Frequently Asked Questions (FAQ) – Elektor ESP32 Energy Meter

Q1. How do I initially program the ESP32 Energy Meter without a USB-C port?

The USB-C port was intentionally omitted for safety, complexity, and cost reasons. You can program the ESP32 using an external ESP32 programmer connected to the JP2 header on the board. After initial programming, you can enable OTA (Over-The-Air) updates for convenient future firmware updates.

Q2. Can I add a USB-C port myself?

Yes, it is possible, but you need to source the required SMD components yourself. Elektor does not currently offer a kit for this, but the BOM list is available in the project's GitHub repository.

Q3. What type of OLED display is compatible with the energy meter?

The energy meter supports common I²C OLED displays, typically the 0.96-inch 128×64 OLED screens with the SSD1306 chipset. You can also use larger displays (1.3", 1.9"), but minor firmware adjustments will be needed for layout and resolution.

Q4. How do I connect the OLED display?

Connect your OLED display to the Qwiic-compatible I²C port (K5 connector) on the board. If your OLED screen's pin order differs, two connector options at K5 address this.

Q5. Does the OLED display require programming?

Yes. The initial sketch provided has basic OLED support built-in, and the ESPHome firmware fully integrates OLED functionality. You can customize the display by using the Adafruit_SSD1306 & Adafruit_GFX libraries.

Q6. How can I set up Wi-Fi connectivity for Home Assistant integration?

Initially, configure your ESP32 using ESPHome's web interface with basic setup parameters. After initial configuration, copy and paste the detailed YAML configuration from our GitHub repository into your device settings and upload it.

Q7. Is it possible to use the energy meter without ESPHome or MQTT?

Yes, the meter can function entirely offline, showing real-time data on the OLED screen without integration. You can modify the provided MQTT-based sketch to remove MQTT functions and add SD card logging functionality via the I²C SD card module if desired.

Q8. What power supply should I use?

The required transformer should provide up to 300 mA at 12 V, sufficient to power the ESP32-S3 and peripherals such as sensors and the OLED display.

Q9. How accurate is the energy meter?

The ESP32 Energy Meter provides stable and consistent readings sufficient for residential use. Although not industrial-grade, the ATM90E32 calibration features ensure acceptable accuracy suitable for home monitoring purposes.

Q10. Can I recover the ESP32 if it stops responding?

Yes. If the module is responsive, reflash it using a proper 3.3 V ESP32 programmer. If damaged, you can replace the ESP32-S3 module or connect another ESP32 module directly to the IO header.

Q11. Are there any known limitations or compatibility notes I should be aware of?

Ensure all interfacing and programming tools used provide a 3.3 V logic level. The ESP32S3 is not tolerant to 5 V signals and could be damaged if connected to incompatible equipment.

Q12. What if my OLED display has reversed VCC and GND pins?

The board provides two connector options on K5 specifically to accommodate OLED displays that have reversed VCC and GND pins, common in some OLED screens.

Q13. Can I log energy data to an SD card?

Yes, you can connect an I²C SD card module through the Qwiic connector. You will need to modify and extend the provided sketch or firmware to support data logging.

Q14. Does the energy meter include a built-in webserver?

Yes, the energy meter project includes a built-in webserver hosted on the ESP32. This web interface mirrors the OLED display data, offering users another convenient method to monitor energy usage remotely.

Q15. What should I do if my device does not connect to Wi-Fi?

Check your YAML configuration carefully. Ensure the correct SSID and password are entered, and verify that the static IP address and subnet settings match your network.

Q16. What is the recommended resistor setup for voltage and current sensing?

The meter uses a 1:101 voltage divider for safety and flexibility, resulting in about ±200 mV at the ADC for a 20 V peak input. For current sensing, a 5 Ω burden resistor provides about 250 mV, which balances resolution and thermal performance effectively. You can adjust these resistors for higher ADC utilization if desired.

Q17. Can I use different programmers like FTDI or Arduino boards for flashing?

Use only ESP32-compatible programmers at 3.3 V logic levels. Avoid using 5 V logic devices like some FTDI and Arduino boards, as they can damage the ESP32-S3 module.

Q18. Is pre-installed firmware provided?

The energy meter is intentionally left without pre-installed firmware to allow users the flexibility to select and configure their preferred firmware environment (ESPHome, MQTT, etc.).

Q19. What happens if I accidentally used 5V logic and damaged the ESP32-S3?

If damage occurs, the ESP32-S3 module can be desoldered and replaced. Alternatively, a separate ESP32-S3 module can be connected directly via the IO headers.

Q20. Where can I find comprehensive documentation and firmware examples?

Comprehensive documentation, firmware examples, and the full Bill of Materials (BOM) are available on the official Elektor GitHub repository.