University of Applied Sciences

HOCHSCHULE EMDEN·LEER

Manual

Counter Detection with ESP32

Author 1:	Nisha Kumari
Matriculation No.:	7023211
Author 2:	Dereck Alpizar
Matriculation No.:	7023782
Author 3:	Neeranjan Jayamurugan Karthikeyani
Matriculation No.:	7023553
Course of Studies:	Business Intelligence and Data Analytics

First examiner:	Prof. Dr. Elmar Wings
Submission date:	January 28, 2023

University of Applied Sciences Emden/Leer · Faculty of Technology · Mechanical Engineering Department
Constantiaplatz 4 · 26723 Emden · http://www.hs-emden-leer.de

Contents

List of Figures

1 Getting started with ESP32

ESP32 is a microcontroller that has built-in WiFi and Bluetooth capabilities. It can be used to create a digital meter using various sensors and displays.

To get started with using an ESP32 for a digital meter, you will first need to set up the development environment and install the necessary software, such as the Arduino IDE. Once this is done, you can upload a sketch (program) to the ESP32 that will control the sensors and display.

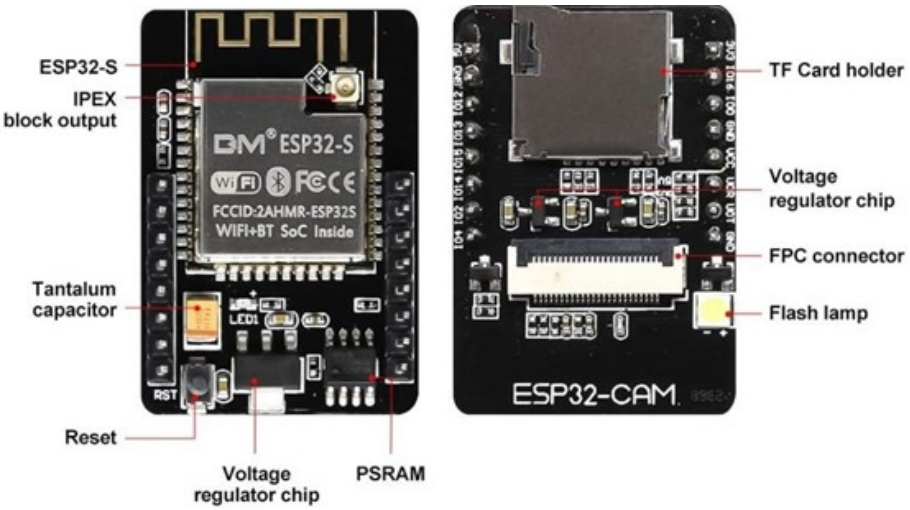

Figure 1.1: Development board components in ESP32-CAM
Reference: [**Lab:2021**]

Following figure shows the output and power pins. There are three GND pins and two pins for power, it could be 3.3V or 5V. To upload the code, two serial pins GPIO1 and GPIO3 are used. GPIO0 is used to check the flash mode. To be able to upload code, GPIO0 should be connected to GND [**randomnerdtutorials**].

Figure 1.2: ESP32 CAM PINOUT
Reference: [**randomnerdtutorials**]

1.1 Functions

The main function of a digital meter created using an ESP32 would be to read sensor data, process that data, and display it on a screen. The sketch for the digital meter would likely include code to handle communication with the sensors, code to process the data, and code to display it on the screen.

The GUI or UI of the digital meter would depend on the specific implementation. It could include a simple numerical display, or it could include more advanced features such as graphs or charts.

In this project, we prefer a numerical display.

1.2 Specifications

- Processing power of up to 240 MHz dual core
- Wi-Fi - 802.11b/g/n/e/i
- Bluetooth 4.2 BR/EDR and BLE standards

- Up to 520 KB SRAM
- Up to 4MB flash memory
- 40-pin GPIO header

More detailed specifications and hardware decsriptions are already mentioned in ESP32 section under DomainKnowledge chapter of the team report.

1.3 Independence of internals

The independence of the internals would depend on the specific implementation of the digital meter using the ESP32. It could be designed to operate independently or to communicate with other devices.

The internal peripherals, such as the Analog-to-Digital Converter or the Inter-Integrated Circuit controller operate independently of the main processor. The digitization of analog meter readings without the need for additional external components is possible.

2 How to use

2.1 Hardware Setup

Requirements are as follows:

- ESP32 development board: This will be the main microcontroller that will read the analog input from the analog meter and send the data to your computer or the internet.
- Analog water/electricity meter: This is the device that will measure the amount of volume/energy consumed. It will output the number that corresponds to the amount of electricity/water consumed.
- USB cable: This will be used to connect the ESP32 to your computer to upload the code and monitor the data.
- Power supply: The ESP32 can be powered through USB or an external power supply, a typical voltage range is 3.3V
- Connecting wires or ESP MB can be used to connect the ESP32 to the digital electricity meter.

2.1.1 Steps to get Hardware started

- Install the ESP32 cam in front of the analog meter and connect it to the meter using appropriate connection wires.
- Install the necessary libraries in the Visual Studio: Open the IDE, go to the library manager, and install any libraries necessary to interact with the meter and the ESP32 board.
- Test the already uploaded code by opening the serial monitor and sending commands to the ESP32.

2.2 Software Setup

Requirements:

- Visual Studio

- Python (version 3.9)
- ESP-IDF
- Required Python libraries

To setup the code Visual Studio code is required. Also, ESP-IDF plugin is required to develop, build, flash, monitor and debug the code loaded in to ESP32 devices.

Steps to install the Visual Studio code are mentioned in our team report under DomainKnowledge chapter in section Visual Studio Code.

2.2.1 Steps to setup code

Important files Install the necessary libraries mentioned in requirements text file at location : ../ML23-05-Counter-Detection-EPS32/Code/Docs/Requirements.txt.

The main.py python file is at location : ../ML23-05-Counter-Detection-EPS32/Code/main.py.

The original dataset (folder- ImagesOriginal), python files for image augmentation, for modifying brightness and for scaling data are stored at location : /Users/nishakumari/Documents/GitHub/ML23-05-Counter-Detection-EPS32/Code. Also, this location has the final images after data tranformation in the folder ImagesFinal and the validation images in folder ImagesValidation.

To run.py files To run a .py file in Visual Studio Code, you will need to have Python installed on your computer. Once you have Python installed, you can follow these steps to run a .py file in Visual Studio Code:

- Open the .py file in Visual Studio Code.
- Open the integrated terminal by clicking on the terminal tab or by pressing ctrl + .
- In the terminal, type "python" followed by the name of the .py file and press enter. For example, if the name of the file is "hello.py", you would type "python hello.py" and press enter.
- The code in the .py file will now run and the output will be displayed in the terminal.
- Alternatively, you can also run the code by using the VS Code extension "Python" which allows you to run the code and have better debugging capabilities and also have intellisense for python.

- It's worth mentioning that you can also use the built-in debugging features to debug your Python code in Visual Studio Code by installing the Python extension, then set a breakpoint and start debugging.

3 Maintenance and Troubleshooting

3.1 Maintenance

Maintenance for a digital meter created using an ESP32 would include regular checks of the sensors to ensure they are working properly, and replacing any sensors that are not working.

Ensure that the ESP32 cam and the meter surfce is dust free to ensure correct and accurate readings.

Also, to make sure that ESP32 cam can read the meter correctly, proper lighting and temperature conditions should be maintained.

3.2 Troubleshooting

Troubleshooting for a digital meter created using an ESP32 would involve checking for common issues such as incorrect wiring or programming errors.

Firstly it should be identified whether the issue is related to hardware or software. If it is a hardware issue then the device would need to be replaced and the model would need to be transferred to the new hardware device.

Troubleshooting the software issue on ESP32 would require connecting it to the laptop/desktop to debug and reinstall the model code.

3.2.1 Steps to troubleshoot

- Connect the ESP32 to the PC: Attach the ESP32-CAM to the ESP32-MB
- Connect the ESP32-MB with the USB cable to the PC/laptop USB port.
- Install the necessary drivers: Check if your PC has the necessary drivers installed to recognize the ESP32. If not, you can download the latest drivers from the official website of the ESP32 manufacturer.

- Open the Arduino IDE: Launch the ESP-IDF on your PC and make sure that it recognizes the ESP32.
- Upload code to the ESP32: Use the upload button in the ESP-IDF to upload code to the ESP32. Make sure that you have selected the correct board and port in the tools menu before uploading.
- Open the serial monitor in the ESP-IDF to check for any error messages or issues. This will help you troubleshoot any issues with the code or the connection between the ESP32 and the PC.
- Check the connections: Make sure that all the connections between the ESP32 and the PC are secure, and that the connections between the ESP32 and other devices are also secure.
- Try a different USB cable or USB port: Sometimes, the issue may be with the USB cable or the USB port. Try using a different cable or a different USB port on your PC.
- Check the board settings: Make sure that the board settings in the Arduino IDE match the ESP32 board you are using.
- Check the power supply: Make sure that the ESP32 is receiving power. It could be that the USB cable is not providing enough power, and you may need to use an external power supply.
- Check online resources: If you are still unable to connect your ESP32 to your PC, check online resources such as forums, documentation, or FAQs for troubleshooting tips and solutions.

3.3 Safety Precautions

- Store the ESP32 device in a 3D printed case.
- Avoid touching the circuit board and wires with bare hands to avoid electric shocks.
- For proper grounding make sure that the ESP32 is properly grounded to avoid damage from voltage transients and other electrical disturbances.
- Observe ESD protection: Electrostatic discharge (ESD) can damage the ESP32, so it is important to observe ESD protection when handling or working with the device.

3.4 FAQs

For frequently asked questions, please visit ESP-FAQ