Smart water meter with AI on the edge

by Ainul Danish Zulhusni Bin Azrin (28977), Shreevarshan Sivakumar (31230), Yacine Aider (30913)

1. Introduction:

Integrating artificial intelligence (AI) with edge devices has changed the game in the quickly developing IoT (Internet of Things) industry. Smart water meter reading, which enables effective monitoring and management of water consumption in homes and businesses, is one of the promising applications of AI on the Edge. In this context, we'll look at how to use the ESP32-CAM, a powerful and reasonably priced microcontroller with a camera module, to enable real-time water meter reading with artificial intelligence capabilities and seamlessly integrate it with Home Assistant, a well-liked home automation platform, using the MQTT (Message Queuing Telemetry Transport) protocol. The ESP32-CAM is a flexible microcontroller with built-in Wi-Fi, Bluetooth, and a camera module. It is the perfect device to use in a variety of IoT projects due to its small size and low power consumption. We can do on-device AI operations like image identification and data processing directly at the edge by utilizing the processing capacity of ESP32-CAM, eliminating the need on cloud services and assuring faster and more dependable real-time data analysis.

Manually reading water meters the old-fashioned way is laborious and time-consuming. The ESP32-CAM can take pictures of the water meter dial using AI on the Edge technology, and by employing AI algorithms, it can analyze the readings intelligently and autonomously. Home Assistant is a well-liked open-source platform for home automation that unifies the management of smart devices and services. The ESP32-CAM can easily connect with Home Assistant by utilizing MQTT, a compact messaging standard. The MQTT broker hosted on Home Assistant can receive the meter reading data that has been processed on the edge by the AI model.

The ESP32-CAM and Home Assistant communicate through the MQTT broker. It makes message exchange more effective and guarantees dependable data transport between devices. As an MQTT broker, Home Assistant can receive the water meter readings sent by the ESP32-CAM and show the data on its user interface in real time. Customers may easily keep an eye on how much water they use and create automated rules based on the results.

2.Materials:

- ### ESP 32-CAM module

The ESP32-CAM module (Fig 1) is a small development board that supports cameras and Wi-Fi and is based on the ESP32 microcontroller. It has GPIO pins, a camera interface, and a USB-to-TTL converter for programming. It is frequently utilized for IoT initiatives, security cameras, and AI edge applications.

Fig1: ESP-32 module. Source: own image - Shreevarshan

- **ESP32-CAM AI-Thinker MB Programmer**

A shield called the ESP32-CAM AI-Thinker MB programmer (Fig 2) can be connected to the GPIOs on the ESP32-CAM board. The CH340C USB to serial chip is included with the programmer. As a result, the use of the shield's USB interface to program the ESP32-CAM is possible. The shield also includes BOOT (IO0) and RESET (IO0) buttons. This was helpful for quickly flashing the ESP32-CAM or resetting it.
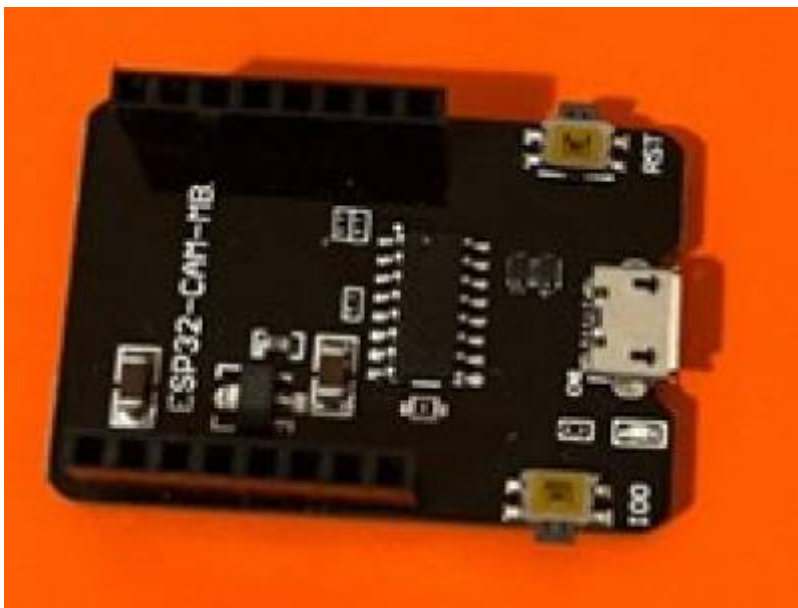

Fig 2: AI Thinker MB Programmer. Source: own image - Shreevarshan

- **16 Gb SD card and SD card reader:**

For purposes such as Storage of AI Models and Data, Resource Constraints, Flexibility and Upgradability, Offline AI Processing, Data Logging and Retrieval an SD card was used. A reader was used to read the and write to the SD card as the laptops used were not having an SD card slot.

- **Micro USB to USB A cable:**

The cable (Fig 3) was necessary to connect to the programmer to the USB slots in the laptop to perform actions needed for examples, flashing the firmware and accessing AI on the edge.

Fig 3: Micro USB to USB-A. Source: own image - Shreevarshan

- **3D housing:**

A design inspired from the web was taken and modified to fit the needs of the water meter sample provided. The output of the printed file is seen in Fig 4.
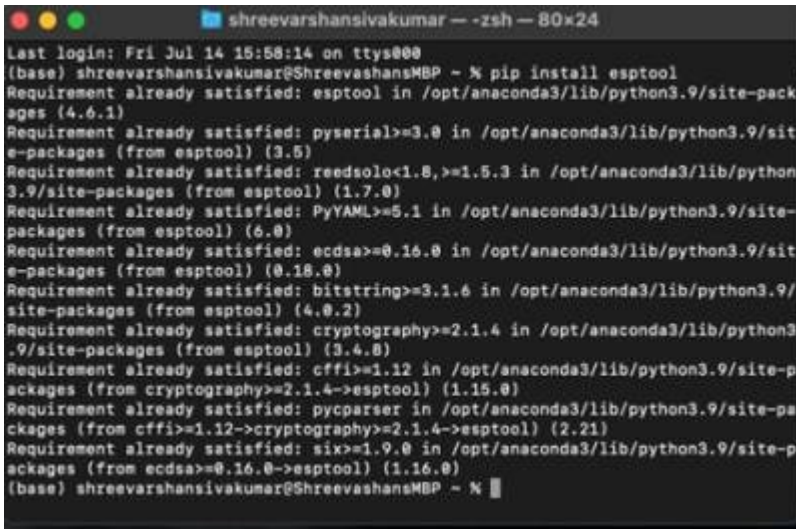

Fig 4: 3D Housing to support the cam module. Source: own image - Shreevarshan

2.1Methods:

1. Downloading the AI on the edge source code zip from github:

This was the necessary file that was used to work on. This was unzipped into a location as per needs. Ensured was that the SD card folder in the version downloaded was not empty and downloaded was also partitions.bin, bootloader.bin and firmware.bin from the github separately. Link to github can be found here: https://github.com/jomjol/AI-on-the-edge-device

2. Flashing using the python based esptool:
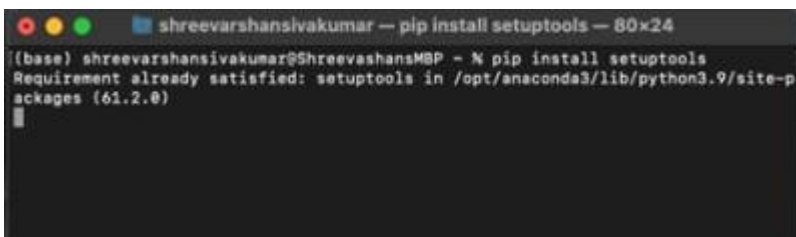
• A python environment is needed, used for this was anaconda prompt. Installed was esptool (Fig 5). Code snippet: pip install esptool

Fig 5: Installing esptool. Source: own image - Shreevarshan

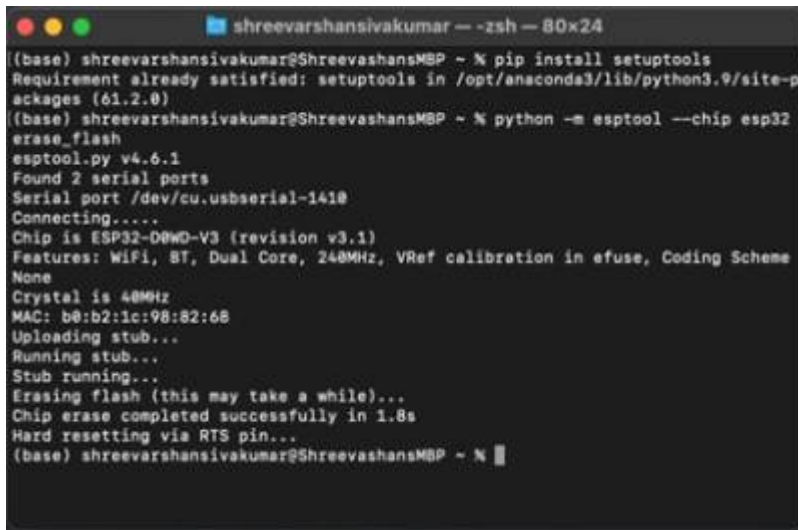• Setup tools was installed using code: pip install setup tools (Fig 5).

Fig 5: Installing setup tools. Source: own image - Shreevarshan

• The ESP 32 module was connected to the MB programmer to put it in boot mode and then the following commands were used to erase the flash (Fig 6) and write the flash bootloader, partitions and firmware (Fig 7). For writing these files, it is necessary to make sure to be on the correct directory to call the function, where these files reside. (Fig 7).

The code snippets for:

Erasing: python -m esptool –chip esp32 erase_flash (uses python to use esptool to erase flash on esp 32 module). Flashing: python -m esptool –chip esp32 write_flash 0x01000 bootloader.bin 0x08000 partitions.bin 0x10000 firmware.bin

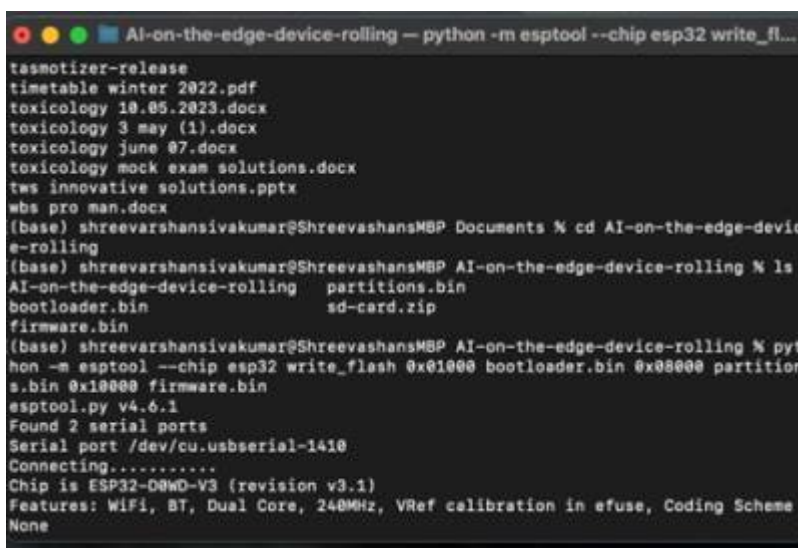Fig 6: Erasing flash. Source: own image - Shreevarshan

Fig 7: Writing flash after choosing the correct directory. Source: own image - Shreevarshan

3. Fomatting the SD card and copying the SD card files:

• The SD card was supposed to be formatted in the FAT 32 type. Then the files from the SD folder on the AI on the edge was copied into the SD card. The was opened using a text editor and the SSID and password of the network was given (Fig 8) in this case, used was a hotspot from the PC which was worked on to get easier connections and to get the IP address of the watermeter i.e., ESP module for accessing AI on the edge.
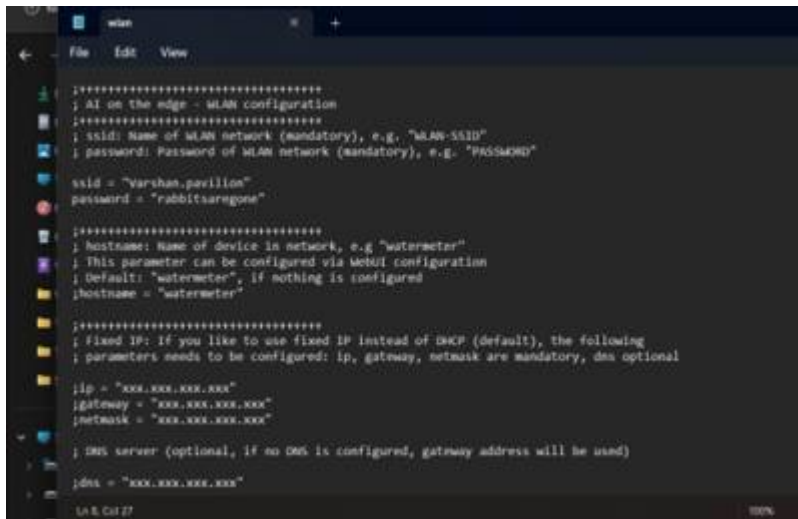
Fig 8: Edited SSID and password on "wlan" file. Source: own image - Shreeevarshan

4. Getting the IP of the "watermeter"- ESP module:

The SD card is replaced into the ESP module and the Micro USB cable is connected to the laptop and the ESP module. It connected to the credentials provided, when it connects to the hotspot, the LED on the ESP lights up. Under the settings of Windows, the IP of the connected devices are available under the personal hotspot and from there the IP can be taken to open AI on the edge (Fig 9).
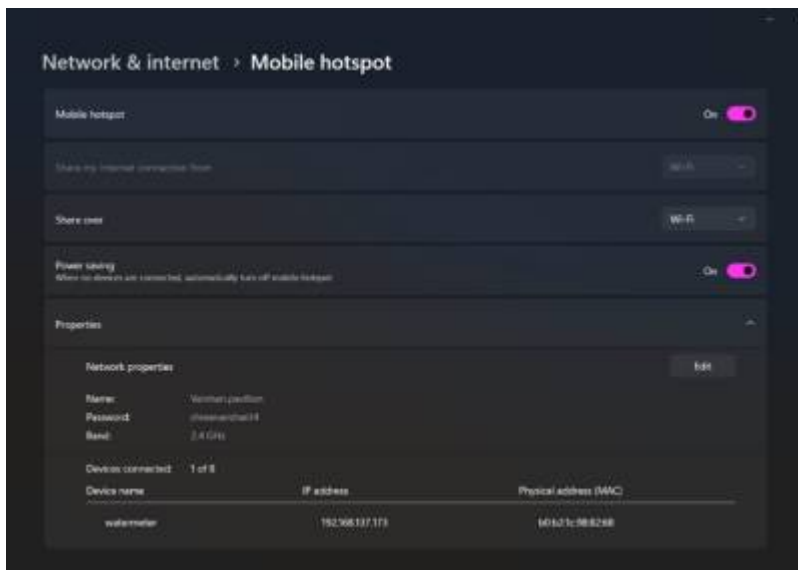
Fig 9: Getting the IP of the "watermeter". Source: own image - Shreevarshan

5. AI on the edge and New reference creation:

• The obtained IP of the "watermeter" is pasted on the search bar of the browser and the AI on the edge is accessed. When the page is accessed, It already had a pre available reference image of the water meter reading. Next step was to create an own reference image by clicking "create new reference". This accesses the camera on the ESP to take a picture of the reference that is provided (Fig 10). The ESP setup as above was fixed to the 3D mount which was printed to take pictures of the reference readings (Fig 11).

Fig 10: Water meter reference image. Source:https://hidroconta.com/en/water-meter-reading/

Fig 11: Creating a new reference image. Source: own image - Shreevarshan

6. Creating Alignment marks, ROI and analog references:

• The next step was to create the alignment marks, in this case used was" m3 " from the image provided as reference(Fig 12) and click on the update reference image and then save, so this process is updated (Fig 13).
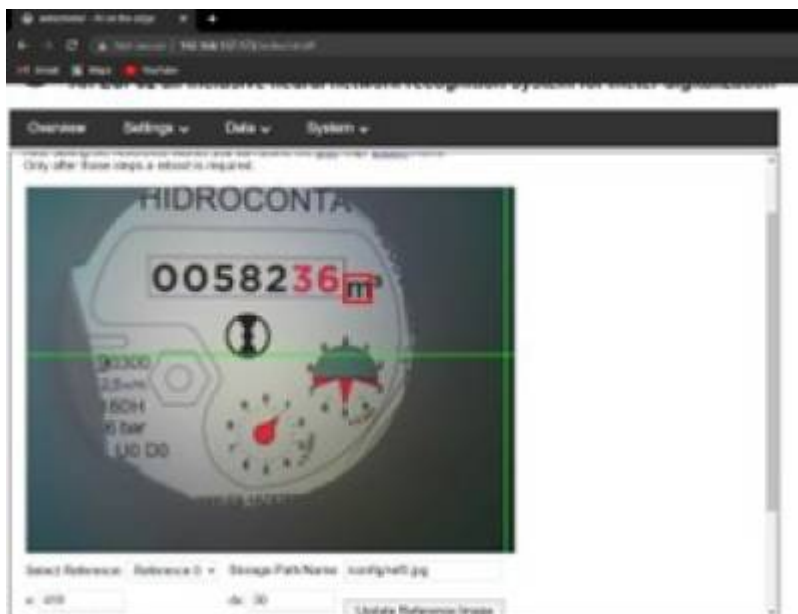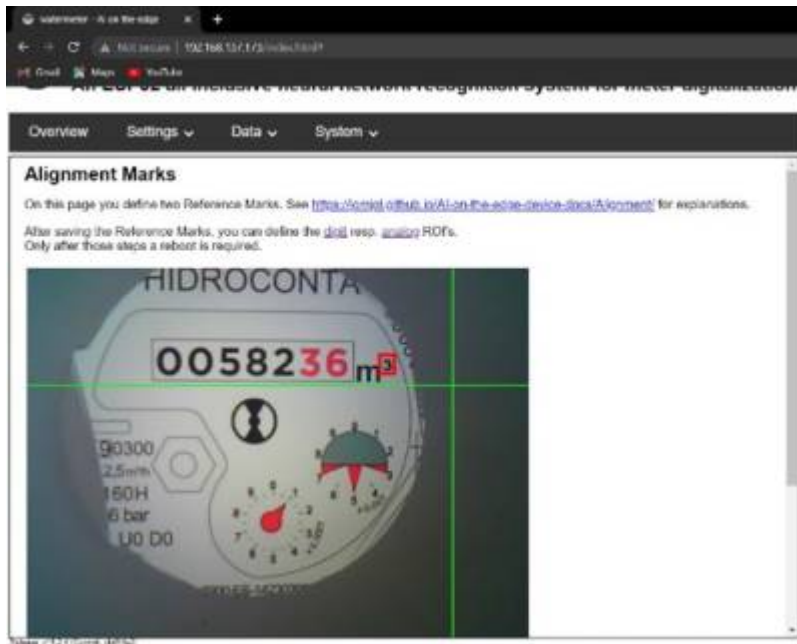
Fig 12: Creating Alignment marks. Source: own image- Shreevarshan
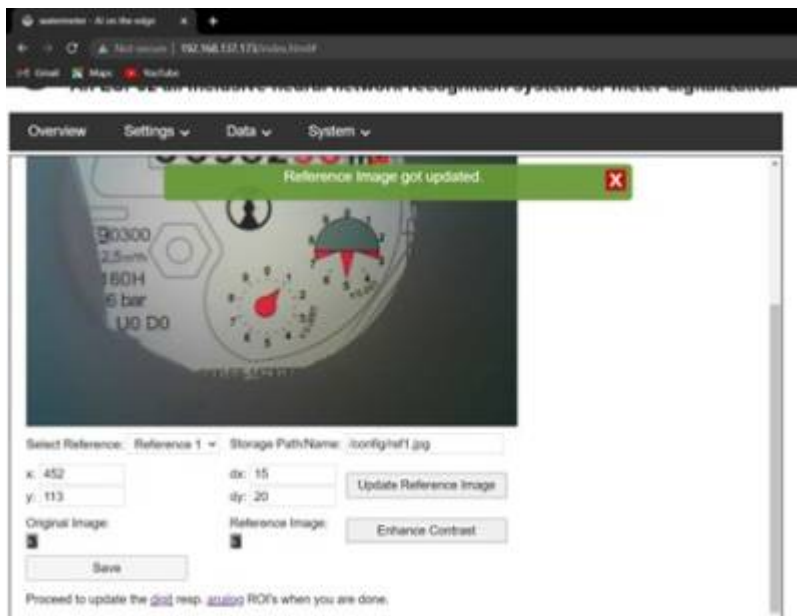
Fig 13: Updating the reference. Source: own image - Shreevarshan

• **Proceeding to analyse digits and Analog:**

There were three digits that could be added as references, further more Number references could be added by clicking "New" and "new ROI after current". It was not working to provide correct results when decimal places were to be analyses by adding the rew ROI(Fig 14). Further was to analyse the analog by creating a reference. This was also updated and saved (Fig 15). When these steps were completed, proceeded was to reebot the device to activate the changes and the page reebots in 60 seconds.

Fig 14: Referencing digits. Source: own image- Shreevarshan
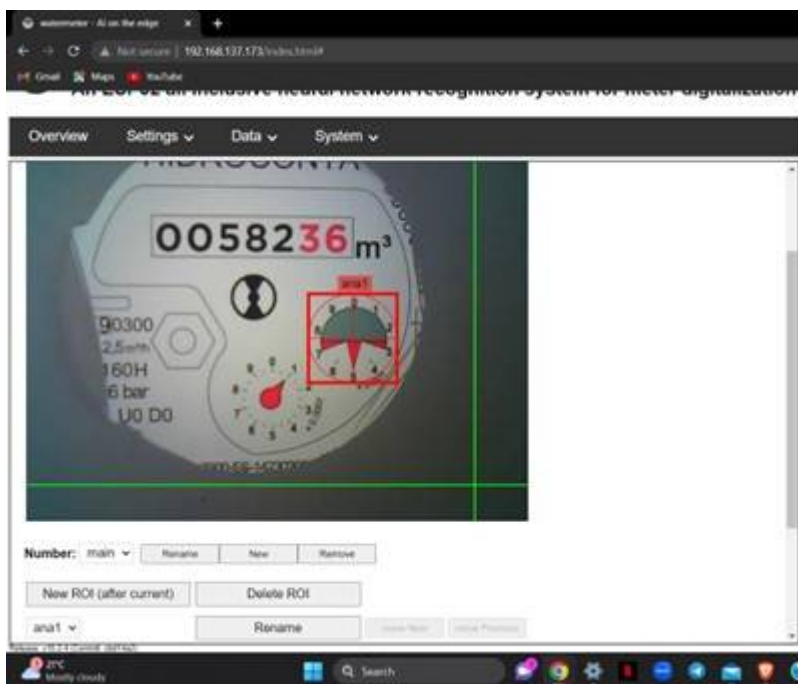
Fig 15: Referencing the analogs. Source: own image - Shreevarshan

7. Analysing new image readings with time frames:

The device was completely configured and set with the reference image provided. This setup was later used to read the image to check if it gives correct readings. To do this, "Data" dropdown menu was 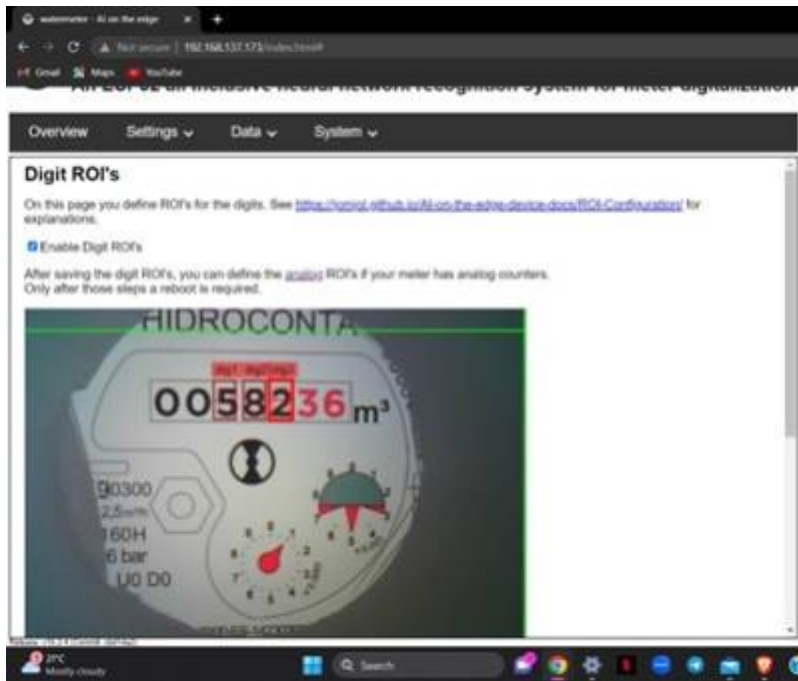chosen and then recognition was clicked to take a new picture to get the readings. It first initialises the device, then takes the image, then aligns and then the data it reads can be available in the overview (Fig 16). This Process of image processing was configured to every 3 minutes under the "Settings" dropdown and then clicking on "configuration". When scrolled to the last, the time interval can be set to take pictures. Also here the MQTT and Home Assistant could be set up to send this data to the dashboard (Fig 17). In this case, the page was very unresponsive to do the configurations of the MQTT and Home Assistant as the work was done on different laptops and the ESP module did not

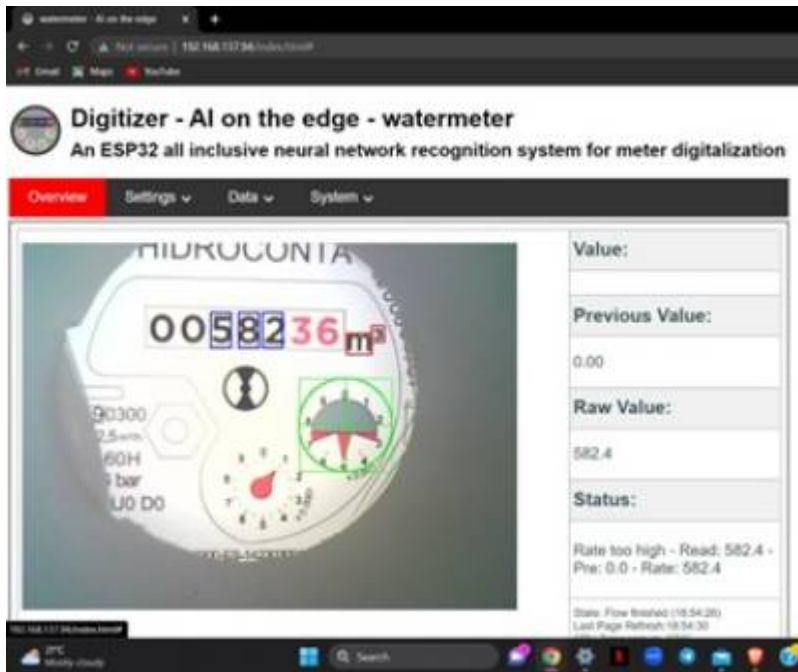work on the other laptop which had the Home Assistant installed.

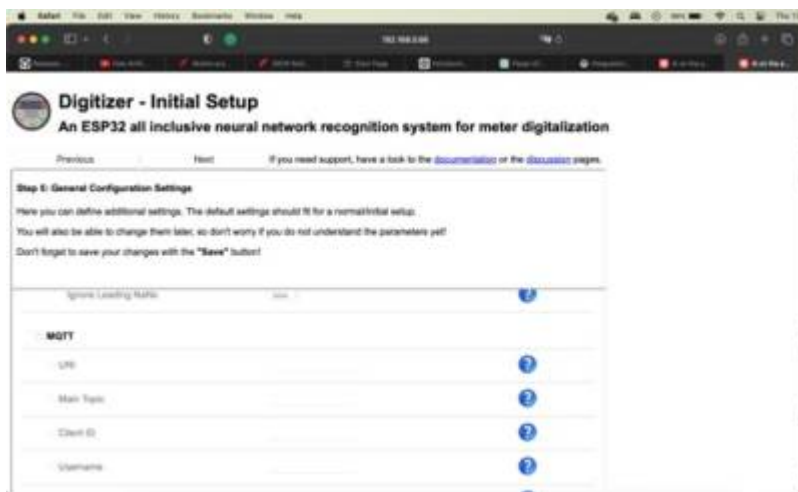Fig 16: Final reading of the values in the water meter reference. Source: own image - Shreevarshan

Fig 17: Configuring the MQTT and Home Assistant. Source: own image - Shreevarshan

8. Working with Home Assistant

This project focuses to utilize the many benefits of Home Assistant while also using MQTT to relay data from AI On the Edge water meter. • Since Home Assistant wasn't already available, first step was to download Home Assistant into the system. Below are the steps needed to run Home Assistant and MQTT.

Among the various methods of installation of Home assistant, Oracle VM was used(Virtual Machine) to operate and get Home Assistant in the system. Oracle VM (Virtual Box) allows the user to enable cross-platform virtualization software. This means that the user can extend their existing computer to run multiple operating systems than the already built-in software, with this, we were able to use Linux, to download, and operate Home Assistant to the PC. Firstly, Oracle VM was downloaded (Virtual Box) and the settings were configured. Then, starting the VM. The IP Address for Home Assistant becomes available (Fig 18, Fig 19). Later, an account was created and the setup was started.

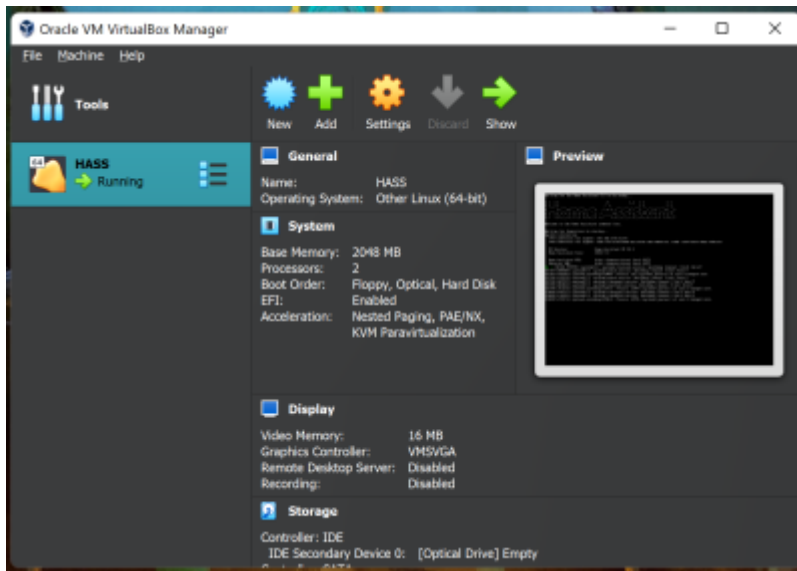Fig 18: Configuration settings of Oracle VM, source: own image – Ainul Danish
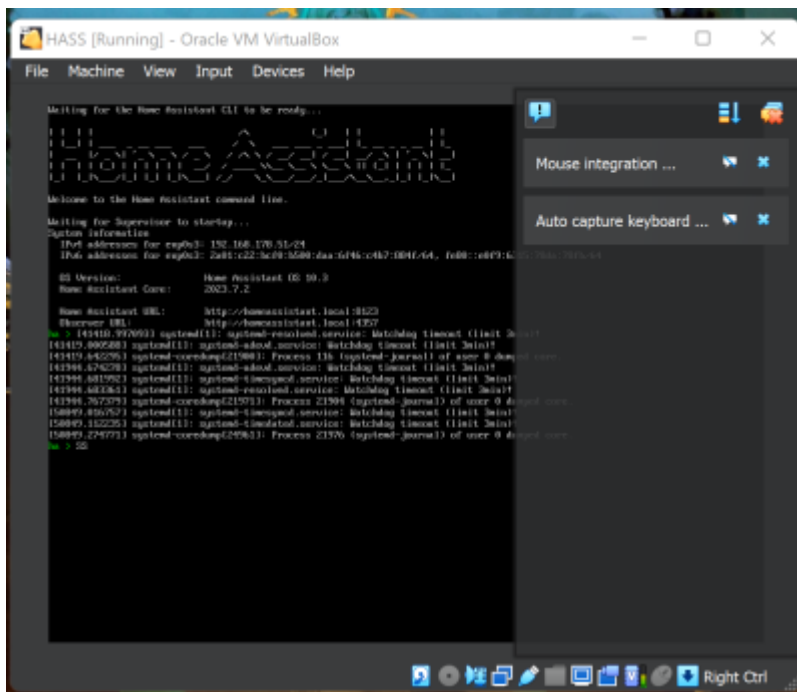

Figure 19: IP Address for Home Assistant, source: own image – Ainul Danish

Home Assistant can be configured to view all relative home devices in any home. In Fig 20 are all the relative devices that are connected and available at home. More integration can be added to the Home Assistant for more advance smart home services.
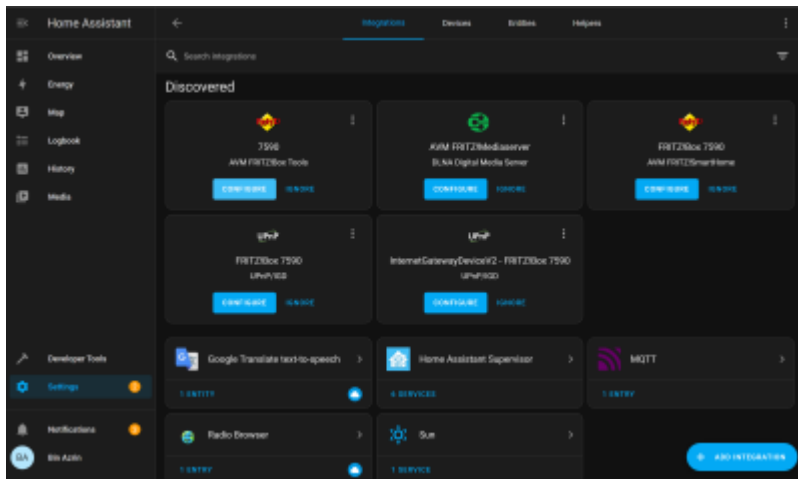
Fig 20: Devices available in-Home Assistant, source: own image – Ainul Danish

Next, with Home Assistant installed and running, MQTT is next to be set up and configured.

- **MQTT:** When Home Assistant installed and running, proceeded with was MQTT. It acts as the medium to send data from the publisher (ESP32 cam water meter) to the subscriber (Home Assistant). This enables MQTT to send data reading of the water meter. This can be achieved by a central message broker, acting as the server that manages the communication between devices. To acquire the broker, the Mosquito broker was downloaded in "Add-Ons" (Fig 21).
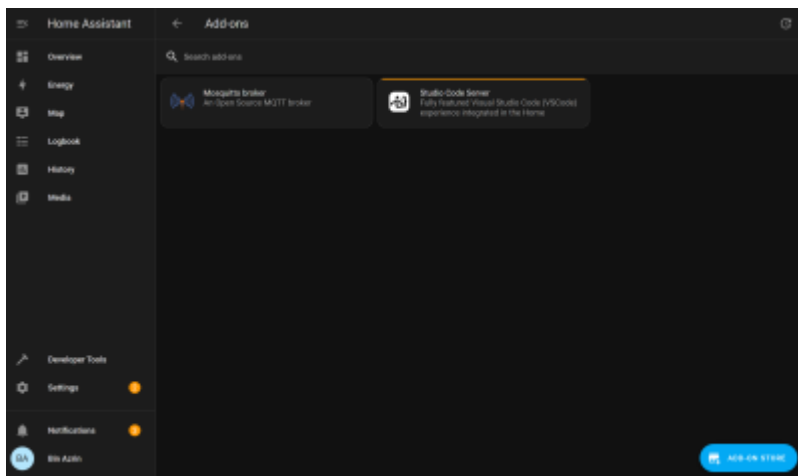
Figure 21: Downloading Mosquito broker in Add-Ons, source: own image – Ainul Danish

- **Testing the broker:** Any sample topic was added in the publish and listening field (ensure that the topic is the same in both the fields). Proceeded with clicking on Start Listening. In the payload field, any message was typed in and clicked on Publish. The same message received and appearing at the bottom (Fig 22) confirms the working of the broker.
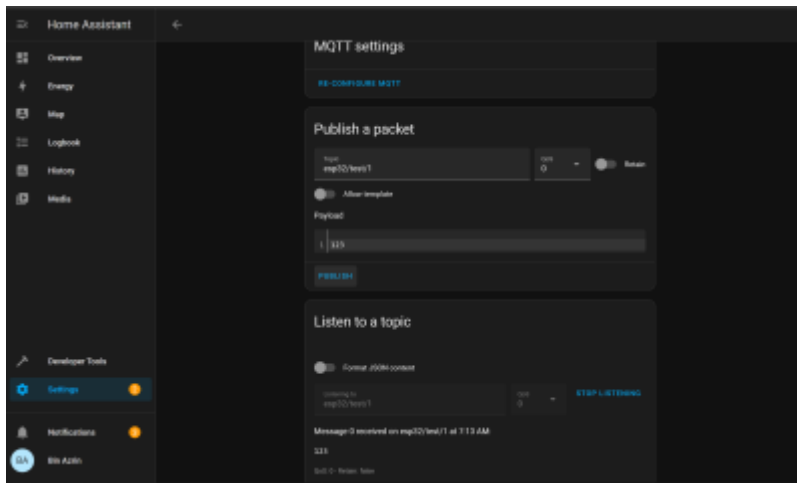
Figure 22: Testing whether the MQTT broker is functioning, source: own image – Ainul Danish

- **Uploading codes:** After MQTT being set, uploaded was the relative codes into the Home Assistant for ESP32 Cam to read and send the data. Stated here are the Home Assistant configurations and finding the "configuration.yaml" file. One of the best ways is to download the "Studio Code Server" add-on from Home Assistant, as it helps to navigate around Home Assistant such as searching for files much easier. This leads to the configuration files (Fig 23). Unfortunately, development with Home Assistant was put to a halt here, as the ESP32 CAM wasn't able to connect to the laptop with Home Assistant installed.
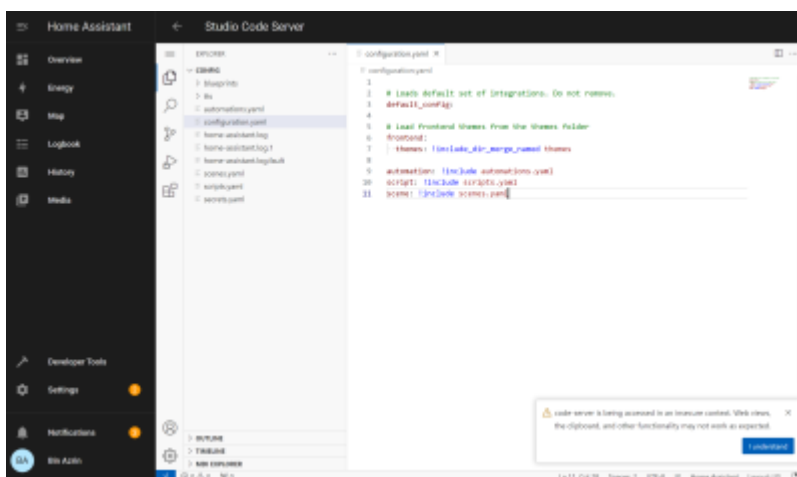

Figure 23: "Configuration.yaml" with the help of Studio Code Server to upload the code. Source: own image – Ainul Danish

Results and Discussion:

The results were successfully obtained and the data of readings is also saved to use these later for post processing. During every time interval set, the device takes the image and does the recognition and saves the data and displays it as well on the overview. As previously mentioned in Step 7, because the AI on the edge and Home Assistant was done on separate laptops, further coding for Home Assistant had to be halt only until Configuration.yaml part. Unfortunately, the ESP32 CAM was unable to connect to the laptop that had Home Assistant installed and set up. Therefore, no further progress could be made. However, we firmly believe that the project would still be successful if everything wasn't done separately, as we had no issue with setting up Home Assistant.

Issues Detected:

An issue that frequently occurred was an error called "Brownout Trigger detected". This error essentially means that, there is a temporary drop in voltage in the power supply, which is usually below the normal operating voltage range of the system. So, the programmer/machine is limiting and inhibiting itself from uploading/running the code, as it would potentially cause issues for the ESP32Cam. Therefore, the cam was unable to work. While this error kept happening with using the cam with the UARTsBee programmer. The possible explanations as to why this error occur were: 1) the USB cable is of poor quality, and possibly too long. 2) The computer's USB port cannot supply enough power to the board. 3) The ESP32 Cam itself is defective. 4) Other components in the circuit are not correctly wired, affecting the power supply. One of the solutions tried was to connect ESP32-CAM power supply from 3.3V to 5V from the UARTsBee, which still had no impact. The AI on the edge was not always very responsive while working, often crashed. If it worked on Windows it didn't work on a Mac. The ESP also didn't connect often to the WLAN at home, that was an additional reason to use the personal hotspot. In addition to that, the addition of new ROI fields in the digits and analog references did not provide correct results when rebooted and the page was unresponsive during recognition when any new ROI was introduced. Further issues regarding Home Assistant were discussed in Step 7 of methods, above.

References:

https://github.com/jomjol/AI-on-the-edge-device

https://jomjol.github.io/AI-on-the-edge-device-docs/

https://www.hackster.io/harshkc2000/esp32-and-toit-integrating-home-assistant-through-mqtt-d090a2

https://hidroconta.com/en/water-meter-reading/

https://www.youtube.com/watch?v=Uap_6bwtILQ&pp=ygUgaG9tZSBhc3Npc3RhbnQgc21hcnQgd2F0ZXIgbWV0ZXI%3D