Smart gas metering with computer vision

by Luca Alexander Fröhling (30588), Til Scholz (?) and Alice Peter (?)

1. Introduction

1.2 Why should we use smart meters?

2 Materials and Methods

2.1 Materials

Within the scope of this project, we used a commercially available gas meter from Pipersberg, a 3D-printed housing for the camera and a computer as well as a number of hardware parts, which are explained in detail in Section 2.2.

2.1.1 Housing

For the project it was necessary to design a housing for the camera, which should fulfill various functions. The most important two functions were on the one hand a stable and precise alignment of the camera and on the other hand the shadowing of the counter from other, external light sources to prevent possible reflections on the images of the camera. 

2.2 Hardware

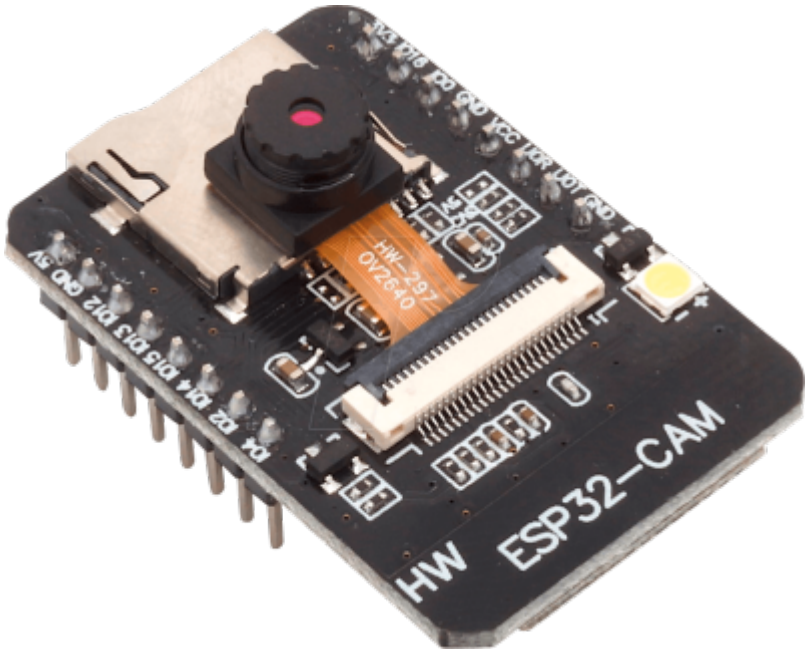
For the installation of the AI-on-the-edge program within the scope of this project a number of

hardware parts were required. The most important components are the ESP32-CAM, the UartSBee v5' programmer and an SD card. In addition, a power connection and jumper cables are needed to connect the individual components. }

2.2.1 ESP32-Cam

The ESP32- Cam is a very versatile and helpful developer board for creating, processing and transmitting image files. It combines an ESP-32 microcontroller, which is a low-cost and low-power 32-bit controller, with a small camera module. With its compact design of less than 4.05×2.7×0.45cm and its low supply voltage of 5V, it is versatile in the field of IoT projects that require real-time transmission and processing of images. Examples of applications include smart metering, surveillance systems and general projects in remote monitoring and remote data analysis.

For more technical data follow this link:
https://loboris.eu/ESP32/ESP32-CAM%20Product%20Specification.pdf#:~:text=Module%20Model%20ESP32-CAM%20Package%20DIP-16%20Size%2027%2A40.5%2A4.5%02Ä±0.2%02Åmm%20SPI,Storage%20Environment%20-40%20%01%07~%2090%20%01%07%2C%20%3C%2090%25RH



2.2.2 Uartsbee programmer

The UartSBee v5 is a communication module developed by Seeed Studio. It is designed to facilitate serial communication between a computer and other devices or microcontrollers. The module is FTDI cable compatible USB to Serial adapter and contains an BEE socket(20pin 2.0mm). Bee sockets are designed to provide a convenient and standardized way to connect XBee modules to other devices or microcontrollers. These sockets typically consist of a physical connector and corresponding pin layout that matches the XBee module's form factor. The UartSBee v5 provides a convenient way to connect and communicate with devices that use UART (Universal Asynchronous Receiver-Transmitter) protocol. It features a USB interface on one end and a 6-pin header on the other, which can be used to connect to other devices using jumper wires or a suitable connector. By using the UARTSBee v5, we could establish a virtual serial port connection between a computer and the target device, the ESP32-

Cam module, enabling data transmission and debugging.

For more technical data follow this link: https://wiki.seeedstudio.com/UartSBee_v5/



2.2.3 SD Card

An SD card, also known as a Secure Digital card, is a type of portable storage device commonly used in various electronic devices such as digital cameras, smartphones, tablets, and other devices that require additional storage capacity. The standard SD card is the largest in physical size and it and can have a storage capacity in the range of megabytes up to terabytes. In addition, different SD cards differ in their maximum bus speed which plays an important role in data transmission. The Intenso MicroSDHC Class10 8GB, which was used for this project, guarantees transfer rates of up to 25 MB/s and at least 10MB/s.

For more technical data follow this link:
https://www.intenso.de/produkte/speicherkarten/sd-karte-class-10



2.2.4 Hardware setup

The UartSBee programmer is connected to the power source and is set to 5V (there is a small regulator on the board for this). The board is in turn connected to the camera with jumper cables whereas here the VCC is connected to PIN 5V. The GND is connected to GND and the transmit pin of the programmer is connected to the receive pin of the camera. The other way around, the receive pin of the programmer is connected to the transmit pin of the camera. The last connection with jumper cables is the pins GPIO0 and GND on the camera.

2.3 Methods

For this project, in addition to the lectures given by Professor Dr. Becker, research was conducted almost exclusively online about the hardware used and application-related solutions for the software. For the implementation, we largely relied on the documentation of AI-on-the-edge, through which we had access to a largely working code. With small changes as well as working on a local server and an existing interface, the data could be read out successfully.

For the housing of the camera, on the other hand, a 3D print was made in the Green Fablab with the help of Jeff Josu. In this, the camera was fixed with the help of a PVC pipe and a rubber, so that it has an optimal alignment and a fixed distance to the meter. In addition, an access for the jumper cable was placed on the side to be able to maintain the connection.

3.0 Results and Discussion

3.1.1 Software

Since the program was already available to us, we did not have to insert any new code. However, we will briefly explain the principle of the AI-on-the-edge program. By uploading the given code, the camera takes a photo at a certain interval while the LED is activated. This image is now read by the previously specified reference image, alignment markers and ROI's by an artificial intelligence. In addition, the web server function and Wifi compatibility of the espcam is used to start a web server. The collected data and images are now stored on the SD card and displayed on the web server.

3.1.2 Internal Setup

If you have connected the hardware correctly you can start the setup. Here we followed the documentation of AI-on-the-edge. First the correct code had to be downloaded. We could download it from the releases page of the AI-on-the-edge GitHub. Then the camera had to be flashed to upload new code. There were two ways to do this. We decided to flash the camera using the Python based console. For this we used Anaconda PowerShell Prompt (anaconda3). If you navigate to the right directory, you can flash the camera with the command esptool erase_flash and upload the code with esptool write_flash 0x01000 bootloader.bin 0x08000 partitions.bin 0x10000 firmware.bin. But for this you need the bibliotheca esptool, which could be installed with python -m pip install esptool. The next

step of the setup was to upload the necessary files to the SD card. In addition, the Wi-Fi data had to be adjusted so that the Espcam could connect to the local Wi-Fi. Once all the files on the SD card were correct, all that was left to do was to insert them into the Espcam. After that the connection of the pin GPIO0 with GND was disconnected and the reset button was pressed to run the previously uploaded code. To find the IP of the web server we used the serial monitor of Arduino. This shows you the correct IP address and also a general address with a corresponding port. This is set by default to http://Wartermeter:Port (in our case 80). After this step the camera can be disconnected from the PC and now only needs a power source. The first time the camera is started, the setup mode is started. A few things have to be set here. This includes the creation of a reference image, the definition of the alignment markers, the setting of the Regions of Interest (for us only digital markers, for water meters or other meters also analog markers) and the settings. After the setup is completed, the camera is restarted and set to normal mode. This includes an interface where a current image of the camera is uploaded, with the numbers read out. Possible errors are also displayed. In the settings you can now also change the ROI's, the reference image, or the alignment markers. In addition, you can make individual settings for your camera. In our case we set the frequency of the images to every 3 minutes, changed the hostname to Gas meter and shifted the output value by 3 comma digits. By changing the hostname to Gas meter the interface of the webserver now says Gas meter and the IP changes to http://Gasmeter:Port . Since the gas meter is specified with a value with 3 decimal places, we have also changed this. Under the heading Data you can now display the readout values in csv files or graphs and upload your own files. In addition, the individual images of the numbers that were read out during the last measurement are displayed here. If you change the settings, you only have to save them and restart the server. Under the heading System the server can be restarted manually, and the latest version of the program can be uploaded if an update has been released.

3.1.3 Code

As mentioned before, this project relied on a pre-built program including code, which was uploaded as a complete file to the SD card of the ESP32-Cam after flashing it in advance.

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

3.1.4 Interface Web-Server

As described in 3.1.1 and 3.1.2 the used program already provides a pre-made interface, where we can access the data after an easy and relatively quick registration of over camera. Within the scope of this documentation, a few excerpts are presented below, which show some of the most important functions.





These two pictures show the “Overview” where the latest image taken by the camera is shown. Additionally we are able to read the latest values directly on the right part of the page.



On this screenshot the automated acquisition of the digit ROIs can be seen.



This screenshot shows the history of the read values. This file can also be downloaded as csv.file to process data in another form.

4.0 Discussion

4.1 Problems

During setup and installation, we encountered many problems that unfortunately stalled the process at times. Initially, we had difficulty working with the camera itself, as we were not used to the process of flashing and uploading new code. Often, we didn't know if the code was uploaded, the camera was flashed correctly or just the connection to Uartsbee programmer was wrong. After we could fix these problems we started with the setup of the AI-on-the-edge program. There were some good tutorials on YouTube but most of them were several years old and based on older versions of the AI-on-the-edge program. With the documentation that was available on the GitHub of the program we had at least some written help. Following the documentation, we managed to start the web server, where the next problem occurred. Without a serial monitor we did not know which IP address was assigned. After we found out the correct IP address through Arduino we went to the setup of the camera. Where we encountered the next problem, which was the focus setting. This could be changed by turning the lens of the camera, which is fixed with glue when delivered. By removing the glue, the focus should be changeable. However, during this process, the connection from the board to the camera broke and we had to get a new one. When it worked with the second camera the next step was to build a device that is in the right distance to the gas meter, is removable and also the camera should be removable.

Within all these requirements, the camera always had to be in the same place, otherwise the program would have problems reading the numbers. For our purposes, we have removed the glass cover of the gas meter, as this can easily cause reflections of the LED and built a cover that is light protected to avoid further reflections. However, this change deprived us of the possibility to use our project for private purposes, since the removal of the glass on gas meters is not allowed. Despite the above-mentioned difficulties, we were happy with our project in the end, even though it cannot be used for private purposes. For this, however, you would only have to build a new device over the glass cover.

4.2 Conclusion

The objective of this project was to develop a functional smart meter for a commercial gas meter and thus make remote reading and remote monitoring possible. This can be interesting especially in smarthome applications to detect consumption and potentials. This task was successfully achieved despite the problems outlined in 4.1. As shown in the previous sections, the concept is reliable and easy to use. In addition, the data can currently be retrieved via a local WIFi. A possible goal in the further development of the project is the integration of the smart meter into a smarthome concept in order to optimize not only water and electricity consumption but also gas consumption. For this purpose, it would be necessary to transmit the data to another server. In addition to these findings, legal issues must also be clarified, which have already been explained in the preliminary phase.

5.0 Sources

Picture 1:
https://www.reichelt.de/de/de/entwicklerboards-esp32-kamera-2mp-25--debo-cam-esp32-p266036.html?r=1

Picture 2: https://wiki.seeedstudio.com/UartSBee_v5/

Picture 3:
https://www.amazon.de/Intenso-Micro-Class-Speicherkarte-SD-Adapter/dp/B008RDCCFS/ref=asc_df_B008RDCCFS/?tag=googshopde-21&linkCode=df0&hvadid=214439841299&hvpos=&hvnetw=g&hvrand=11130761054592799 51&hvpone=&hvptwo=&hvqmt=&hvdev=c&hvdvcmdl=&hvlocint=&hvlocphy=9044847&hvtargid=pla-391053427092&psc=1&th=1&psc=1

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

Permanent link:
https://student-wiki.eolab.de/doku.php?id=amc:ss2023:group-x:start&rev=1689775765

Last update: **2023/07/19 16:09**