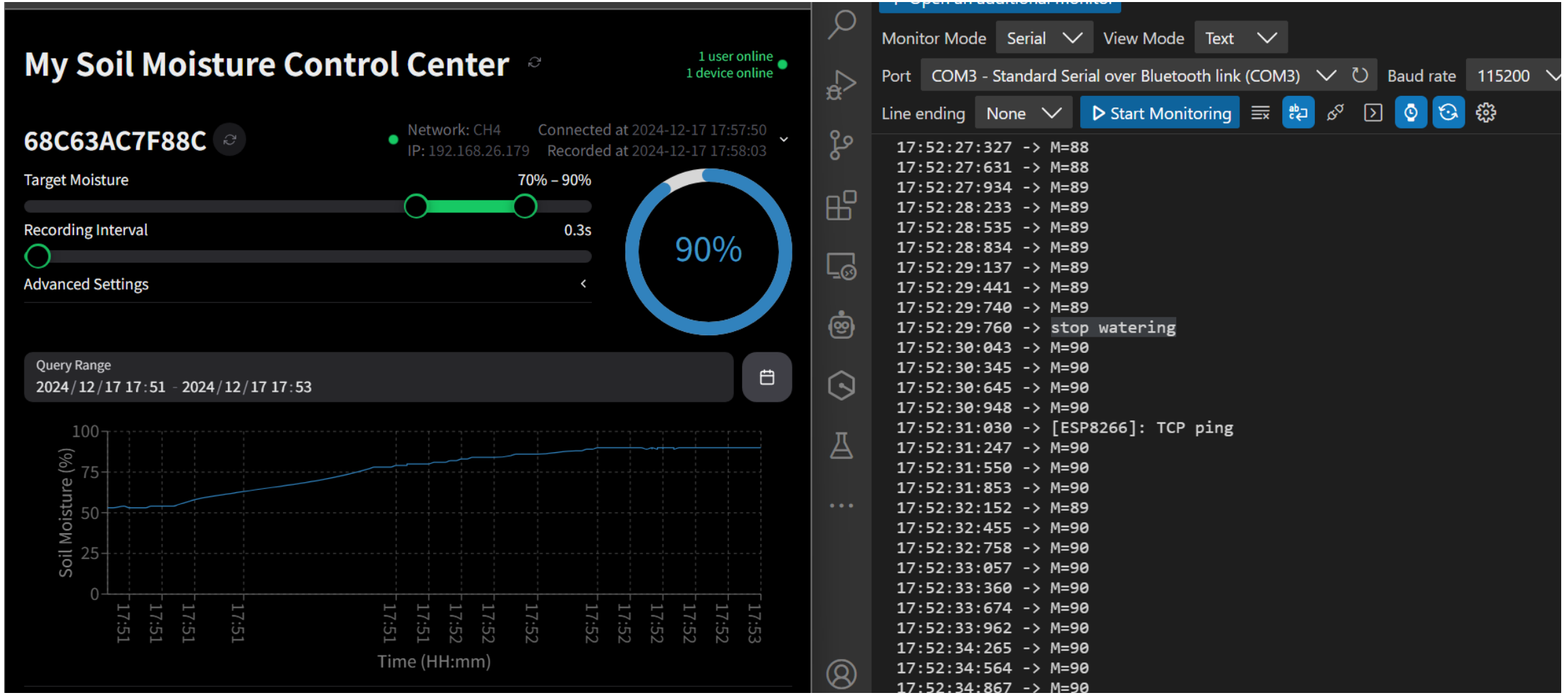


Demo

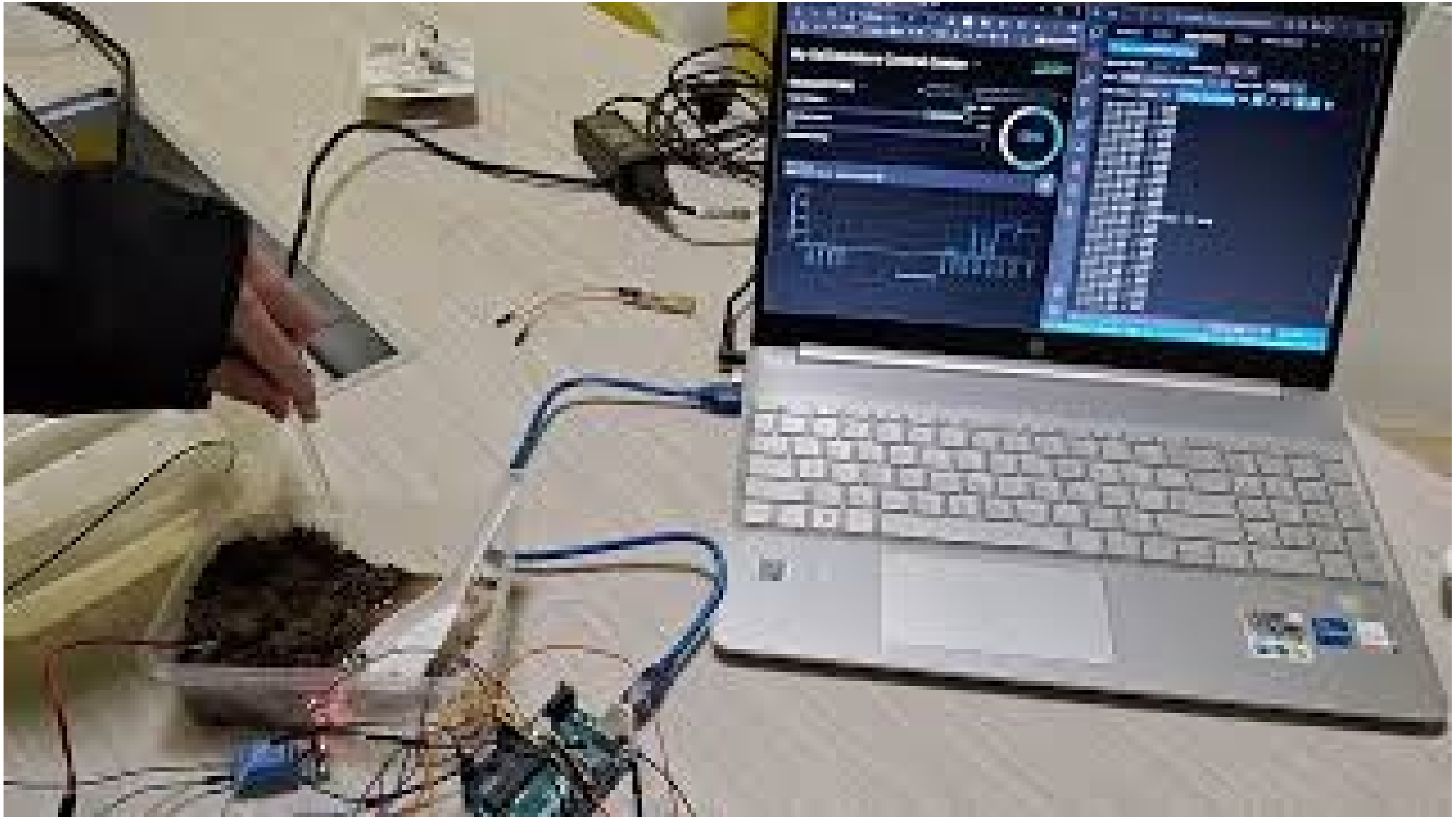
Demo – 裝置照片 (放照片)



Demo – 網頁展示 (放我們成功完的網頁)



Demo – 影片展示



Demo now

Thank you for listening