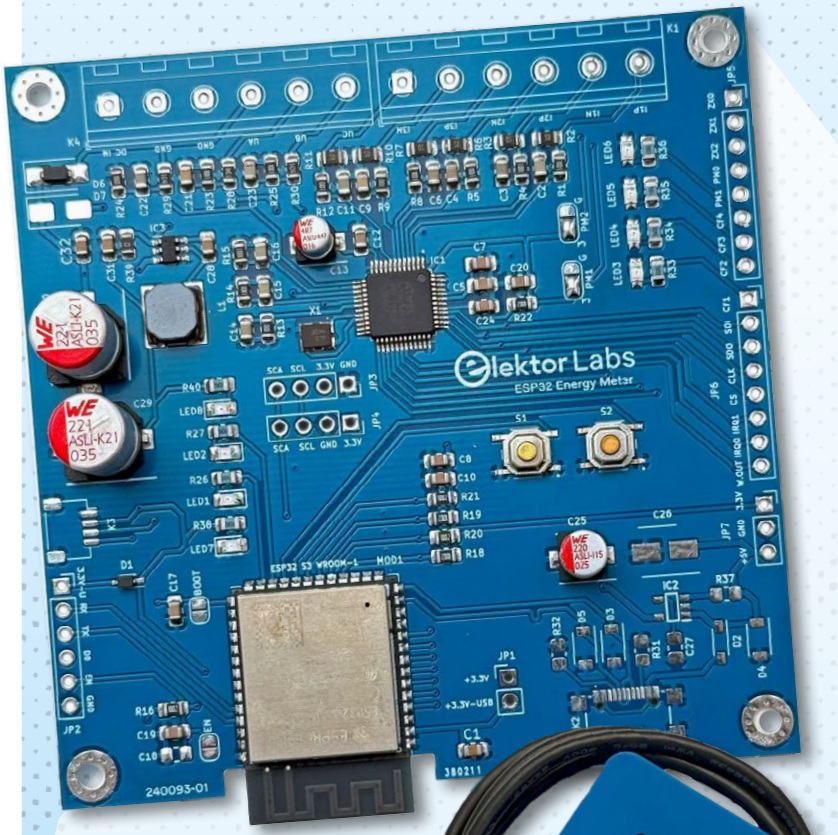
SPECIAL elektor SPECIAL

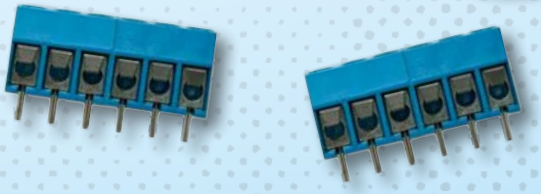
design › share › earn

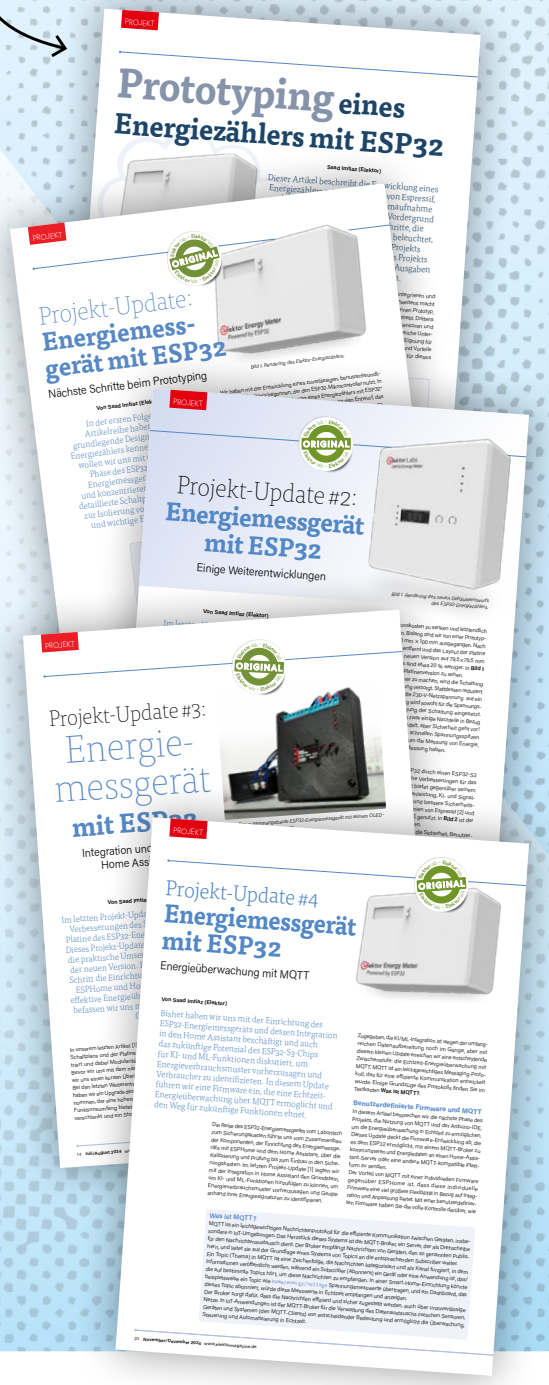
Elektor ESP32-Energiemessgerät

Elektor lab · Elektor lab · ORIGINAL · Elektor lab · Elektor lab

IN DIESER AUSGABE







Prototyping eines Energiezählers mit ESP32

Saad Imtiaz (Elektor)

Dieser Artikel beschreibt die Entwicklung eines Energiezählers mit einem ESP32 von Espressif, wobei die Überwachung der Stromaufnahme in Echtzeit und die Sicherheit im Vordergrund stehen. Es werden die ersten Schritte, die Anforderungen und Überlegungen beleuchtet, die beim Start eines Embedded-Projekts anfallen. Mit dem Fortschreiten des Projekts werden wir in den nächsten Elektor-Ausgaben über die Ergebnisse berichten.

Bild 1. Test-Rendering, wie unser Energiezähler aussehen könnte.

In der Elektronik kann die richtige Kombination der richtigen Technologien zu bedeutenden Fortschritten führen. Dieses Projekt zielt darauf ab, einen Energiezähler zu entwickeln, der den ESP32-Mikrocontroller von Espressif und das Energiemess-IC ATM90E32AS von Microchip verwendet. Dieser Artikel beschreibt die Auswahl der Komponenten bis hin zum ersten Prototyping. Das Ziel ist einfach: ein zuverlässiges System für eine genaue Energieerfassung im heimischen Zählerkasten oder in der Werkstatt zu schaffen. Mit diesem Messgerät können die Benutzer ihren Stromverbrauch in Echtzeit verfolgen und so Erkenntnisse gewinnen, die zu einer effizienteren Energienutzung führen können.

Design und Anforderungen

Das Projekt hat klare Ziele und Entwicklungsanforderungen: Echtzeit-Überwachung des einphasigen Stroms mit drei Stromwandlern (CTs), Bezahlbarkeit und Benutzerfreundlichkeit. Die Wahl der Hauptbestandteile ESP32 und ATM90E32AS orientierte sich an diesen Zielen und bot sowohl Kosteneffizienz als auch zuverlässige Leistung. Ein weiteres Ziel war es, die Abmessungen unter 100×80×30 mm (L×B×H) zu halten, damit das Gerät in einem Schaltschrank untergebracht werden kann. Um die Benutzerfreundlichkeit zu erhöhen, ist auch eine mobile Schnittstelle für die Fernüberwachung sowie ein OLED-Display mit Tasten für die direkte Interaktion vorgesehen. Das Design ermöglicht auch künftige Software-Updates, so dass ein langfristiger Nutzen für den Verbraucher gewährleistet ist. In **Bild 1** ist ein (gerendertes) Bild des aktuellen Gehäuses des Prototyps dargestellt.

Auswahl des Mikrocontrollers

Die Wahl des ESP32-Mikrocontrollers beruhte auf einer detaillierten Analyse seiner Fähigkeiten. Der Chip zeichnet sich in mehreren Punkten aus, die für den Erfolg dieses Projekts von Belang waren. Erstens lässt er sich leicht in verschiedene Schaltungsentwürfe integrieren und bietet dadurch Flexibilität in der Entwicklungsphase. Zweitens macht ihn seine Kosteneffizienz zu einer attraktiven Wahl für einen Prototyp, der ein Gleichgewicht zwischen Leistung und Kosten anstrebt. Drittens bietet die Kompatibilität mit einer breiten Palette von Sensoren und ICs erhebliche Vorteile. Und schließlich trägt die umfangreiche Unterstützung der Community für den ESP32-Chip zu seiner Eignung für dieses Projekt bei. **Bild 2** hebt die wichtigsten Merkmale und Vorteile des ESP32-D0WD-V3 hervor, die dazu führten, dass er für dieses Projekt ausgewählt wurde.

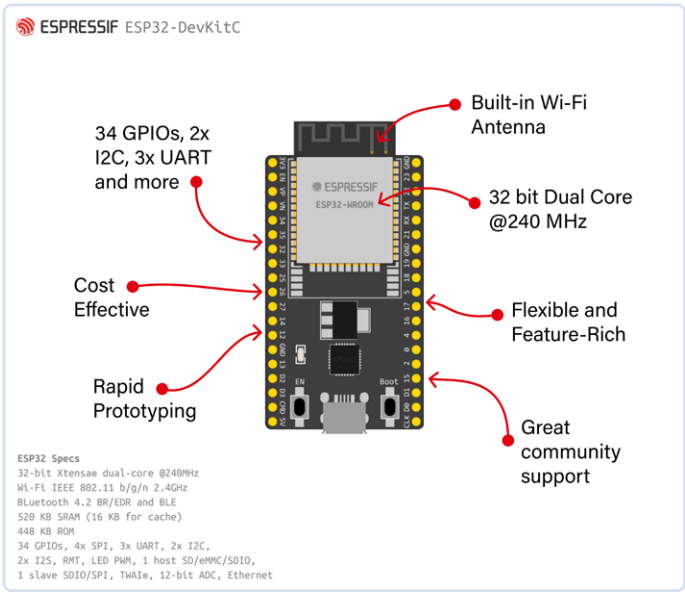


Bild 2. Hauptmerkmale und Vorteile des ESP32.

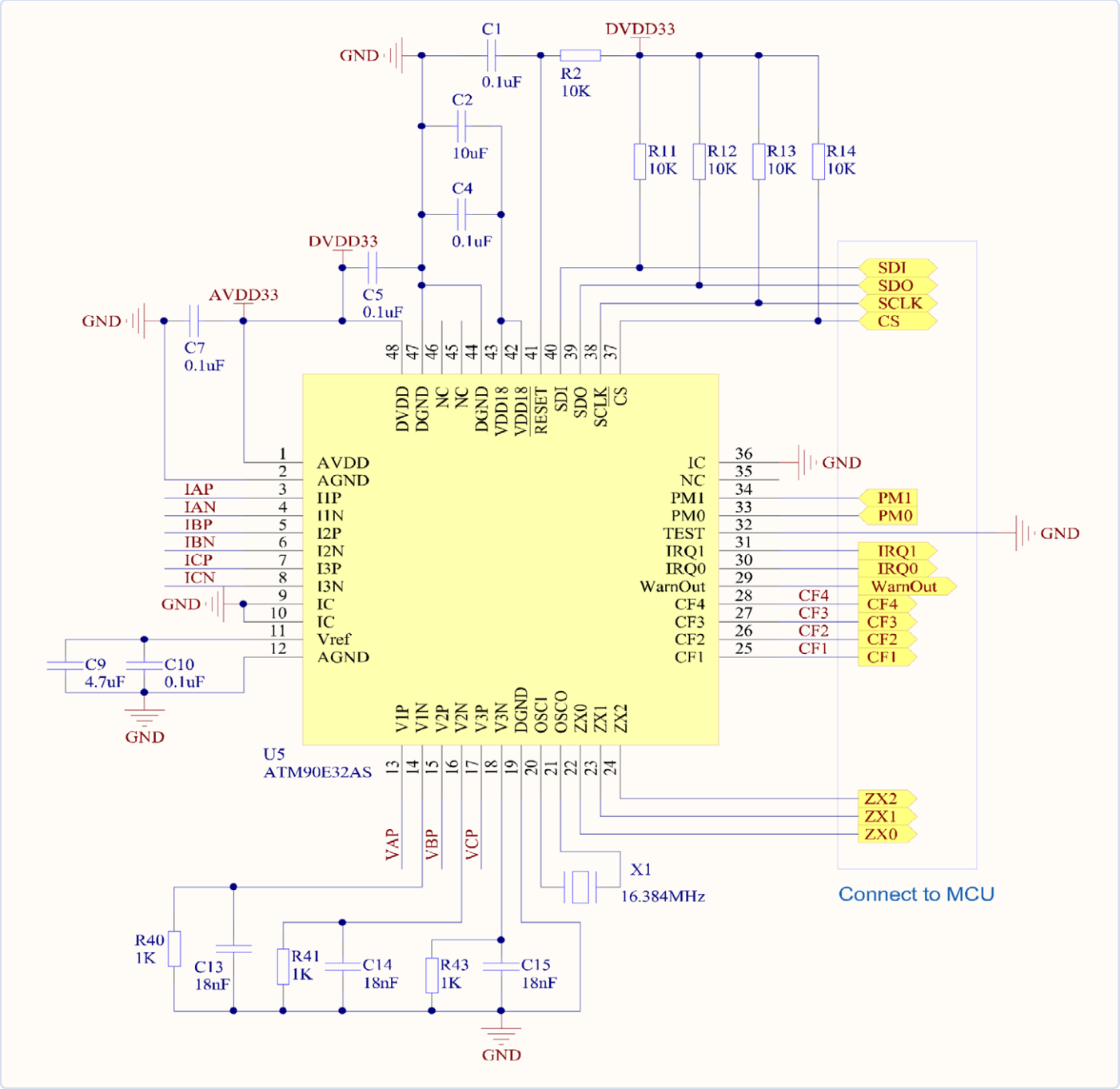


Bild 3. Der Energiezähler basiert auf einer Application Note von Atmel [2]. Hier sieht man die Schaltung rund um das Zähler-IC.

Integration des Zähler-ICs

Der ATM90E32AS-IC von Microchip wurde gemäß den Anwendungshinweisen [2] des Herstellers verwendet. Das Dokument diente als Eckpfeiler, damit das Energiemess-IC nahtlos mit dem ESP32-Mikrocontroller kommunizierte konnte. Dennoch war diese Phase nicht frei von Problemen und Herausforderungen. Die Beschaffung der richtigen Bauteile (unter Einhaltung des Budgetrahmens) erforderte angesichts der eingeschränkten Verfügbarkeit eine akribische Planung. In **Bild 3** ist die von Atmel (jetzt Microchip) zur Verfügung gestellte Application Note dargestellt.

Entwurfsphase und elektrische Sicherheitsstandards

Die Entwurfsphase ist in der Tat ein entscheidender Teil des Entwicklungsprozesses, vor allem wenn die Sicherheit ein unverzichtbarer Aspekt ist. Bei einem Gerät, das für die Interaktion mit Netzwechselspannungen ausgelegt ist, muss die Einhaltung etablierter Sicherheitsnormen genauestens beachtet werden. In **Bild 4** ist das Blockdiagramm des Projekts dargestellt.

Um die elektrische Sicherheit zu gewährleisten, wurden mehrere spezielle Bauteile in das Design aufgenommen: Metalloxid-Varistoren (MOVs) wurden zur Unterdrückung transienter Spannungen eingesetzt, um die Schaltung vor Spannungsspitzen zu schützen, wesentliche Sicherungsbauteile sollen Überstrombedingungen verhindern.

Neben der Bauteile-Auswahl wurden bei der Schaltungsentwicklung auch Überlegungen zum Layout angestellt, um die Sicherheitsnormen einzuhalten. Zwischen den leitenden Elementen auf der Platine wurden angemessene Kriech- und Luftstrecken eingehalten, um Lichtbögen und ähnliche unliebsame Erscheinungen zu verhindern. Die Leiterbahnbreiten für die Wechselspannungsverbindungen wurden sorgfältig berechnet, damit sie die Nennströme bewältigen, wobei die IPC-2221-Normen [1] beachtet wurden. Dies war entscheidend,

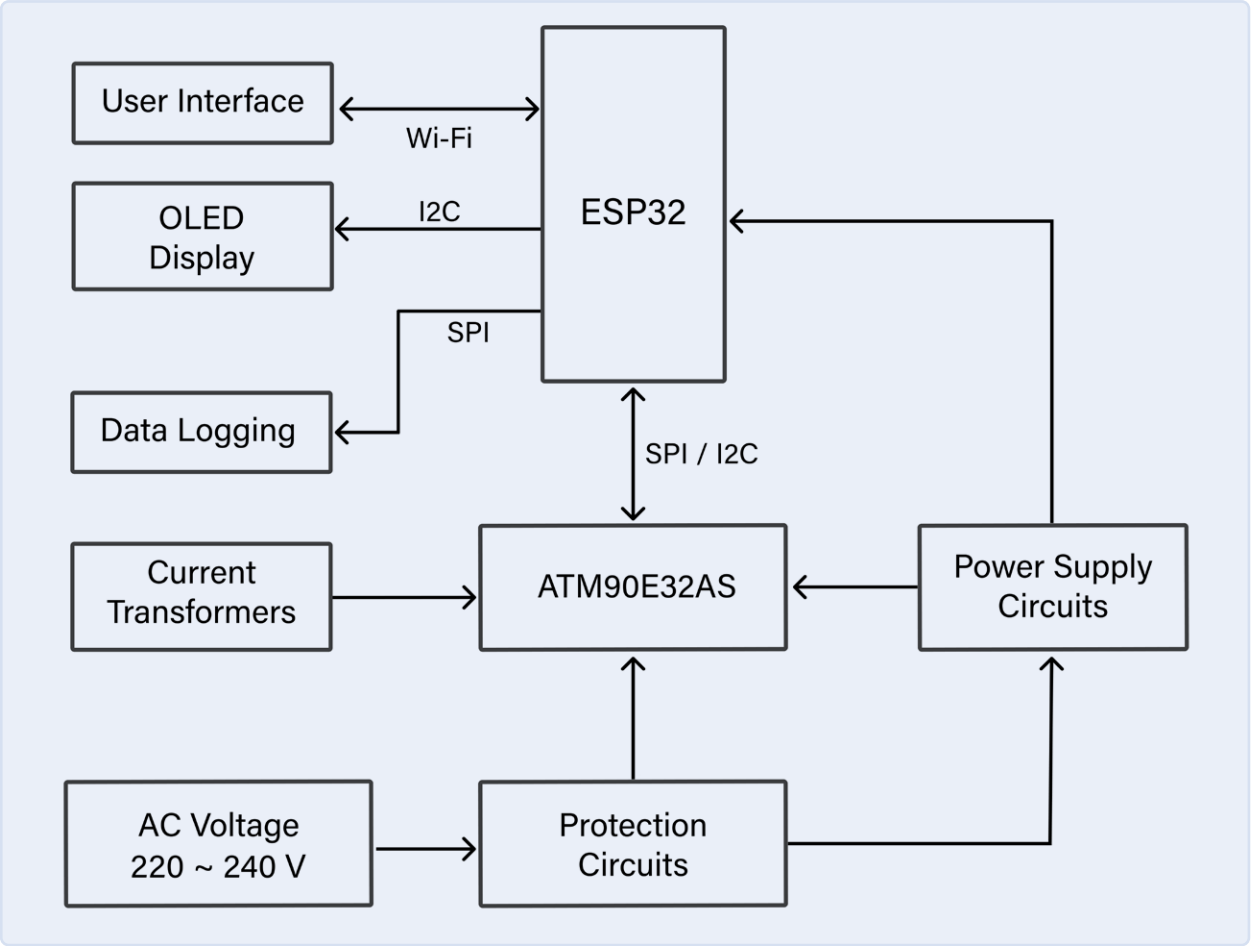


Bild 4. Blockschaltbild unseres Energiezählerprojekts.

um die thermische Leistung der Platine unter Volllastbedingungen zu gewährleisten. Für eine saubere Masseverbindung wurde eine solide Massefläche geplant. Besonderes Augenmerk wurde auf die Ausgestaltung von Differenzialpaaren gelegt, um die Signalintegrität zu gewährleisten, wobei darauf geachtet wurde, dass das Routing einer präzisen Geometrie folgt, um elektromagnetische Störungen zu minimieren.

Die Entscheidung für JLC PCB

Nach Prüfung verschiedener Platinen-Services und -Bestückungsdienste fiel die Wahl auf JLC PCB. Der Hauptgrund für diese Wahl war die Ausgewogenheit von Kosteneffizienz und Zuverlässigkeit, die das Unternehmen bietet. Diese Entscheidung war wichtig, um das Projekt im Rahmen unseres Budgets zu halten, ohne Kompromisse bei der Qualität der bestückten Platine eingehen zu müssen. Derzeit werden der Schaltplan des Prototyps und die Leiterplattenentwürfe fertiggestellt, so dass die Prototypen bald in die Produktion geschickt werden können.

Ein Blick zurück und einer nach vorne

Rückblickend zeigt dieses Projekt, was erreicht werden kann, wenn sorgfältige Planung auf gute Technik trifft. Die Hürden, mit denen wir konfrontiert wurden, haben uns geholfen, unseren Entwurf zu verbessern. Wir gehen davon aus, dass wir von der Herstellung eines Prototyps zu einer möglichen Massenproduktion übergehen können, und vor allem, dass das Gerät einen echten Unterschied im Umgang der Menschen mit ihrem Energiebedarf bewirken wird. Über den Fortgang dieses Projekts werden wir in den nächsten Elektor-Ausgaben ausführlich berichten - wir sind ja noch dabei, den Prototyp herzustellen, zu testen und an der Software zu basteln, mit der er betrieben werden soll. Es wird also noch mehr kommen, bleiben Sie dran! Wir werden in der kommenden Januar/Februar-Ausgabe 2024 von Elektor, die dem Thema Strom und Energie gewidmet ist, über die Fertigstellung berichten.

RG – 230646-02

Haben Sie Fragen oder Kommentare?

Wenn Sie technische Fragen zu diesem Artikel haben, wenden Sie sich bitte an die Elektor-Redaktion unter redaktion@elektor.de.

Passende Produkte

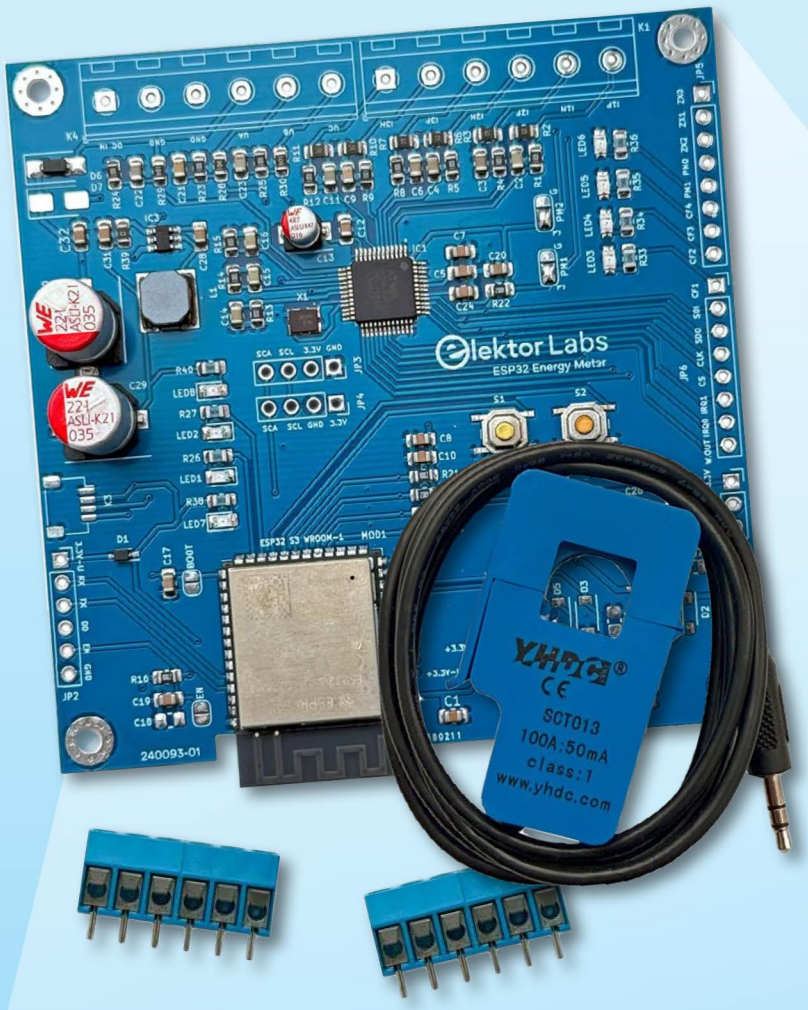
- **ESP32-DevKitC-32E**
  www.elektor.de/20518
- **ESP32-C3-DevKitM-1**
  www.elektor.de/20324

WEBLINKS

[1] IPC-2221A-Standards: https://t1p.de/wjwop
[2] Application Note ATM90E32AS : https://t1p.de/i5axy

NEU

Elektor ESP32-Energiemessgerät



- SMD-vorbestückte Platine
- Inkl. Stromtransformer
- Kompatibel mit ESPHome, Home Assistant und MQTT
- Zugehörige Elektor-Artikel stehen zum Gratis-Download

Jetzt bestellen:

www.elektor.de/21044

elektor
design > share > earn

Projekt-Update: Energiemessgerät mit ESP32

Nächste Schritte beim Prototyping

Bild 1. Rendering des Elektor-Energiezählers.

Von Saad Imtiaz (Elektor)

In der ersten Folge dieser Artikelreihe haben wir das grundlegende Design des Elektor-Energiezählers kennengelernt. Nun wollen wir uns mit der nächsten Phase des ESP32-basierten Energiemessgeräts befassen und konzentrieren uns dabei auf detaillierte Schaltpläne, Strategien zur Isolierung von Schaltkreisen und wichtige Erweiterungen.

Wir haben mit der Entwicklung eines zuverlässigen, benutzerfreundlichen Energiezählers begonnen, der den ESP32-Mikrocontroller nutzt. In unserem letzten Beitrag „Prototyping eines Energiezählers mit ESP32" [1], haben wir die grundsätzlichen Anforderungen an den Entwurf, das Blockschaltbild und den Plan für den Start dieses Projekts besprochen. Bevor wir ein Update dazu geben, lassen Sie uns noch einmal rekapitulieren. Das Äußere des Energiezählers ist in **Bild 1** zu sehen. Unser Schwerpunkt lag auf der Energieüberwachung in Echtzeit, wobei ein Hauptaugenmerk auf Sicherheit und Erschwinglichkeit lag. Um die Energiemessung präzise zu gestalten, haben wir uns für ein mehrphasiges Energieüberwachungs-IC ATM90E32AS von Atmel [2] entschieden. Dieses IC bezieht die einphasige Spannung aus dem Netz und verwendet Split-Coil-Transformatoren, um den Strom sicher zu messen. Als Mikrocontroller für das Gerät wurde der ESP32 ausgewählt, da er über integriertes WLAN verfügt und im Vergleich zu anderen MCUs sehr kostengünstig ist. In **Bild 2** ist das aktualisierte Blockschaltbild des Projekts dargestellt.

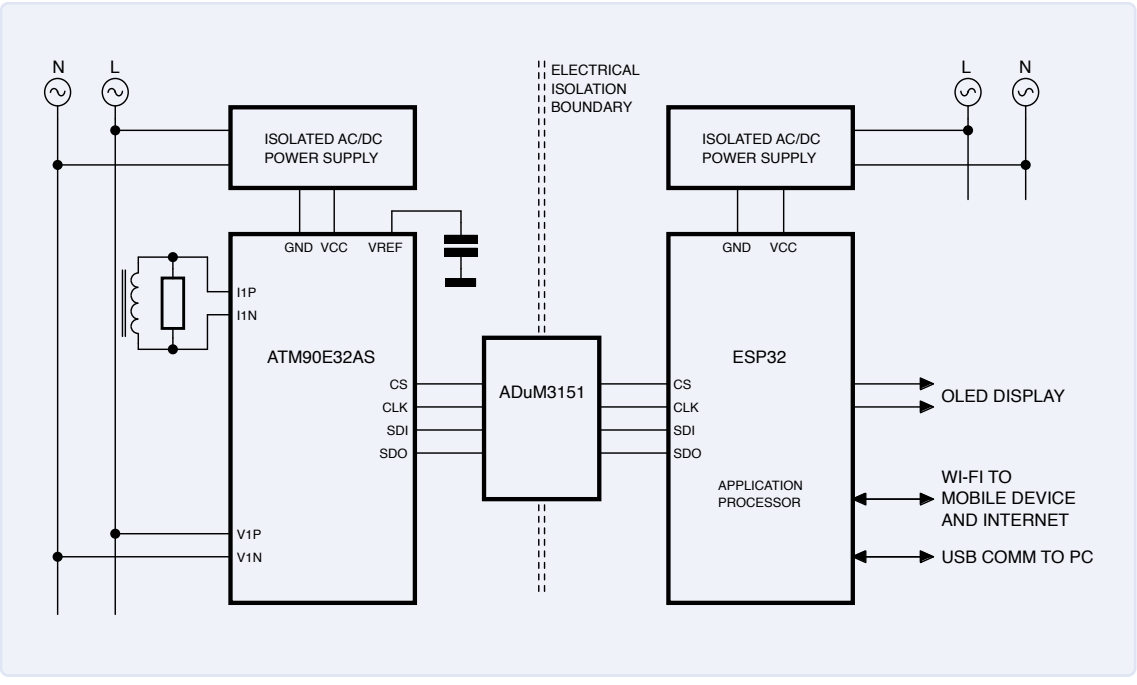


Bild 2. Blockschaltbild des Elektor-Energiezählers.

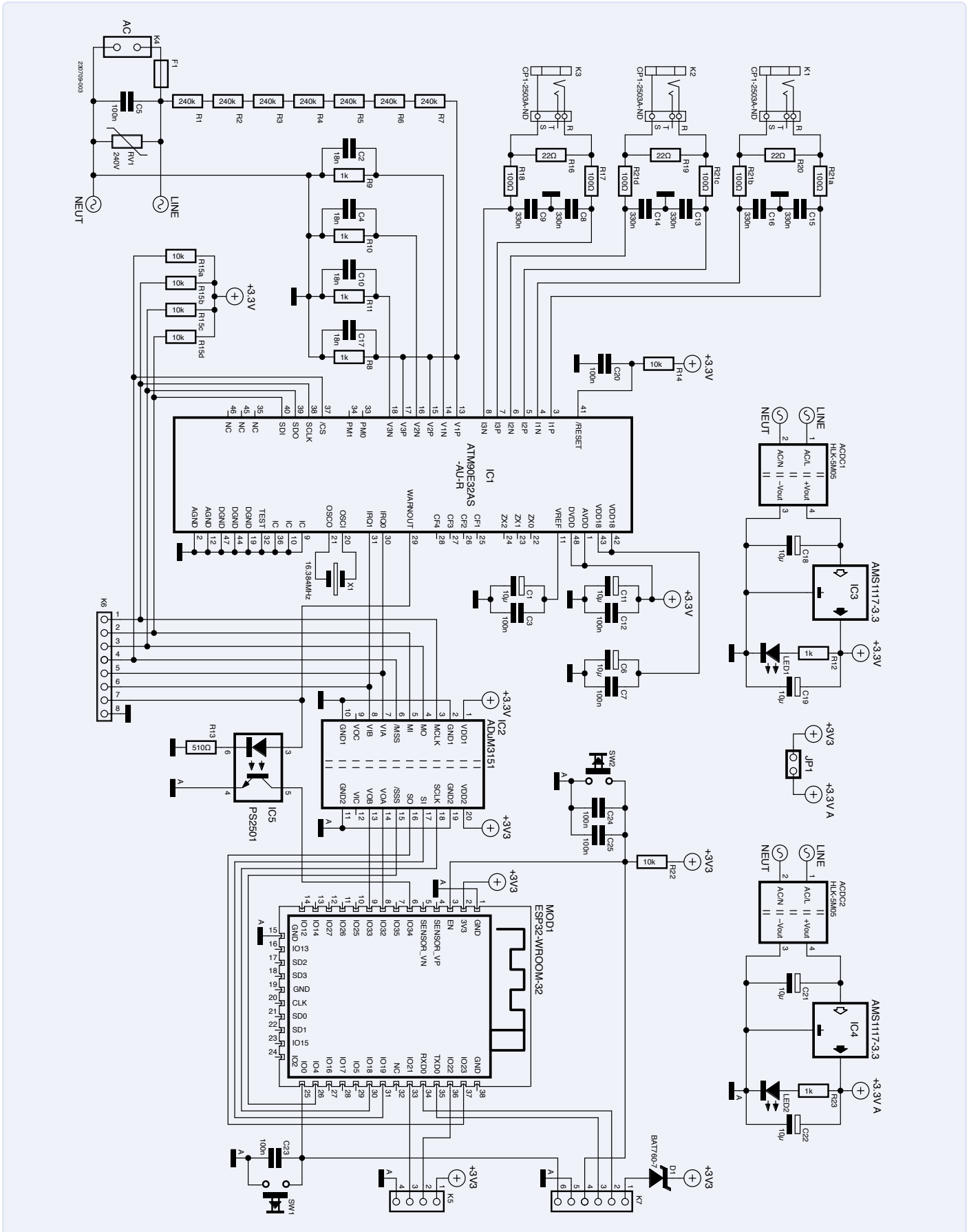


Bild 3. Schaltbild des Energiezähler-Projekts.

Bild 4. Der Stromsensor SCT013 von YHDC. (Quelle: YHDC)

Die geplante Größe des endgültigen Energiezählers beträgt 100 × 80 × 30 mm (L×B×H), aber für den Prototyp ist unsere Platine mit 100 mm × 100 mm noch zu groß. Unser Ziel ist es aber, diesen Prototyp als Proof-of-Concept zu verwenden. Wenn wir damit erst einmal erfolgreich sind, können wir die Größe für die endgültige Version auf 100 × 80 mm oder sogar weniger reduzieren. Der Hauptzweck der Entwicklung dieses Energiemessgeräts bestand darin, ein IoT-fähiges, preisgünstiges Gerät zu entwickeln, das genaue Energiemessungen vornehmen und dem Benutzer über ein mobiles Gerät (Handy, Computer) Energiedaten in Echtzeit zur Verfügung stellen kann, damit dieser seinen Stromverbrauch in Echtzeit verfolgen und in der Folge energieeffizienter handeln kann.
In diesem Artikel gehen wir näher auf die Entwicklung des Projekts ein und beleuchten den Schaltplanentwurf, die Implementierung einer Isolierung der Stromkreise und die wichtigsten Verbesserungen, die wir seit unserem ursprünglichen Konzept aufgenommen haben.

Entwurf des Schaltbildes

Der ESP32-Mikrocontroller bleibt das Herzstück unseres Geräts und bildet eine nahtlose Schnittstelle zum ATM90E32AS für die Energiemessung. Unser aktualisierter Schaltplan spiegelt einen schlankeren Ansatz wider, mit reduziertem Rauschen und Verbesserungen in der Signalintegrität, der Isolierung der Schaltkreise und vieles mehr. In **Bild 3** sehen Sie den vollständigen Schaltplan des Projekts.
Der ATM90E32AS (IC1) ist das Gehirn des gesamten Projekts. Er verbindet die Netzspannung über sieben in Reihe geschaltete 240-k-Widerstände (R1...R7) mit den Pins V1P, V2P und V3P. Der Einfachheit halber werden alle diese Pins mit einer Phasenspannung aus dem Netz versorgt. Sie fragen sich vielleicht, warum nicht ein Transformator anstelle dieser Reihe von Widerständen? Weil wir aufgrund des von uns gewählten Ansatzes Größen- und Kostenbeschränkungen haben. Abgesehen von der geringen Größe der Widerstände gibt es einen weiteren Vorteil, nämlich eine geringere Phasenverzögerung. Transformatoren können eine Phasenverzögerung zwischen den Primär- und Sekundärwicklungen verursachen, die sich auf das Timing und die Genauigkeit der Spannungswerte bei Energiemessungen auswirken kann. Bei der Verwendung von Widerständen wird diese Phasenverzögerung erheblich reduziert, was zu genaueren Echtzeit-Spannungsmessungen führen kann. Die Verwendung dieser Reihenwiderstände hat jedoch einen großen Nachteil, nämlich die fehlende galvanische Isolierung. Wir werden später in diesem Artikel darauf eingehen.
Kommen wir nun zur Strommessung: Hierfür verwenden wir Split-Coil-Stromtransformatoren des Typs SCT013 von YHDC, die an die Audio-Klinkenbuchsen K1...K3 angeschlossen werden. Es handelt sich um Stromwandler mit geteiltem Kern, wie in **Bild 4** dargestellt. Der Grund für die Wahl des Stromwandlers war, dass er kostengünstig ist, einfach anzuwenden und vor allem, dass die Außenleiter bei der Montage nicht aufgetrennt/abgeklemmt werden müssen, da sich die obere Kernhälfte des Wandlers einfach aufklappen lässt.
Der Energiezähler wird von den zwei HLK-5M05-Modulen ACDC1 und ACDC2 der Firma Hi-Link versorgt, um eine galvanische Trennung zwischen der MCU und der Schaltung des Energiezählers zu gewährleisten und vor Risiken durch die Netzspannung zu schützen. Die beiden Regler AMS1117-3.3 liefern stabile 3,3-V-Spannungen, die für den ESP32 und andere Niederspannungsbauteile unerlässlich sind. Die Sicherheit wird durch Sicherungen (F1) für den Überstromschutz und R23, einen Metalloxid-Varistor (MOV) gegen Spannungsspitzen weiter verstärkt. Zu Diagnosezwecken zeigen LED1 und LED2 die Stromversorgung und den Betriebsstatus an. Über den Anschluss K6 werden alle Ausgänge der MCU für das Debugging angeschlossen.

Isolierung der Schaltung

Im Schaltplan haben Sie vielleicht zwei DC-Massen bemerkt, GND und GNDA. Der Masseanschluss GND ist mit dem IC1 verbunden und außerdem mit dem Neutralleiter des Wechselstromnetzes. GNDA ist eine isolierte Masseklemme, die mit dem ESP32-WROOM-32D (MOD1) verbunden ist. Um die Sicherheit zu gewährleisten, muss der ESP32 unbedingt vom Neutralleiter des Wechselstromnetzes isoliert werden. Da IC1 über keine galvanische Trennung verfügt, muss er andersweitig von der Controllerabteilung isoliert werden. Dabei stellt sich natürlich die Frage, wie beide Chips über das SPI überhaupt kommunizieren sollen. Hier kommt IC2, ein ADuM3151 von Analog Devices, ins Spiel. Der ADuM3151 ist von zentraler Bedeutung für die sichere Kommunikation zwischen IC1 und dem ESP32-WROOM-32D, da er eine galva-

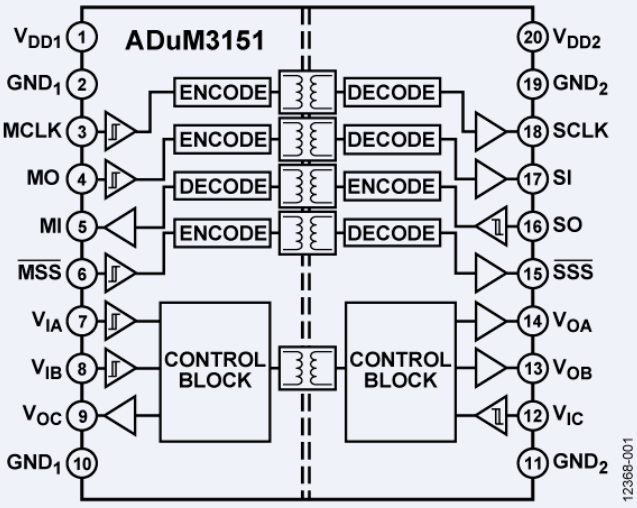


Bild 5. Funktionsschaltbild des SPIsolators ADuM3151. (Quelle: Analog Devices [3])

nische Trennung der SPI-Leitungen gewährleistet. In **Bild 5** sehen Sie, wie IC2 im Inneren aufgebaut ist [3]. Das IC verwendet induktive Koppler, um digitale Signale über eine Isolationsbarriere zu übertragen. Dabei werden die Hochspannungstransienten des Wechselstromnetzes effektiv vom ESP32 und dem eventuell daran angeschlossenen Computer ferngehalten. Dies ist entscheidend, um Schäden während der Programmierung und der Fehlersuche zu vermeiden. Gleichzeitig ist die sichere und zuverlässige SPI-Kommunikation über mehrere isolierte Kanäle gewährleistet, was die Datenintegrität aufrechterhält und mit den Sicherheits- und Leistungszielen des Projekts übereinstimmt.

Benutzeroberfläche und Interaktion

Die Benutzeroberfläche des ESP32-Energiemessers ist informativ und benutzerfreundlich gestaltet. Als primäre und lokale Anzeige dient ein reaktionsschnelles und klar ablesbares OLED-Display, das an K5 und somit an den I²C-Pins des ESP32 angeschlossen ist. Dieses Display zeigt dem Benutzer alle relevanten Daten in Echtzeit an, einschließlich Kennwerte zur Energieaufnahme und zum Systemstatus.
Zusätzlich zum Hardware-Display umfasst das Projekt einen Webserver, der auf dem ESP32 gehostet ist. Diese Weboberfläche spiegelt die auf dem OLED-Bildschirm angezeigten Daten wider und bietet den Nutzern eine alternative Möglichkeit zur Überwachung ihres Energiebedarfs. Das Entwicklungsteam hat es sich zur Aufgabe gemacht, eine benutzerfreundliche und zugleich detaillierte Web-Oberfläche zu schaffen, die eine barrierefreie und umfassende Datendarstellung gewährleistet. Dieser Ansatz mit zwei Schnittstellen ermöglicht es den Benutzern, sowohl lokal als auch aus der Ferne mit dem Energiezähler zu interagieren, was die Benutzerfreundlichkeit des Systems insgesamt erhöht.

Nächste Schritte und Zukunftspläne

Während das Projekt voranschreitet, wurde das ursprüngliche Platinenlayout an einen PCB-Service verschickt. Nach der Rückkehr der Schaltung wird sich der Schwerpunkt auf die Firmware-Seite des Projekts verlagern. Bei der Entwicklung der Firmware geht es darum, den ESP32 so zu programmieren, dass er die Energieverbrauchsdaten genau verarbeitet und anzeigt, den Webserver verwaltet und eine reibungslose Kommunikation zwischen allen Komponenten gewährleistet.
Für die Zukunft ist geplant, weitere Funktionen zu integrieren, um die Funktionalität des Energiezählers zu verbessern. Dazu könnten gehören:

- Fernüberwachungsfunktionen: Die Nutzer können ihre Energieverbrauchsdaten von jedem beliebigen Ort aus über die Webschnittstelle überprüfen.
- Warnungen und Benachrichtigungen: Implementierung eines Systems, das die Nutzer über ungewöhnliche Muster der Energieaufnahme oder mögliche Systemprobleme informiert.
- Werkzeuge zur Datenanalyse: Integration von Analysetools in die Weboberfläche, um den Nutzern zu helfen, ihre Energieverbrauchstrends zu verstehen und Möglichkeiten zur Verbesserung der Effizienz auszumachen.

Wir bemühen uns um kontinuierliche Innovation, wobei wir uns auch auf das Feedback der Nutzer freuen, um in Zukunft Verbesserungen vorzunehmen. Ziel ist es, nicht nur ein zuverlässiges Instrument zur Energieüberwachung bereitzustellen, sondern den Nutzern auch Einblicke in ihren Energieverbrauch zu geben und so das Bewusstsein um die Effizienz zu fördern.

RG – 230709-02

Haben Sie Fragen oder Kommentare?

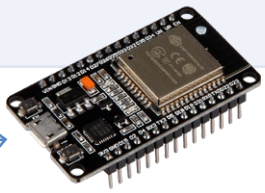
Wenn Sie technische Fragen oder Anmerkungen zu diesem Artikel haben, wenden Sie sich bitte an den Autor unter saad.imtiaz@elektor.com oder an die Elektor-Redaktion unter redaktion@elektor.de.

Über den Autor

Saad Imtiaz (Senior Engineer, Elektor) ist ein Mechatronik-Ingenieur mit fünf Jahren Erfahrung in den Bereichen Embedded Systems, mechatronische Systeme und Produktentwicklung. Er hat mit zahlreichen Unternehmen, von Start-ups bis hin zu Weltkonzernen, bei der Entwicklung von Produktprototypen zusammengearbeitet. Saad hat auch einige Zeit in der Luftfahrtindustrie verbracht und ein Technologie-Startup-Unternehmen geleitet. Seit kurzem ist er bei Elektor tätig und treibt die Projektentwicklung sowohl im Bereich Software als auch Hardware voran.

Passende Produkte

- **ESP32-S3-Entwicklungsboard LILYGO T-Display-S3**
  www.elektor.de/20299
- **ESP-C3-12F-Kit Entwicklungsboard (4 MB Flash)**
  www.elektor.de/19855
- **Joy-IT NodeMCU ESP32**
  www.elektor.de/19973



WEBLINKS

[1] Saad Imtiaz, „Prototyping eines Energiezählers mit ESP32", Elektor Gast-Ausgabe 2023: https://www.elektormagazine.de/230646-02
[2] Mehrphasen-Energiemess-IC ATM90E32AS: https://t1p.de/uaf1f
[3] Digitale Isolatoren ADuM3151 SPIsolator von Analog Devices: https://t1p.de/6sxkn

Projekt-Update #2: Energiemessgerät mit ESP32

Einige Weiterentwicklungen

Bild 1. Rendering des neuen Gehäuseentwurfs des ESP32-Energiezählers.

Von Saad Imtiaz (Elektor)

Im letzten Update dieses Projekts haben wir die Schaltpläne und die Strategien zur Isolationsbarriere des ESP32-Energiezählers besprochen. Es gibt aber weitere Verbesserungen, einen Entwurf für ein Platinenlayout und einiges mehr zu behandeln.

Im Jahr 2023 begannen wir das Projekt mit dem Ziel, einen zuverlässigen und benutzerfreundlichen Energiezähler auf der Basis eines ESP32-Mikrocontrollers zu entwickeln. In unserem letzten Artikel im Januarheft [1] haben wir das Blockschaltbild, die detaillierte Schaltung, die Maßnahmen zur elektrischen Isolation, die Funktionen und die Projektstrategie besprochen. Rekapitulieren wir dies kurz, bevor wir mit dem nächsten Projekt-Update beginnen.

Das wichtigste Ziel bestand darin, einen präzisen und effizienten Energiezähler zu entwickeln, der die Fähigkeiten des ESP32-Mikrocontrollers von Espressif und des ATM90E32AS zur Energiemessung nutzt. Das Projekt sollte durch einen sorgfältigen Schaltungsentwurf nicht nur benutzerfreundlich und zuverlässig sein, sondern durch eine elektrische Trennung der verschiedenen Schaltungsteile in Form eines ADuM3151 eine sichere Kommunikation zwischen dem ESP32 und dem ATM90E32AS von Atmel (jetzt Microchip) gewährleisten. Der Schwerpunkt lag auf Sicherheit und Effizienz durch die Einbeziehung von Techniken zur Rausch- und Störunterdrückung, Verbesserung der Signalintegrität und Schutzmechanismen wie Sicherungen und Varistoren. Durch Einbeziehung zukunftssicherer Funktionen zur Integration von Fernüberwachungs- und Datenanalysetools sollte das Gerät ein verbessertes Energiemanagement ermöglichen und Einblicke in die Effizienz von elektrischen Verbrauchern gewähren.

In diesem Update wollen wir von den Hauptzielen nicht abrücken, aber es wurden zahlreiche Änderungen vorgenommen, um das Projekt sicherer zu machen, die Produktionskosten zu senken und letztendlich die Größe des Geräts zu reduzieren. Bislang sind wir von einer Prototyp-Platine mit den Abmessungen 100 mm × 100 mm ausgegangen. Nach den Tests wurden einige Bauteile entfernt und das Layout der Platine optimiert, so dass die Größe der neuen Version auf 79,5×79,5 mm reduziert werden konnte – und das sind etwa 20 % weniger. In **Bild 1** ist das neue Gehäuse für die neue Platinenversion zu sehen.

Um den ESP32-Energiezähler sicherer zu machen, wird die Schaltung nicht mehr direkt aus der Netzspannung versorgt. Stattdessen reduziert ein Niederspannungstransformator die 230-V-Netzspannung auf ein sicheres Maß von 12 V. Diese Spannung wird sowohl für die Spannungsabtastung als auch für Stromversorgung der Schaltung eingesetzt. Durch den Transformator hat man sich zwar einige Nachteile in Bezug auf die Phasenverzögerung eingehandelt, aber Sicherheit geht vor! Und da es nicht um die Messung von schnellen Spannungsspitzen oder Überspannungen geht, sondern um die Messung von Energie, sollte dies keinen Einfluss auf unsere Messung haben.

Der aktualisierte Schaltplan

Als erste Maßnahme haben wir den ESP32 durch einen ESP32-S3 ersetzt, der mehr Potenzial und erhebliche Verbesserungen für das Energiemeter bereitstellt. Der ESP32-S3 bietet gegenüber seinem Vorgänger ESP32 eine verbesserte Rechenleistung, KI- und Signalverarbeitungsfunktionen, mehr Speicher und bessere Sicherheitsfunktionen. Dazu wurden die Design-Richtlinien von Espressif [2] und andere nützliche Internet-Ressourcen [3…6] genutzt. In **Bild 2** ist der modifizierte Schaltplan des Projekts zu sehen.

Auch das Platinenlayout wurde optimiert, um die Sicherheit, Benutzerfreundlichkeit und Effizienz des ESP32-S3 zu verbessern. Es wurden wesentliche Anpassungen vorgenommen, um die Platine zu verkleinern und dadurch eine kompaktere Grundfläche zu erhalten. Zu den Anpassungen gehört der Gebrauch eines Netztrafos für die Stromversorgung, um die Sicherheit zu erhöhen. Das Gerät ist jetzt für ein- und dreiphasige Messungen vorbereitet und damit viel vielseitiger. Die Verwendung des effizienteren Abwärtswandlers AP63203WU-7 anstelle von Hi-Link-Modulen sowie benutzerfreundlicher USB-C- und Qwiic-Erweiterungsanschlüsse trugen zur Weiterentwicklung des Projekts

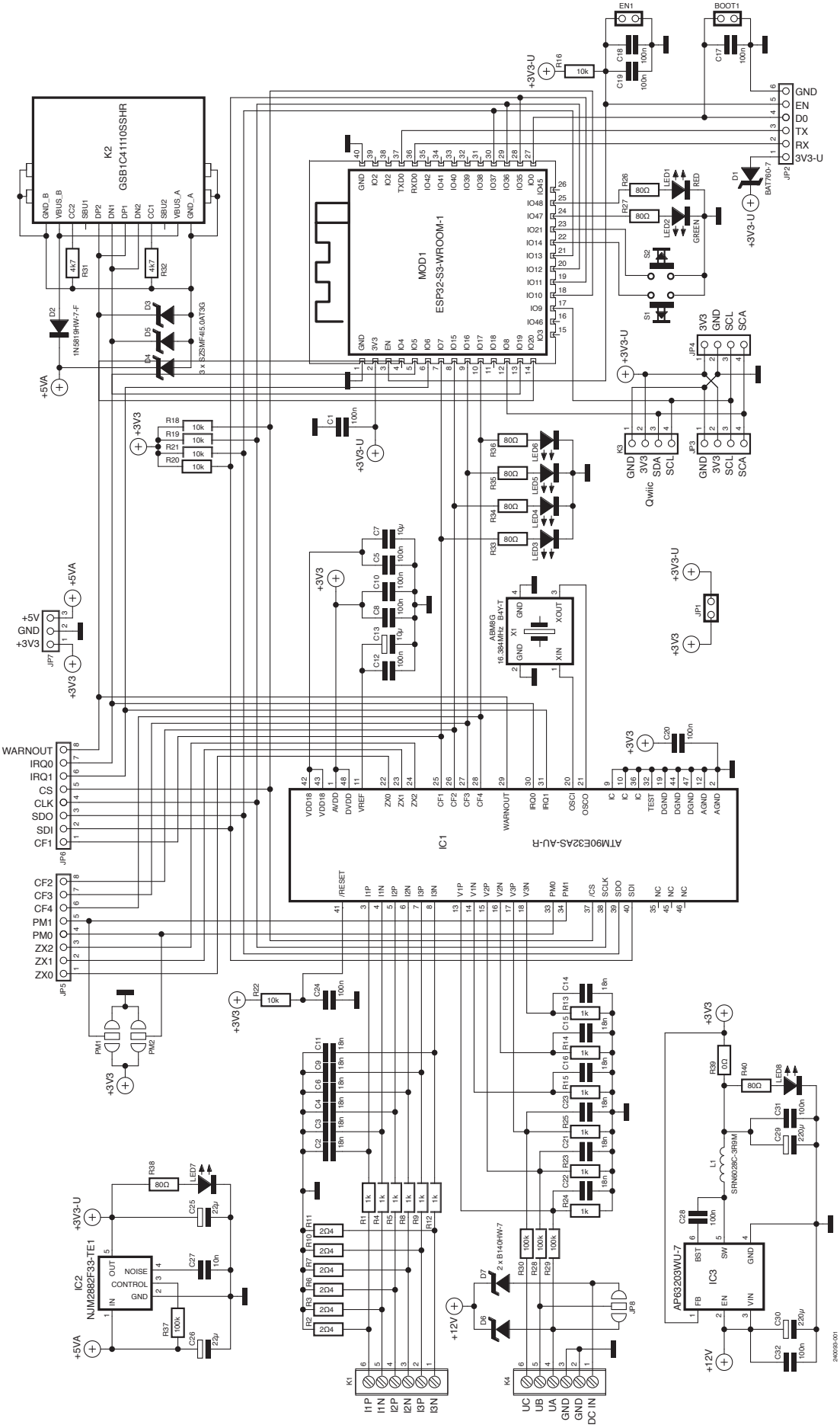


Bild 2. Schaltung des Projekts.

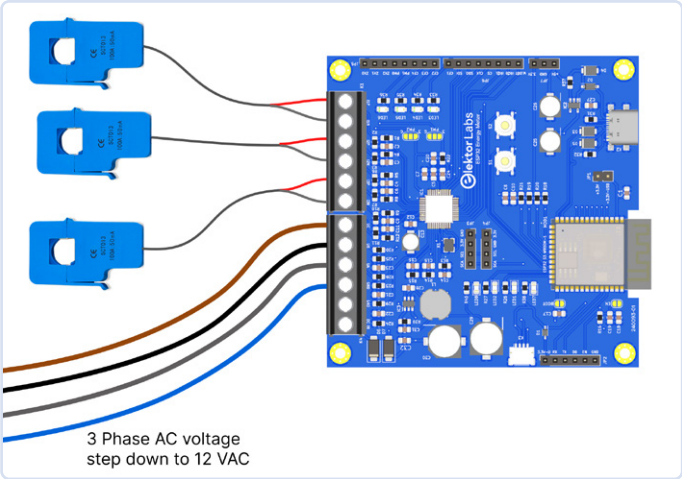


Bild 3. Gesamtverdrahtung eines dreiphasigen Messsystems mit Stromwandlern und dem ESP32-Energiezähler.

bei. Diese Verbesserungen bauen auf den Fähigkeiten des ESP32-S3 auf und konzentrieren sich auf die Bereitstellung einer praktischen, anpassungsfähigen und sicheren Lösung zur Energieüberwachung.

Verfeinerte Spannungs- und Stromabtastung

IC1 ist nach wie vor derselbe ATM90E32AS, aber die Änderung besteht darin, dass sich zwischen der gefährlichen Netzspannung und dem Energie-Monitoring-IC jetzt ein Netztransformator mit einer 12-V-Sekundärwicklung befindet. Durch diese Änderung kann das Projekt sicherer getestet und eingesetzt werden, da 12 V nicht nur ungefährlich sind, sondern der Trafo auch eine galvanische Trennung zur Netzspannung gewährleistet. Bei meinen Tests konnte ich keinen nennenswerten negativen Einfluss des Trafos ausmachen.

Für jeden der Spannungsabtastungseingänge von IC1 gibt es jetzt also nur noch einen 100-kΩ-Widerstand (R28 bis R30). Beim letzten Mal haben wir die Spannungen aller Außenleiter in einem Eingang zusammengefasst, aber viele Leser haben sich die Möglichkeit gewünscht, den Eingang bei Bedarf sowohl in einem dreiphasigen als auch mit einphasigen Netz zu verwenden. Wir haben darüber nachgedacht, und jetzt können wir die Schaltung mit beiden Fällen verwenden. Standardmäßig ist der dreiphasige Modus eingestellt, aber wenn man ihn einphasig machen möchte, muss der Jumper JP8 kurzgeschlossen werden. **Bild 3** zeigt die prinzipielle Verdrahtung für ein dreiphasiges System. Die Spannungen auf den Außenleitern (gegen den blauen Neutralleiter) werden dabei von kleinen (Klingel-) Trafos auf 12 VAC herabgesetzt. Nicht, dass Sie auf dumme Gedanken kommen: Auch der Neutralleiter wird vom Trafo getrennt!

Für die Stromabtastung und -messung wird anstelle der Kopfhörerbuchse der Schraubklemmenblock K1 mit einem Raster von 5,08 mm verwendet. Dies trägt zur allgemeinen Robustheit des Energiemessgeräts bei. Als Stromwandler wurde der Typ SCT013 (100 A / 50 mA) von der Firma YHDC gewählt; die Widerstände R1...R12 an den drei Stromsensoreingängen setzen gemäß Datenblatt den Ausgangsstrom in eine entsprechende Spannung um.

Optimierung der Stromversorgung

Der Energiezähler wird jetzt vom Abwärtsschaltregler IC3, einem AP63203WU-7 von Diodes Incorporated versorgt. Die ursprünglichen Hi-Link-Module HLK5M05 haben sich als viel zu sperrig und viel zu teuer erwiesen. Der Abwärtswandler ist effizienter als die Hi-Link-Module, kostet weniger und ist viel kleiner. Zu Entwicklungs- und Testzwecken kann man den Schaltregler auch mit 12 VDC an K4 versorgen, während im Normalbetrieb die Betriebsspannung von L1 (UA an K4) stammt.

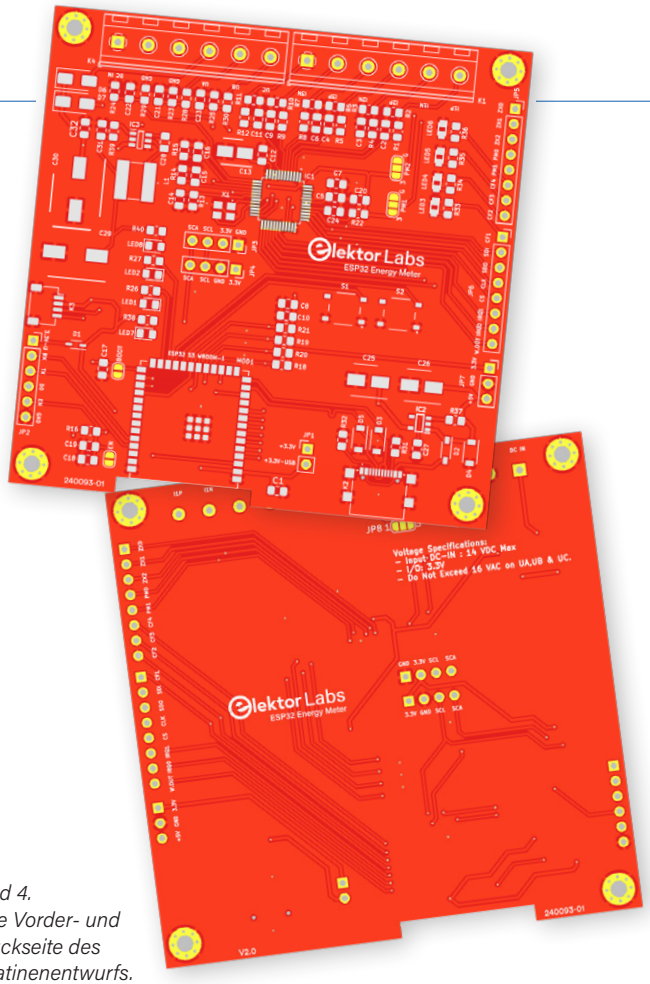


Bild 4. Die Vorder- und Rückseite des Platinenentwurfs.

Interaktive und modulare Funktionen

An die Ausgänge CF1 bis CF4 (für Wirkleistung, Blindleistung, Grundwellenleistung und Harmonische Leistung) wurden LEDs angeschlossen [7][8]. Für die Auswahl des Leistungsmodus von IC1 wurden die Jumper PM1 und PM2 hinzugefügt.

In dieser Version werden alle Ausgangspins des ATM90E32AS auf die Klemmen JP5 und JP6 der MCU zur Verfügung gestellt. Dadurch kann der Energiezähler auch als Modul mit einer anderen MCU verwendet werden, wenn der ESP32-S3 nicht zum Einsatz kommen soll.

Der ESP32-S3 besitzt USB-Funktionaltät, so dass es sehr bequem ist, die MCU auf diese Weise zu programmieren, weshalb wir den USB-C-Anschluss K2 hinzugefügt haben. Für die Fehlersuche wurde der Anschluss JP2 hinzugefügt. Die Status-LEDs LED1 und LED2, die vom ESP32-S3 gesteuert werden, und die Drucktasten S1 und S2, um mit dem OLED-Display zu interagieren, stellen die Benutzerschnittstelle dar. Aufgepasst, bei einigen I²C-OLED-Displays liegt auf Pin 1 Masse, bei anderen ist es die 3,3-V-Versorgung. Mit JP3 und JP4 ist das Energiemesssystem für beide Varianten gerüstet

Schließlich gibt es den Qwiic-Anschluss K3, über den sich die Funktionalität des Energiemessers erweitern lässt, falls man zusätzliche Sensoren oder Module zu diesem Projekt hinzufügen möchte.

Das Platinenlayout

Das Platinenlayout wurde sorgfältig auf Kompaktheit und einfaches Löten hin optimiert, wie in **Bild 4** bewiesen wird. Auf der Oberseite sind die Anschlüsse für die Spannungs- und Strom-Sampling strategisch an einer Stelle positioniert, die eine Integration in ein DIN-Hutschienengehäuse zulässt. Auf der rechten Seite erleichtern Stiftleisten im Raster 2,54 mm den Anschluss eines externen Mikrocontroller-Boards, was einen modularen Aufbau des Messgeräts erleichtert. In der Mitte befindet

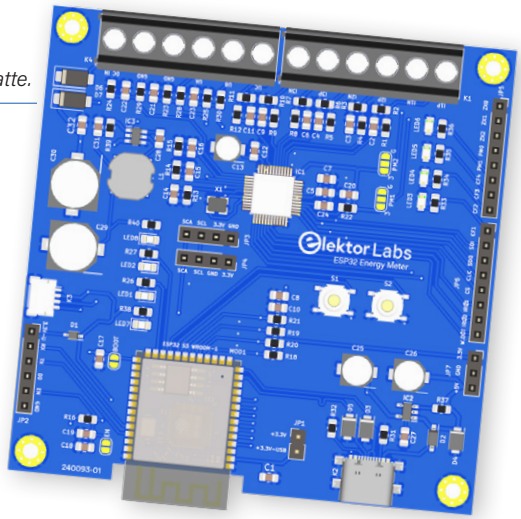


Bild 5. 3D-Modell der bestückten Leiterplatte.

sich der Anschluss für ein OLED-Display, flankiert von Drucktasten für eine intuitive Bedienung. Neben dem OLED-Display sorgen Stromversorgungs- und Status-LEDs für eine unmittelbare visuelle Rückmeldung, während die Energie-Impulsausgangs-LEDs zur direkten Überwachung bequem in der Nähe der MCU-Ausgangsklemmen angeordnet sind.
Der USB-C-Anschluss und das ESP32-S3-Modul sind von den Bereichen mit Wechselspannung weit entfernt, was der Sicherheit zugutekommt. Ein Keramikkondensator neben dem 3-V-Eingang des ESP32-S3 dient der Entkopplung und reduziert eventuelles Rauschen deutlich. Außerdem sind Elektrolytkondensatoren in den Entwurf aufgenommen, die die Stromversorgung weiter stabilisieren und die Zuverlässigkeit und Leistungsfähigkeit der Schaltung gewährleisten. Dieses Layout erleichtert den Montageprozess und verbessert die Funktionalität und Benutzerfreundlichkeit durch eine klare und logische Anordnung der Bauteile. In **Bild 5** sehen Sie das Rendering der bestückten Platine.

Achtung, in diesem Projekt haben Sie es mit gefährlicher Netzspannung zu tun! Personen, die keine Erfahrung mit Netzspannungen haben, sollten dieses Projekt nicht in Angriff nehmen oder jemanden mit Erfahrung fragen, der ihnen bei diesem Projekt helfen kann!

Nächste Schritte, weitere Aussichten

Nach der Prototypen-Phase mit dem ursprünglichen Schaltplan haben wir mehrere Verbesserungen vorgenommen, um die Zuverlässigkeit des ESP32-Energiemessers zu erhöhen. Derzeit konzentrieren wir uns auf die Weiterentwicklung der Firmware.
Das neueste Platinenlayout wurde zur Produktion verschickt, und wir erwarten es bald zurück, um umfangreiche Tests zur Zuverlässigkeit des Systems durchzuführen. Gleichzeitig wird die Softwareentwicklung vorangetrieben, um die Fähigkeiten des ESP32-S3-Moduls in unserem Energiezähler zur Gänze auszuschöpfen.
Für die Zukunft planen wir, den ESP32-Energiezähler in den Home Assistant einzugliedern, um die Benutzerfreundlichkeit zu erhöhen. Nichtsdestotrotz sind wir auch bestrebt, eine maßgeschneiderte Firmware zu entwickeln, um das Potenzial des Geräts voll auszuschöpfen. Zusammenfassend lässt sich sagen, dass das Projekt sowohl mit Hardware- als auch mit Softwareverbesserungen vorankommt. Unser Ziel bleibt es, eine zuverlässige und effiziente Lösung für die Energiemessung bereitzustellen. Dieses Projekt ist auch auf Elektor-Labs [9] zu finden, also zögern Sie nicht, dort Kommentare und Beiträge zu hinterlassen!

RG — 240093-02

Haben Sie Fragen oder Kommentare?

Wenn Sie technische Fragen zu diesem Artikel haben, senden Sie bitte eine E-Mail an den Autor unter saad.imtiaz@elektor.com oder an das Elektor-Redaktionsteam unter redaktion@elektor.de.

Über den Autor

Saad Imtiaz (Senior Engineer, Elektor) ist ein Mechatronik-Ingenieur mit Erfahrung in den Bereichen Embedded-Systems, mechatronische Systeme und der Produktentwicklung. Er hat mit zahlreichen Unternehmen, von Startups bis hin zu Weltkonzernen, bei der Entwicklung von Produktprototypen zusammengearbeitet. Saad hat auch einige Zeit in der Luftfahrtindustrie verbracht und ein Technologie-Startup-Unternehmen geleitet. Seit kurzem ist er bei Elektor tätig und treibt die Projektentwicklung sowohl im Bereich Software als auch Hardware voran.

Passende Produkte

- **Qoitech Otii Arc - Stromversorgung, Leistungsmesser und Datenerfassung**
  www.elektor.de/19270
- **ESP-Terminal**
  www.elektor.de/20526
- **Arduino Nano ESP32**
  www.elektor.de/20562

WEBLINKS

[1] Saad Imtiaz, „Projekt-Update: Energiemessgerät mit ESP32“, Elektor 1/2024: https://www.elektormagazine.de/magazine/elektor-325/62700
[2] Blockschaltbild ESP32 S3 DevKitC-1 (PDF): https://t1p.de/oa83w
[3] ESP32 S3 Pinout Hilfe: https://luisllamas.es/en/which-pins-can-i-use-on-esp32-s3
[4] Schaltplan V2.1ESP32-S3-USB-Bridge (PDF): https://tinyurl.com/usbbridgeschematic
[5] ESP32-S3 Pinbelegung: http://wiki.fluidnc.com/en/hardware/ESP32-S3_Pin_Reference
[6] ESP32-S3: Welche Pins soll ich verwenden?: https://atomic14.com/2023/11/21/esp32-s3-pins.html
[7] Applikationsschrift AN_46103 Poly-Phase Energy Metering IC: https://t1p.de/vyuph
[8] Atmel-Datenblatt M90E32AS: https://www.mouser.de/c/?q=M90E32AS
[9] ESP32-Energiemessgerät auf Elektor Labs: https://elektormagazine.de/labs/esp32-energy-meter-an-open-source-solution-for-real-time-energy-monitoring

Projekt-Update #3: Energie-messgerät mit ESP32

Integration und Test mit Home Assistant

Von Saad Imtiaz (Elektor)

Im letzten Projekt-Update haben Sie von den Verbesserungen des Schaltplans und der Platine des ESP32-Energiezählers erfahren. Dieses Projekt-Update konzentriert sich auf die praktische Umsetzung und Integration der neuen Version. Es erläutert Schritt für Schritt die Einrichtung des Messgeräts mit ESPHome und Home Assistant für eine effektive Energieüberwachung. Außerdem befassen wir uns mit der Kalibrierung des Geräts.

In unserem letzten Artikel [1] haben wir uns auf Verbesserungen des Schaltplans und der Platine des ESP32-Energiemessgeräts konzentriert und dabei Modularität und Sicherheitsfunktionen verbessert. Bevor wir uns mit dem nächsten Projekt-Update befassen, gönnen wir uns einen kurzen Überblick.

Bei den letzten Weiterentwicklungen des ESP32-Energiemessgeräts haben wir ein Upgrade des Mikrocontrollers auf einen ESP32-S3 vorgenommen, der eine höhere Verarbeitungsleistung und einen größeren Funktionsumfang bietet. Mit dem neuen Design wurde die Platine verschlankt und ein Stromversorgungssystem mit Transformatoren eingeführt, das die „Hochspannung" von 230 V von der Elektronik fernhält und sowohl für die Spannungsabtastung als auch für den Betrieb des Systems harmlose 12 V verwendet. Dadurch wurde die Sicherheit erheblich verbessert und gleichzeitig die Flexibilität für ein- und dreiphasige Installationen beibehalten.

Bild 1. Das zusammengebaute ESP32-Energiemessgerät mit seinem OLED-Display und den Statusanzeigen.

Weitere Verbesserungen betrafen die Verwendung eines effizienteren Abwärtswandlers des Typs AP63203WU-7, die Modularisierung der Leiterplatte, die Kalibrierung der Stromwandler-Abtastschaltung und vieles mehr. Dadurch wurden nicht nur die Leistung und die Funktionalität des Energiezählers optimiert, sondern auch die Kosten verringert und die Größe reduziert.

In diesem dritten Projekt-Update gehen wir auf die Schritte ein, die unternommen wurden, um dieses Energiemessgerät in Betrieb zu nehmen, und auf den Weg, den er vom Labortisch bis zum Zählerkasten genommen hat. Darüber hinaus wird erläutert, wie das Gerät eingerichtet und kalibriert und wie es mit ESP Home in den Home Assistant integriert wird, um die gesammelten Daten des Energiemessers anzuzeigen und zu überwachen. In **Bild 1** ist ein Schnappschuss des ESP32-Energiemessers in einem 3D-gedruckten Gehäuse mit OLED-Display zu sehen. Das Bild zeigt auch die Statusanzeigen, anhand der sich die Stromaufnahme in Echtzeit verfolgen lässt.

Zusammenbau

Die neue Platine wurde so entworfen, dass sie kompakter und einfacher zu löten ist. Das Layout weist nun angemessene Abstände um jedes Bauteil auf, was den Lötprozess erleichtert. Um die Verbreitung des Projekts in der Community und Änderungen durch Enthusiasten und Fachleute gleichermaßen zu erleichtern, stehen die komplette Stückliste (BOM) im Mouser-Format und die Produktionsdateien auf dem GitHub-Repository von Elektor Labs [2] bereit.

Für die Spannungs- und Stromabnahmeanschlüsse wurden Schraubklemmen von *CUI Devices* verwendet. Die Qualität dieser Klemmen ist viel höher als die der billigen blauen Klemmen, die man bei den meisten Sensormodulen sieht. Da wir es mit Wechselspannungen

und Energiemessung zu tun haben, sind sichere und zuverlässige Verbindungen unerlässlich.
Die Rauschunterdrückung ist ein wichtiger Aspekt des Platinenlayouts. Durch die Aufnahme von Elektrolyt- und Keramikkondensatoren um den Energiemess-Chip ATM90E32S herum wird sowohl nieder- als auch hochfrequentes Rauschen herausgefiltert, was eine genauere und stabilere Energiemessung gewährleistet. Die Platine ist in **Bild 2** abgebildet.
Wie bereits erwähnt, haben wir uns für die Verwendung eines 12-V-Transformators für die Spannungsabtastung und die Energieversorgung des gesamten Systems entschieden. Einen solchen Transformator zu finden ist einfach und billig, aber die meisten dieser Transformatoren benötigen viel Platz, wenn sie in einem kundenspezifischen Gehäuse verwendet werden, wie **Bild 3** beweist. Daher sollten Sie am besten DIN-Schienen-Transformatoren (zum Beispiel Klingeltrafos) verwenden, um den Aufbau sauberer und sicherer zu gestalten. Solche Transformatoren lassen sich leicht im Netz finden. Außerdem hängt die Genauigkeit der Spannungsmessungen von den Eigenschaften der Transformatoren ab, einschließlich der Genauigkeit ihres Spannungsverhältnisses, der Phasenverschiebung und der Linearität.
Vielleicht haben Sie auf den Bildern bemerkt, dass nur ein Transformator an das Energiemessgerät angeschlossen ist. Der Energiemesser wurde nämlich hier für einphasigen Betrieb konfiguriert (Lötjumper JP8 auf der Rückseite der Platine angebracht, wie in **Bild 4** zu sehen). Um den Energiezähler im dreiphasigen Modus zu betreiben und die Spannung jeder Phase in einem dreiphasigen System mit drei Transformatoren abzufragen, müssen Sie die Primärseiten der drei Transformatoren an die jeweiligen Außenleiter (L1, L2, L3) gegen den Neutralleiter N anschließen. Auf der Sekundärseite schließen Sie ein Ende der Wicklung jedes Transformators an einen gemeinsamen Nullpunkt auf der Platine an und bilden so eine Sternkonfiguration (Y). Die freien Enden der Sekundärwicklungen (V1, V2, V3) liefern dann die Spannungsausgänge (UA, UB und UC) auf der Platine für jede Phase. Unbedingt muss sichergestellt sein, dass die Transformatoren für Netzspannung und den Strom des Systems ausgelegt sind und dass Primär- und Sekundärstromkreise strikt voneinander isoliert sein müssen, aus Sicherheitsgründen und um einen stabilen und ausgeglichenen Neutralleiter-Anschluss und damit eine hohe Messgenauigkeit zu gewährleisten.

Einrichten mit ESPHome und Home Assistant

Als Teil der Entwicklung wird eine spezielle Firmware erstellt, um die Fähigkeiten des Energiemess-Chips und die fortschrittlichen KI-Funktionen des ESP32-S3 zu nutzen.
Obwohl die Entwicklung einer solchen maßgeschneiderten Firmware viel Zeit in Anspruch nimmt und zur Zeit noch im Gange ist, schränkt dies die Verwendbarkeit des Energiemessgeräts nicht ein. Das Gerät kann schon jetzt mit bestehenden Plattformen wie Home Assistant voll funktionsfähig sein und bietet eine sofortige Lösung für die Energieüberwachung. Daher liegt der Schwerpunkt in diesem Artikel auf der Integration des ESP32-Energiemessers in das Smart Home mit Home Assistant [3] und ESPHome [4]. Dieser Abschnitt führt Sie zunächst durch die Einrichtung des Energiemessgeräts in der Home-Assistant-Umgebung, damit Sie seine vollständige Funktionalität nutzen können.

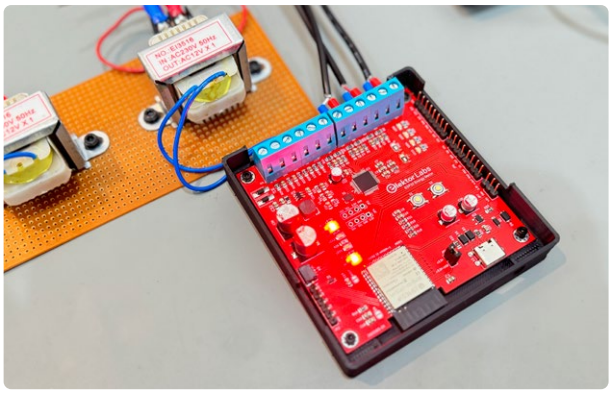


Bild 2. Platine des vollständig montierten ESP32-Energiemessers.

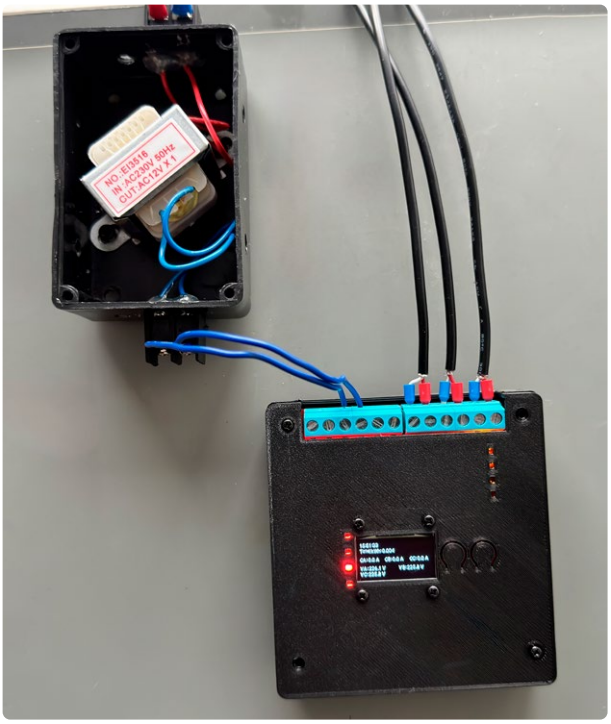


Bild 3. Der Trafo in einem kundenspezifischen Gehäuse verringert 230 V auf 12 V.

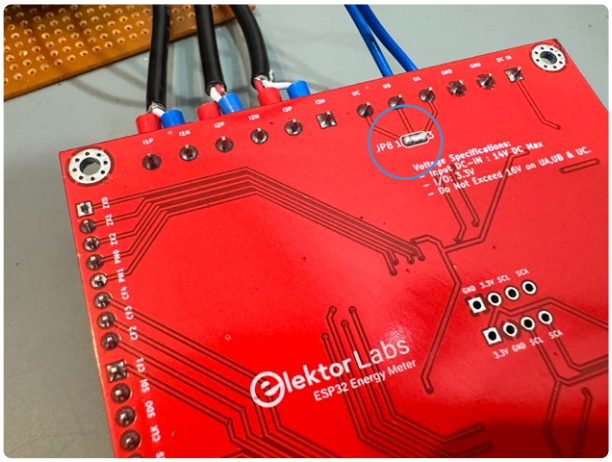


Bild 4. Jumper-Konfiguration auf der Platine für den Betrieb des Energiemessgeräts im Einphasenbetrieb.

Um das ESP32-Energiemessgerät mit der ESPHome-Firmware einzurichten und es in Home Assistant zu integrieren, müssen Sie natürlich zuerst Home Assistant installieren. Eine ausführliche Anleitung dazu finden Sie in einem Artikel meines Elektor-Kollegen Clemens Valens [5]. Ist die Installation gelungen, fügen Sie die ESPHome-Integration aus dem *Add-on Store* hinzu und erstellen ein neues Projekt in ESPHome für Ihr ESP32-Gerät. Dadurch wird automatisch eine grundlegende YAML-Konfigurationsdatei erzeugt, die ein eigenes ESPHome-Projekt mit allen verwendeten Sensoren und vielen anderen Optionen spezifiziert. Sie müssen diese Standarddatei herunterladen, bevor Sie die nächsten Schritte ausführen.

Verbinden Sie Ihren ESP32 mit Ihrem Computer und wählen Sie den richtigen COM-Port für den ESP32-S3 aus. Klicken Sie im ESPHome-Dashboard auf *Install* und wählen Sie die *.bin*-Datei aus, um die Firmware zu flashen.

Sobald die Firmware erfolgreich hochgeladen wurde und Ihr ESP32-Energiemessgerät vom Home Assistant erkannt wird, können Sie die ursprüngliche YAML-Konfiguration bearbeiten. Öffnen Sie dazu das ESPHome-Dashboard in Home Assistant, suchen Sie Ihr Gerät und klicken Sie auf die Option *Edit* auf der Karte des Energiezählers. Ersetzen Sie die vorhandene Konfiguration durch den YAML-Inhalt aus dem GitHub-Repository [2]. Anschließend konfigurieren Sie in dieser neuen YAML-Konfiguration die API-, OTA- und WLAN-Anmeldedaten. Installieren Sie die neue Konfiguration drahtlos auf Ihrem ESP32-Energiemeter. Sobald dies geschehen ist, wird das Gerät aktiv und verbindet sich. Um die Daten des Energiezählers auf Ihrem Home-Assistant-Dashboard anzuzeigen, weisen Sie das ESPHome-Gerät einfach einem bestimmten Bereich in Home Assistant zu. Dies hilft Ihnen, Ihr Dashboard zu organisieren, indem Sie die Geräte nach ihrem

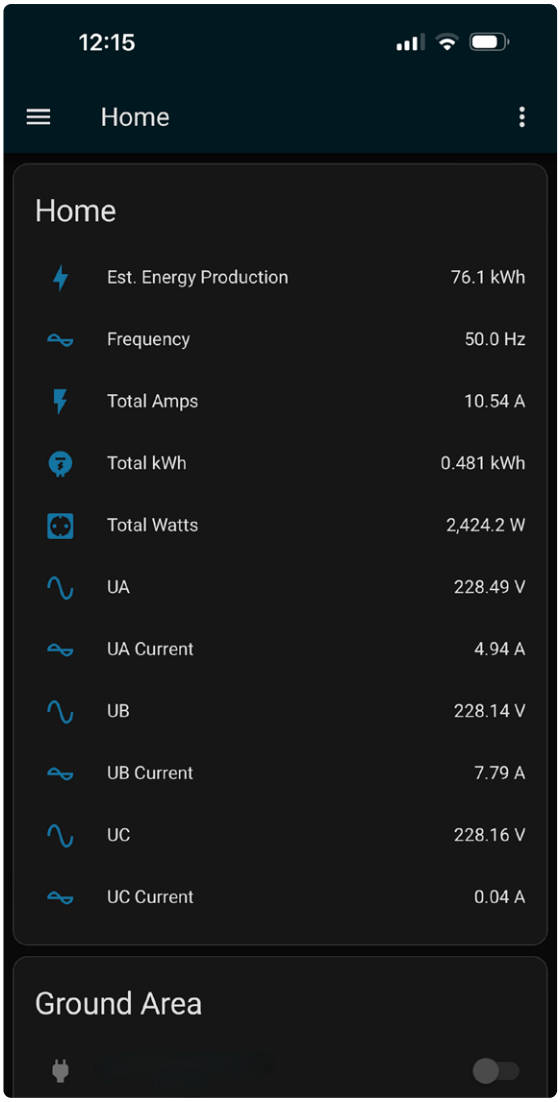


Bild 5. Dashboard des Home Assistant mit Echtzeit-Energiedaten des ESP32-Energiemeters.

Bild 6. Detaillierte Grafiken zum Energieverlauf im Home Assistant.

physischen oder logischen Standort in Ihrem Haus gruppieren. Ein Beispiel, wie eine solche Anzeige der Energiemessdaten auf dem Home-Assistant-Dashboard aussehen könnte, zeigt **Bild 5**.
Die Integration des ESP32-Energiemessgeräts in den Home Assistant vereinfacht nicht nur die Überwachung des Energieverbrauchs, sondern schaltet auch eine Reihe von leistungsstarken Funktionen der Plattform frei. Home Assistant bietet eine intuitive Schnittstelle für die Visualisierung von Daten in Echtzeit, Steuerungsautomatisierung und nahtlose Integration mit anderen intelligenten Geräten in Ihrem Haus.
Diese Integration ermöglicht detaillierte Verlaufsgrafiken und Analysen innerhalb von Home Assistant, die einen detaillierten Einblick in das Muster des Stromverbrauchs im Laufe der Zeit geben, wie in **Bild 6** dargestellt. Dadurch kann man fundierte Entscheidungen über den Energiebedarf treffen, potenzielle Einsparmöglichkeiten ermitteln und die Energieeffizienz des Hauses optimieren.
Mit dieser Einrichtung können die Nutzer alle Vorteile dieser Funktionen nutzen und das ESP32-Energiemeter zu einem zentralen Baustein des Smart-Home-Ökosystems machen. Durch die Integration wird nicht nur die Funktionalität des Energiemessgeräts erweitert, sondern auch das gesamte Smart Home mit umfassenden Energieüberwachungs- und -management-Tools bereichert.

YAML-Konfiguration

Die YAML-Konfiguration richtet das ESP32-Energiemessgerät mit ESPHome ein und ermöglicht die Überwachung wichtiger elektrischer Parameter wie Spannung, Strom und Leistung aller drei Phasen. Sie nutzt die Fähigkeiten des ATM90E32-Sensors, mit detaillierten Definitionen für die SPI-Kommunikation und individuelle Sensoren für jede Phase. Diese Konfiguration misst nicht nur, sondern berechnet auch den Gesamtverbrauch, wobei ein Energiezähler für die Tagesaufnahme in Kilowattstunden und ein OLED-Display zur Visualisierung der Daten in Echtzeit vorhanden sind. Diese Konfigurationen werden gemäß den Anweisungen auf der ESPHome-Seite für den ATM90E32-Sensor [6] vorgenommen.

Bild 7. Aufbau zum Testen und Kalibrieren des ESP32-Energiezählers mit einer variablen Last.

Bild 8. Einsatz des Zangen-Strommessers bei der Kalibrierung.

Damit die vom ESP32-Energiemessgerät gemeldeten Daten ausreichend genau sind, ist eine Kalibrierung unabdingbar. Wie man die Verstärkungseinstellungen für Stromwandler und Spannungseingänge vornimmt, wird im nächsten Abschnitt beschrieben.

Testaufbau und Kalibrierung

Für den Testaufbau wurde ein Haartrockner mit mehreren Heizstufen und Geschwindigkeiten als Last verwendet, der einen Bereich von 0,7...8 A abdeckt. Das Netzkabel einer Verlängerungssteckdose wurde abisoliert, um den klappbaren Stromwandler am stromführenden oder am neutralen Leiter anzubringen, was eine direkte Überwachung unter verschiedenen Bedingungen ermöglichte, wie in **Bild 7** dargestellt.
Ich habe für die Strom- und Spannungskalibrierung des ESP32-Energiemessgeräts mein Digitalmultimeter UT201+ zu Hilfe genommen. Es bietet eine Auflösung von 0,001 A bei einem Fehler von ±4% +10 Digits für den Strom und eine Auflösung von 0,001 V bei einem Fehler von ±1% +5 Digits für die Spannung. Diese Präzision ist für die meisten Projekte ausreichend, aber etwas weniger genau als die meisten professionellen Messgeräte.
Während der Kalibrierung wurde ein Offset der Strommesswerte beobachtet: Der Messwert der Stromzange betrug 1,692 A (siehe **Bild 8**), während die vom Energiemessgerät berechneten Messwerte nach der Kalibrierung 1,70...1,73 A anzeigten (siehe **Bild 9**). In Anbetracht der Spezifikationen des UT201+ und des SCT-013-000, eines Stromwandlers mit geteiltem Kern der Klasse 1 mit einem Fehler von 1% des tatsächlichen Wertes liegt diese kleine Diskrepanz innerhalb der erwarteten Fehlerspanne. Um eine noch höhere Genauigkeit zu erreichen, müsste ein präziseres Zangenmessgerät verwendet werden.
Um eine optimale Messgenauigkeit des ESP32-Energiemeters zu

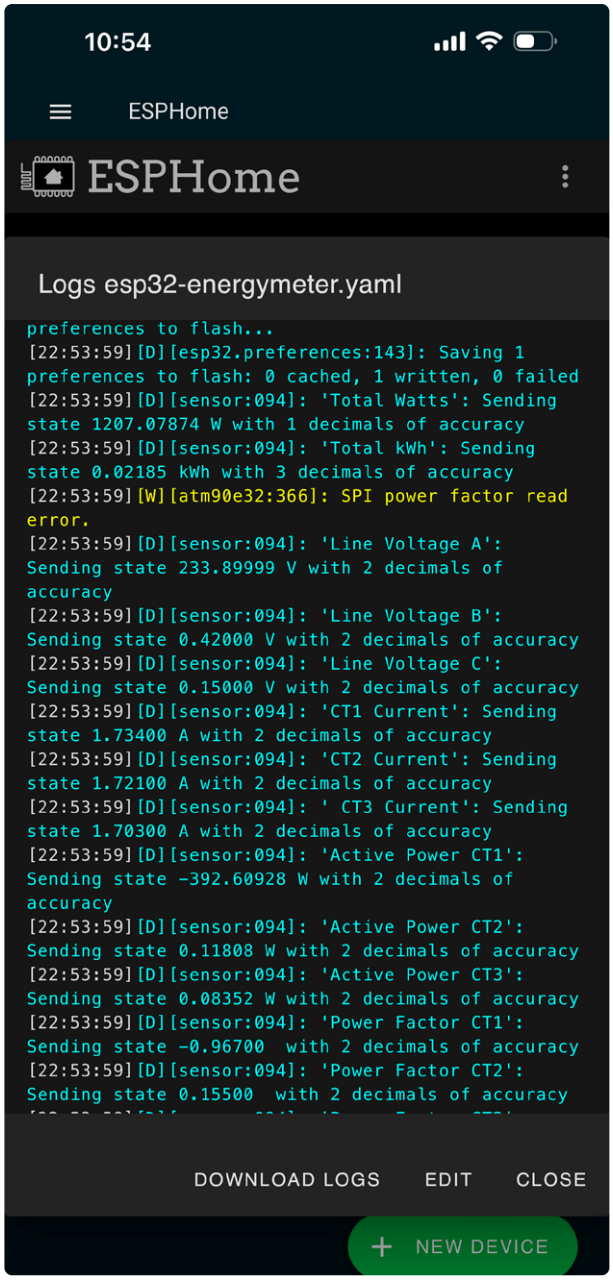


Bild 9. Die endgültigen Ergebnisse der Kalibrierung zeigen eine bessere Genauigkeit der Messung.

WARNUNG: Die Arbeit am und im Zählerkasten birgt Risiken, einschließlich der Gefahr eines elektrischen Schlags oder eines Brandes. Schalten Sie unbedingt den Strom ab, bevor Sie mit der Installation beginnen. In den meisten Ländern darf diese Arbeit ohnehin nur von einem qualifizierten Elektriker durchgeführt werden!

erreichen, wurden die Verstärkungseinstellungen für die Spannungs- und Strommessungen angepasst. Für die Spannung wurde der Sensor mit der folgenden Formel kalibriert:

Neue gain_voltage = (Ihr Spannungsmesswert / ESPHome-Spannungsmesswert) × vorhandener Wert für gain_voltage

Ähnlich verhält es sich mit der Stromeinstellung:

Neue gain_ct = (Ihr aktueller Messwert / ESPHome-Strommesswert) × vorhandener gain_ct-Wert

Diese neuen Verstärkungswerte wurden dann in der YAML-Konfigurationsdatei von ESPHome eingetragen, anschließend neu kompiliert und die Firmware hochgeladen. Dieser Vorgang kann bei Bedarf wiederholt werden, um die Präzision jederzeit „nachzuschärfen". Dies ist für eine genaue Berichterstattung und Analyse in jeder Einrichtung zur Energieüberwachung entscheidend.

Das ESP32-Messgerät im Zählerkasten

Die Installation des ESP32-Energiemessgeräts in meinem Zählerkasten erwies sich zwar als eine überschaubare Aufgabe, die jedoch eine sorgfältige Beachtung der Details erforderte, um sowohl Sicherheit

Bild 10. ESP32-Energiemessgerät, installiert in einer Schalttafel zur Überwachung des Stromverbrauchs in Echtzeit.

als auch Funktionalität zu gewährleisten. Ich begann mit der Auswahl des am niedrigsten abgesicherten Stromkreises, da mir dies einen Sicherheitspuffer bot. Der Schutzschalter würde bei unerwarteten Überspannungen oder Transformatorausfällen auslösen und so das System schützen.

Die Verwendung von klappbaren Stromwandlern war besonders vorteilhaft, weil sie schnell an jede beliebige Last geklemmt werden, aber es war entscheidend, auf die Richtung des Stromflusses zu achten, um genaue Messwerte zu gewährleisten. Wenn die Stromrichtung und der Stromwandler nämlich nicht korrekt ausgerichtet sind, erscheinen die Leistungsmesswerte negativ.

Bild 10 zeigt das in einen Zählerkasten eingebaute ESP32-Energiemessgerät in Aktion mit Anzeige von Strom- und Spannungsmessungen in Echtzeit sowie die entsprechende Last in Kilowatt.

Entwicklung und Aussichten

Während die derzeitige Software auf ESPHome läuft, werden die Möglichkeiten des ESP32-Energiemeters weiter ausgebaut. Die neue Firmware freut sich darauf, in das Projekt integriert zu werden, die speziell dafür entwickelt wurde, das volle Potenzial des ESP32-S3-Chips zu nutzen. Es wird erwartet, dass diese zukünftige Firmware fortschrittliche Funktionen wie detaillierte Energieanalysen und möglicherweise bahnbrechende KI/ML-Funktionen enthält, die Energieverbrauchsmuster vorhersagen und sogar ein Gerät anhand seines Lastprofils identifizieren könnten.

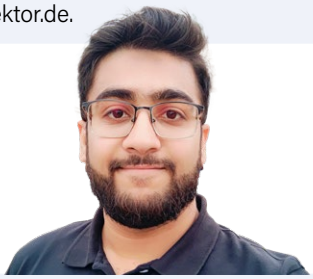
Obwohl die grundlegenden Design- und Betriebsaspekte des Projekts bereits abgeschlossen sind, ist die Entwicklung dieser anspruchsvollen Funktionen ein komplexes und zeitaufwändiges Unterfangen. Ich bin begeistert von den Möglichkeiten und möchte die Grenzen dessen, was dieses Energiemessgeräts erreichen kann, weiter ausreizen.

Das ESP32-Energiemessgerät-Projekt wird kontinuierlich weiterentwickelt und mit jedem Update werden weitere Funktionen hinzugefügt. Maker der Elektor-Community, die sich für die kommenden KI- und ML-Funktionen interessieren oder die zur Entwicklung beitragen möchten, sind herzlich eingeladen, sich zu beteiligen. Die Zusammenarbeit wird dazu beitragen, eine noch robustere und funktionsreichere Energieüberwachungslösung zu entwickeln.

RG — 240244-02

Haben Sie Fragen oder Kommentare?

Wenn Sie Fragen zu diesem Artikel haben, wenden Sie sich bitte an den Autor unter saad.imtiaz@elektor.com oder an die Elektor-Redaktion unter redaktion@elektor.de.



Über den Autor

Saad Imtiaz (Senior Engineer, Elektor) ist Mechatronik-Ingenieur mit Erfahrung in eingebetteten Systemen, mechatronischen Systemen und Produktentwicklung. Er hat mit zahlreichen Unternehmen, von Start-ups bis hin zu Weltkonzernen, bei der Prototypenerstellung und Entwicklung zusammengearbeitet. Saad hat auch einige Zeit in der Luftfahrtindustrie verbracht und ein Technologie-Startup-Unternehmen geleitet. Bei Elektor treibt er die Projektentwicklung sowohl im Bereich Software als auch Hardware voran.

Passende Produkte

- **Stromzange 4350 von PeakTech**
  www.elektor.de/18161
- **Multimeter SDM3045X von Siglent**
  www.elektor.de/17892

WEBLINKS

[1] Saad Imtiaz, „Projekt-Update #2: Energiemeter mit ESP32“, Elektor 5-6/2024: https://www.elektormagazine.de/magazine/elektor-342/62845
[2] Github Repository des Energiemessgeräts mit ESP32: https://github.com/ElektorLabs/esp32-energymeter
[3] Home Assistant: https://home-assistant.io/
[4] ESPHome: http://esphome.io
[5] Clemens Valens, „Hausautomatisierung leicht gemacht“, Elektor 9-10/2020: https://www.elektormagazine.de/magazine/elektor-154/58936
[6] Power Sensor ATM90E32: https://esphome.io/components/sensor/atm90e32.html

Projekt-Update #4
Energiemessgerät mit ESP32

Energieüberwachung mit MQTT

Von Saad Imtiaz (Elektor)

Bisher haben wir uns mit der Einrichtung des ESP32-Energiemessgeräts und dessen Integration in den Home Assistant beschäftigt und auch das zukünftige Potenzial des ESP32-S3-Chips für KI- und ML-Funktionen diskutiert, um Energieverbrauchsmuster vorherzusagen und Verbraucher zu identifizieren. In diesem Update führen wir eine Firmware ein, die eine Echtzeit-Energieüberwachung über MQTT ermöglicht und den Weg für zukünftige Funktionen ebnet.

Die Reise des ESP32-Energiemessgeräts vom Labortisch zum Sicherungskasten führte uns vom Zusammenbau der Komponenten, der Einrichtung des Energiemessgeräts mit ESPHome und dem Home Assistant, über die Kalibrierung und Prüfung bis zum Einbau in den Sicherungskasten. Im letzten Projekt-Update [1] legten wir mit der Integration in Home Assistant den Grundstein, um KI- und ML-Funktionen hinzufügen zu können, um Energieverbrauchsmuster vorherzusagen und Geräte anhand ihrer Energiesignaturen zu identifizieren.

Zugegeben, die KI/ML-Integration ist wegen der umfangreichen Datenaufbereitung noch im Gange, aber mit diesem kleinen Update erreichen wir eine entscheidende Zwischenstufe: die Echtzeit-Energieüberwachung mit MQTT. MQTT ist ein leichtgewichtiges Messaging-Protokoll, das für eine effiziente Kommunikation entwickelt wurde. Einige Grundzüge des Protokolls finden Sie im Textkasten **Was ist MQTT?**.

Benutzerdefinierte Firmware und MQTT

In diesem Artikel besprechen wir die nächste Phase des Projekts, die Nutzung von MQTT und der Arduino-IDE, um die Energieüberwachung in Echtzeit zu ermöglichen. Dieses Update deckt die Firmware-Entwicklung ab, die es dem ESP32 ermöglicht, mit einem MQTT-Broker zu kommunizieren und Energiedaten an einen Home-Assistant-Server oder eine andere MQTT-kompatible Plattform zu senden.

Der Vorteil von MQTT mit einer individuellen Firmware gegenüber ESPHome ist, dass diese individuelle Firmware eine viel größere Flexibilität in Bezug auf Integration und Anpassung bietet. Mit einer benutzerdefinierten Firmware haben Sie die volle Kontrolle darüber, wie

Was ist MQTT?

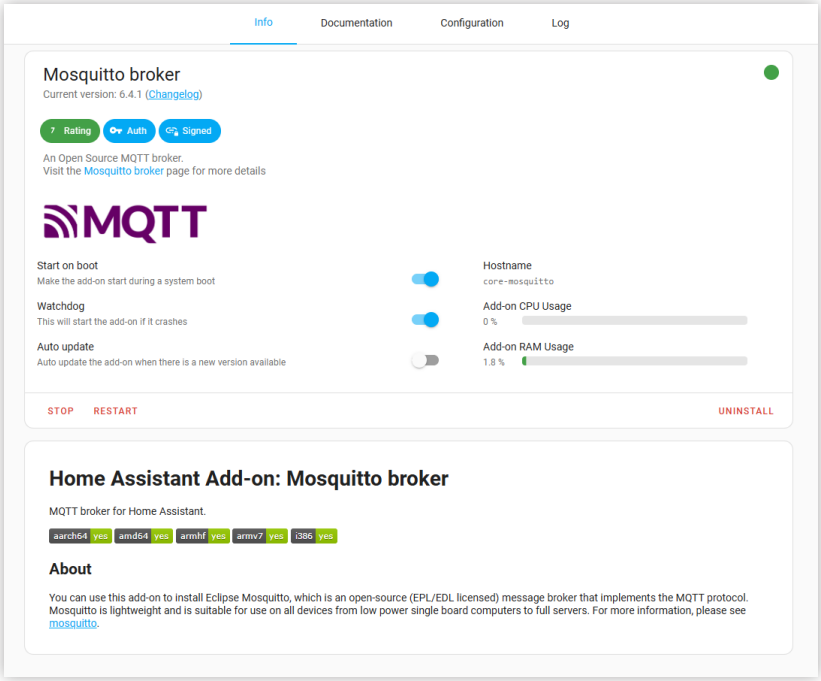
MQTT ist ein leichtgewichtiges Nachrichtenprotokoll für die effiziente Kommunikation zwischen Geräten, insbesondere in IoT-Umgebungen. Das Herzstück dieses Systems ist der MQTT-Broker, ein Server, der als Drehscheibe für den Nachrichtenaustausch dient. Der Broker empfängt Nachrichten von Geräten, den so genannten Publishern, und leitet sie auf der Grundlage eines Systems von Topics an die entsprechenden Subscriber weiter.

Ein Topic (Thema) in MQTT ist eine Zeichenfolge, die Nachrichten kategorisiert und als Kanal fungiert, in dem Informationen veröffentlicht werden, während ein Subscriber (Abonnent) ein Gerät oder eine Anwendung ist, das/die auf bestimmte Topics hört, um diese Nachrichten zu empfangen. In einer Smart-Home-Einrichtung könnte beispielsweise ein Topic wie `home/energy/voltage` Spannungsmesswerte übertragen, und ein Dashboard, das dieses Topic abonniert, würde diese Messwerte in Echtzeit empfangen und anzeigen.

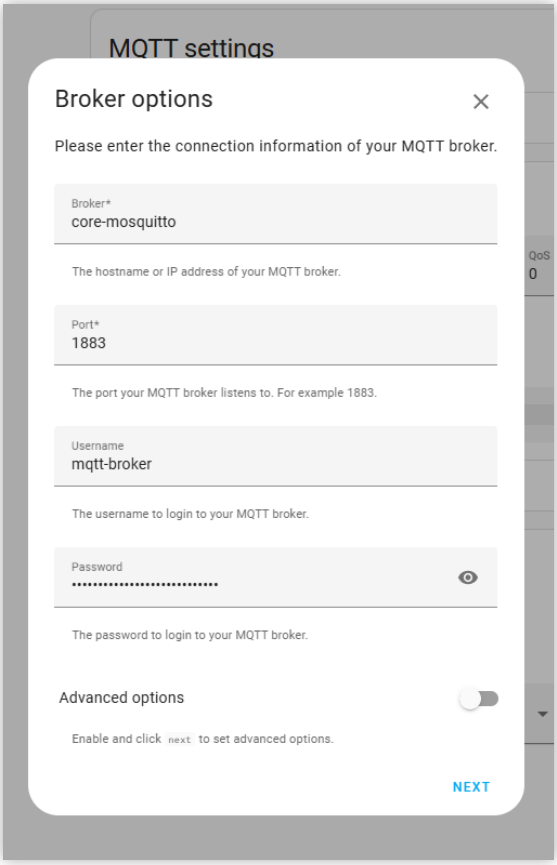
Der Broker sorgt dafür, dass die Nachrichten effizient und sicher zugestellt werden, auch über unzuverlässige Netze. In IoT-Anwendungen ist der MQTT-Broker für die Verwaltung des Datenaustauschs zwischen Sensoren, Geräten und Systemen (den MQTT-Clients) von entscheidender Bedeutung und ermöglicht die Überwachung, Steuerung und Automatisierung in Echtzeit.

Listing 1: Die Firmware (Auszug)

```
#include <WiFi.h>
#include <SPI.h>
#include <ATM90E32.h>
#include <MQTTPubSubClient.h>
#include <config.ino> // Include configuration file for WiFi and MQTT details

// WiFi Credentials
const char* ssid = WIFISSID;       // Your WiFi SSID
const char* pass = WIFIPASSWORD;   // Your WiFi Password

WiFiClient client;
MQTTPubSubClient mqtt;

ATM90E32 energymeter{};

void setup() {

...

  /* Initialize the ATM90E32 energy meter with the specified parameters */
  energymeter.begin(CS_PIN, LINEFREQ, PGA_GAIN, VOLTAGE_GAIN, GAIN_CT_A, GAIN_CT_B, GAIN_CT_C);

...

  /* Begin the WiFi connection using the provided SSID and password */
  WiFi.begin(ssid, pass);

  /* Initialize the MQTT client */
  mqtt.begin(client);

  /* Connect to WiFi, MQTT broker, and Home Assistant */
  connect();

...
 }

void loop() {
  /* Keep the MQTT client updated */
  mqtt.update();

  /* Reconnect to the MQTT broker if the connection is lost */
  if (!mqtt.isConnected()) {
    connect();
  }

  /* Check and send energy data to Home Assistant every 3 seconds */
  static uint32_t prev_ms = millis();
  if (millis() > prev_ms + 3000) {
    prev_ms = millis();
    getEnergyData(); // Retrieve energy data and send via MQTT
  }
}

void getEnergyData() {

  // Retrieve system status from the ATM90E32
  unsigned short sys0 = energymeter.GetSysStatus0();   //EMMState0
  unsigned short sys1 = energymeter.GetSysStatus1();   //EMMState1
  unsigned short en0 = energymeter.GetMeterStatus0();  //EMMIntState0
  unsigned short en1 = energymeter.GetMeterStatus1();  //EMMIntState1

  // Print system and meter status for debugging
  Serial.println("Sys Status: S0:0x" + String(sys0, HEX) + " S1:0x" + String(sys1, HEX));
  Serial.println("Meter Status: E0:0x" + String(en0, HEX) + " E1:0x" + String(en1, HEX));
  delay(10);

  // Check if the MCU is not receiving data from the energy meter
  if (sys0 == 65535 || sys0 == 0) Serial.println("Error: Not receiving data
                                                  from energy meter - check your connections");

  // Retrieve all parameters from the ATM90E32
  float lineVoltageA = energymeter.GetLineVoltageA();
  float lineVoltageB = energymeter.GetLineVoltageB();
  float lineVoltageC = energymeter.GetLineVoltageC();
       ...

  // Send all the collected energy data via MQTT to Home Assistant
  mqtt.publish("esp32energymeter/lineCurrentA", String(lineCurrentA).c_str());
  mqtt.publish("esp32energymeter/lineCurrentB", String(lineCurrentB).c_str());
  mqtt.publish("esp32energymeter/lineCurrentC", String(lineCurrentC).c_str());
  mqtt.publish("esp32energymeter/totalCurrent", String(totalCurrent).c_str());

...
}
```



▲ *Bild 1. Konfigurationsoptionen für den MQTT-Broker im Home Assistant, einschließlich der Option* ***Start on Boot****, die einen automatischen Start auslöst.*



▶ *Bild 2. Einrichten der MQTT-Integration in Home Assistant mit IP-Adresse, Port und Anmeldeinformationen des Benutzers.*

die Daten erfasst, verarbeitet und übertragen werden, so dass Sie das System an die spezifischen Projektanforderungen anpassen können. Dieses Maß an Kontrolle ist besonders vorteilhaft für komplexe Anwendungen, bei denen Sie die Leistung optimieren, nicht standardisierte Hardware integrieren oder erweiterte Funktionen wie KI und maschinelle Lernalgorithmen implementieren wollen. Darüber hinaus ermöglicht die benutzerdefinierte Firmware eine einfachere Integration in eine Vielzahl von Plattformen, die über den Home Assistant hinausgehen, etwa Cloud-basierte Dienste, benutzerdefinierte Dashboards oder andere IoT-Systeme. Sie können auch detailliertere Sicherheitsmaßnahmen implementieren, zum Beispiel benutzerdefinierte Verschlüsselungsprotokolle oder erweiterte Authentifizierungsmechanismen, damit Ihre Daten im gesamten Netzwerk sicher sind. Darüber hinaus kann eine solche Firmware für bestimmte Anwendungsfälle optimiert werden, um Überflüssiges zu eliminieren und so die Systemeffizienz zu verbessern, was besonders in Umgebungen mit eingeschränkten Ressourcen wichtig ist.

Die Software

Die Firmware wurde geschrieben, um den ESP32 mit einem WLAN zu verbinden und MQTT für die Kommunikation zu nutzen, so dass die Energiedaten zu Überwachungs- und Automatisierungszwecken an einen Home-Assistant-Server gesendet werden können. In **Listing 1** ist ein Auszug des Codes zu sehen. Der gesamte Code und alle Hardware-Dateien finden Sie auf dem GitHub-Repository dieses Projekts [2].

Das Projekt stützt sich auf zwei wichtige Bibliotheken: die Bibliothek *ATM90E32* von CircuitSetup [3] wickelt die Kommunikation mit dem Energiemess-IC ab, und die Bibliothek *MQTTPubSubClient* [4], die die Kommunikation als MQTT-Client verwaltet. Die *ATM90E32*-Bibliothek ist für das Sammeln von Daten Spannung, Strom und Leistung vom Energiezählerchip unerlässlich. Während des Entwicklungsprozesses stellte die Integration von MQTT mit Benutzername und Passwort eine große Herausforderung dar. Während viele MQTT-Bibliotheken solch grundlegende Aufgaben bewältigen können, zeichnete sich *MQTTPubSubClient* als eine der wenigen Bibliotheken aus, die mit dem ESP32 kompatibel war und die notwendigen Authentifizierungsfunktionen unterstützte. In der Software werden zunächst die notwendigen Bibliotheken für die Netzwerkkonnektivität, die SPI-Kommunikation, die Verbindung mit dem Energiezähler-IC und die Verwaltung der MQTT-Kommunikation eingebunden. Die Konfigurationsdetails für WLAN und MQTT werden in einer separaten Konfigurationsdatei gespeichert. Die `setup()`-Funktion initialisiert die serielle Schnittstelle für das Debugging, richtet den ATM90E32-Energiezähler mit den angegebenen Parametern ein und stellt die Verbindungen zum WLAN und zum MQTT-Broker her. Dies sorgt dann für eine sichere die Kommunikation des ESP32 mit Home Assistant und anderen MQTT-kompatiblen Plattformen.

In der Hauptschleife `loop()` wird der MQTT-Client ständig aktualisiert, um die Verbindung mit dem Broker aufrechtzuerhalten. Wenn die Verbindung zu irgendeinem Zeitpunkt unterbrochen wird, versucht der Code automatisch, die Verbindung wiederherzustellen. Außerdem ruft die Software die Energiedaten vom ATM90E32-Chip ab und sendet sie an den MQTT-Broker. Auf diese Weise kann der Home Assistant den Energieverbrauch nahezu

Home Assistant: Definieren von MQTT-Daten als Sensoren

Um die von Ihrem ESP32-Energiemessgerät über MQTT gesendeten Daten zu überwachen, müssen Sie diese Datenpunkte als Sensoren in Home Assistant definieren:

Öffnen der Konfigurationsdatei

Öffnen Sie Ihre Home-Assistant-Konfigurationsdatei (*configuration.yaml*) mit dem Datei- oder einem beliebigen Texteditor.

Definieren der MQTT-Sensoren

Fügen Sie in der Datei *configuration.yaml* die folgende Konfiguration hinzu, um Ihre MQTT-Sensoren zu definieren:

```
mqtt:
  sensor:
    - name: Line Voltage A
      unique_id: esp32_voltage_a
      state_topic: "esp32energymeter/lineVoltageA"
      unit_of_measurement: "V"

    - name: Line Current A
      unique_id: esp32_current_a
      state_topic: "esp32energymeter/lineCurrentA"
      unit_of_measurement: "A"
```

Anpassung der Sensoren

Ersetzen Sie `Line Voltage A`, `Line Current A` und so weiter durch Ihnen genehme Namen. Sorgen Sie dafür, dass das `state_topic` mit dem Thema übereinstimmt, das in Ihrer ESP32-Firmware für die Veröffentlichung der Daten verwendet wird. Die `unique_id` sollte jeden Sensor eindeutig bezeichnen, damit der Home Assistant dies richtig verfolgen und verwalten kann.

Speichern und Neustart des Home Assistant

Nachdem Sie die Sensor-Definitionen hinzugefügt haben, speichern Sie die Datei *configuration.yaml* und starten Sie Home Assistant neu, so dass die Änderungen übernommen werden.

Erfreuen Sie sich der Sensor-Anzeige

Nach dem Neustart von Home Assistant sollten Ihre MQTT-Sensoren im Dashboard angezeigt werden. Jetzt können Sie die Energiedaten in Echtzeit überwachen!

in Echtzeit überwachen und wertvolle Erkenntnisse für die Hausautomatisierung liefern.

Die Funktion `getEnergyData()` erfasst die verschiedenen Energiemesswerte vom ATM90E32-Chip wie Netzspannung, Strom, Leistung (Wirk-, Blind- und Scheinleistung), Leistungsfaktor, Phasenwinkel, Frequenz und Temperatur. Die gesammelten Daten werden dann in spezifischen MQTT-Topics veröffentlicht, so dass sie für die Überwachung und Analyse im Home Assistant zugänglich sind. Entwickler und Benutzer können die Energiedaten auch auf dem seriellen Monitor ausgeben, wenn sie Debugging aktivieren, was eine einfache Fehlersuche und Validierung der Daten ermöglicht.

Die Funktion `connect()` sorgt dafür, dass der ESP32 eine stabile Verbindung zum Netzwerk und zum MQTT-Broker aufrechterhält. Diese Funktion kümmert sich um die Wiederherstellung einer unterbrochenen WLAN- oder MQTT-Verbindung. Debugging-Meldungen geben in Echtzeit Auskunft über den Verbindungsstatus, und eine LED zeigt erfolgreiche Verbindungen an.

Einrichten eines MQTT-Brokers im Home Assistant

Damit das ESP32-Energiemessgerät das MQTT-Protokoll zum Senden der Energiemesswerte verwenden kann, müssen Sie zunächst einen MQTT-Broker einrichten. Ein MQTT-Broker kann auf fast jedem Computer eingerichtet werden, der mit Ihrem Heimnetzwerk verbunden ist. Er kann auf Ihrem PC mit einem Docker-Containers konfiguriert, direkt auf einem Raspberry Pi installiert oder sogar auf einem Cloud-Server für den Fernzugriff gehostet werden. Um die Dinge jedoch einfach zu halten und eine nahtlose Integration in Ihre Smart Home-Einrichtung zu ermöglichen, werden wir es hier im Home Assistant einrichten.

Um das MQTT-Add-on im Home Assistant zu installieren, rufen Sie zunächst Ihr Home-Assistant-Dashboard auf, indem Sie zu der URL navigieren, unter der Ihre Instanz läuft. Gehen Sie in der Seitenleiste auf *Settings*, dann auf *Add-ons* und öffnen Sie den *Add-on Store*. Wenn Sie *MQTT* suchen und den offiziellen Mosquitto-Broker in den Ergebnissen finden, klicken Sie darauf und wählen

Bild 3. Testen der MQTT-Verbindung durch Abonnieren aller Themen, damit die Nachrichten von Home Assistant empfangen werden.

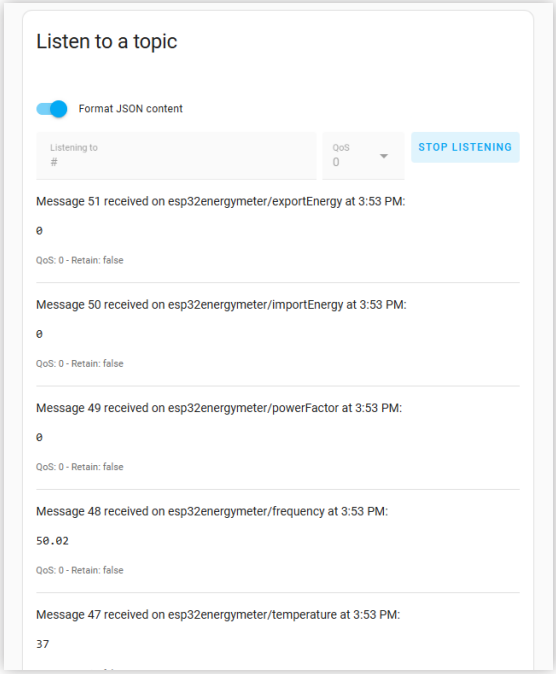
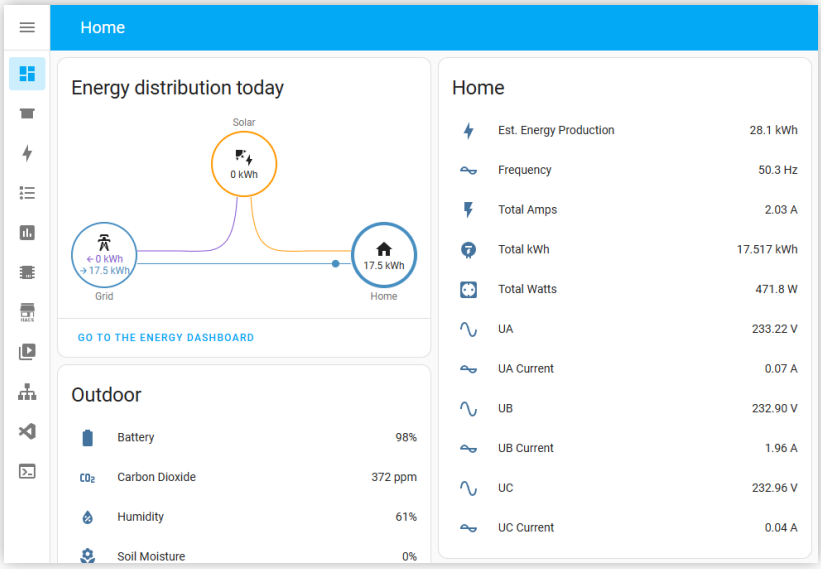


Bild 4. Benutzerdefiniertes Dashboard in Home Assistant, das die Echtzeit-Energiedaten des ESP32-Energiemessgeräts anzeigt.



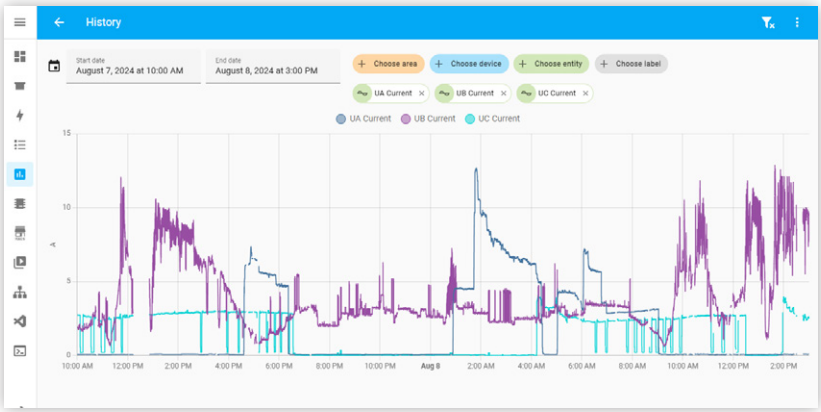


Bild 5. Beispiel für in Home Assistant erstellte Verlaufs-Diagramme zur Überwachung des Energieaufnahme.

Sie dann *Install*. Die Installation kann einige Augenblicke dauern, aber dann können Sie den MQTT-Broker konfigurieren, indem Sie die Konfigurationsoptionen bearbeiten. In der Regel sind die Standardeinstellungen ausreichend, aber Sie können und sollten einen bestimmten Benutzernamen und ein Kennwort festlegen. Klicken Sie nach der Konfiguration auf die Schaltfläche *Start*, um den MQTT-Broker zu aktivieren, und aktivieren Sie *Start on boot option*, damit er bei jedem Neustart von Home Assistant automatisch gestartet wird (siehe **Bild 1**).

Als Nächstes müssen Sie die MQTT-Integration in Home Assistant konfigurieren. Gehen Sie zu *Settings*, dann *Devices & Services* und wählen Sie *Integrations*. Klicken Sie auf *Add Integration*, suchen Sie nach *MQTT*, und wählen Sie es aus. Home Assistant erkennt automatisch den laufenden MQTT-Broker. Wenn Sie dazu aufgefordert werden, konfigurieren Sie die MQTT-Einstellungen wie etwa die IP-Adresse des Brokers - normalerweise *localhost*, wenn er auf demselben Gerät wie Home Assistant läuft, aber Sie können auch *core-mosquito* mit demselben Ergebnis eingeben. Behalten Sie für den Port den Standardwert *1833* bei, und kontrollieren Sie, ob es sich beim Benutzernamen und beim Kennwort um die Anmeldedaten eines aktuellen Benutzers in Home Assistant handelt, wie in **Bild 2** gezeigt. Sie können in diesem Fall auch einen separaten Benutzer in Home Assistant nur für MQTT einrichten.

Jetzt können Sie kontrollieren, ob Home Assistant so eingerichtet ist, dass er Nachrichten von Ihrem ESP32-Energiemessgerät empfangen kann. Abonnieren Sie dazu ein Topic „#“, um alle an Ihren MQTT-Broker gesendeten Nachrichten zu empfangen, wie in **Bild 3** gezeigt. Sie können auf diese Seite zugreifen, indem Sie im Menüpunkt MQTT-Integration auf *Configure in Integration entities* klicken.

Wenn der MQTT-Broker auf Home Assistant läuft, können Sie nun Ihr ESP32-Energiemessgerät anschließen. In der Datei *config.ino* Ihrer ESP32-Firmware setzen Sie die `HOMEASSISTANT_IP` auf die IP-Adresse Ihrer Home-Assistant-Instanz. Konfigurieren Sie dann `DEVICE_NAME`, `USER_ID` und `PASSWORD`, wenn Sie die Authentifizierung auf dem MQTT-Broker eingerichtet haben. Danach können Sie den ESP32 mit der Firmware flashen und kontrollieren, ob er sich erfolgreich mit dem MQTT-Broker verbindet. Sobald die Verbindung hergestellt ist, beginnt das ESP32-Energiemessgerät mit der Veröffentlichung von Energiedaten an den MQTT-Broker, die Sie im Home Assistant überwachen können, indem Sie sich bei den entsprechenden MQTT-Topics anmelden.

Um Energiedaten in Home Assistant anzuzeigen, werden die MQTT-Entitäten automatisch für jedes Topic erstellt, das Ihr ESP32-Energiemessgerät veröffentlicht. Falls nicht, was manchmal vorkommen kann, müssen Sie die MQTT-Entitäten in der Datei *configurations.yaml* in Home Assistant definieren. Sie können den Anweisungen folgen, die unter **Definieren von MQTT-Daten als Sensoren** erwähnt werden.

Danach finden Sie die Sensorentitäten im Home Assistant unter *Settings*, *Devices & Services* und *Entities*. Verwenden Sie diese Entitäten, um in Home Assistant benutzerdefinierte Dashboards zu erstellen, mit denen Sie Energiedaten in Echtzeit anzeigen, Diagramme erstellen und Warnmeldungen auf der Grundlage von Verbrauchsschwellenwerten einrichten können (siehe **Bild 4** und **Bild 5**). Mit MQTT und Home Assistant können Sie auch Aktionen auf der Grundlage von Energiedaten automatisieren (siehe **Bild 6**), mit anderen intelligenten Geräten integrieren und wertvolle Einblicke in den Energieverbrauch Ihres Hauses gewinnen.

Einsteigern empfehle ich die Lektüre des *Getting Started Guide* von Home Assistant [5], der eine umfassende Einführung in die Einrichtung und Verwendung von Home Assistant bietet. Außerdem finden Sie in der Dokumentation zur MQTT-Integration von Home Assistant [6][7] detaillierte Hinweise zur Konfiguration von MQTT und zur effektiven Integration Ihrer Geräte. Diese Ressourcen helfen Ihnen, Home Assistant und MQTT in Betrieb zu nehmen und Ihre Smart Home-Einrichtung effizienter und benutzerfreundlicher zu gestalten.

RG – 240349-02

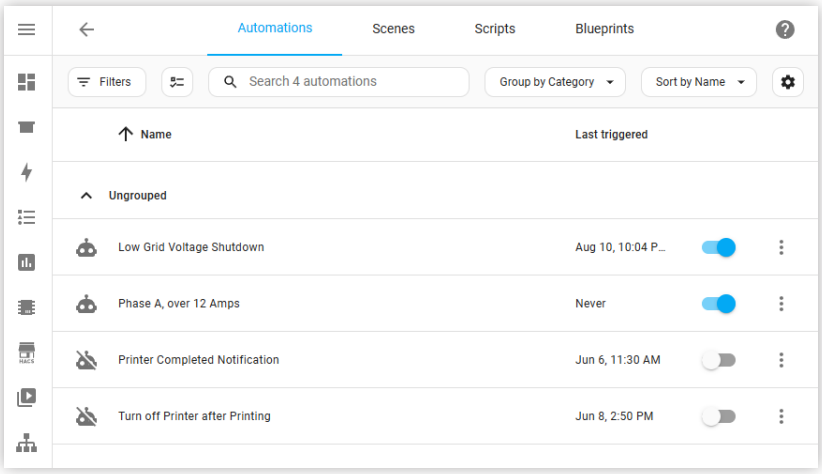


Bild 6. Automatisierte Aktionen in Home Assistant, basierend auf den Energiedaten des ESP32-Energiemessgeräts, die über MQTT empfangen werden.

Über den Autor

Saad Imtiaz, Senior Engineer bei Elektor, ist Mechatronik-Ingenieur mit einer umfassenden Erfahrung in eingebetteten Systemen und Produktentwicklung. In seinem Berufsleben hat er mit einer Vielzahl von Unternehmen zusammengearbeitet, vom innovativen Start-up bis hin zum etablierten Weltkonzern, und zukunftsweisende Prototyping- und Entwicklungsprojekte vorangetrieben. Mit seinem umfangreichen Hintergrund, der eine Tätigkeit in der Luftfahrtindustrie und die Leitung eines Technologie-Startups umfasst, bringt Saad eine einzigartige Mischung aus technischem Fachwissen und Unternehmergeist in seine Rolle bei Elektor ein. Hier trägt er zur Projektentwicklung sowohl im Bereich Software als auch Hardware bei.

Sie haben Fragen oder Kommentare?

Wenden Sie sich bitte an den Autor unter saad.imtiaz@elektor.com oder an die Elektor-Redaktion unter redaktion@elektor.de.

WEBLINKS

[1] Saad Imtiaz, „Projekt-Update #3: Energiemessgerät mit ESP32“, Elektor 7-8/2024: https://elektormagazine.de/240244-02
[2] Github-Repository ESP32 Energy Meter: https://github.com/ElektorLabs/esp32-energymeter
[3] Arduino-Bibliothek ATM90E32 von CircuitSetup: https://github.com/CircuitSetup/ATM90E32
[4] Bibliothek MQTTPubSubClient von hideakitai: https://github.com/hideakitai/MQTTPubSubClient
[5] Einführung in Home Assistant: https://www.home-assistant.io/getting-started/
[6] MQTT-Integration in Home Assistant: https://www.home-assistant.io/integrations/mqtt/
[7] MQTT-Sensor in Home Assistant: https://www.home-assistant.io/integrations/sensor.mqtt/