Περιγραφή Project

καλωσόρισμα

Σκοπός της Συσκευής - 00:30

Η συσκευή που σχεδιάστηκε, έχει ως σκοπό:

- την καταγραφή της κατανάλωσης ρεύματος μίας συσκευής, στην συγκεκριμένη περίπτωση έναν οικιακό server φτιαγμένο από ένα παλιό PC
- την πρόσβαση των δεδομένων που καταγράφτηκαν μέσω internet
- την επιπλέον χρήση του αισθητήρα για την αυτόματη εκκίνηση του server (π.χ. αφού πέσει το ρεύμα, να μην χρειαστεί να πατήσει κάποιος το κουμπί)

Επιλογή Μικροελεγκτή - 01:30

Για project βασισμένα σε αισθητήρες, συνηθίζονται μικροελεγκτές τύπου Arduino, εφόσον έχουν χαμηλό κόστος, μεγάλο εύρος χρήσεων και είναι εύκολα στον προγραμματισμό τους.

Προτιμήθηκε αντί αυτού να χρησιμποιηθεί DevBoard βασισμένο στο ESP32, επειδή:

- Έχει ενσωματωμένο WiFi και Bluetooth, το οποίο θα είναι χρήσιμο στην συγκεκριμένη εφαρμογή
- Ειναι **πολύ** γρηγορότερο από το το ‘κλασσικό’ Arduino, άρα μπορούμε να τρέξουμε πιο περίπλοκο λογισμικό.
- Προγραμματίζεται και αυτό μέσω Arduino IDE, άρα δεν αυξάνεται η δυσκολία μαζί με την απόδοση.
- Περιέργως, το κόστος δεν είναι καν υψηλότερο, με ένα FireBeetle 2-C6 που επιλέχθηκε να είναι φθηνότερο από ένα (αυθεντικό) Arduino. Μόνο μειονέκτημα είναι η κακή ποιότητα Analog to Digital Converter, το οποίο θα αναλυθεί περεταίρω αργότερα.

Επιλογή Αισθητήρα - 2:30

Εφόσον η βασική λειτουργία είναι η μέτρηση κατανάλωσης ενέργειας, είναι σημαντική η επιλογή σωστού αισθητήρα. Τρεις μέθοδοι για την μέτρηση έντασης ξεχώρισαν στην έρευνά μας:

- Αντιστάτης Shunt: ένας αντιστάτης υπερβολικά μικρής αντίστασης που συνδέεται **απευθείας** στο δίκτυο ενέργειας για να μετρηθεί η ένταση που τον διαρρέει. Απορρίπτεται για προφανείς λόγους
- Αισθητήρας Hall Effect: χρησιμοποιεί το ηλεκτρομαγνητικό πεδίο που δημιουργεί το καλώδιο. Καλή επιλογή, και οικονομική, αλλά ενώ ασφαλέστερος από τον αντιστάτη, θέλει πάλι ένωση με το καλώδιο το δικτύου.
- Αισθητήρας CT (Μετασχηματισμού): Επιλέχθηκε επειδή ήταν η ασφαλέστερη επιλογή και λόγω της ευκολία της στην χρήση.

Κύκλωμα Μετασχηματιστή - 3:00

Ο αισθητήρας CT χρησιμοποιέι πρακτικά ένα κύκλωμα μετασχηματιστή, για να δημιουργήσει μία ένταση ανάλογη σε αυτή που μετριέται.

[δείχνει κύκλωμα]

Εάν θυμηθούμε λίγο ηλεκτρικά κυκλώματα, μπορούμε να δούμε πως η ένταση στο δευτερεύον κύκλωμα (του αισθητήρα) θα είναι αντιστρόφως ανάλογη των στροφών του πηνίου του αισθητήρα. Εάν το πηνίο έχει π.χ. 1000 στροφές, μία “τεράστια” για έναν μικροελεγκτή ένταση των 20A, θα γίνει 20mA, πολύ πιο διαχειρήσιμη.

Το πρωτεύον πηνίο είναι απλά το καλώδιο της συσκευής, άρα έχει μία “στροφή”. Εάν το τυλίγαμε δύο φορές θα μετρούσαμε την διπλάσια ένταση κ.ο.κ.

Εξωτερικό ADC

Το ενσωματομένο ADC του ESP32 κατέληξε να δημιουργεί προβλήματα, οπότε αποφασίστηκε να αγορασθεί εξωτερικό ADC, συγκεκριμένα ADS1115, το οποίο εκτός από καθαρότερες μετρήσεις προσφέρει και μεγαλύτερη ανάλυση, 16 bit αντί για 12. Επικοινωνεί με το ESP32 μέσω I2C bus.

Καλάθι αγορών - 3:30

Εδώ φαίνονται τα βασικά εξαρτήματα του project. Από αριστερά προς τα δεξιά έχουμε:

- Αισθητήρας έντασης
- μικροελεγκτής
- ADC

Συναρμολόγηση

Και εδώ φαίνονται όλα μαζί δεμένα σε ένα breadboard. Αν δείτε τυλίξαμε το καλώδιο που μετράμε το ρεύμα πολλές φορές για να αυξήσουμε την ευαισθησία του μετρητή.

Server-side Software

Αφού λοιπόν τον συναρμολογήσουμε, προγραμματίστηκε μέσω Arduino IDE ώστε να απλά επιστρέφει την τωρινή τιμή του αισθητήρα κάθε φορά που του κάνουμε HTTP Request. Ο server ζητά αυτή την τιμή κάθε 2 δευτερόλεπτα.

Με όλες αυτές τις τιμές, εμφανίζεται το πρόβλημα της παρουσίασης αυτών των δεδομένων. Χρησιμοποιήθηκε το λογισμικό Node-RED, που κάνει αρκετά ευκολότερη την διαδικασία, αφού δεν απαιτεί προγραμματισμό η χρήση του.

Server-side Software ii - 4:15

Κάπως έτσι γίνεται το στήσιμο της σελίδας. Χρησιμοποιώντας κόμβους, επεξργάζονται τα δεδομένα και στήνεται στο Dashboard

Server-side Software iii 5:00

Το οποίο μοιάζει κάπως έτσι. Με reverse proxy μπορούμε να το κάνουμε διαθέσιμο και online. Ας το δούμε και ζωντανά

Showcase Εδώ

Ροή Δεδομένων

Συνοπτικά, κάπως έτσι πηγαίνουν τα δεδομένα από τον αισθητήρα στο internet.

Δυσκολίες στην Εκτέλεση

Για να δούμε και που υπήρξαν προβλήματα στην εκτέλεση του project

Ενσωματωμένο ADC - 5:30

Όπως είπαμε και προηγουμένως, το ESP32 μας δημιούργησε προβλήματα λόγω του κακού του ADC.

Συγκεκριμένα, τιμές **κοντά** στο 0 φαινόντουσαν μηδενικές, με αποτέλεσμα να νομίζουμε πως υπήρχε πρόβλημα με τον αισθητήρα, χάνοντας πολύτιμο χρόνο.

Δοκιμάσαμε το εξωτερικό ADC μετά από επικοινωνία με τους κατασκευαστές του αισθητήρα, με την προτασή τους να είναι εύστοχη.

Θα πρότεινα σε όσους κάνουν project με ESP32 να έχουν ένα, μονάχα για να μπορούν να διαγνώσουν τέτοια προβλήματα συντομότερα.

Εμφάνιση Δεδομένων 6:30

Είχαμε πολλές επιλογές για να εμφανίσουμε τα δεδομένα, και ήταν δύσκολο να επιλέξουμε την καταλληλότερη

- Μια επιλογή θα ήταν να φτιάχναμε απλό shell script όπου θα έφτιαχνε ένα απλό αρχείο κειμένου με όλες τις μετρήσεις και θα έβγαζε μέσο όρο, δεν θα ήταν και πολύ ‘όμορφο’ όμως
- Θέλαμε να δοκιμάσουμε το Thingsboard, που φαίνεται από κάτω, γιατί χρησιμοποιείται και από τη βιομηχανία, αλλά θα ήταν πολύ περίπλοκο και δεν μπορούσαμε να το εγκαταστήσουμε, ακόμα και με docker (?!)
- Τελικά το Node-RED ήταν πολύ καλή επιλογή, βοήθησε το γεγονός πως έχει ‘άγνοια’ των δεδομένων, δηλαδή δεν το ένοιαζε αν ήταν μετρητής ενέργειας, απλά διάβαζε αριθμούς, οπότε μπορεί να χρησιμοποηθεί με πρακτικά οτιδήποτε.

Ανακρίβεια Μετρήσεων 7:00

Σημαντική σημείωση είναι πως ο αισθητήρας μετά **μόνο** ένταση.

Θέλουμε όμως να ξέρουμε και **τάση** και **συντελεστή ισχύος**

Υποθέσαμε πως η τάση είναι σταθερή στα 240V (πράγμα πρακτικά αληθές), και πως ο συντελέστής ισχύος είναι ίσος με τη μονάδα, καθώς τα σύγχρονα τροφοδοτικά υπολογιστή κάνουν καλή δουλειά στο να τον κρατάνε υψηλό.

Ερωτήσεις;