Patryk Lamentowicz (29481), Sabin Khatiwada (29907), Chee Yan San (28996)

Smart Watermeter

1. Introduction

Do-It-Yourself (DIY) is a concept that encourages us individuals to create, repair, or modify things on their own, without relying on professionals or experts. In the last few years, these DIY projects are getting more popular. In this project, a DIY approach was taken to create a smart watermeter with cheap and easily available components and some help from the internet with codes and tutorials. This project also has an artificial intelligence (AI) kind of edge to it that it has a remote, constant monitoring feature i.e. it automates manual human tasks. Although this project focuses on the watermeter, this approach can be applied to create other kinds of remote monitoring systems as well. This smart watermeter monitors the water flow within a gauge by taking a picture of the meter reading and converting that picture into a digital number using optical character recognition technology, then it saves the data and can also transmit the data into the cloud for remote observation using its WiFi internet connection.

With its features, this device can help the user recognize their usage patterns and norms, and easily detect abnormalities (leaks, overuse of children or tenants). This can enable the user to establish their conservation goals to minimalize consumption in order to keep track of sustainability goals.

2. Components

The components required for this project are listed below:

a. ESP32-CAM

The ESP32-CAM is a popular and cheap development board based on the ESP32 microcontroller. Its dimensions of 27mm*40.5mm*4.5mm make it small but a powerful tool. It also comes with a built-in camera module and microSD card slot and has Wi-Fi and Bluetooth connectivity feature with a dual-core processor, making it suitable for a wide range of IoT (Internet of Things) applications, remote controlling, or inspection systems. More details at: https://docs.ai-thinker.com/en/esp32-cam



Figure 1. ESP32-CAM
Source: https://www.nutsvolts.com/magazine/article/build-a-video-camera-using-the-esp32-cam-board

b. UartSBee v5.0 Module

The ESP32-CAM used in this project does not come with its own programmer hence UartSBee v5.0 module was used. The UartSBee v5.0 module is a compact and versatile UART (universal asynchronous receiver-transmitter)-to- USB (universal serial bus) adapter. It enables easy communication between microcontrollers (like ESP32-CAM in our case) or other UART-enabled devices and a computer (PC/laptop) through USB. This module is commonly used in electronics projects for debugging, programming, and data exchange, making it a convenient tool for developers or hobbyists. For this project UART of FT232 was used. More details at: https://wiki.seeedstudio.com/UartSBee_v5/



Figure 2. UartSBee v5.0 Module
Source: https://wiki.seeedstudio.com/UartSBee_v5/

c. MicroSD Card

The microSD card is a small, portable memory storage device used in various electronic devices. It serves as an external storage medium, commonly found in smartphones, digital cameras, tablets, and other devices. Its sizes range from small and large storage capacity (32, 64 GB, etc) making it ideal for storing photos, videos, document files, and other types of data. For the purposes of our project, firstly the program required is flashed into the microcontroller from it. Later the device data like pictures taken, logs, etc are also saved into it.

d. USB A-micro USB cable

The USB A to micro USB cable is a common data and charging cable used with smartphones and other portable devices. It enables easy connection and data transfer between devices with USB Type-A and micro USB ports. For this project, it is required to send the codes to the programmer. It is also the source of power for the programmer and the microcontroller.

e. Watermeter

The watermeter, usually found at home, is a device used to measure the amount of water consumption. It is typically installed by utility companies to track water usage in residential properties. The meter provides essential data for billing purposes, helping homeowners and utility providers monitor and manage water consumption efficiently.

In the ideal case, a watermeter provided by the local authorities is used. For this project, the device was trained on the pictures of water meters with different reading levels on them.



Figure 3. Example Watermeter
Source: https://wasser.badenovanetze.de/wasser-allgemein/wasserzaehler/

f. Microcontroller housing

To mount the setup onto the watermeter, a cardboard housing of length 10 cm and 9 cm diameter was designed. This housing helps to create a controlled environment with a fixed distance from the watermeter and also no external light coming in. Also, the lens was focused on the object by anticlockwise rotation of the lens. The focus calibration was performed hit and trial method, as different focal length requires different calibration.



Figure 4 (a). Housing without the ESP32-CAM

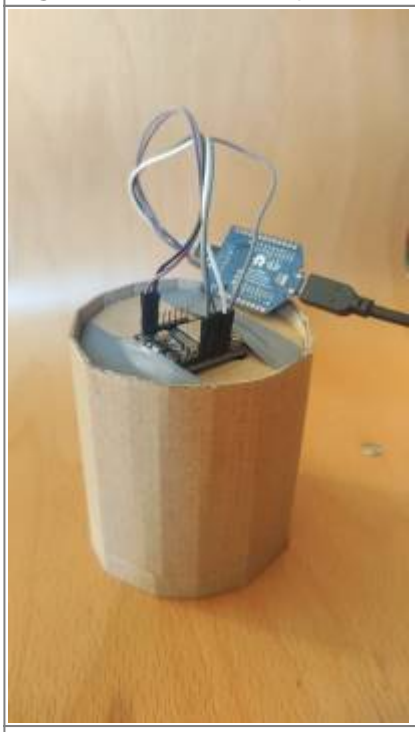


Figure 4 (b). Housing with the ESP32-CAM

3. Method

a. Hardware Setup

Using jumper cables, ESP32-CAM was connected to the FTDI programmer. The connection setup can be seen in Figure 5 below. Figure 6 is the actual experimental setup of this project. Whereas in Figure 7, the schematic representation of the ESP32-CAM and FTDI programmer is shown. GPIO 0 and GND are both connected in ESP32-CAM to upload the code (Program). After the programming code is uploaded, Transfer-receive and programming cables can be unplugged. Only 5v/Vcc and GND connection is required now for the device to function.

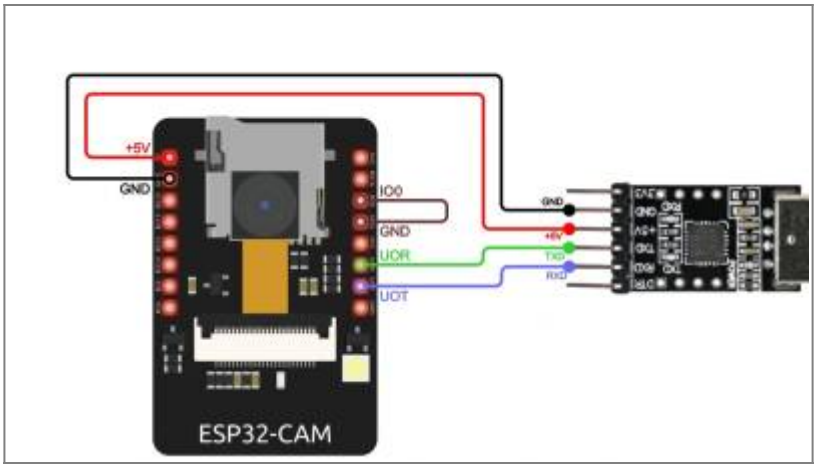


Figure 5. ESP32-CAM and Programmer connection Sketch

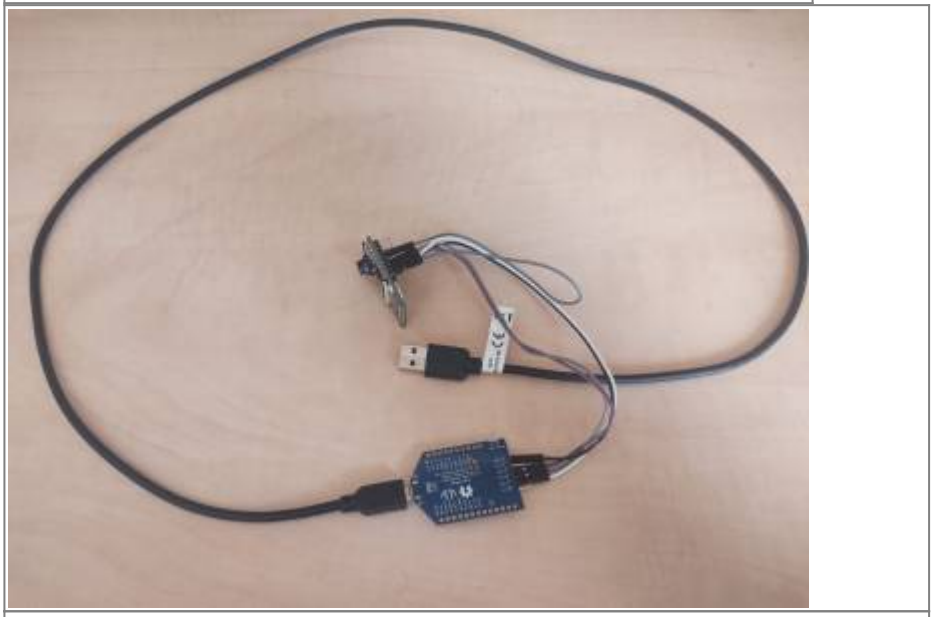


Figure 6. ESP32-CAM and Programmer connection with USB-A Cable

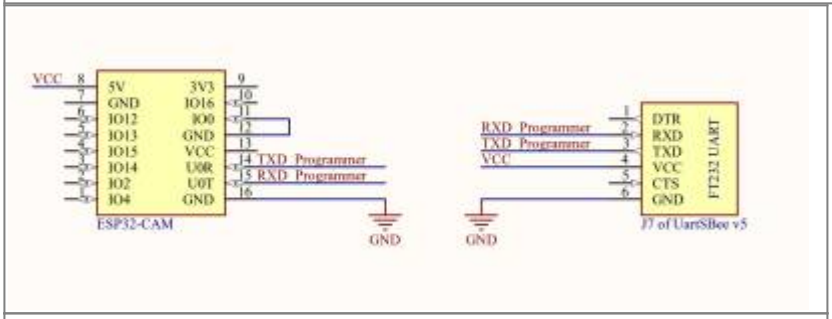


Figure 7. Schematic drawing of connection for programming

b. Software Setup

For this project, the software provided in the AI on the edge device GitHub repository was implemented. The complete documentation provided by the author can be found under the following link. **https://jomjol.github.io/AI-on-the-edge-device-docs/**

Steps to flash the device with the above-mentioned software are given below:

1. Download the program from the following link. https://github.com/jomjol/AI-on-the-edge-device
2. Then, open the device in Bootloader mode.
3. Microcontroller can be flashed by either using the Flash Tool from Espressif (GUI) or by using

the Python-based esptool (Console). The linked documentation above provided detailed instructions on it.

4. In the case of this project, some hurdles were encountered mainly, there was no access to the web page. To solve this problem, the new releases provided by the author from the website,https://github.com/jomjol/AI-on-the-edge-device/releases were manually updated by copying the content to the SD card.
5. In the SD card, edit the file wlan.ini, to let the device connect to the WiFi network.
6. Access the device by entering the IP address of the device in a browser window, only Edge and Chrome browsers are accepted.
7. In case of any error, the log page should be checked.

Now the device can be accessed in the browser window and initial setup should be performed.

c. Initial Setup/Calibration

Now in the case of our project, sample watermeter images with different readings were printed as shown in Figure 8. The full initial setup and calibration process are described in detail in the following link.https://jomjol.github.io/AI-on-the-edge-device-docs/initial-setup/ But, important (not to miss) steps to initiate and calibrate the device are listed below:

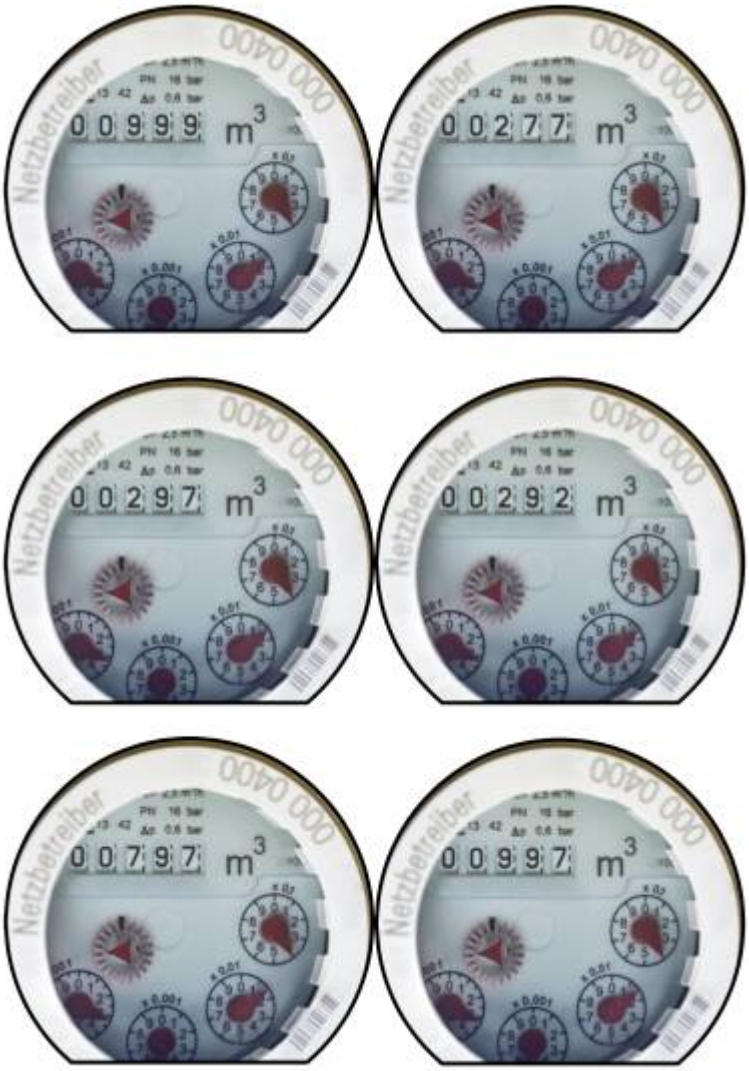


Figure 8. Sample images of watermeter with different readings

1. The first step of the setup is to set a reference image. For that, a new reference image has to be captured. It should also be calibrated in terms of brightness, contrast, and saturation. Moreover, the image should be rotated so that the numerical values are horizontally aligned as seen in Figure 9.

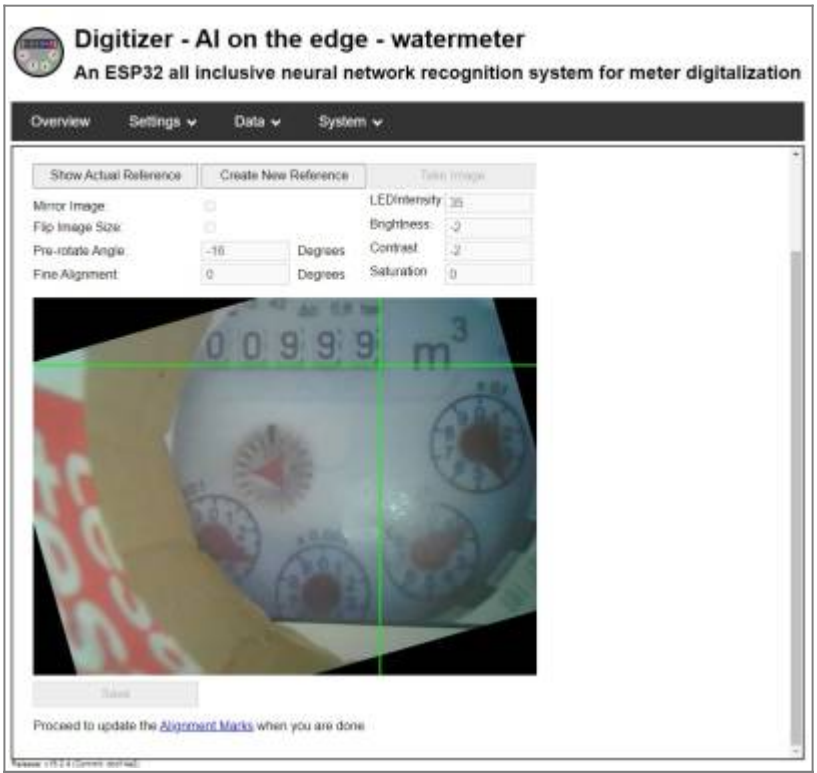


Figure 9. Reference Image setup

2. After that, alignment marks should be set up. It is a reference position for the software to search for the analog and digit regions of interest, which is the unit of measurement in the watermeter's case. These alignment marks are always the same, hence first of all this is recognized by the program, and then the analog and digits in fixed x and y displacements from these marks. The alignment setup of m^3 as individual 'm' and $^{'3'}$ is observed in Figure 10 below.

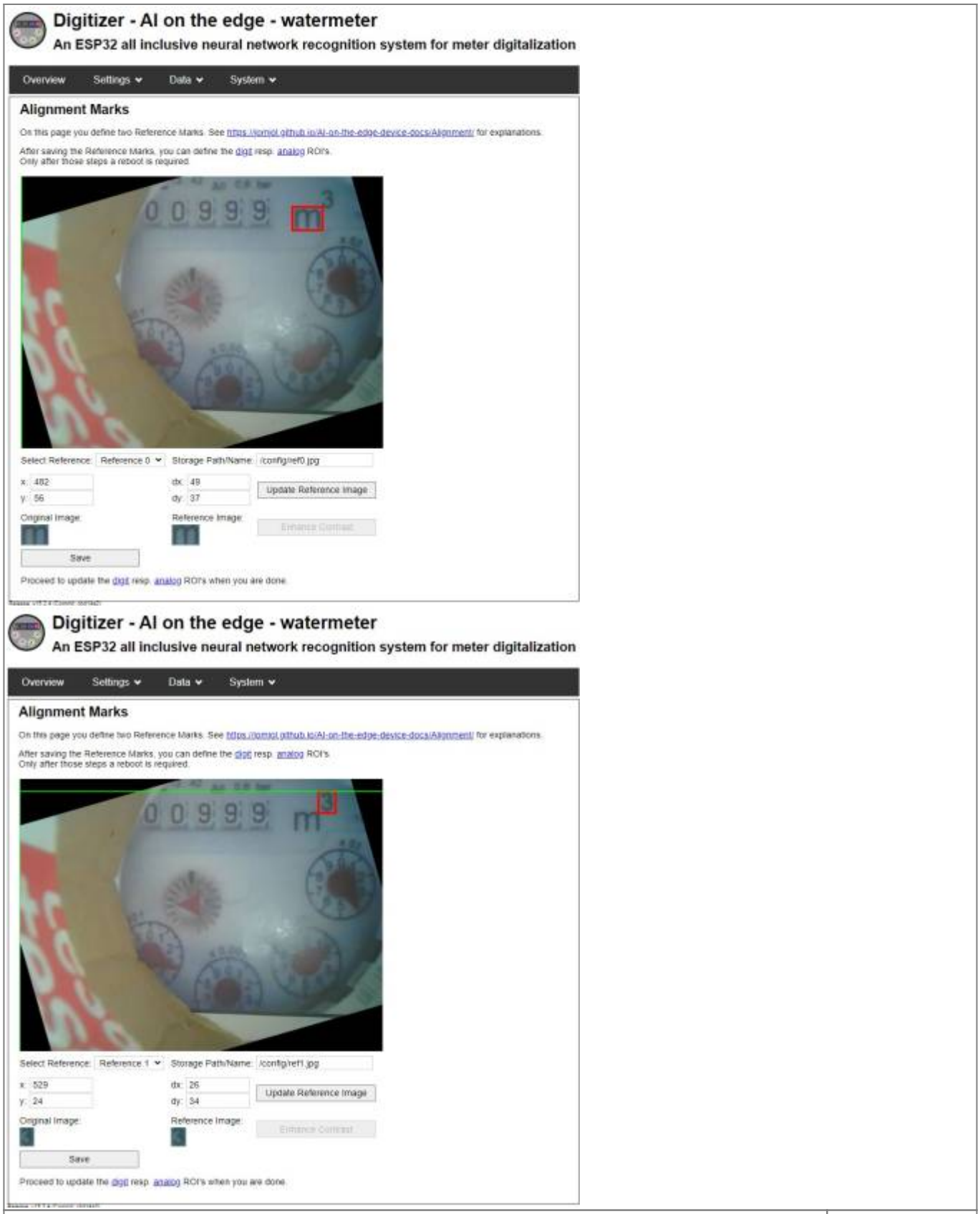


Figure 10. Alignment marks setup for 'm' and '3' as constant reference position

3. Then, the region of interest for the digits (numbers) has to be set up, it is done by selecting x and y coordinates for every numeric digit of interest. It is seen in Figure 11 below.



Figure 11. Digit region of interest setup

4. Likewise, the analog region of interest is marked, as shown in Figure 12. Analog needles give the numbers after the comma for a reading, hence the more needles we introduce the more precise the reading gets. In this project, we are only interested in one digit after decimal. For more precise measurement, other analog meters can also be marked as regions of interest.

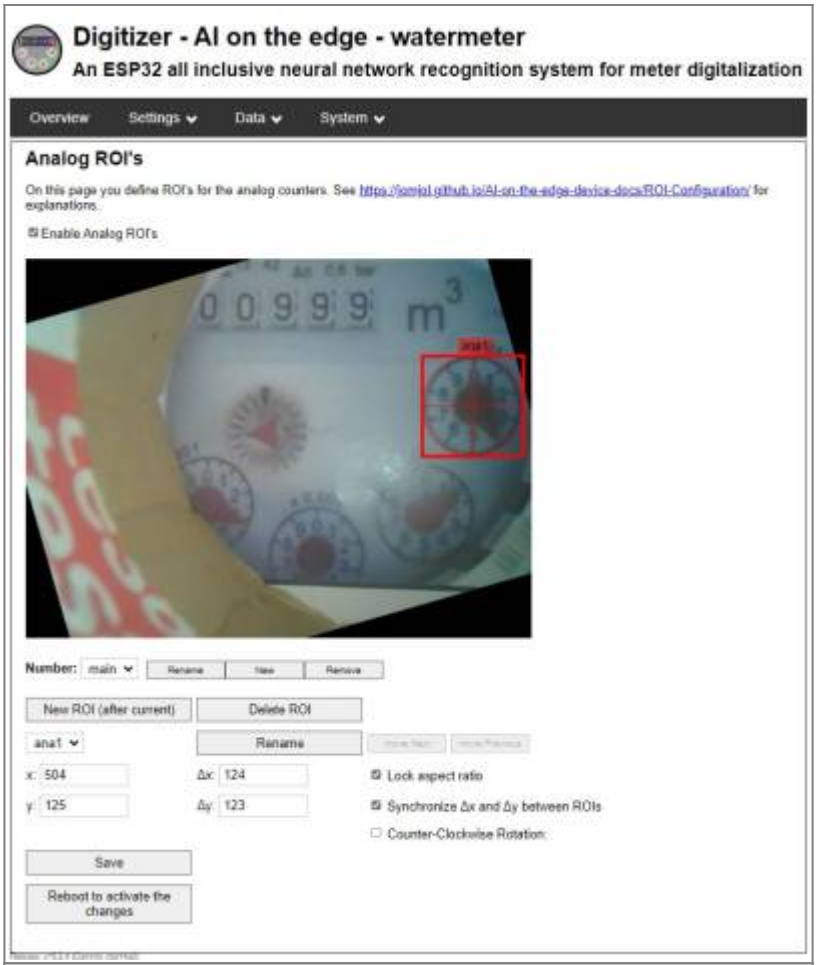


Figure 12. Analog region of interest setup

5. Now the setup is complete, the device is ready to be rebooted with changes activated.

4. Results

After reboot, the device now automatically captures the image and using its optical character recognition technology converts regions of interest into data readings. The device read the different samples provided as follows. Figure 13 shows the experimental setup of the device over the sample image.



Figure 13. Experimental setup of the device over the sample image

Figure 14 shows the different readings when the device was placed over the different sample images. The value on the top right of the image is the actual post-processed value, and above the individual digits and the analog needle are the pre-processing initial read values.

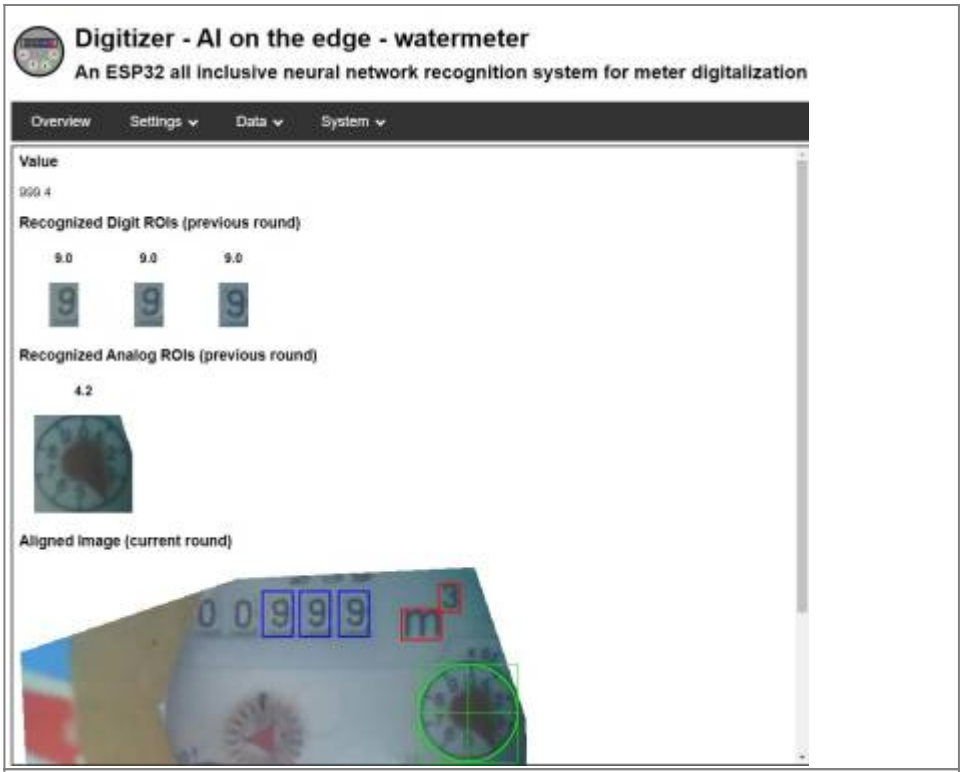


Figure 14. (a) Device reading the reference image as a sample image

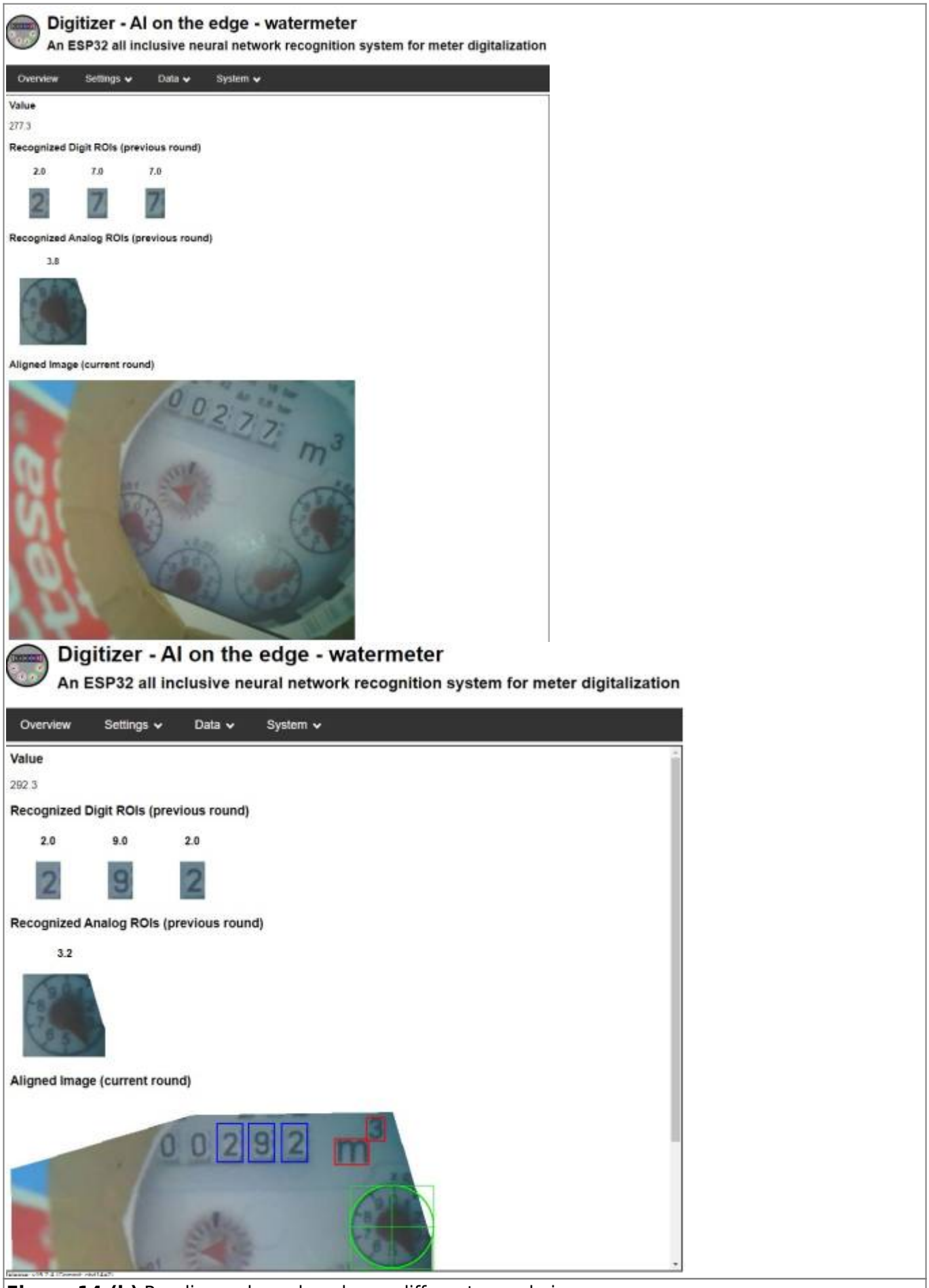


Figure 14.(b) Readings when placed over different sample images

Under the configuration setup of the software setting menu and the device has been instructed to capture the image and process it for a digital value every minute and the recorded data is saved in the SD card for 3 days as seen in Figure 15. These results can be viewed as a whole in the data log. This is shown in Figure 16.

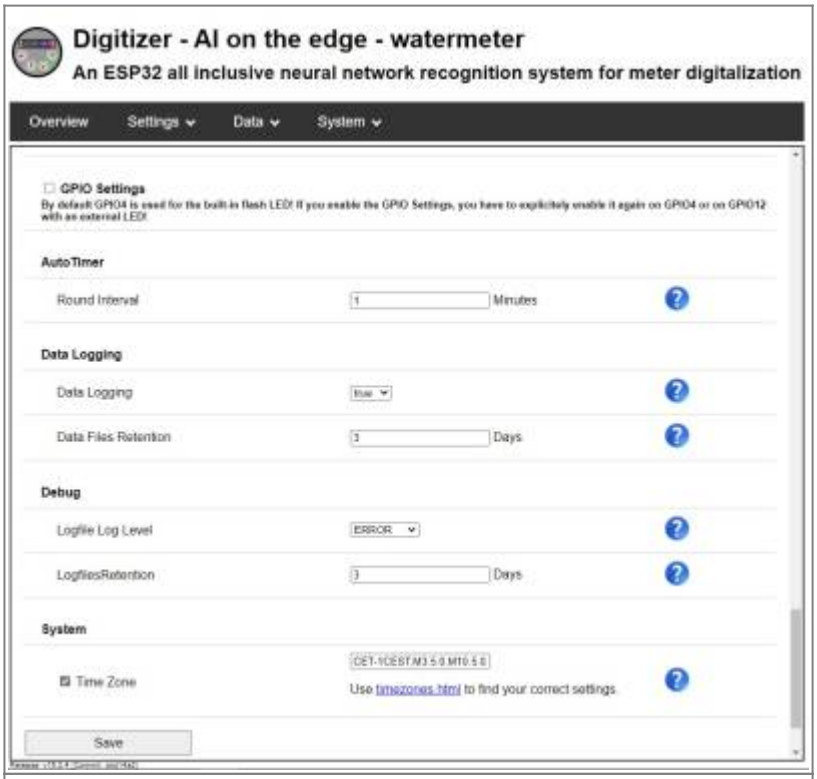


Figure 15. Interval and data retention configuration

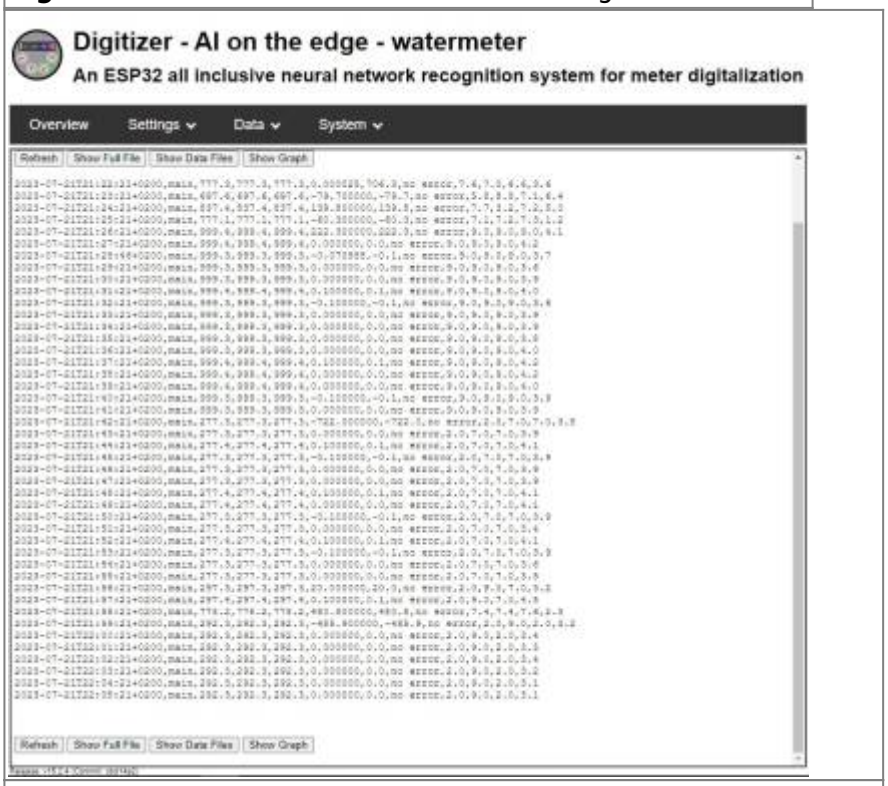


Figure 16. Logs representing one-minute interval data collection

5. Remote data sharing with MQTT

Now that the data is ready, it is only saved locally for 3 days. This limits our scope of communication. This is why, this project was further extended to remote monitoring using the Message Queuing Telemetry Transport (MQTT) protocol. MQTT is a lightweight publish-subscribe machine-to-machine network protocol for message queuing services. As it is an Internet of Things (IoT) protocol, it is suitable for the application of this project. There are 2 roles in this protocol i.e., a client and a broker. In MQTT any conventional server is called an MQTT broker and the clients can connect with it. They can have different roles like a subscriber or publisher. In the scenario of this project, watermeter is a publisher, as it shares its readings with the broker. Now, any client that subscribes to the relevant topics would get the data forwarded from the broker. The MQTT setup in the device can be configured under configuration in the settings menu, this has been shown in Figure 17, where the reading has been forwarded to the broker and credentials for access have been set up. MQTTX is an application used in the demonstration of this project. In Figure 18, an MQTT deployment can be seen where the watermeter and the MQTTX application both connect to the broker. In Figure 19, Remote data reception by the MQTTX application according to the subscription can be seen. The data is the published data by the watermeter under various topics. The client application can subscribe to any of these topics and the complete list of topics of the watermeter is documented on the MQTT API page: https://jomjol.github.io/AI-on-the-edge-device-docs/MQTT-API/

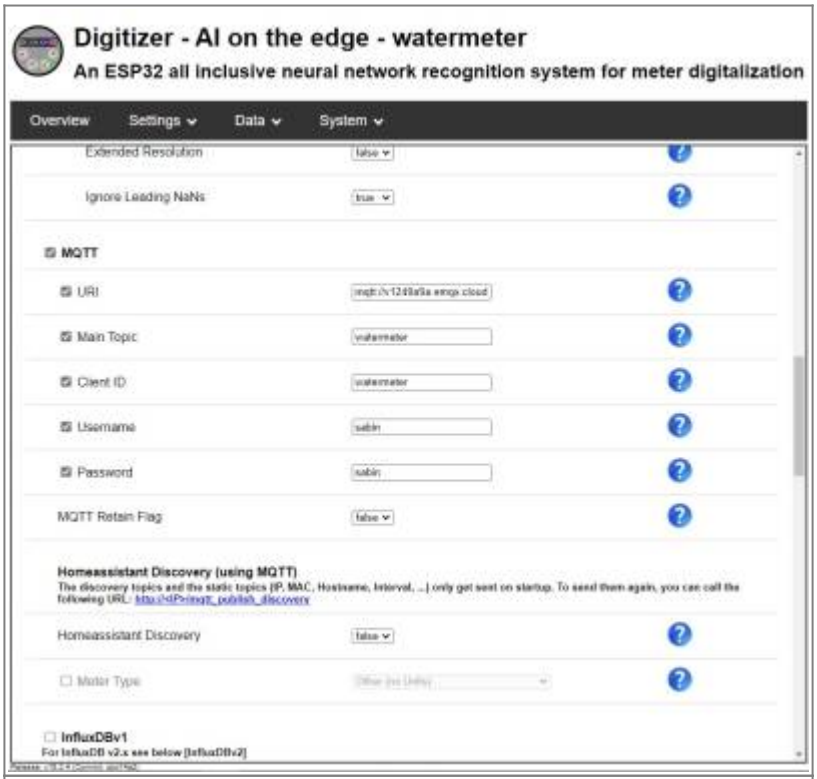


Figure 17. MQTT setup in the device software

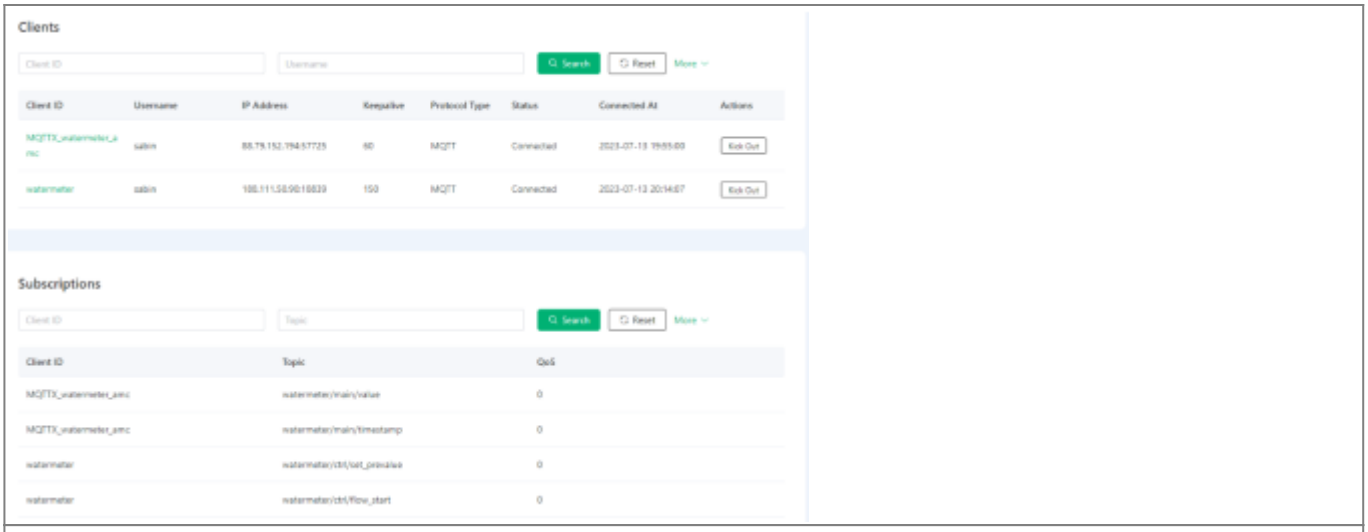


Figure 18. Two client sessions (watermeter and MQTTX application subscriber) inside a single broker

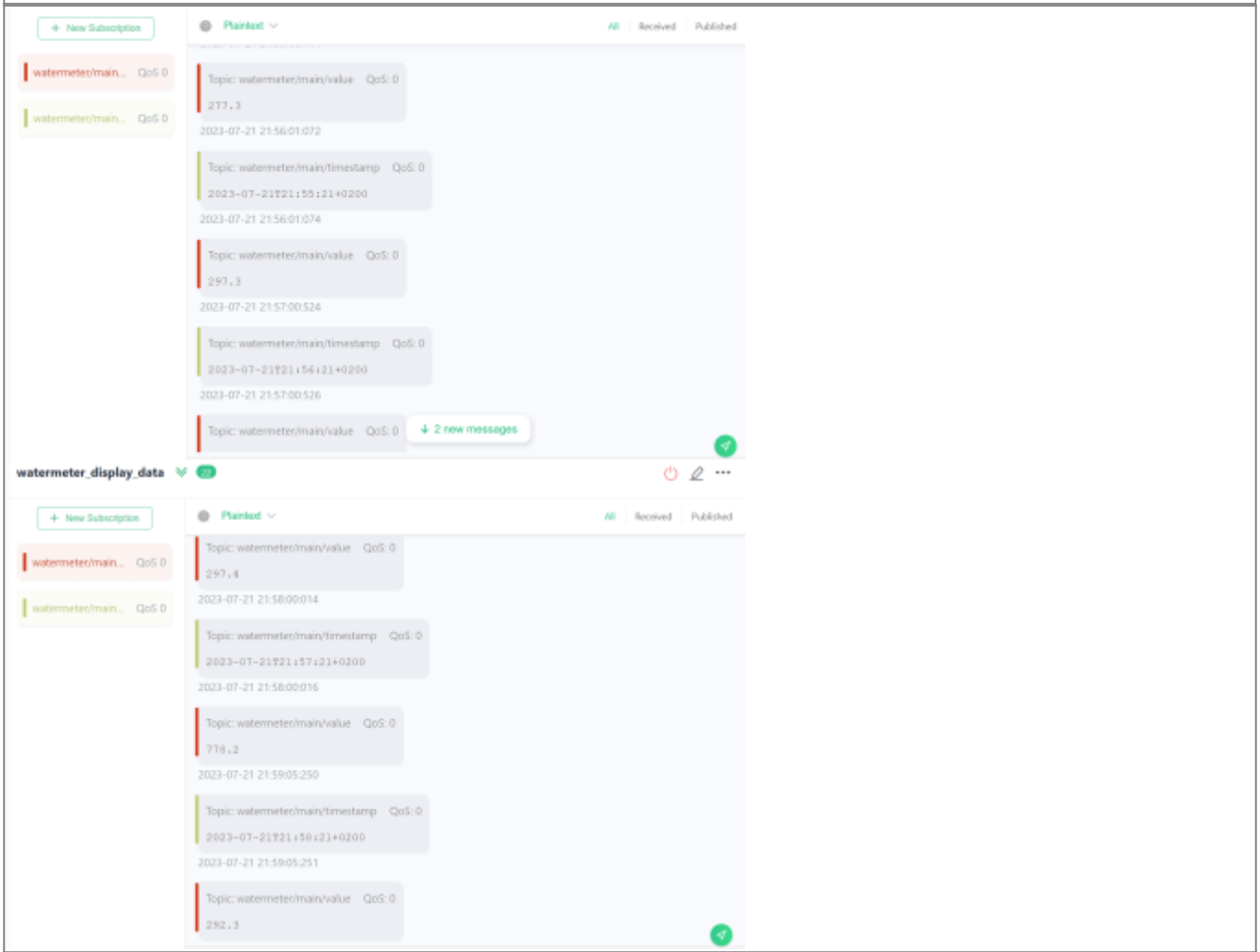


Figure 19. Remote data reception according to the subscription (main value and timestamp)

For this project, only two topics i.e., the reading after post-processing and time stamp were pursued. Some other topics that could be also subscribed hence, remotely monitored are listed with their respective subscription topic and description in Table 1.

Table 1: Some topics and their description

Topic	Description
watermeter/main/error	Informs about the flow status. If there is unreadable value it informs it as error.

watermeter/main/raw	Gives the initial value read before post processing.
watermeter/main/rate	Shows flow per minute.
watermeter/main/changeabsolut	Gives the difference between the previous and actual read value.

6. Best practice suggestions

1. Camera focus should be optimized according to the distance between the device and the object. Focus can be changed by rotating the lens.
2. After setting up the reference image, ideally, the device position not should be changed.
3. Tapes can be placed over the LED of the microcontroller to diffuse the brightness when required, or the brightness value can be changed during the setup of the reference image. Also, reference images should be set up so that, the reflection falls on the empty or irrelevant area of the watermeter.
4. For the accurate reading of the analog needles, the focus has to be adjusted accordingly.
5. The pixelation of the object should be of the best quality possible. In this project, the device could read the digits while the object image was shown on the big LED screen but couldn't do so while using a normal 14 inches laptop screen.
6. WiFi connection for the ESP32-CAM should be set up to a network without strong firewall security like the WiFi in the student dormitory or the EDUROAM of the university. The easiest way to solve this problem would be to create a hotspot from the laptop connecting to any internet with a 2.4 GHz network bandwidth.

7. Discussions and Conclusion

This project demonstrated the use of artificial intelligence in everyday devices and introduced a watermeter device that utilizes AI to measure and monitor water flow. The device captures images of the gauge and reads the analog numbers using AI, converting them into digital format for easy access and analysis via a standard protocol like MQTT. The subscribers can not just access the data as the device reads them in real-time, but also use it for further processing. This makes it possible for a large-scale implementation like the water companies that get the meter reading automatically from all the customers and generate the bills. Another implementation can be the use of other AI models to detect patterns and detect leaks as well as provide predictions and conversation strategies. The device can therefore be used to promote efficient water usage in response to global demands.

The main issue is still the reliability and accuracy because it cannot be always made sure that the device reads the correct values from the meter. In the above examples, we could see that the digits could be properly recognized but there was always a discrepancy with the analog needle values. Other than that, there is also the question of reliability in terms of maintaining the service, as the AI software doesn't always successfully detect the meter reading. In the course of this project, the image provided was high in contrast, and the possibility of reflection is minimal due to printed samples. But in a real-world scenario, the meter might not always be read or read correctly, due to reflections, brightness level, moisture, or even low contrast in the meter numbers, even if the reference image is calibrated to the best possible level.

These issues can be tackled by using fully digital flow meters that can share their reading via numerous digital data transfer methods. Infrared reading is common with radiators and electric meters, where the values can be accurately transmitted. But it cannot be done in real-time. Another thing that could be further looked into in the future would be to introduce a battery-based current

source and further introduce deep sleep for sustainable energy consumption.

All in all, if the user is looking for a cheap solution that works with existing infrastructure and is able to maintain the accuracy and reliability of the readings, then this method will definitely be suitable.

8. References

1. https://docs.ai-thinker.com/en/esp32-cam
2. https://www.nutsvolts.com/magazine/article/build-a-video-camera-using-the-esp32-cam-board
3. https://wiki.seeedstudio.com/UartSBee_v5/
4. https://wasser.badenovanetze.de/wasser-allgemein/wasserzaehler/
5. https://jomjol.github.io/AI-on-the-edge-device-docs
6. https://github.com/jomjol/AI-on-the-edge-device
7. https://github.com/jomjol/AI-on-the-edge-device/releases
8. https://jomjol.github.io/AI-on-the-edge-device-docs/initial-setup/
9. https://jomjol.github.io/AI-on-the-edge-device-docs/MQTT-API/
10. https://mqttx.app/?utm_source=mqttx&utm_medium=referral&utm_campaign=logo-to-homepage
11. https://mqttx.app/downloads
12. https://www.youtube.com/watch?v=d_u8c3bu-zg

From:
https://student-wiki.eolab.de/ - **HSRW EOLab Students Wiki**

Permanent link:
https://student-wiki.eolab.de/doku.php?id=amc:ss2023:group-d:start&rev=1690308715

Last update: **2023/07/25 20:11**