ΔΙΕΘΝΕΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΤΗΣ ΕΛΛΑΔΟΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Σύστημα μέτρησης και απεικόνισης λειτουργιών αυτοκινήτου

Του φοιτητή: Μπρεδάκη Ιωάννη

Αρ. Μητρώου: Ele518094

Επιβλέπων
Ονοματεπώνυμο: Δρ.Τσιακμάκης Κυριάκος

Ημερομηνία 19/01/2025

Τίτλος Δ.Ε.: Σύστημα μέτρησης και απεικόνισης λειτουργιών αυτοκινήτου
Κωδικός Δ.Ε.: 23238
Ονοματεπώνυμο φοιτητή: Μπρεδάκης Ιωάννης
Ονοματεπώνυμο εισηγητή: Τσιακμάκης Κυριάκος
Ημερομηνία ανάληψης Δ.Ε.: 05/09/2023
Ημερομηνία περάτωσης Δ.Ε.: 19/01/2025

Βεβαιώνω ότι είμαι ο συγγραφέας αυτής της εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, έχω καταγράψει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών, εικόνων και κειμένου, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επιπλέον, βεβαιώνω ότι αυτή η εργασία προετοιμάστηκε από εμένα προσωπικά, ειδικά ως διπλωματική εργασία, στο Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του ΔΙ.ΠΑ.Ε.

Πρόλογος

Το ταμπλό του αυτοκινήτου αποτελεί ένα από τα πιο σημαντικά σταθερά τμήματα ενός αυτοκινήτου. Καταλαμβάνει ολόκληρο το μπροστινό μέρος στο εσωτερικό του οχήματος και σε αυτό συναντάμε μεγάλο αριθμό οργάνων, διακοπτών και άλλων συστημάτων. Επομένως είναι σημαντικό να επενδύουμε στην εξέλιξή του με σκοπό την καλύτερη εμπειρία του οδηγού. Ένας τρόπος να επιτευχθεί αυτό είναι η αντικατάσταση του κλασσικού ταμπλό από ένα ψηφιακό με τη βοήθεια του αναπτυξιακού συστήματος Arduino, κάτι το οποίο αποτελεί και το σκοπό της παρούσας εργασίας. Στόχος είναι να κατανοηθεί σε βάθος η λειτουργία του Arduino και να αποκτηθεί μεγαλύτερη εξοικείωση με τον προγραμματισμό του.

Ο λόγος που επέλεξα αυτό το θέμα ήταν γιατί θεωρώ σημαντικό να έχει ο οδηγός ενός αυτοκινήτου τη δυνατότητα να αποκτήσει ένα πιο προσωποποιημένο ταμπλό, το οποίο θα του δίνει όλες τις απαραίτητες και αναγκαίες για τον ίδιο πληροφορίες.

Το όφελος από την ολοκλήρωση της εργασίας αυτής ήταν η γνώση που αποκτήθηκε γύρω από το λογισμικό και το υλικό που χρησιμοποιήθηκε αλλά και η γνώση του τρόπου αντιμετώπισης και επίλυσης των διάφορων προβλημάτων που εμφανίζονται κατά τη διάρκεια κατασκευής οποιασδήποτε συσκευής ή κυκλώματος.

Περίληψη

Στην παρούσα εργασία γίνεται η μελέτη και η κατασκευή ενός συστήματος μέτρησης και απεικόνισης λειτουργιών αυτοκινήτου. Η μελέτη περιλαμβάνει την κατανόηση του αναπτυξιακού συστήματος Arduino, τόσο σε επίπεδο υλικού όσο και σε επίπεδο λογισμικού. Ειδικότερα, χρησιμοποιείται ειδικό πρόγραμμα μέσω του οποίου δημιουργείται ο κώδικας και στη συνέχεια φορτώνεται στη συσκευή.

Η κατασκευή ολοκληρώνεται με τη σύνδεση των επιμέρους στοιχείων του συστήματος, δηλαδή ενός Arduino, ενός πολυπλέκτη και 2 οθονών, με τη βοήθεια ενός breadboard. Τέλος, το σύστημα συνδέεται σε αυτοκίνητο μέσω της θύρας OBD που βρίσκεται στο όχημα και μετά την εκκίνηση του λαμβάνουμε τις 4 ενδείξεις που έχουν επιλεγεί στις οθόνες. Οι ενδείξεις αυτές είναι η ταχύτητα του αυτοκινήτου, οι στροφές του κινητήρα, η θερμοκρασία του ψυκτικού υγρού και η τάση της μπαταρίας.

Τα αποτελέσματα που προέκυψαν είναι πολύ ικανοποιητικά καθώς το σύστημα είναι πλήρως λειτουργικό, παρ’ όλες τις δυσκολίες και τα προβλήματα που εμφανίστηκαν κατά τη διαδικασία του προγραμματισμού του.

Car function measurement and display system

Bredakis Ioannis

Abstract

This thesis studies the design and construction of a car function measurement and display system. The study involves understanding the Arduino development system, both in terms of hardware and software. In particular, a specific program is used where the code is generated and then loaded into the device.

The construction is completed by connecting the individual components of the system, which are an Arduino, a multiplexer and 2 displays, by using a breadboard. Finally, the system is connected to the car through the OBD port located on the vehicle and after starting the system we get the 4 values selected on the displays. These values are the speed of the car, the engine RPM, the coolant temperature and the battery voltage.

The results obtained are very satisfactory as the system is fully functional, despite all the difficulties and problems encountered during the programming process.

Εισαγωγή

Το ταμπλό αποτελεί βασικό τμήμα του αυτοκινήτου και είναι αυτό το οποίο μπορεί να δώσει στον οδηγό όλες τις πληροφορίες που χρειάζεται. Ως μοναδικό μέσο σύνδεσης οδηγού-οχήματος είναι αναγκαίο να εξελιχθεί και να δώσει τη δυνατότητα στον οδηγό να επιλέγει ο ίδιος τις ενδείξεις που θα εμφανίζονται.

Σκοπός της εργασίας είναι να μελετηθεί και να κατασκευαστεί ένα σύστημα μετρήσεων των λειτουργιών ενός αυτοκινήτου, το οποίο θα μπορεί να αντικαταστήσει το ταμπλό. Η κατασκευή θα στηρίζεται πάνω στο αναπτυξιακό σύστημα Arduino και θα περιλαμβάνει το σύστημα επικοινωνίας με τον εγκέφαλο του αυτοκινήτου και ένα σύστημα απεικόνισης των ενδείξεων.

Στο 1ο κεφάλαιο δίνονται όλες οι απαραίτητες πληροφορίες για το σύστημα Arduino, καθώς αποτελεί βασικό στοιχείο της συσκευής που υλοποιείται.

Στο 2ο κεφάλαιο παρουσιάζονται τα υλικά που συνέθεσαν το σύστημα μέτρησης και απεικόνισης.

Στο 3ο κεφάλαιο παρουσιάζεται τόσο το λογισμικό που χρησιμοποιήθηκε για τη δημιουργία του κώδικα όσο και τα βασικά κομμάτια του κώδικα.

Στο 4ο κεφάλαιο εξηγούνται οι συνδέσεις που έγιναν για να τεθεί σε λειτουργία το σύστημα.

Στο 5ο κεφάλαιο περιγράφονται τόσο τα προβλήματα που εμφανίστηκαν και ο λόγος που εμφανίστηκαν, όσο και ο τρόπος που αντιμετωπίστηκαν.

Στη συνέχεια αναφέρονται τα συμπεράσματα που προέκυψαν με την ολοκλήρωση της εργασίας.

Τέλος, παρατίθεται ολόκληρος ο κώδικας που χρησιμοποιήθηκε αλλά και ο αντίστοιχος κώδικας για την απεικόνιση των ενδείξεων σε μια μόνο οθόνη.

Περιεχόμενα

Κατάλογος Σχημάτων

Κατάλογος πινάκων

Κεφάλαιο 1ο: Arduino

1.1 Ορισμός και βασικά στοιχεία

Το Arduino είναι ένας μικροελεγκτής ανοικτού κώδικα ο οποίος προγραμματίζεται εύκολα και μπορεί να ενημερωθεί ανά πάσα στιγμή. Το πρώτο Arduino παρουσιάστηκε το 2005 και ο αρχικός σχεδιασμός του είχε σκοπό να εξυπηρετήσει επαγγελματίες και φοιτητές ώστε να μπορούν να αναπτύξουν συσκευές με δυνατότητα αλληλεπίδρασης με το περιβάλλον χρησιμοποιώντας αισθητήρες και ενεργοποιητές.

Βασισμένο σε απλές πλακέτες μικροελεγκτών, αποτελεί μια υπολογιστική πλατφόρμα ανοιχτού κώδικα που χρησιμοποιείται για την κατασκευή και τον προγραμματισμό ηλεκτρονικών συσκευών. Μπορεί να λειτουργήσει ως μίνι υπολογιστής όπως και άλλοι μικροελεγκτές, λαμβάνοντας εισόδους και ελέγχοντας τις εξόδους. Επιπλέον, είναι σε θέση να λαμβάνει και να στέλνει πληροφορίες μέσω του διαδικτύου[1].

Το Arduino χρησιμοποιεί για λογισμικό για την ανάπτυξη του κώδικα το Arduino IDE. Είναι κατασκευασμένο είτε με μικροελεγκτές 8-bit Atmel AVR είτε με έναν 32-bit Atmel ARM, και προγραμματίζεται εύκολα χρησιμοποιώντας τη γλώσσα C ή C++ στο Arduino IDE. Η μεταφόρτωση ενός κώδικα στην πλακέτα Arduino γίνεται με τη χρήση ενός καλωδίου USB[2].

Οι λόγοι που κάνουν το Arduino πολύ διαδεδομένο είναι:

•Ενεργή κοινότητα χρηστών: Καθώς οι χρήστες δουλεύουν με παρόμοιο υλικό, είναι πολύ εύκολο μέσω κοινοτήτων να μοιραστούν πληροφορίες ώστε να λύσουν προβλήματα ή/και ιδέες.

•Ανάπτυξη του Arduino: Όπως αναφέρθηκε νωρίτερα, το Arduino αναπτύχθηκε αρχικά για φοιτητές και επαγγελματίες. Έτσι είναι ένας εύκολος τρόπος για να κάνει κάποιος το ξεκίνημά του.

•Φθηνό υλικό: Πρόκειται για πλατφόρμα ανοικτού κώδικα επομένως δεν αγοράζεται. Το μοναδικό κόστος που προκύπτει είναι η αγορά της πλακέτας. Εναλλακτικά προκύπτει το κόστος της αγοράς των εξαρτημάτων της με σκοπό κάποιος να φτιάξει μόνος του την πλακέτα αφού τα σχέδια του υλικού προδφέρονται χωρίς κάποιο κόστος στο διαδίκτυο.

•Η πλακέτα Arduino ως προγραμματιστής: Είναι εφικτό η πλακέτα να προγραμματιστεί οποιαδήποτε στιγμή και γι' αυτό οι πλακέτες Arduino συνοδεύονται με ένα καλώδιο USB, οπότε καθίσταται πολύ εύκολη η διαδικασία φόρτωσης του κώδικα.

•Περιβάλλον πολλαπλών πλατφορμών: Η κοινότητα με χρήστες Arduino συνεχώς αυξάνεται καθώς το IDE Arduino μπορεί να λειτουργήσει σε διάφορα περιβάλλοντα όπως Microsoft, Linux και Mac OS X [1][3].

1.2 Στοιχεία Arduino

Τα στοιχεία ενός Arduino μπορούν να διαχωριστούν σε 2 κατηγορίες:

- Υλικό

Ο πίνακας ανάπτυξης Arduino αποτελείται από πολλά εξαρτήματα που μαζί τον κάνουν να λειτουργεί. Εδώ είναι μερικά από αυτά τα κύρια συστατικά στοιχεία που βοηθούν στη λειτουργία του:

- Μικροελεγκτής: Αυτή είναι η καρδιά της πλακέτας, η οποία λειτουργεί ως ένας μίνι υπολογιστής. Μπορεί να λαμβάνει καθώς και να στέλνει πληροφορίες ή εντολές στις

περιφερειακές συσκευές που είναι συνδεδεμένες σε αυτήν. Ο μικροελεγκτής που χρησιμοποιείται διαφέρει από πλακέτα σε πλακέτα και έχει επίσης τις δικές του διάφορες προδιαγραφές.

- Εξωτερική τροφοδοσία ρεύματος: Χρησιμοποιείται για την τροφοδοσία της αναπτυξιακής πλακέτας Arduino με μια ρυθμιζόμενη τάση που κυμαίνεται από 9 - 12 Volt.
- Βύσμα USB: Αυτό το βύσμα είναι μια πολύ σημαντική θύρα σε αυτή την πλακέτα. Χρησιμοποιείται για τη μεταφόρτωση ενός προγράμματος στον μικροελεγκτή χρησιμοποιώντας ένα καλώδιο USB. Διαθέτει επίσης ρυθμιζόμενη τάση 5V η οποία τροφοδοτεί επίσης την πλακέτα Arduino σε περιπτώσεις που το εξωτερικό τροφοδοτικό δεν υπάρχει.
- Εσωτερικός προγραμματιστής: Ο κώδικας λογισμικού που αναπτύχθηκε μπορεί να μεταφορτωθεί στον μικροελεγκτή μέσω της θύρας USB, χωρίς εξωτερικό προγραμματιστή.
- Κουμπί επαναφοράς: Αυτό το κουμπί υπάρχει στην πλακέτα και μπορεί να χρησιμοποιηθεί για την επαναφορά του μικροελεγκτή.
- Αναλογικές ακίδες: Υπάρχουν ορισμένες αναλογικές ακίδες εισόδου που κυμαίνονται από A0 - A7 (τυπικά). Αυτές οι ακίδες χρησιμοποιούνται για την αναλογική είσοδο/έξοδο. Ο αριθμός των αναλογικών ακροδεκτών ποικίλλει από πλακέτα σε πλακέτα.
- Ψηφιακές ακίδες I/O: Υπάρχουν επίσης κάποιες ψηφιακές ακίδες εισόδου που κυμαίνονται από 2 έως 16 (τυπικά). Αυτές χρησιμοποιούνται για την ψηφιακή είσοδο/έξοδο. Ο αριθμός αυτών των ψηφιακών ακροδεκτών ποικίλλει από πλακέτα σε πλακέτα.
- Ακίδες τροφοδοσίας και GND: Υπάρχουν ακίδες στην πλακέτα ανάπτυξης που παρέχουν 3.3V, 5V και γείωση[4].

➢ Λογισμικό

Ο κώδικας προγράμματος που γράφεται για το Arduino είναι γνωστός ως σκίτσο. Το λογισμικό που χρησιμοποιείται για την ανάπτυξη τους είναι κοινώς γνωστό ως Arduino IDE. Αυτό το IDE περιέχει τα εξής μέρη:

- Επεξεργαστή κειμένου: Εδώ μπορεί να γραφτεί ο κώδικας χρησιμοποιώντας μια απλουστευμένη έκδοση της C ή C++ γλώσσας προγραμματισμού.
- Περιοχή μηνυμάτων: Εμφανίζει το σφάλμα και δίνει επίσης ανατροφοδότηση για την αποθήκευση και την εξαγωγή του κώδικα.
- Κείμενο: Η κονσόλα εμφανίζει κείμενο που εξάγεται από το περιβάλλον Arduino, και περιλαμβάνει σφάλματα, μηνύματα και άλλες πληροφορίες.
- Γραμμή εργαλείων της κονσόλας: Αυτή η γραμμή εργαλείων περιέχει διάφορα κουμπιά όπως Verify, Upload, New, Open, Save και Serial Monitor. Στην κάτω δεξιά γωνία του παραθύρου εμφανίζεται το Development Board και η σειριακή θύρα που χρησιμοποιείται[4].

1.3 Τύποι Arduino

Στην αγορά υπάρχει μεγάλη ποικιλία πλακετών Arduino, κάθε μία με τα δικά της χαρακτηριστικά. Στη συνέχεια θα γίνει η παρουσίαση ορισμένων τύπων πλακετών Arduino και η αναφορά μερικών άλλων. Καθώς η ανάλυση όλων των τύπων ξεφεύγει από τα όρια της παρούσας εργασίας, έχουν επιλεχθεί οι πιο απλοί τύποι πλακετών, οι οποίες είναι εύκολες στη χρήση, με προσιτή τιμή και ευελιξία, επομένως είναι ιδανικές για εκμάθηση και βασική δημιουργία πρωτοτύπων[1][2][5].

- Arduino UNO R3

Το Arduino UNO είναι η πιο δημοφιλής και ευρέως χρησιμοποιούμενη αναπτυξιακή πλακέτα. Αυτή η πλακέτα βασίζεται στον μικροελεγκτή ATmega328P και διαθέτει 14 ψηφιακές ακίδες εισόδου/εξόδου, 6 αναλογικές εισόδους και συχνότητα ρολογιού 16MHz. Διαθέτει επίσης βασικές θύρες επικοινωνίας όπως SPI, I2C και UART, σύνδεση USB, κουμπί επαναφοράς και υποδοχή τροφοδοσίας. Ο ελεγκτής στην πλακέτα διαθέτει μνήμη flash 32KB (με 0,5KB να χρησιμοποιούνται για τον bootloader), 2KB SRAM και 1KB EEPROM.

Σχήμα 1. 1: Arduino UNO R3 [5]

- Arduino UNO R4 Minima

Το Arduino UNO R4 είναι η τελευταία έκδοση της δημοφιλούς πλακέτας UNO R3 με πολλά βελτιωμένα χαρακτηριστικά. Διαθέτει ισχυρό μικροεπεξεργαστή 32-bit RA4M1 της Renesas με αυξημένη επεξεργαστική ισχύ. Η πλακέτα διαθέτει αυξημένη μνήμη 32KB SRAM, 256KB μνήμη flash και 8KB EEPROM και ταχύτερη συχνότητα ρολογιού 48MHz.

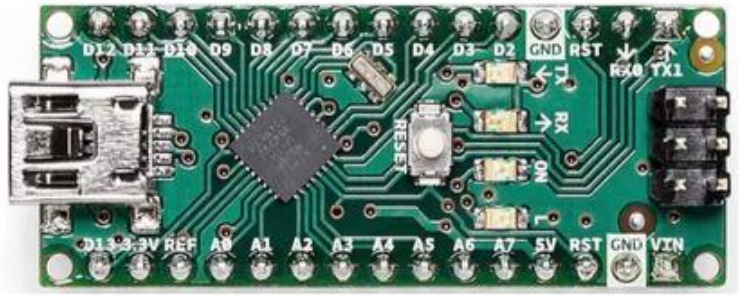
Η πλακέτα διαθέτει 14 ψηφιακές εισόδους/εξόδους και 6 αναλογικές εισόδους. Περιλαμβάνει επίσης μια σειρά από ενσωματωμένα περιφερειακά, όπως ένα 12-bit DAC, CAN BUS και OP AMP. Διαθέτει επίσης ενσωματωμένη υποστήριξη HID (Human Interface Device) που της επιτρέπει την προσομοίωση ποντικιού ή πληκτρολογίου.

Σχήμα 1. 2: Arduino UNO R4 Minima [5]

- Arduino Nano

Το Arduino Nano είναι επίσης μια δημοφιλής πλακέτα μεταξύ των προγραμματιστών λόγω του μικρού μεγέθους του. Βασίζεται στον μικροελεγκτή ATmega328 και έχει παρόμοια χαρακτηριστικά με το Uno, αλλά μικρό μέγεθος σε σύγκριση με την πλακέτα UNO.

Αυτή η πλακέτα διαθέτει 8 αναλογικές ακίδες, 22 ψηφιακές ακίδες εκ των οποίων οι 6 είναι ακίδες PWM. Της λείπει μόνο μια υποδοχή τροφοδοσίας DC και λειτουργεί με καλώδιο Mini USB αντί για τυπικό καλώδιο USB. Είναι καλύτερη για έργα που χρειάζονται λιγότερο χώρο μνήμης και λιγότερες ακίδες GPIO για σύνδεση.

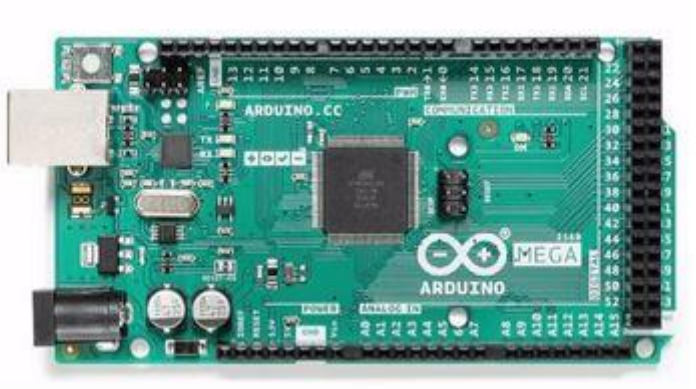


Σχήμα 1. 3: Arduino Nano [5]

- Arduino Mega 2560 Rev3

Το Arduino Mega είναι μια πιο ισχυρή και μεγαλύτερη πλακέτα από τα Uno και Nano. Βασίζεται στον ATmega2560 που είναι ένας μικροελεγκτής 8-bit και διαθέτει 54 ψηφιακές ακίδες εισόδου/εξόδου (εκ των οποίων οι 15 μπορούν να χρησιμοποιηθούν ως έξοδοι PWM), 16 αναλογικές εισόδους, 4 UARTs (σειριακές θύρες υλικού), έναν ταλαντωτή κρυστάλλου 16MHz, μια σύνδεση USB, μια κεφαλίδα ICSP, ένα κουμπί επαναφοράς και μια υποδοχή τροφοδοσίας. Ο ελεγκτής στην πλακέτα διαθέτει 256KB μνήμης flash (εκ των οποίων τα 8KB χρησιμοποιούνται από τον bootloader), 8KB SRAM και 4KB EEPROM.

Είναι ιδανικό για εφαρμογές όπου απαιτούνται πολλά I/O ή περιφερειακά, περισσότερη επεξεργαστική ισχύς και μνήμη. Υποστηρίζει επίσης την κεφαλίδα ICSP, η οποία χρησιμοποιείται για τον προγραμματισμό της πλακέτας χωρίς να την αποσυνδέσετε από το κύριο κύκλωμα.



Σχήμα 1. 4: Arduino Mega 2560 Rev3 [5]

- Arduino Leonardo

Το Arduino Leonardo βασίζεται στον μικροελεγκτή ATmega32U4. Διαθέτει 20 ψηφιακές ακίδες εισόδου/εξόδου, 7 από τις οποίες μπορούν να χρησιμοποιηθούν ως έξοδοι PWM και 12 ως αναλογικές είσοδοι. Διαθέτει επίσης έναν κρυσταλλικό ταλαντωτή 16MHz, μια υποδοχή micro USB, μια κεφαλίδα ICSP και ένα κουμπί επαναφοράς.

Το μοναδικό χαρακτηριστικό του Leonardo είναι η ικανότητά του να εξομοιώνει ένα πληκτρολόγιο ή ένα ποντίκι USB. Αυτό επιτρέπει τη χρήση του σε έργα όπου απαιτείται μια συσκευή εισόδου χρήστη, όπως ένας ελεγκτής παιχνιδιών ή ένας ελεγκτής συντόμευσης πληκτρολογίου. Επιπλέον, διαθέτει ενσωματωμένη επικοινωνία USB, ώστε να μπορεί να λειτουργήσει ως σειριακή συσκευή USB ή συσκευή USB MIDI χωρίς την ανάγκη πρόσθετων προγραμμάτων οδήγησης. Το Leonardo υποστηρίζει επίσης μια σειρά από γλώσσες προγραμματισμού, συμπεριλαμβανομένων των C, C++ και Python.

Σχήμα 1. 5: Arduino Leonardo [5]

- Arduino Micro

Το Arduino Micro είναι παρόμοιο με το Arduino Leonardo. Βασίζεται στον μικροελεγκτή ATmega32U4. Διαθέτει 20 ψηφιακές ακίδες εισόδου/εξόδου (εκ των οποίων οι 7 μπορούν να χρησιμοποιηθούν ως έξοδοι PWM και οι 12 ως αναλογικές είσοδοι), μια σύνδεση micro USB, έναν κρυσταλλικό ταλαντωτή 16 MHz, μια κεφαλίδα ICSP και ένα κουμπί επαναφοράς. Η μόνη διαφορά είναι η έλλειψη υποδοχής εισόδου DC.

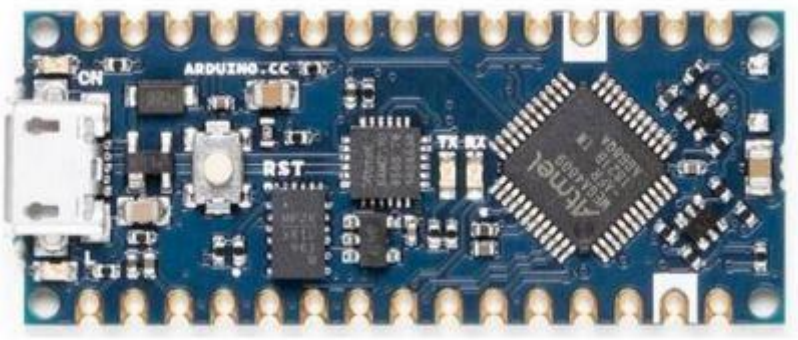
Η πλακέτα έχει σχεδιαστεί για να έχει μικρό μέγεθος και να μπορεί να ενσωματωθεί εύκολα σε διάφορα έργα. Διαθέτει επίσης ενσωματωμένη επικοινωνία USB, η οποία καθιστά εύκολη τη σύνδεση της πλακέτας με έναν υπολογιστή για προγραμματισμό και επικοινωνία. Η πλακέτα Arduino Micro είναι μια δημοφιλής επιλογή για τη δημιουργία μικρών, φορητών έργων που απαιτούν χαμηλή κατανάλωση ενέργειας και συμπαγές μέγεθος.

Σχήμα 1. 6: Arduino Micro [5]

- Arduino Nano Every

Το Arduino Nano Every είναι μια πιο ισχυρή και αναβαθμισμένη έκδοση της δημοφιλούς πλακέτας Arduino Nano, με βελτιωμένη απόδοση και μεγαλύτερο εύρος λειτουργιών. Διαθέτει πολύ ισχυρότερο επεξεργαστή ATMega4809 και έχει 50% περισσότερη μνήμη προγράμματος και η μνήμη RAM είναι 200% μεγαλύτερη που μας επιτρέπει να δημιουργούμε μεγαλύτερα προγράμματα.

Είναι κατάλληλο για φορητές εφευρέσεις και ρομποτική χαμηλού κόστους. Μπορεί να χρησιμοποιηθεί σε breadboard κατά την τοποθέτηση κεφαλίδων ακροδεκτών ή ως SMT απευθείας συγκολλημένο σε PCB.



Σχήμα 1.7: Arduino Nano Every [5]

Στη συνέχεια θα παρουσιαστούν τύποι πλακετών με προηγμένα χαρακτηριστικά και μεγαλύτερη επεξεργαστική ισχύ για πιο σύνθετα έργα. Είναι χρήσιμες σε έργα όπου απαιτούνται προηγμένες λειτουργίες και υψηλότερη απόδοση.

- Arduino Due

Το Arduino Due είναι μια πανίσχυρη πλακέτα που βασίζεται στη CPU Atmel SAM3X8E ARM Cortex-M3. Αποτελείται από 54 ψηφιακούς ακροδέκτες εισόδου/εξόδου (εκ των οποίων οι 12 μπορούν να χρησιμοποιηθούν ως έξοδοι PWM), 12 αναλογικές εισόδους, 4 UART (σειριακές θύρες υλικού), ένα ρολόι 84 MHz, μια σύνδεση USB, 2 DAC (ψηφιακά σε αναλογικά), 2 TWI, μια υποδοχή τροφοδοσίας, μια κεφαλή JTAG, μια κεφαλή SPI, ένα κουμπί επαναφοράς και ένα κουμπί διαγραφής.

Λάβετε υπόψη ότι η πλακέτα Arduino Due λειτουργεί στα 3,3V και η εφαρμογή τάσεων υψηλότερων από 3,3V σε οποιοδήποτε ακροδέκτη I/O θα μπορούσε να προκαλέσει βλάβη στην πλακέτα.

Σχήμα 1. 8: Arduino Due [5]

- Arduino Zero

Το Arduino Zero είναι μια ισχυρή επέκταση 32-bit του Arduino UNO με τον ίδιο τύπο. Αποτελεί εξαιρετική επιλογή για καινοτόμα έργα στον τομέα του IoT, της φορητής τεχνολογίας, του αυτοματισμού υψηλής τεχνολογίας και της ρομποτικής. Τροφοδοτείται από την Atmel SAMD21 MCU και διαθέτει πυρήνα ARM Cortex® M0+ 32-bit με ταχύτητα ρολογιού 48 MHz . Λειτουργεί με τάση 3,3V και διαθέτει 20 ψηφιακές ακίδες I/O, 10 ακίδες PWM, 6 αναλογικές ακίδες εισόδου και 1 αναλογική ακίδα εξόδου.

Ένα από τα χαρακτηριστικά που ξεχωρίζουν είναι ο ενσωματωμένος αποσφαλματωτής της Atmel (EDBG) που παρέχει μια πλήρη διεπαφή αποσφαλμάτωσης χωρίς πρόσθετο υλικό. Η πλακέτα μπορεί να βελτιστοποιηθεί για να μειώσει την κατανάλωση ενέργειας χρησιμοποιώντας τη βιβλιοθήκη Arduino Low Power.

Σχήμα 1. 9: Arduino Zero [5]

- Arduino MKR Zero

Το Arduino MKR Zero είναι μια συμπαγής πλακέτα μικροελεγκτή που βασίζεται στον επεξεργαστή SAMD21G18A 32-bit ARM Cortex M0+. Έχει σχεδιαστεί για να είναι μια εξαιρετικά αποδοτική και χαμηλής κατανάλωσης ενέργειας πλακέτα που είναι ιδανική για ένα ευρύ φάσμα εφαρμογών του Διαδικτύου των Πραγμάτων (IoT).

Η πλακέτα διαθέτει 22 ψηφιακούς ακροδέκτες εισόδου/εξόδου, συμπεριλαμβανομένων 12 εξόδων PWM και 7 αναλογικών ακροδεκτών εισόδου, μια θύρα Micro-USB, SPI, μια διεπαφή UART και μια διεπαφή I2C, που της επιτρέπει να επικοινωνεί με άλλες συσκευές.

Αποτελεί ένα εξαιρετικό εκπαιδευτικό εργαλείο για την εκμάθηση της ανάπτυξης εφαρμογών 32-bit. Ένα άλλο πλεονέκτημα αυτής της πλακέτας είναι ότι μπορείτε να παίξετε με αρχεία MUSIC χωρίς επιπλέον υλικό, καθώς διαθέτει υποδοχή SD με ειδικές διεπαφές SPI (SPI1).

Σχήμα 1. 10: Arduino MKR Zero [5]

Επιπρόσθετα, υπάρχουν οι πλακέτες Arduino IoT. Οι πλακέτες αυτές διαθέτουν ενσωματωμένες μονάδες Wi-Fi, που τους επιτρέπουν να συνδέονται ασύρματα στο διαδίκτυο και σε

τοπικά δίκτυα. Ορισμένες πλακέτες περιλαμβάνουν επίσης δυνατότητες Bluetooth, επιτρέποντας την επικοινωνία με κοντινές συσκευές. Επιπλέον, οι προηγμένες πλακέτες IoT μπορεί να υποστηρίζουν και άλλα πρωτόκολλα όπως το LoRa ή το NB-IoT, τα οποία είναι χρήσιμα για επικοινωνία μεγάλης εμβέλειας και χαμηλής κατανάλωσης ενέργειας.

Αυτές οι πλακέτες μπορούν εύκολα να ενσωματωθούν με διάφορους αισθητήρες και ενεργοποιητές, καθιστώντας απλή τη συλλογή δεδομένων από το περιβάλλον και τον έλεγχο εξωτερικών συσκευών. Επιτρέπουν στους ερασιτέχνες να δημιουργήσουν μια ποικιλία έξυπνων συσκευών, όπως συστήματα οικιακού αυτοματισμού, σταθμούς παρακολούθησης του περιβάλλοντος και τηλεχειριζόμενες μικροσυσκευές. Ας εξερευνήσουμε μερικές δημοφιλείς πλακέτες IoT.

Τέτοιου είδους πλακέτες είναι οι:

- Arduino UNO WIFI Rev2
- Arduino GIGA R1 WiFi
- Arduino Nano 33 BLE
- Arduino UNO R4 WiFi
- Arduino Nano ESP32
- Arduino Nano 33 IoT
- Arduino Nano RP2040 Connect
- Arduino MKR WiFi 1010
- Arduino MKR WAN 1310
- Arduino MKR NB 1500
- Arduino MKR Vidor 4000

Τέλος, υπάρχουν οι πλακέτες Arduino με δυνατότητα AI. Αυτές συνδυάζουν την ευελιξία του Arduino με τη δύναμη της AI, παρέχοντας μια πλατφόρμα για την ενσωμάτωση δυνατοτήτων AI σε διάφορα έργα. Ένα από τα πιο σημαντικά χαρακτηριστικά αυτών των πλακετών είναι η ενισχυμένη επεξεργαστική τους ισχύς. Συχνά είναι εξοπλισμένες με πιο ισχυρούς μικροελεγκτές, όπως αυτοί της σειράς ARM Cortex-M και αυξημένη χωρητικότητα μνήμης.

Επιπλέον, αυτές οι πλακέτες συχνά υποστηρίζουν διάφορες επιλογές συνδεσιμότητας, όπως Wi-Fi, Bluetooth και LoRa. Έρχονται επίσης με υποστήριξη λογισμικού για δημοφιλή πλαίσια και βιβλιοθήκες AI, διευκολύνοντας την ανάπτυξη και την εγκατάσταση εφαρμογών AI. Αυτό τις καθιστά ιδανικές για προγραμματιστές και χομπίστες που θέλουν να εφαρμόσουν μηχανική μάθηση, ανάλυση δεδομένων και έξυπνη αυτοματοποίηση στα έργα τους.

Τέτοιου είδους πλακέτες είναι οι:

- Arduino Nano 33 BLE Sense Rev2
- Nicla Sense ME
- Nicla Voice
- Nicla Vision
- Portenta H7
- Portenta H7 Lite
- Portenta H7 Lite Connected
- Portenta X8
- Portenta C33

Κεφάλαιο 2ο: Υλικό συσκευής (Hardware)

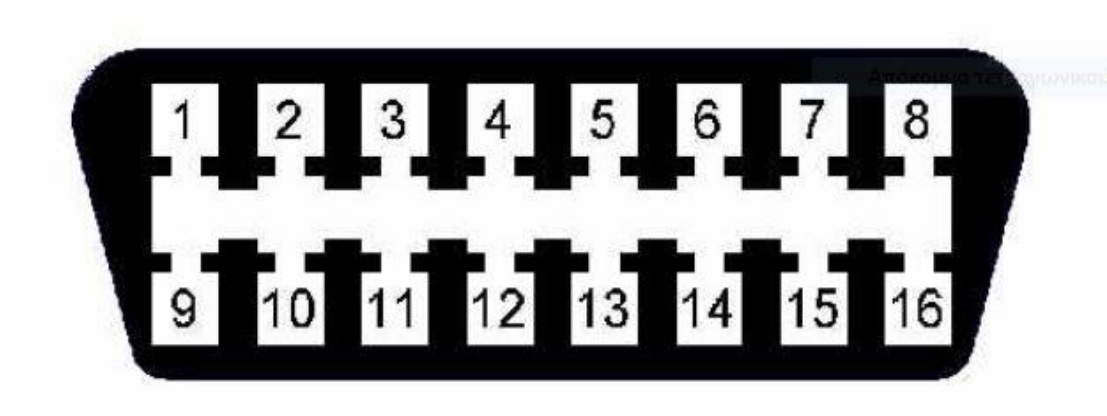
Τα βασικά εξαρτήματα που χρησιμοποιήθηκαν για την κατασκευή του συστήματος είναι τα εξής:

1. ESP32 OBD DEV BOARD
2. 1.3-inch OLED Display I2C SH1106
3. PCA9548A - I2C Multiplexer

Για να γίνει κατανοητή η λειτουργία του κυκλώματος και η επιλογή των συγκεκριμένων στοιχείων θεωρείται απαραίτητη η επεξήγηση τους καθώς και η παρουσίαση των βασικών χαρακτηριστικών τους.

2.1 ESP32 OBD DEV BOARD

Το OBD (On Board Diagnostics) αποτελεί το βασικό πρωτόκολλο που διαθέτουν τα οχήματα με σκοπό να ανακτηθούν πληροφορίες διαγνωστικού ελέγχου. Οι πληροφορίες αυτές δημιουργούνται μέσω της μονάδας ECU που υπάρχει στο όχημα. Συνήθως η θύρα OBDI-16 ακίδων βρίσκεται κάτω από το ταμπλό, στην πλευρά του οδηγού[6].



Σχήμα 2. 1: Θύρα OBD [8]

Κάθε ακίδα αντιστοιχεί στον έλεγχο ορισμένης λειτουργίας.

Pin 1 : Χρησιμοποιείται από τον κατασκευαστή

Pin 2 : Χρησιμοποιείται από SAE J1850 PWM και VPW

Pin 3 : Χρησιμοποιείται από τον κατασκευαστή

Pin 4 : Γείωση

Pin 5 : Γείωση

Pin 6 : Χρησιμοποιείται από ISO 15765-4 CAN

Pin 7 : Η γραμμή K του ISO 9141-2 και του ISO 14230-4

Pin 8 : Χρησιμοποιείται από τον κατασκευαστή

Pin 9 : Χρησιμοποιείται από τον κατασκευαστή

Pin 10 : Χρησιμοποιείται μόνο από SAE J1850 PWM

Pin 11 : Χρησιμοποιείται από τον κατασκευαστή

Pin 12 : Ασύνδετο

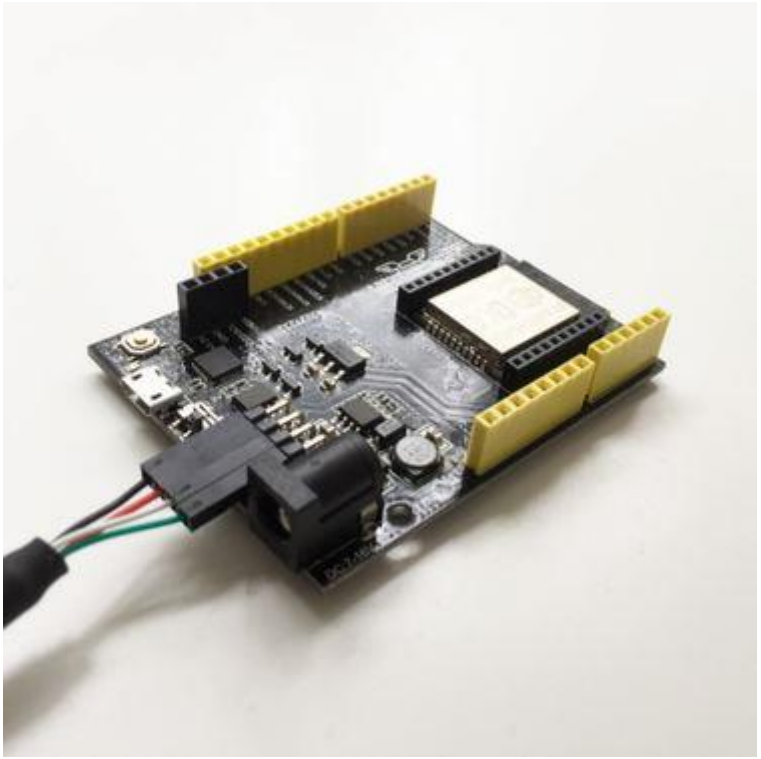
Pin 13 : Χρησιμοποιείται από τον κατασκευαστή

Pin 14 : Χρησιμοποιείται από το ISO 15765-4 CAN

Pin 15 : Η γραμμή L ISO 9141-2 και ISO 14230-4

Pin 16 : Τροφοδοσία από την μπαταρία του αυτοκινήτου

Για τις ανάγκες τις παρούσας εργασίας χρησιμοποιήθηκε το ESP32 OBD DEV BOARD. Πρόκειται για μια πλακέτα ανάπτυξης συμβατή με Arduino. Βασίζεται στο Espressif ESP32 SoC και οδηγεί σχεδόν όλες τις ακίδες I/O από το ESP32 σε τυπικό παράγοντα Arduino[7][8][9].

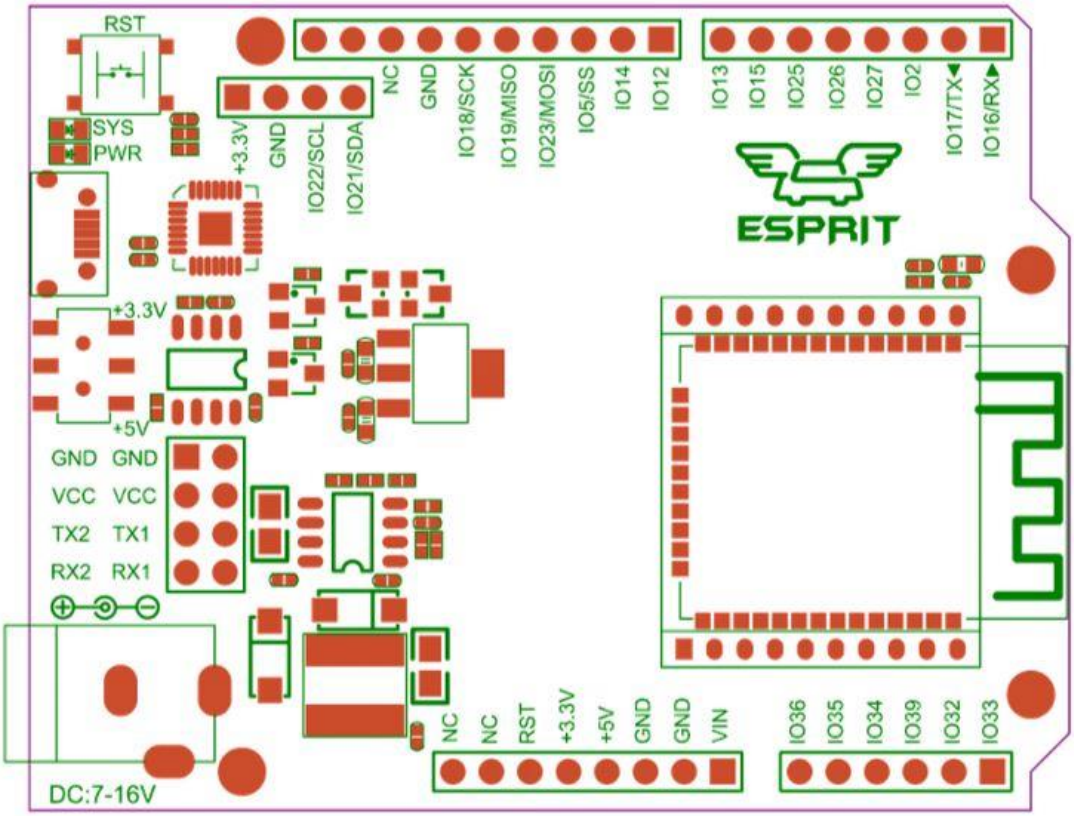


Σχήμα 2. 2: ESP32 OBD DEV BOARD [9]

Τα χαρακτηριστικά που εμφανίζει είναι:

- Συμβατό με τα pinouts του Arduino UNO
- Συχνότητα ρολογιού έως 240MHz
- 520kB εσωτερική SRAM, 4MB εσωτερική Flash
- Ενσωματωμένος πομποδέκτης WiFi 802.11 BGN
- Ενσωματωμένο Bluetooth διπλής λειτουργίας (κλασικό και BLE)
- Ενσωματωμένη υποδοχή xBee με εναλλασσόμενη τάση VCC

- Πρόσθετες διατάξεις ακροδεκτών I2C, 2x σειριακή UART



Σχήμα 2. 3: ESP32 OBD DEV BOARD PCB [9]

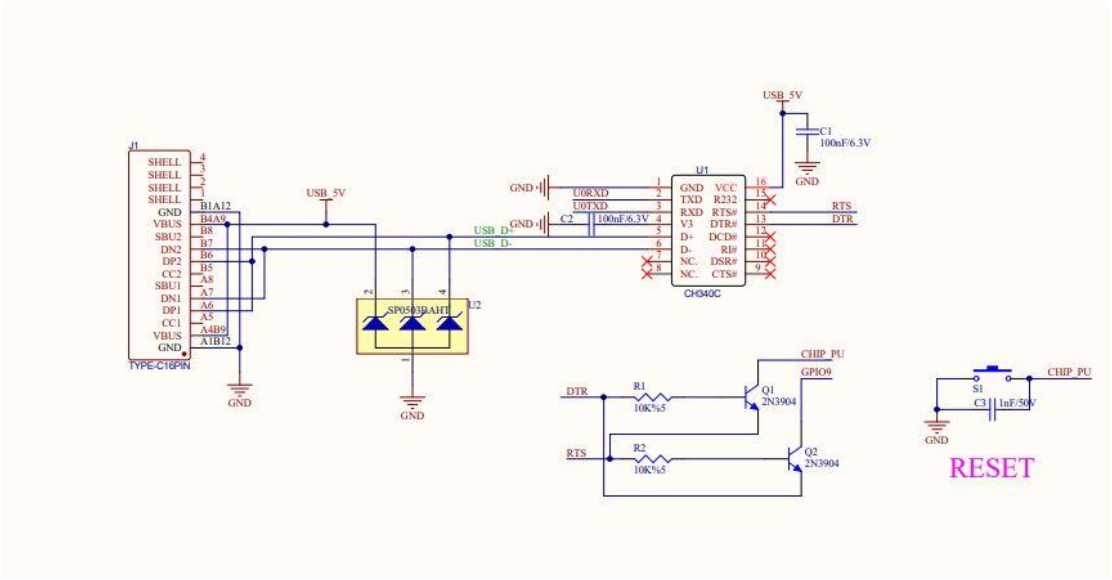
Εκτός από τους τυπικούς ακροδέκτες Arduino, διατίθενται οι ακόλουθες πρόσθετες διεπαφές:

- Κεφαλή 4 ακίδων I2C: SCL(GPIO22), 3.3V, GND
- Κεφαλή σειριακής UART 4 ακίδων #1: Rx(GPIO16), Tx(GPIO17), VCC (3.3V/5V), GND
- Σειριακή κεφαλή UART 4 ακίδων #2: Rx(GPIO32), Tx(GPIO33), VCC (3.3V/5V), GND
- Υποδοχή xBee: Rx(GPIO32), Tx(GPIO33), VCC (3.3V/5V), GND

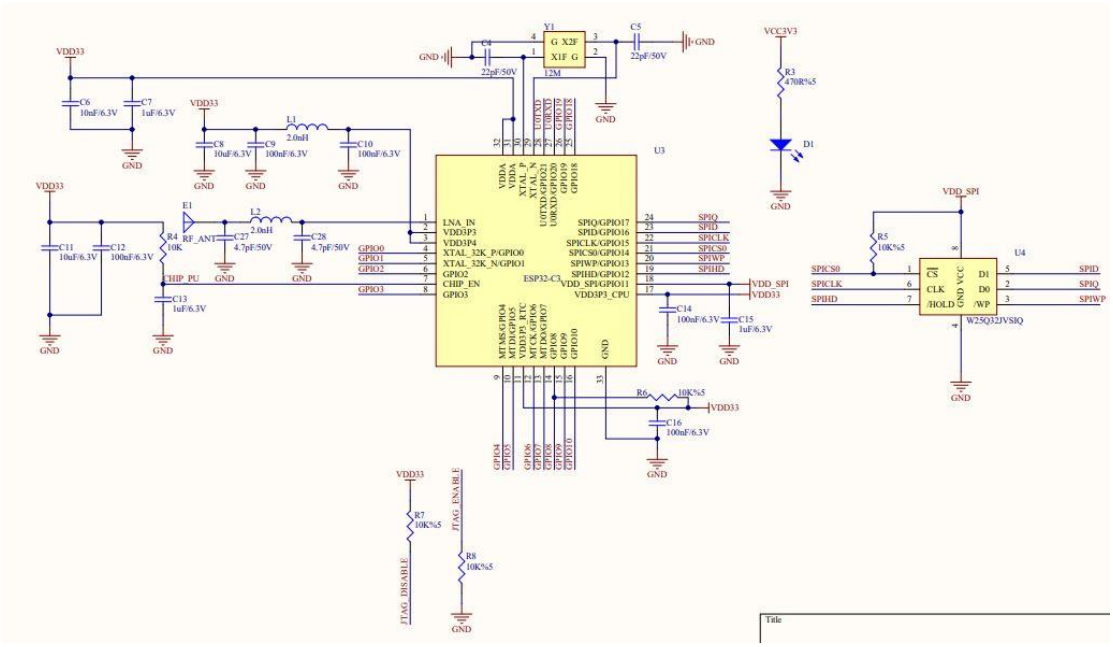
Στην πραγματικότητα, οι ακίδες GPIO του ESP32 είναι πλήρως διαμορφώσιμες, οπότε οι παραπάνω πρόσθετες διατάξεις ακροδεκτών είναι απλώς για προεπιλεγμένη/συνιστώμενη χρήση και κυριολεκτικά μπορούν να αναδιαμορφωθούν για οποιουσδήποτε σκοπούς εισόδου/εξόδου.

Ο διακόπτης VCC εναλλάσσεται μεταξύ 3,3V και 5V για τις ακίδες VCC τόσο της σειριακής κεφαλής UART όσο και της υποδοχής xBee. Οι ακίδες VCC/GND στην κεφαλή UART μπορούν να χρησιμοποιηθούν για την τροφοδοσία της πλακέτας.

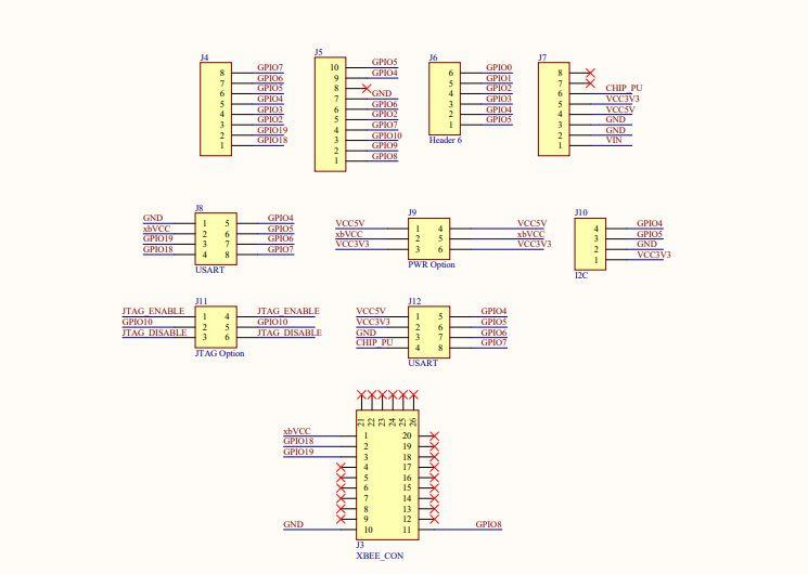
Τα επιμέρους σχηματικά διαγράμματα που απαρτίζουν την πλακέτα δίνονται στη συνέχεια[7].



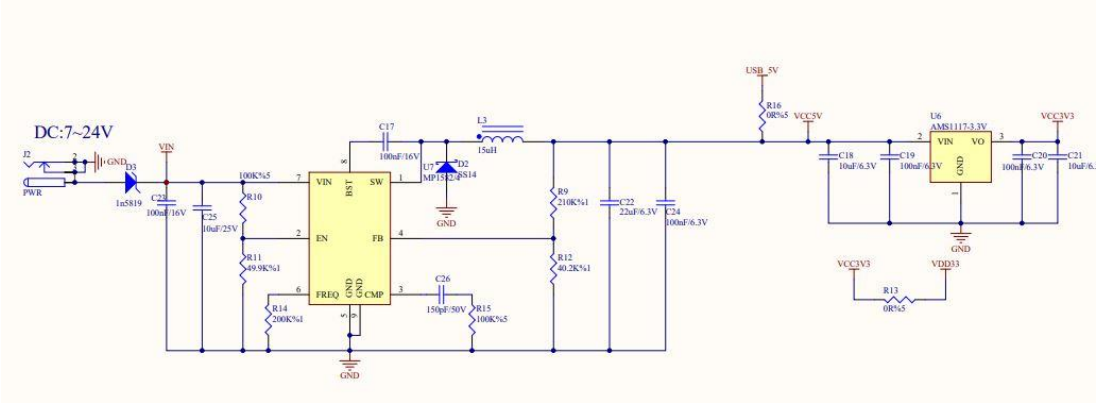
Σχήμα 2. 4: Σχηματικό διάγραμμα ESP32 OBD DEV BOARD [9]



Σχήμα 2. 5: Σχηματικό διάγραμμα ESP32 OBD DEV BOARD [9]



Σχήμα 2. 6: Σχηματικό διάγραμμα ESP32 OBD DEV BOARD [9]



Σχήμα 2.7: Σχηματικό διάγραμμα ESP32 OBD DEV BOARD [9]

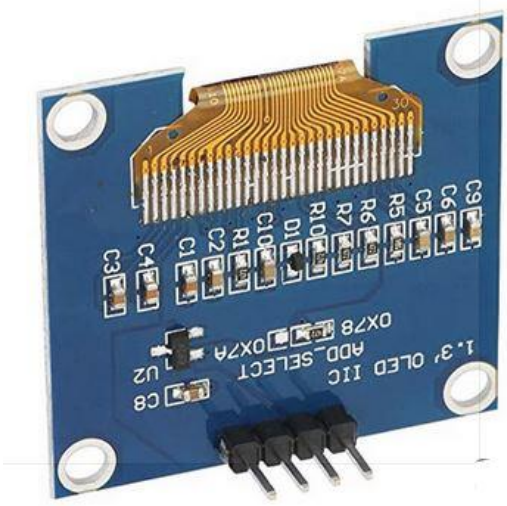
2.2 1.3-inch OLED Display I2C SH1106

Η οθόνη που χρησιμοποιήθηκε για την απεικόνιση των ενδείξεων είναι η 1.3-inch OLED Display I2C SH1106. Λόγω του μεγέθους της και για την καλύτερη απεικόνιση στην πραγματικότητα χρησιμοποιήθηκαν 2 οθόνες[10].

Σχήμα 2.8: OLED display SH1106 (μπροστά όψη)

Τα χαρακτηριστικά της είναι:

- Driver: IC SH1106
- Χρώμα οθόνης: μπλε
- Ανάλυση: 128x64 pixels
- Μέγεθος οθόνης: 1.3 inch.
- Τάση λειτουργίας: DC 3.3V - 5V
- Διάσταση: 35.40 X 33.50 X 2.4mm (L*W*H)
- Διεπαφή επικοινωνίας: lIC(GND.VCC,SCL,SDA)



Σχήμα 2.9: OLED display SH1106 (πίσω όψη)

Οι ακροδέκτες της οθόνης χρησιμοποιούνται για τις παρακάτω λειτουργίες.

Pin VCC: Τάση εισόδου

Pin GND: Γείωση

Pin SCL: Σειριακό ρολόι

Pin SDA: Σειριακά δεδομένα

2.3 PCA9548A - I2C Multiplexer

Όπως αναφέρθηκε νωρίτερα, χρησιμοποιήθηκαν 2 οθόνες. Για να μπορέσουν τα δεδομένα να απεικονιστούν στις οθόνες ήταν απαραίτητη η χρήση πολυπλέκτη ώστε τα δεδομένα να μοιραστούν στις οθόνες. Ο πολυπλέκτης που χρησιμοποιήθηκε είναι ο PCA9548A. Με τα οκτώ κανάλια του, το PCA9548A καθιστά δυνατή τη λειτουργία οκτώ συσκευών I2C με την ίδια διεύθυνση σε έναν δίαυλο I2C[11].

Σχήμα 2. 10: Πολυπλέκτης PCA9548A

Περαιτέρω τεχνικά χαρακτηριστικά του πολυπλέκτη:

- Τροφοδοσία: 2.3 - 5,5 βολτ
- Μέγιστη συχνότητα διαύλου I2C: 400 kHz
- Καρφίτσα επαναφοράς χαμηλής ενεργότητας
- Όλες οι είσοδοι ανεκτικές στα 5V
- Πολλαπλά ή όλα τα κανάλια μπορούν να ενεργοποιηθούν ταυτόχρονα

Ακολουθεί πίνακας με την επεξήγηση των ακροδεκτών του πολυπλέκτη.

PIN			TYPE	DESCRIPTION
NAME	TSSOP, VSSOP (PW, DGS)	VQFN (RGE)		
A0	1	22	I	Address input 0. Connect directly to V_{CC} or ground
A1	2	23	I	Address input 1. Connect directly to V_{CC} or ground
A2	21	18	I	Address input 2. Connect directly to V_{CC} or ground
GND	12	9	—	Ground
$\overline{RESET}$	3	24	I	Active-low reset input. Connect to V_{CC} or V_{DPUM} (1) through a pull-up resistor, if not used
SD0	4	1	I/O	Serial data 0. Connect to V_{DPU0} (1) through a pull-up resistor
SC0	5	2	I/O	Serial clock 0. Connect to V_{DPU0} (1) through a pull-up resistor
SD1	6	3	I/O	Serial data 1. Connect to V_{DPU1} (1) through a pull-up resistor
SC1	7	4	I/O	Serial clock 1. Connect to V_{DPU1} (1) through a pull-up resistor
SD2	8	5	I/O	Serial data 2. Connect to V_{DPU2} (1) through a pull-up resistor
SC2	9	6	I/O	Serial clock 2. Connect to V_{DPU2} (1) through a pull-up resistor
SD3	10	7	I/O	Serial data 3. Connect to V_{DPU3} (1) through a pull-up resistor
SC3	11	8	I/O	Serial clock 3. Connect to V_{DPU3} (1) through a pull-up resistor
SD4	13	10	I/O	Serial data 4. Connect to V_{DPU4} (1) through a pull-up resistor
SC4	14	11	I/O	Serial clock 4. Connect to V_{DPU4} (1) through a pull-up resistor
SD5	15	12	I/O	Serial data 5. Connect to V_{DPU5} (1) through a pull-up resistor
SC5	16	13	I/O	Serial clock 5. Connect to V_{DPU5} (1) through a pull-up resistor
SD6	17	14	I/O	Serial data 6. Connect to V_{DPU6} (1) through a pull-up resistor
SC6	18	15	I/O	Serial clock 6. Connect to V_{DPU6} (1) through a pull-up resistor
SD7	19	16	I/O	Serial data 7. Connect to V_{DPU7} (1) through a pull-up resistor
SC7	20	17	I/O	Serial clock 7. Connect to V_{DPU7} (1) through a pull-up resistor
SCL	22	19	I/O	Serial clock bus. Connect to V_{DPUM} (1) through a pull-up resistor
SDA	23	20	I/O	Serial data bus. Connect to V_{DPUM} (1) through a pull-up resistor
VCC	24	21	Power	Supply voltage

Σχήμα 2. 11: Ακροδέκτες πολυπλέκτη PCA9548A[7]

Για τη σύνδεση των παραπάνω εξαρτημάτων αλλά και τη διασύνδεση με το αυτοκίνητο χρησιμοποιήθηκαν επιπλέον:

- Ένα breadboard
- OBD II αντάπτορας
- Καλώδιο USB (για τη φόρτωση του κώδικα)
- Βραχυκυκλωτήρες (για τη σύνδεση στο breadboard)

Το συνολικό κόστος της συσκευής φαίνεται στον παρακάτω πίνακα:

Πίνακας 2.1: Πίνακας κόστους κατασκευής

ΕΞΑΡΤΗΜΑ	**ΤΙΜΗ**
ESP32 dev board	30€
2 x 1.3-inch OLED Display I2C SSH1106	13€
PCA9548A - I2C Multiplexer	3,5€
Breadboard	12€
OBD II αντάπτορας	75€
Καλώδιο USB	2€
Βραχυκυκλωτήρες	Ελάχιστο κόστος
ΣΥΝΟΛΟ	135,5€

2.4 Σύγκριση ESPrit με Arduino UNO

Όπως αναφέρθηκε και νωρίτερα, το ESP32 dev board είναι συμβατό με τα pinouts του Arduino UNO. Το ESPrit (μια παραλλαγή του ESP32) και το Arduino Uno είναι δύο πολύ διαφορετικές πλατφόρμες μικροελεγκτών που έχουν σχεδιαστεί για διαφορετικές εφαρμογές. Ακολουθεί μια αναλυτική σύγκριση[1][4][7]:

Οι βασικές διαφορές που εμφανίζουν τα ESPrit και Arduino Uno είναι:

1. Επεξεργαστική Ισχύς

ESPrit: Διαθέτει διπλό πυρήνα (dual-core) επεξεργαστή Xtensa LX6 στα 240 MHz.

Πολύ πιο ισχυρό από το Arduino Uno, κατάλληλο για επεξεργασία δεδομένων και απαιτητικές εφαρμογές (π.χ., IoT, επεξεργασία σημάτων).

Arduino Uno: Διαθέτει μονό πυρήνα ATmega328P στα 16 MHz.

Κατάλληλο για βασικές εφαρμογές όπως έλεγχος LED, αισθητήρων ή μοτέρ.

2. Μνήμη

ESPrit: Flash: 4-16 MB (ανάλογα με το μοντέλο).

RAM: 520 KB SRAM.

Πολύ μεγαλύτερη μνήμη, επιτρέποντας την εκτέλεση μεγαλύτερων και πιο σύνθετων προγραμμάτων.

Arduino Uno: Flash: 32 KB.

RAM: 2 KB SRAM.

Κατάλληλο μόνο για απλές εφαρμογές.

3. Συνδεσιμότητα

ESPrit: Ενσωματωμένο Wi-Fi και Bluetooth (BLE).

Κατάλληλο για IoT εφαρμογές που απαιτούν ασύρματη επικοινωνία.

Arduino Uno: Δεν διαθέτει ενσωματωμένη συνδεσιμότητα.

Χρειάζεται πρόσθετα εξαρτήματα (shields) για Wi-Fi ή Bluetooth.

4. Ενέργεια

ESPrit: Υποστηρίζει λειτουργίες χαμηλής κατανάλωσης ενέργειας (low-power modes).

Κατάλληλο για εφαρμογές με μπαταρία που απαιτούν αποδοτική κατανάλωση.

Arduino Uno: Καταναλώνει περισσότερη ενέργεια σε σχέση με το ESPrit για αντίστοιχες λειτουργίες.

Λιγότερο αποδοτικό για μπαταριοκίνητες εφαρμογές.

5. Είσοδοι/Έξοδοι (GPIO Pins)

ESPrit: Διαθέτει περισσότερα GPIO (έως και 34 pins).

Υποστηρίζει πολλαπλές λειτουργίες (π.χ., PWM, I2C, SPI, UART, ADC, DAC).

Arduino Uno: Διαθέτει 14 ψηφιακά pins (6 από αυτά μπορούν να χρησιμοποιηθούν ως PWM).

Υποστηρίζει βασικές λειτουργίες όπως I2C, SPI, UART και ADC (6 αναλογικές είσοδοι).

6. Προγραμματισμός

ESPrit: Υποστηρίζει πολλές γλώσσες:

C/C++ μέσω Arduino IDE ή ESP-IDF.

MicroPython.

JavaScript μέσω πλατφορμών όπως Espruino.

Πολύ ευέλικτο για προχωρημένες εφαρμογές.

Arduino Uno: Προγραμματίζεται μόνο σε C/C++ μέσω Arduino IDE.

Περιορισμένες δυνατότητες σε σχέση με το ESPrit.

7. Μορφή (Form Factor)

ESPrit: Συνήθως πιο συμπαγές και με ενσωματωμένα περιφερειακά (κεραία, αισθητήρες).

Arduino Uno: Μεγαλύτερο, με τυποποιημένα headers για εύκολη σύνδεση shields.

8. Τιμή

ESPrit: Συχνά φθηνότερο ή στην ίδια κατηγορία τιμής με το Arduino Uno, αλλά παρέχει πολύ περισσότερες δυνατότητες.

Arduino Uno: Ελαφρώς πιο ακριβό σε σχέση με το ESPrit για την προσφερόμενη απόδοση.

Τα πλεονεκτήματα του ESPrit είναι:

- Ισχυρότερος επεξεργαστής.
- Μεγαλύτερη μνήμη και αποθήκευση.
- Ενσωματωμένο Wi-Fi και Bluetooth.
- Κατάλληλο για IoT και πιο σύνθετες εφαρμογές.

Τα πλεονεκτήματα του Arduino Uno:

- Απλότητα και ευκολία χρήσης για αρχάριους.
- Μεγάλη κοινότητα υποστήριξης για βασικές εφαρμογές.
- Ιδανικό για βασικά projects hardware.

Κεφάλαιο 3ο: Λογισμικό συσκευής (Software)

3.1 Λογισμικό Arduino IDE

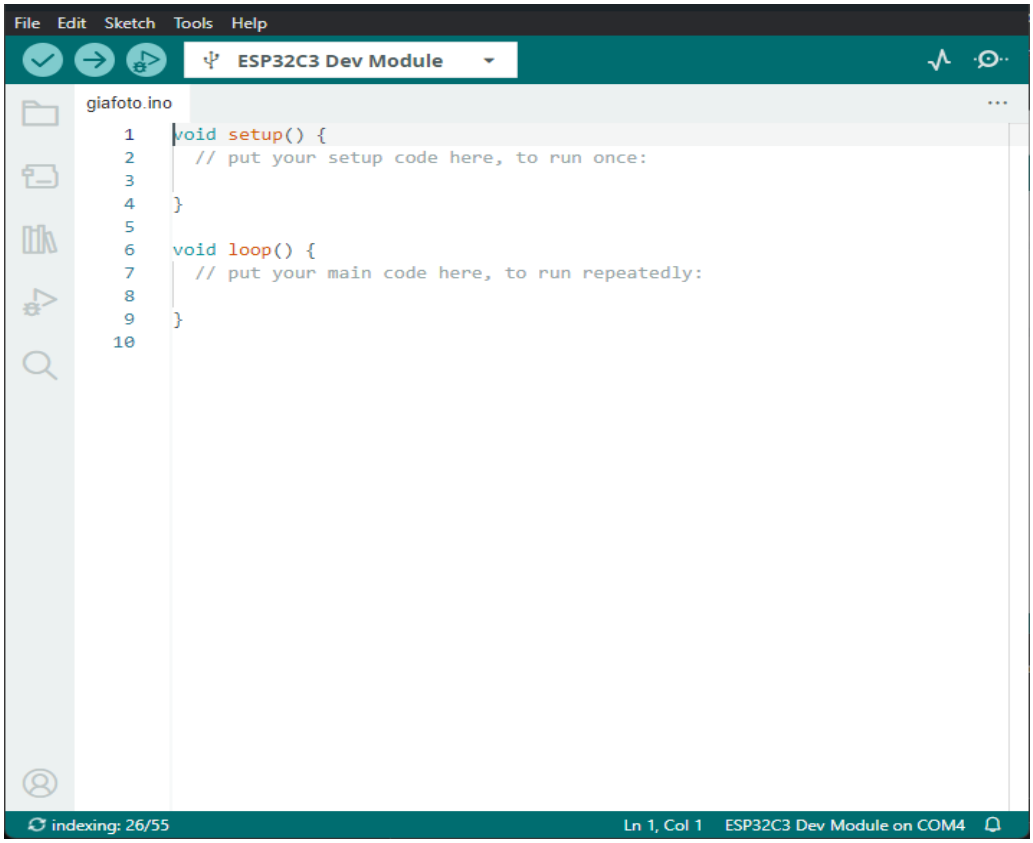
Το λογισμικό που χρησιμοποιήθηκε για να προγραμματιστεί η συσκευή είναι το Arduino IDE. Πρόκειται για το πιο συνηθισμένο περιβάλλον προγραμματισμού, στο οποίο χρησιμοποιείται η γλώσσα C. Υπάρχει η δυνατότητα πρόσβασης σε μεγάλο αριθμό βιβλιοθηκών Arduino με την μελέτη των οποίων μπορεί κάποιος να γράψει τον δικό του κώδικα.

Τα εργαλεία που περιλαμβάνονται σε ένα IDE (Integrated Development Environment) είναι:

- Επεξεργαστής πηγαίου κώδικα
- Μεταγλωττιστής
- Εργαλεία αυτόματης παραγωγής κώδικα
- Αποσφαλματωτής
- Συνδέτης
- Σύστημα ελέγχου εκδόσεων
- Εργαλεία κατασκευής γραφικών διασυνδέσεων χρήστη

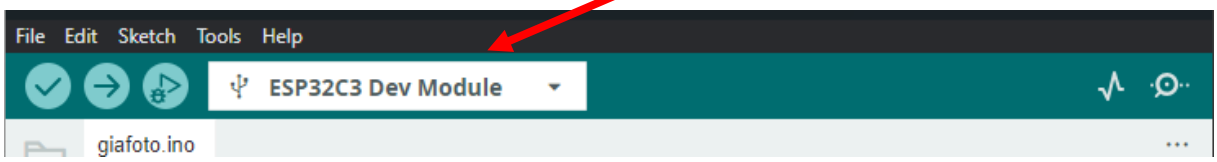
Στη συνέχεια παρουσιάζεται συνοπτικά ο τρόπος χρήσης του λογισμικού αυτού.

Η αρχική οθόνη που βλέπουμε ανοίγοντας το λογισμικό φαίνεται παρακάτω:



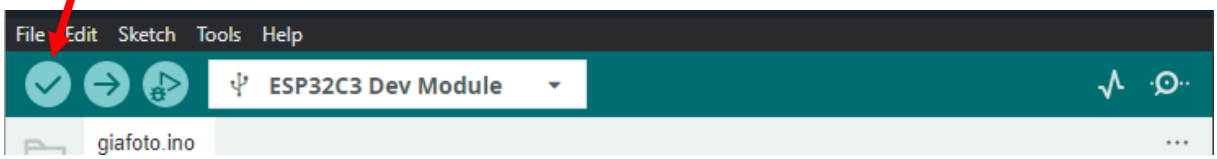
Σχήμα 3. 1: Αρχική οθόνη Arduino IDE

Αρχικά επιλέγουμε το κατάλληλο μοντέλο Arduino που θα χρησιμοποιηθεί.



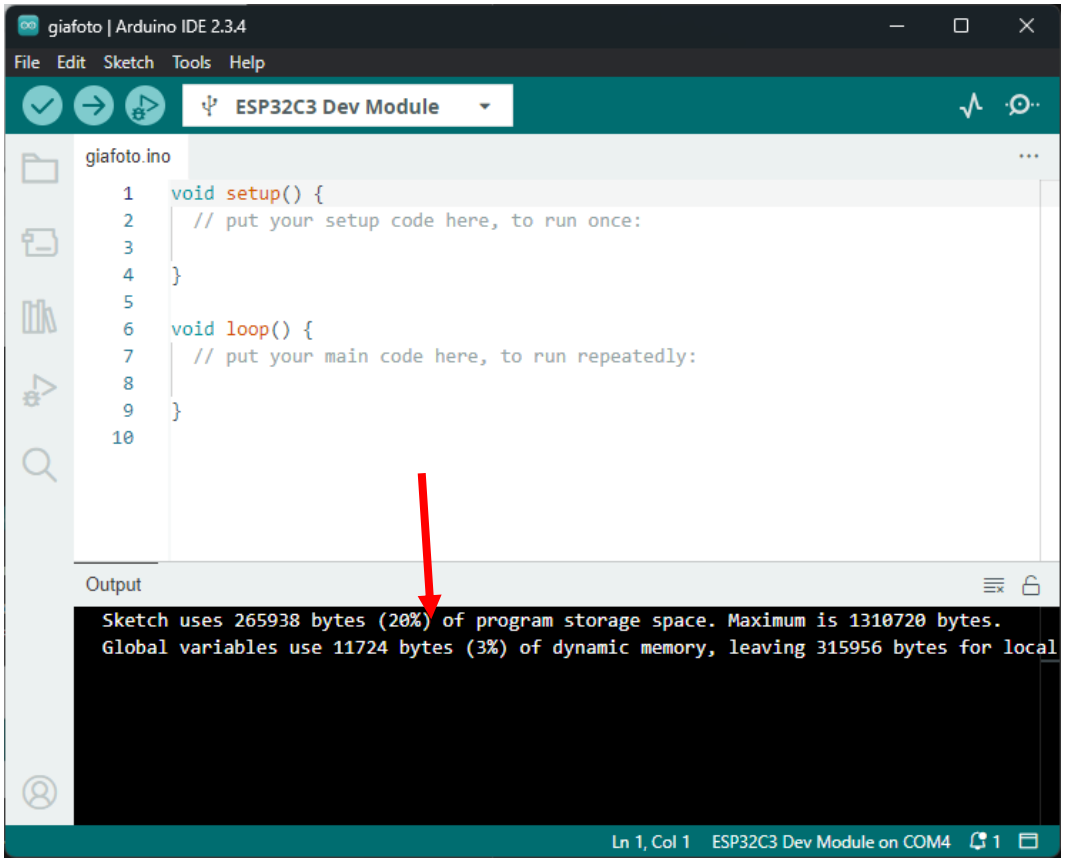
Σχήμα 3.2: Επιλογή μοντέλου Arduino

Στη συνέχεια γράφουμε τον κώδικα, όπως φαίνεται και στην αρχική σελίδα του προγράμματος. Για να μπορέσουμε να φορτώσουμε τον κώδικα θα πρέπει πρώτα να γίνει συντακτικός έλεγχος ο οποίος επιτυγχάνεται πατώντας το εικονίδιο που φαίνεται παρακάτω.



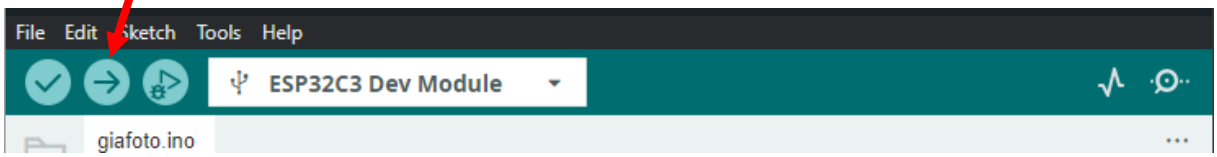
Σχήμα 3. 3: Εικονίδιο συντακτικού ελέγχου

Τα αποτελέσματα του ελέγχου μπορεί να δει ο χρήστης στο κάτω μέρος του παραθύρου.



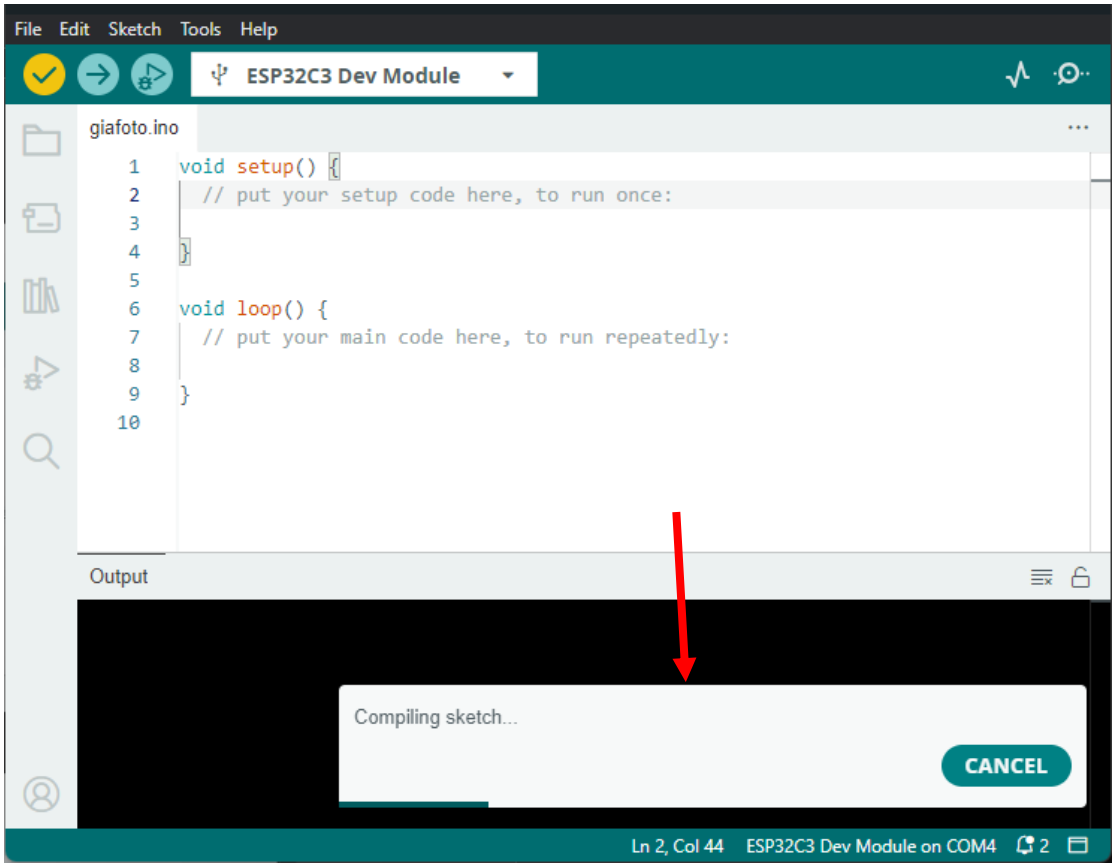
Σχήμα 3. 4: Παράθυρο output

Εφόσον ο κώδικας δεν έχει συντακτικά λάθη, μπορούμε να προχωρήσουμε στη μεταγλώττισή του και έπειτα στη φόρτωση του προγράμματος στη συσκευή μας.

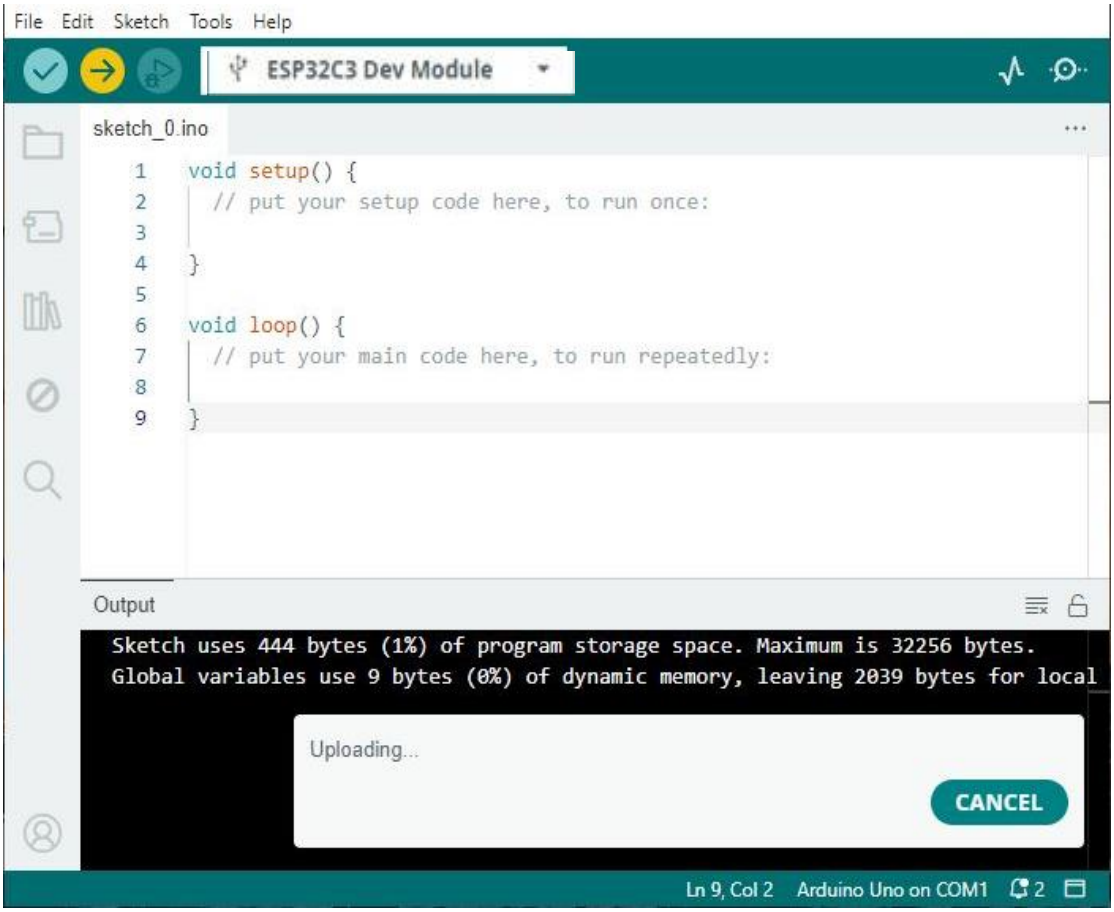


Σχήμα 3. 5: Εικονίδιο μεταγλώττισης

Τα αποτελέσματα και αυτής της ενέργειας φαίνονται στο κάτω μέρος του παραθύρου.



Σχήμα 3. 6: Παράθυρο compiling



Σχήμα 3. 7: Παράθυρο uploading

3.2 Διευθύνσεις I2C

Στη συνέχεια δίνονται ορισμένες πληροφορίες σχετικά με τις διευθύνσεις των ολοκληρωμένων κυκλωμάτων I2C και συγκεκριμένα των οθονών και του πολυπλέκτη.

Το πρωτόκολλο I2C (Inter-Integrated Circuit) είναι μια δημοφιλής μέθοδος επικοινωνίας που χρησιμοποιείται από μικροελεγκτές για τη σύνδεση με περιφερειακές συσκευές, όπως οθόνες και αισθητήρες. Κάθε συσκευή στο δίκτυο I2C έχει μια μοναδική διεύθυνση I2C που χρησιμοποιείται για την ταυτοποίησή της[10].

3.2.1 Διευθύνσεις των οθονών

- Ορισμός διευθύνσεων: Οι οθόνες I2C (π.χ., OLED SH1106 ή SSD1306) έχουν προκαθορισμένες διευθύνσεις που καθορίζονται από τον κατασκευαστή.
- Συνήθεις διευθύνσεις για οθόνες:
  - 0x3C: Προκαθορισμένη διεύθυνση για πολλές οθόνες SSD1306/SH1106.
  - 0x3D: Εναλλακτική διεύθυνση που μπορεί να ενεργοποιηθεί συνήθως με γείωση ή σύνδεση ενός pin (π.χ., ADDR).
- Εντοπισμός διεύθυνσης: Χρησιμοποιώντας έναν I2C σαρωτή, μπορεί να βρεθεί η διεύθυνση της οθόνης που είναι συνδεδεμένη στον μικροελεγκτή. Οι συσκευές εμφανίζονται ως δεκαεξαδικές διευθύνσεις (π.χ., 0x3C).
- Χρήση διευθύνσεων: Στον κώδικα, η διεύθυνση της οθόνης περνά ως παράμετρος στη βιβλιοθήκη οθόνης:

cpp

Αντιγραφή

Επεξεργασία

Adafruit_SSD1306 display(128, 64, &Wire, -1, 0x3C);

Η διεύθυνση 0x3C υποδεικνύει την οθόνη που πρέπει να χρησιμοποιηθεί.

3.2.2 Διευθύνσεις πολυπλέκτη

Ένας πολυπλέκτης I2C, όπως ο PCA9548A, επιτρέπει τη σύνδεση πολλών συσκευών με την ίδια I2C διεύθυνση σε διαφορετικά κανάλια. Αυτό είναι χρήσιμο όταν έχετε πολλές συσκευές με πανομοιότυπες διευθύνσεις[11].

- Βασική διεύθυνση: Ο πολυπλέκτης έχει προκαθορισμένη βασική διεύθυνση (π.χ., 0x70). Αυτή η διεύθυνση μπορεί να αλλάξει συνδέοντας ή αποσυνδέοντας τις θύρες A0, A1, A2 του πολυπλέκτη.

  Παράδειγμα ρυθμίσεων:

  A0: LOW, A1: LOW, A2: LOW → Διεύθυνση 0x70.

  A0: HIGH, A1: LOW, A2: LOW → Διεύθυνση 0x71.

  Χρήση διευθύνσεων του πολυπλέκτη:

Όταν θέλουμε να επικοινωνήσουμε με μια συσκευή σε ένα συγκεκριμένο κανάλι, στέλνουμε εντολή στον πολυπλέκτη για να ενεργοποιήσει το κανάλι:

cpp

Αντιγραφή

Επεξεργασία

Wire.beginTransmission(0x70); // Βασική διεύθυνση του πολυπλέκτη

Wire.write(1 << 3); // Ενεργοποίηση καναλιού 3

Wire.endTransmission();

Μετά την επιλογή καναλιού, η επικοινωνία γίνεται μόνο με τις συσκευές στο κανάλι αυτό.

- Πρακτική Χρήση

Οθόνες: Αν υπάρχουν δύο οθόνες με την ίδια διεύθυνση (π.χ., 0x3C), χρησιμοποιείται ο πολυπλέκτης για να διαχωριστούν:

Συνδέεται η πρώτη οθόνη στο κανάλι 0 και η δεύτερη στο κανάλι 1.

Επιλέγεται το κανάλι πριν σταλούν δεδομένα στην οθόνη.

Πολυπλέκτης: Με τον πολυπλέκτη, μπορούν να χρησιμοποιηθούν έως και 8 κανάλια για διαφορετικές συσκευές ή πολλαπλά αντίγραφα της ίδιας συσκευής.

- Χρησιμότητα
  - Αποφυγή σύγκρουσης διευθύνσεων: Εάν πολλές συσκευές έχουν την ίδια I2C διεύθυνση, ο πολυπλέκτης επιτρέπει την επικοινωνία με μία συσκευή κάθε φορά.
  - Επέκταση I2C: Επιτρέπει τη χρήση περισσότερων συσκευών από αυτές που υποστηρίζει το ενσωματωμένο I2C δίκτυο.

3.3 I2C scanner

Η χρήση ενός I2C scanner είναι απαραίτητη για τη διάγνωση και τη διαχείριση συσκευών που επικοινωνούν μέσω του πρωτοκόλλου I2C. Παρακάτω εξηγούνται οι λόγοι για τους οποίους χρησιμοποιείται ένα I2C scanner[13]:

1. Εύρεση I2C διευθύνσεων:

Κάθε συσκευή που συνδέεται σε ένα I2C δίκτυο έχει μια μοναδική διεύθυνση (7-bit ή 10-bit).

Οι κατασκευαστές μπορεί να έχουν προκαθορίσει αυτές τις διευθύνσεις, αλλά συχνά χρειάζεται να επαληθεύσουμε ποια διεύθυνση χρησιμοποιείται από κάθε συσκευή.

Ένας I2C scanner αναγνωρίζει τις διευθύνσεις όλων των συσκευών που είναι συνδεδεμένες στο δίκτυο.

2. Εντοπισμός συσκευών με συγκρούσεις διευθύνσεων:

Όταν δύο συσκευές στο ίδιο I2C δίκτυο έχουν την ίδια διεύθυνση, δημιουργείται σύγκρουση.

Με τον scanner, μπορούμε να δούμε ποιες διευθύνσεις είναι σε χρήση και να εντοπίσουμε τις συγκρούσεις.

Αν προκύψει σύγκρουση, μπορούμε να χρησιμοποιήσουμε έναν πολυπλέκτη (π.χ., PCA9548A) για τη διαχείριση των συσκευών.

3. Επαλήθευση σύνδεσης υλικού:

Ένας I2C scanner μας επιτρέπει να επαληθεύσουμε ότι το κύκλωμά μας είναι σωστά συνδεδεμένο:

Επιβεβαιώνει ότι οι ακροδέκτες SDA και SCL είναι σωστά ενωμένοι.Ελέγχει αν υπάρχει σωστή αντίσταση pull-up στο δίκτυο I2C.

Εμφανίζει στη σειριακή κονσόλα τις διευθύνσεις των συσκευών που ανταποκρίνονται.

4. Έλεγχος λειτουργικότητας συσκευών:

Αν μια συσκευή δεν ανταποκρίνεται κατά τη σάρωση, μπορεί να σημαίνει:

Πρόβλημα με την τροφοδοσία ρεύματος της συσκευής.

Ακατάλληλη ρύθμιση (π.χ., λάθος επιλογή διεύθυνσης μέσω jumpers ή pins).

Σφάλμα στην ίδια τη συσκευή.

5. Εργαλεία ανάπτυξης και αποσφαλμάτωσης:

Κατά την ανάπτυξη ενός έργου που περιλαμβάνει I2C συσκευές, ένας scanner είναι απαραίτητος για την ταυτοποίηση νέων συσκευών που προστίθενται.

Βοηθάει στην αποσφαλμάτωση κατά τη διάρκεια ανάπτυξης σύνθετων συστημάτων.

Οι εφαρμογές στις οποίες χρησιμοποιείται ένα I2C scanner είναι:

- Αναγνώριση περιφερειακών: Εντοπισμός αισθητήρων, οθονών, ή άλλων I2C συσκευών.
- Ρύθμιση πολυπλεκτών: Επαλήθευση διευθύνσεων σε πολυπλέκτες όπως ο PCA9548A.
- Συντήρηση: Διάγνωση σφαλμάτων ή αποσφαλμάτωση ενός I2C συστήματος.
- Ένας I2C scanner είναι ένα απλό αλλά ισχυρό εργαλείο για να ξεκινήσετε με έργα που βασίζονται σε I2C.

3.4 Βιβλιοθήκες

3.4.1 Βιβλιοθήκη OBD2UART

Η βιβλιοθήκη OBD2UART χρησιμοποιείται για την επικοινωνία ενός μικροελεγκτή (όπως Arduino ή ESP32) με το OBD-II σύστημα ενός αυτοκινήτου μέσω του πρωτοκόλλου UART (Universal Asynchronous Receiver-Transmitter).

Η OBD-II θύρα σε ένα όχημα επιτρέπει την ανάγνωση δεδομένων από τον εγκέφαλο του αυτοκινήτου (ECU). Το OBD2UART μεταφράζει δεδομένα από τη θύρα OBD-II σε σειριακή επικοινωνία UART που μπορεί να διαβαστεί από τον μικροελεγκτή [14][15].

Πρωτόκολλα OBD-II: Υποστηρίζει πολλαπλά πρωτόκολλα, όπως:

- ISO 9141-2
- CAN (Controller Area Network)
- KWP2000 (Keyword Protocol 2000)

Αυτό επιτρέπει στη βιβλιοθήκη να επικοινωνεί με διάφορα οχήματα που χρησιμοποιούν διαφορετικά πρωτόκολλα.

Ανάγνωση PID (Parameter IDs): Το OBD-II χρησιμοποιεί PIDs για την ανάκτηση δεδομένων, όπως:

- RPM κινητήρα.
- Ταχύτητα οχήματος.
- Θερμοκρασία ψυκτικού υγρού.

Η βιβλιοθήκη στέλνει ερωτήματα PIDs και λαμβάνει τις αντίστοιχες τιμές.

Σειριακή Επικοινωνία: Η βιβλιοθήκη χρησιμοποιεί σειριακές εντολές (π.χ., AT εντολές) για να επικοινωνήσει με την OBD-II θύρα.

Η χρήση του γίνεται κυρίως για την ανάγνωση δεδομένων οχήματος όπως RPM κινητήρα, ταχύτητα, θερμοκρασία ψυκτικού υγρού, επίπεδο καυσίμου. Επίσης χρησιμοποιείται ως διαγνωστικό εργαλείο για ανάγνωση και εκκαθάριση διαγνωστικών κωδικών βλαβών (DTCs) και για δημιουργία Custom Εφαρμογών όπως καταγραφή δεδομένων οχήματος για τηλεματικές εφαρμογές ή παρακολούθηση στόλου.

3.4.2 Βιβλιοθήκη SH1106

Η βιβλιοθήκη SH1106 χρησιμοποιείται για τον έλεγχο οθονών OLED που βασίζονται στο driver SH1106. Αυτές οι οθόνες είναι μονόχρωμες και χρησιμοποιούνται ευρέως σε εφαρμογές IoT και ενσωματωμένα συστήματα. Η βιβλιοθήκη υποστηρίζει την επικοινωνία μέσω I2C ή SPI για να στέλνει δεδομένα στην οθόνη[10].

Απεικόνιση Κειμένου και Bitmap:

- Εμφανίζει κείμενο, αριθμούς και bitmap στην OLED.
- Υποστηρίζει διάφορα μεγέθη γραμματοσειράς (π.χ., μικρή, μεσαία, μεγάλη).

Εκκαθάριση και Ενημέρωση Οθόνης:

- Παρέχει λειτουργίες για εκκαθάριση της οθόνης και ανανέωση του περιεχομένου.

Γραφικές Λειτουργίες:

- Ενσωματωμένες λειτουργίες για σχεδίαση γραμμών, κύκλων, ορθογωνίων και άλλων σχημάτων.

Η χρήση του γίνεται για εμφάνιση δεδομένων όπως για εμφάνιση αποτελεσμάτων αισθητήρων ή δεδομένων οχήματος (π.χ., RPM, ταχύτητα) σε πραγματικό χρόνο αλλά και για άλλες εφαρμογές όπως είναι η σχεδίαση μενού, εικονιδίων ή βασικών διαγραμμάτων. Τέλος, χρησιμοποιείται σε IoT εφαρμογές για παράδειγμα σε οθόνες για έξυπνες συσκευές.

Συμπερασματικά:

Το OBD2UART εστιάζει στη διασύνδεση με την OBD-II θύρα ενός οχήματος για ανάγνωση δεδομένων.

Το SH1106 ελέγχει OLED οθόνες για την εμφάνιση αυτών των δεδομένων ή άλλων πληροφοριών.

Ο συνδυασμός τους είναι ιδανικός για projects που απαιτούν παρακολούθηση δεδομένων οχήματος σε πραγματικό χρόνο.

3.5 Επεξήγηση κώδικα

Καθώς δεν θεωρείται από το συγγραφέα απαραίτητη η επεξήγηση ολόκληρου του κώδικα, θα επεξηγηθούν στη συνέχεια τα κομμάτια που συνδέονται με τις επιλεγμένες μετρήσεις/ενδείξεις καθώς και η λειτουργία του πολυπλέκτη.

Για την ολοκλήρωση του κώδικα, χρησιμοποιήθηκαν έτοιμες βιβλιοθήκες οι οποίες τροποποιήθηκαν κατάλληλα ώστε να εμφανίζονται μόνο τα στοιχεία που επιλέχθηκαν και με τον τρόπο που επιλέχθηκε τόσο σε μέγεθος όσο και σε αριθμό ψηφίων.

Παρακάτω φαίνεται το μέρος του κώδικα σύμφωνα με το οποίο οι ενδείξεις της ταχύτητας και της μπαταρίας θα εμφανίζονται στην 1η οθόνη:

```
switch (pid1) { // Επεξεργασία του πρώτου PID
  case PID_SPEED: // Αν το PID είναι η ταχύτητα
    lcd.setCursor(0, 0); // Ορισμός θέσης κειμένου στην αρχή της οθόνης
    lcd.setFontSize(FONT_SIZE_XLARGE); // Ορισμός πολύ μεγάλης γραμματοσειράς
    lcd.print("SPD: "); // Εμφάνιση ετικέτας "SPD"
    lcd.printInt(value1 % 1000, 3); // Εμφάνιση τιμής ταχύτητας (3 ψηφία)
    break;
  case PID_BATTERY_VOLTAGE: // Αν το PID είναι η τάση μπαταρίας
    lcd.setCursor(0, 0); // Ορισμός θέσης κειμένου στην αρχή της οθόνης
    lcd.setFontSize(FONT_SIZE_XLARGE); // Ορισμός πολύ μεγάλης γραμματοσειράς
    lcd.print("VOLT: "); // Εμφάνιση ετικέτας "VOLT"
    lcd.printInt(value1, 2); // Εμφάνιση τιμής τάσης (2 ψηφία)
```

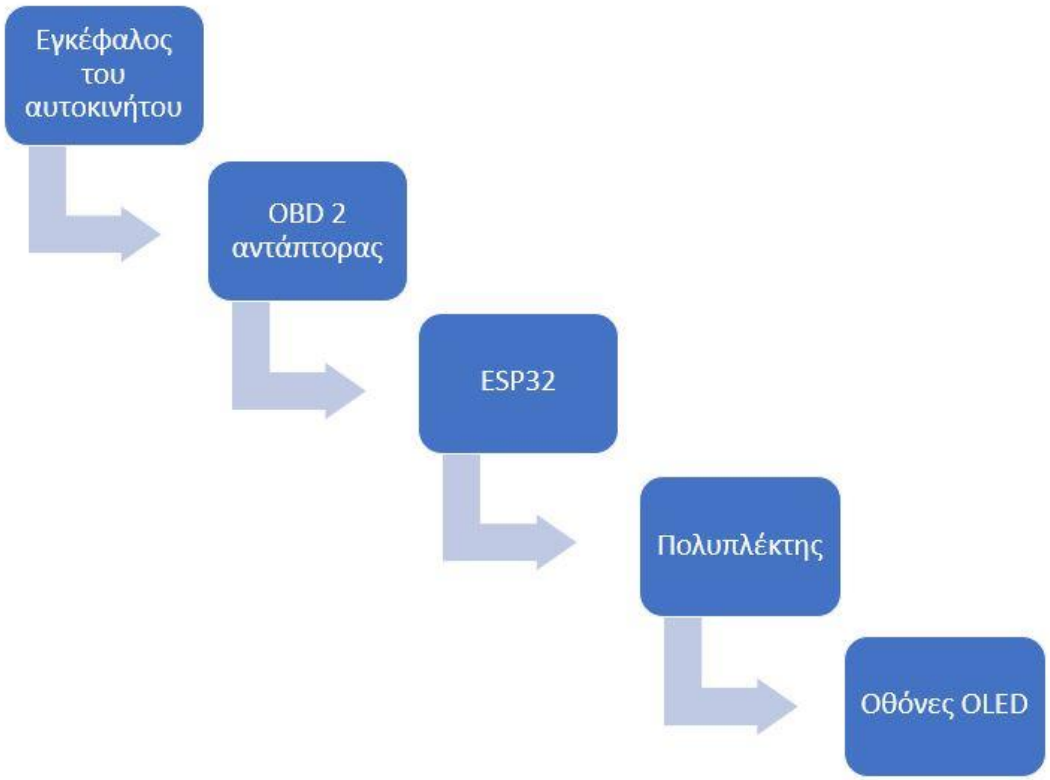
```
    lcd.setFontSize(FONT_SIZE_MEDIUM); // Ορισμός μεσαίας γραμματοσειράς
    lcd.print("V"); // Εμφάνιση μονάδας μέτρησης (Volt)
    break;
```

Παρακάτω φαίνεται το μέρος του κώδικα σύμφωνα με το οποίο οι ενδείξεις της ταχύτητας και της μπαταρίας θα εμφανίζονται στην 2η οθόνη:

```
  switch (pid2) { // Επεξεργασία του δεύτερου PID
    case PID_RPM: // Αν το PID είναι οι στροφές κινητήρα
      lcd.setCursor(0, 5); // Τοποθέτηση κειμένου κάτω από την ταχύτητα
      lcd.setFontSize(FONT_SIZE_LARGE); // Ορισμός μεγάλης γραμματοσειράς
      lcd.print("RPM: "); // Εμφάνιση ετικέτας "RPM"
      lcd.printInt(value2 % 10000, 4); // Εμφάνιση τιμής RPM (4 ψηφία)
      break;
    case PID_COOLANT_TEMP: // Αν το PID είναι η θερμοκρασία ψυκτικού
      lcd.setCursor(0, 5); // Τοποθέτηση κειμένου κάτω από την τάση
      lcd.setFontSize(FONT_SIZE_LARGE); // Ορισμός μεγάλης γραμματοσειράς
      lcd.print("TEMP: "); // Εμφάνιση ετικέτας "TEMP"
      lcd.printInt(value2 % 100, 3); // Εμφάνιση θερμοκρασίας (3 ψηφία)
      lcd.setFontSize(FONT_SIZE_MEDIUM); // Ορισμός μεσαίας γραμματοσειράς
      lcd.print("C"); // Εμφάνιση μονάδας μέτρησης (Celsius)
      break;
```

Κεφάλαιο 4ο: Συνδέσεις συστήματος

Για την κατανόηση των συνδέσεων του συστήματος, παρουσιάζεται το παρακάτω διάγραμμα όπου φαίνεται η πορεία των δεδομένων που λαμβάνονται αρχικά από τον εγκέφαλο και τελικά εμφανίζονται στις οθόνες OLED του συτήματος.

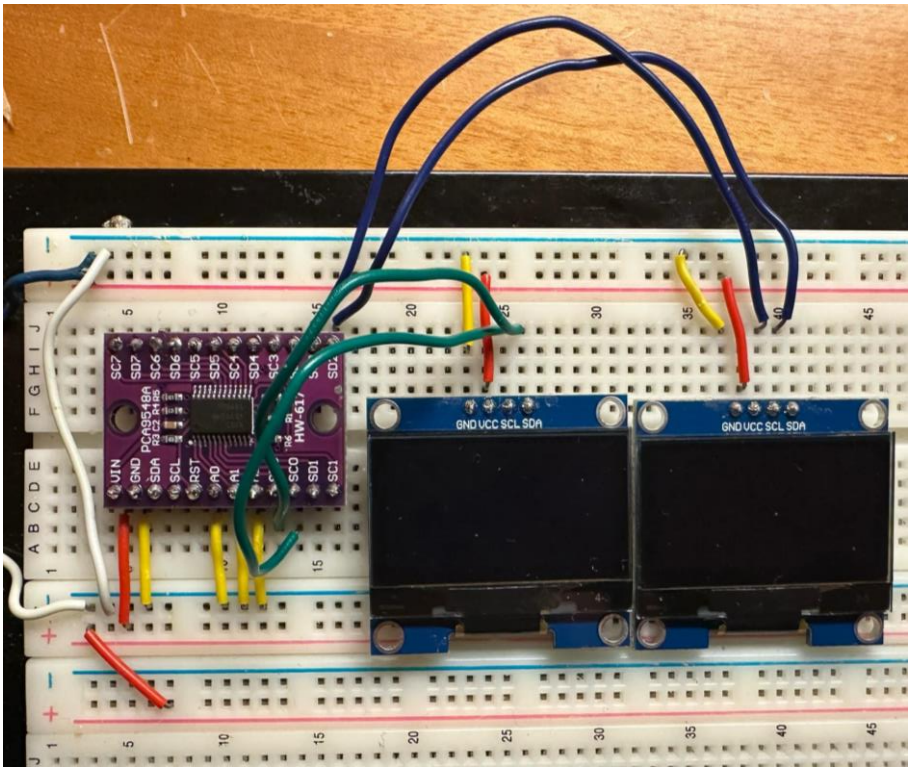


Σχήμα 4. 1: Διάγραμμα ροής δεδομένων

- Αρχικά, μόλις το αυτοκίνητο ξεκινήσει να λειτουργεί, ο εγκέφαλος συλλέγει όλες τις πληροφορίες σχετικά με το αυτοκίνητο.
- Ο αντάπτορας OBD-II συλλέγει δεδομένα όπως η ταχύτητα, η θερμοκρασία, οι στροφές του κινητήρα και η τάση της μπαταρίας από τον εγκέφαλο και τα μεταφέρει στο ESP32 μέσω σειριακής επικοινωνίας.
- Το ESP32 επεξεργάζεται τα δεδομένα και αποστέλλει τα κατάλληλα πακέτα προς τον πολυπλέκτη. Κάθε εντολή συνοδεύεται από τη διεύθυνση της οθόνης που θα εμφανίσει τα δεδομένα.
- Ο πολυπλέκτης ενεργοποιεί το κατάλληλο κανάλι για την OLED οθόνη που θα λάβει τα δεδομένα. Η κατανομή γίνεται σειριακά, αποφεύγοντας συγκρούσεις μέσω του I2C πρωτοκόλλου.
- Οι OLED λαμβάνουν τα δεδομένα και τα εμφανίζουν σε πραγματικό χρόνο, με τις πληροφορίες να προβάλλονται στις αντίστοιχες οθόνες.

Στη συνέχεια γίνεται η περιγραφή των επιμέρους συνδέσεων, προσδιορίζοντας τους ακροδέκτες που χρησιμοποιούνται καθώς και τη λειτουργία που επιτελούν.

Αρχικά γίνεται η σύνδεση των οθονών OLED με τον πολυπλέκτη.



Σχήμα 4. 2: Σύνδεση οθονών με πολυπλέκτη

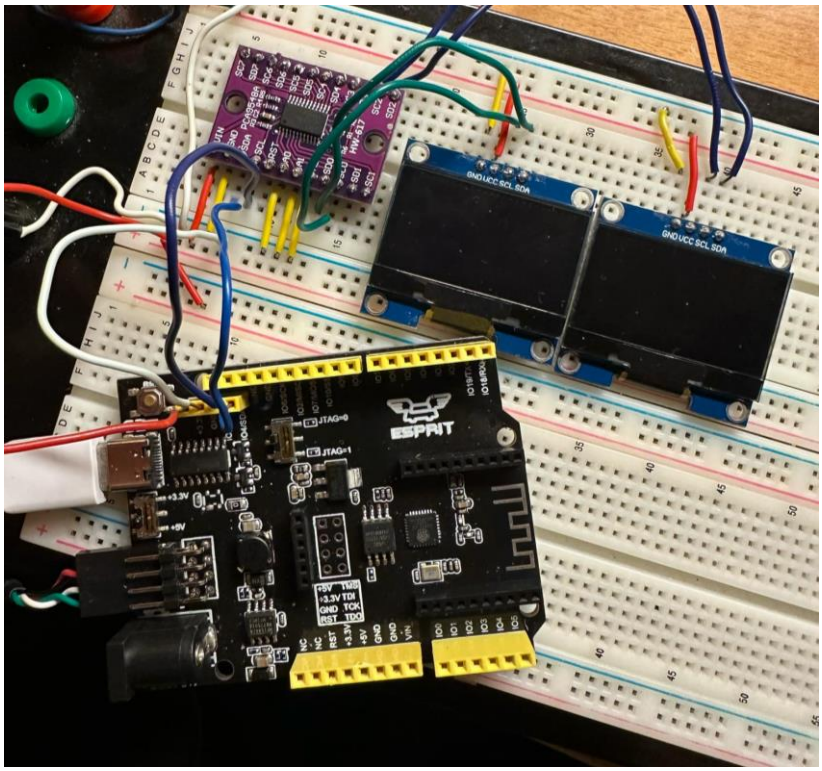
- Το pin VCC της οθόνης 1 συνδέεται στην τροφοδοσία 3.3V (κίτρινο καλώδιο).
- Το pin GND της οθόνης 1 συνδέεται την γείωση (κόκκινο καλώδιο).
- Το pin SCL της οθόνης 1 συνδέεται στην επαφή SC0 του πολυπλέκτη (πράσινο καλώδιο).
- Το pin SDA της οθόνης 1 συνδέεται στην επαφή SD0 του πολυπλέκτη (πράσινο καλώδιο).

Αντίστοιχα,

- Το pin VCC της οθόνης 2 συνδέεται στην τροφοδοσία 3.3V (κίτρινο καλώδιο).
- Το pin GND της οθόνης 2 συνδέεται την γείωση (κόκκινο καλώδιο).
- Το pin SCL της οθόνης 2 συνδέεται στην επαφή SC2 του πολυπλέκτη (μπλε καλώδιο).
- Το pin SDA της οθόνης 2 συνδέεται στην επαφή SD2 του πολυπλέκτη (μπλε καλώδιο).

Επισημαίνεται πως, τα κανάλια ενεργοποιούνται ξεχωριστά, ώστε να μην υπάρχει ομοιότητα στα δεδομένα που θα εμφανίζονται.

Στη συνέχεια γίνεται η σύνδεση του πολυπλέκτη PCA9548A με τον ESP32.

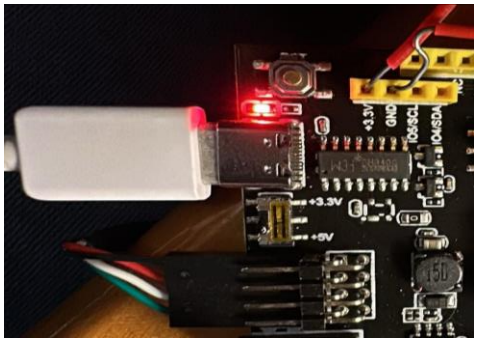


Σχήμα 4. 3: Σύνδεση πολυπλέκτη με ESP32

- Το pin VIN του πολυπλέκτη συνδέεται στην τροφοδοσία 3.3V (κόκκινο καλώδιο).
- Το pin GND του πολυπλέκτη συνδέεται την γείωση (άσπρο καλώδιο).
- Το pin SCL του πολυπλέκτη συνδέεται στην επαφή SCL/IO5 του ESP32 (μπλε καλώδιο).
- Το pin SDA του πολυπλέκτη συνδέεται στην επαφή SDA/IO4 του ESP32 (γαλάζιο καλώδιο).
- Τα pin A0, A1, A2 του πολυπλέκτη (κίτρινα καλώδια) συνδέονται στην γείωση και όπως έχει αναφερθεί νωρίτερα, είναι υπεύθυνα για την εκλογή της διεύθυνσης του πολυπλέκτη.

Επισημαίνεται ότι, το SDA(Data Line) μεταφέρει τα δεδομένα I2C από το ESP32 στον πολυπλέκτη. Το SCL(Clock Line) συγχρονίζει τη μεταφορά δεδομένων μεταξύ του ESP32 και του πολυπλέκτη.

Το καλώδιο USB που φαίνεται να συνδέεται στο ESP32 στο σχήμα 4.4, χρησιμοποιείται μόνο για τη φόρτωση του κώδικα. Δεν απαιτείται οποιαδήποτε σύνδεση με USB στο αυτοκίνητο για να είναι εφικτή η λειτουργία του συστήματος.



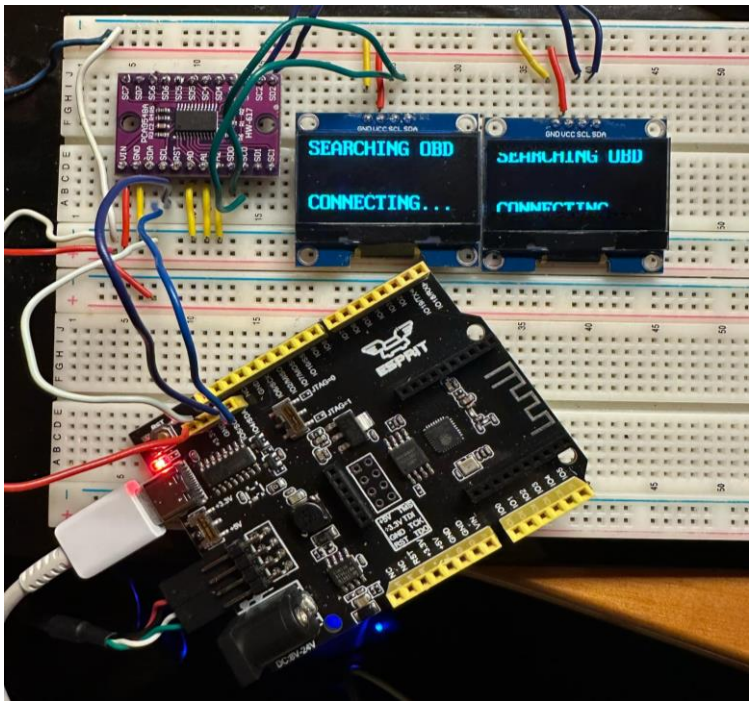
Σχήμα 4. 4: Σύνδεση USB στο ESP32

Τέλος, συνδέεται ο προσαρμογέας OBD-II UART στις σειριακές ακίδες UART1 της πλακέτας. Για τις ανάγκες

Σχήμα 4. 5: OBD-II UART αντάπτορας

Σχήμα 4. 6: OBD-II UART αντάπτορας

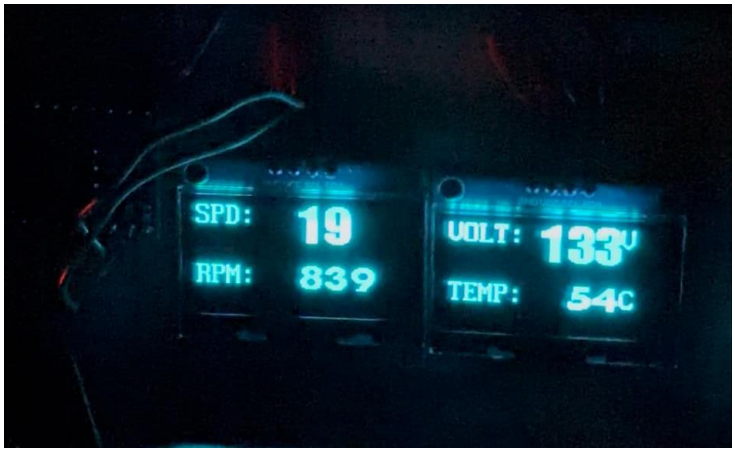
Η σύνδεση των εξαρτημάτων έγινε με τη χρήση ενός breadboard. Χρησιμοποιώντας εξωτερική πηγή τροφοδοσίας, τέθηκε σε λειτουργία το σύστημα και όπως φαίνεται στο σχήμα 4.7, οι οθόνες λειτουργούν και είναι στο στάδιο της ανάγνωσης των δεδομένων από το OBD.



Σχήμα 4. 7: Συνδέσεις στο breadbord

Αφού ολοκληρώθηκε η σύνδεση των επιμέρους υλικών, η συσκευή συνδέθηκε σε αυτοκίνητο, τέθηκε σε λειτουργία και στο σχήμα 4.8 φαίνονται τα δεδομένα που λήφθηκαν.

Σημειώνεται πως μόλις οι οθόνες τροφοδοτηθούν εμφανίζουν αρχικά ένα μήνυμα το οποίο επιβεβαιώνει την ορθή λειτουργία τους και ταυτόχρονα φανερώνονται τα δεδομένα τα οποία πρόκειται να εμφανιστούν. Πιο συγκεκριμένα, η οθόνη 1 εμφανίζει το μήνυμα: “Screen 1: SPD & RPM” και η οθόνη 2 το μήνυμα: “Screen 2: VOLT & TMP”.



Σχήμα 4. 8: Εμφάνιση δεδομένων

Την εργασία συνοδεύει ένα βίντεο όπου φαίνεται η διαδικασία σύνδεσης του συστήματος με το αυτοκίνητο και στη συνέχεια οι ενδείξεις που προβάλλονται στις οθόνες κατά τη λειτουργία του αυτοκινήτου.

Κεφάλαιο 5ο: Προβλήματα και λύσεις

Κατά τη διαδικασία της μελέτης και της κατασκευής της συσκευής παρουσιάστηκαν ορισμένα προβλήματα. Αφού αναγνωρίστηκε η αιτία εμφάνισής τους, διορθώθηκαν και παρουσιάζονται παρακάτω.

1. Αναγνώριση του πολυπλέκτη

Πρόβλημα: Ο πολυπλέκτης (PCA9548A) δεν αναγνωριζόταν από το σύστημα κατά τη διαδικασία αρχικής ρύθμισης.

Αιτία: Λάθος διασύνδεση των ακίδων SDA και SCL μεταξύ ESP32 και πολυπλέκτη.

Η διεύθυνση του πολυπλέκτη (0x70) δεν είχε επιβεβαιωθεί.

Ελλιπής ή λανθασμένη λειτουργία pull-up αντιστάσεων στη γραμμή I2C.

Λύση: Ελέγξαμε και διορθώσαμε τις συνδέσεις SDA και SCL.

Επιβεβαιώσαμε τη διεύθυνση του πολυπλέκτη με σάρωση I2C.

Σταθεροποιήσαμε τα pin του πολυπλέκτη για σίγουρη επαφή.

2. Χρήση διπλής οθόνης μέσω πολυπλέκτη

Πρόβλημα: Οι οθόνες δεν εμφάνιζαν σωστά δεδομένα όταν συνδέονταν μέσω του πολυπλέκτη.

Αιτία: Καθυστερήσεις και σφάλματα κατά την εναλλαγή καναλιών στον πολυπλέκτη.

Μη συμβατές βιβλιοθήκες για την ταυτόχρονη λειτουργία δύο οθονών.

Λύση: Ρυθμίσαμε τη σωστή επιλογή καναλιού με τη συνάρτηση PCA9548A(bus).

Δοκιμάσαμε και προσαρμόσαμε τη βιβλιοθήκη SH1106 για κάθε οθόνη ξεχωριστά.

Προσθέσαμε καθυστερήσεις (delay()) για σταθεροποίηση κατά την εναλλαγή καναλιών.

3. Καθυστέρηση στην απεικόνιση

Πρόβλημα: Παρατηρήθηκε καθυστέρηση στην απεικόνιση των δεδομένων στις οθόνες.

Αιτία: Υπερβολική χρήση της εντολής lcd.clear(), που καθαρίζει ολόκληρη την οθόνη σε κάθε επανάληψη.

Συχνή ανάγνωση δεδομένων από το OBD-II, κάτι που είναι χρονοβόρο.

Στατική καθυστέρηση (delay(500)) στο τέλος του κύριου loop.

Λύση: Περιορίσαμε τη συχνότητα ανάγνωσης δεδομένων από το OBD-II.

4. Ανάγνωση δεδομένων από OBD-II

Πρόβλημα: Ορισμένα PID (π.χ. τάση μπαταρίας, θερμοκρασία ψυκτικού) δεν ανανεώνονταν σωστά ή εμφάνιζαν λάθος τιμές.

Αιτία: Η σύνδεση με το OBD-II δεν ήταν σταθερή.

Οι συχνές κλήσεις στην OBD-II συσκευή οδηγούσαν σε σφάλματα.

Λύση: Ενσωματώσαμε τη συνάρτηση reconnect() για αυτόματη επανασύνδεση όταν προκύπτουν σφάλματα.

5. Δυσκολίες στην εμφάνιση δύο μετρήσεων σε κάθε οθόνη

Πρόβλημα: Η εμφάνιση δύο διαφορετικών τιμών στην ίδια οθόνη προκαλούσε ασυμφωνίες στη θέση των δεδομένων.

Αιτία: Μη σωστή ρύθμιση των συντεταγμένων μέσω της setCursor(x, y).

Λύση: Χρησιμοποιήσαμε τη σωστή ρύθμιση x και y για να τοποθετήσουμε τα δεδομένα στις κατάλληλες θέσεις.

Βεβαιωθήκαμε ότι οι γραμματοσειρές δεν επηρεάζουν την ευθυγράμμιση.

6. Έλεγχος συστήματος μέσω Debugging

Πρόβλημα: Κατά την ανάπτυξη, ήταν δύσκολο να εντοπιστούν σφάλματα στον πολυπλέκτη ή στις οθόνες.

Αιτία: Δεν υπήρχαν αρκετά debugging μηνύματα στον σειριακό monitor.

Λύση: Προσθέσαμε μηνύματα debugging στον σειριακό monitor για κάθε κρίσιμο σημείο (π.χ. αναγνώριση οθονών, εναλλαγή καναλιών, ανάγνωση OBD-II).

7. Συμβατότητα με ESP32

Πρόβλημα: Η χρήση του ESP32-C3 παρουσίασε αρχικά ασυμβατότητες στις γραμμές SDA και SCL.

Αιτία: Οι ακίδες SDA/SCL για το ESP32-C3 διαφέρουν από άλλες εκδόσεις ESP32.

Λύση: Ρυθμίσαμε τις γραμμές SDA και SCL με Wire.begin(4, 5) για να διασφαλίσουμε τη σωστή λειτουργία.

Συμπεράσματα

Η μελέτη των συστημάτων Arduino αλλά και η εξοικείωση με τον προγραμματισμό τους θα μπορούσε να επιφέρει μεγάλη εξέλιξη στην λειτουργία του ταμπλό του αυτοκινήτου. Είναι δυνατό, μέσω του κατάλληλου κώδικα και με τη βοήθεια μιας πλακέτας Arduino, να λαμβάνουμε μέσω της θύρας OBD του αυτοκινήτου τα δεδομένα που στέλνει ο εγκέφαλος και να τα απεικονίζουμε σε μια οθόνη υγρών κρυστάλλων.

Στην παρούσα εργασία, αρχικά έγινε η απαραίτητη μελέτη γύρω από τα χαρακτηριστικά και τις δυνατότητες των διαφόρων μοντέλων Arduino. Με σκοπό να δημιουργηθεί ο κατάλληλος κώδικας, χρησιμοποιήθηκε το λογισμικό Arduino IDE. Παρ' όλο που χρειάστηκαν αρκετές μετατροπές ώστε τα αποτελέσματα να είναι τα επιθυμητά, η διαδικασία της φόρτωσης του κώδικα ήταν αρκετά εύκολη αφού μέσω του λογισμικού με ένα απλό καλώδιο USB ο κώδικας μπορεί να φορτωθεί απευθείας στην πλακέτα.

Αφού έγινε η επιλογή των οθονών οι οποίες θα χρησιμοποιούνταν, επιλέχθηκε και ο κατάλληλος πολυπλέκτης, με σκοπό να λαμβάνουμε τις 2 ενδείξεις στη μία και τις άλλες 2 στην άλλη. Σε αυτό το σημείο χρειάστηκε να δημιουργηθεί επιπλέον κώδικας, μια διαδικασία η οποία εμφάνισε αρκετά εμπόδια στην αρχή.

Τέλος, έγιναν οι συνδέσεις όλων των παραπάνω σε ένα breadboard και συνδέθηκε στο αυτοκίνητο μέσω της θύρας OBD-II UART.

Συμπερασματικά, η υλοποίηση της συσκευής κρίνεται σχετικά απλή, χρειάζεται όμως αρκετή μελέτη για τη σύνταξη του κατάλληλου κώδικα και την επιλογή των υλικών. Το σύστημα είναι πλήρως λειτουργικό, ενώ παράλληλα είναι εφικτή οποιαδήποτε μετατροπή είτε αυτή είναι ο τρόπος απεικόνισης είτε η επιλογή των μετρήσεων που θα απεικονίζονται.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] L. Lewis, "WORKING PRINCIPLE OF ARDUINO AND USING IT AS A TOOL FOR STUDY AND RESEARCH", *International Journal of Control, Automation, Communication and Systems*, Vol.1, No.2, Apr. 2016.

[2] M. Banzi and M. Shiloh, G*etting started with Arduino,* USA: Publisher Make Community, 2022.

[3] K. S. Kaswan, S. P. Singh and S. Sagar, "Role of Arduino in real world applications", *International Journal of scientific & technology research,* Vol. 9, Issue 01, Jan. 2020.

[4] A. S. Ismailov and Z. B. Jo 'rayev, "Study of arduino microcontroller board", *Science and Education Scientific Journal,* Vol. 3, Issue 3, Mar. 2022.

[5] Arduino, "Arduino Hardware," *Arduino,* 2022. [Online]. Available: https://www.arduino.cc/en/hardware.

[6] S. H. Chen, J. S. Pan and K. Lu, "Driving Behavior Analysis Based on Vehicle OBD Information and AdaBoost Algorithms", *International MultiConference of Engineers and Computer Scientists,* Vol. 1, Mar. 2015.

[7] Espressif Systems, "UltraLowPower SoC with RISCV SingleCore CPU," ESP32C3 datasheet, 2021.

[8] A. Rodríguez, J. Vento and R. Inouye, "Implementation of an OBD-II diagnostics tool over CAN-BUS with Arduino", *Sistemas & Telemática*, Vol. 16, no. 45, pp. 45-53, 2018.

[9] Freematics online store, "ESP32 OBD Dev Kit," *Freematics online store*, 2025. [Online]. Available: https://freematics.com/store/index.php?route=product/product&product_id=87.

[10] Sino Wealth, "132x64 Dot Matrix OLED/PLED", SH1106 datasheet, Jun. 2023.

[11] NXP, "8-channel Multiplexer I2C-bus switch with reset," PCA9548A datasheet, Jul. 2009.

[12] M. Niedermaier, T. Hanka, F. Fischer and D. Merli, "A Secure Network Scanner Architecture for Asset Management in Strongly Segmented ICS Networks", *7th International Conference on Information Systems Security and Privacy*, vol.1, pp. 347-355, Feb. 2021.

[13] M. M. Oluwaseyi and A. M. Sunday, "Specifications and Analysis of Digitized Diagnostics of Automobiles: A Case Study of on-Board Diagnostic (OBD II)", *International Journal of Engineering Research & Technology*, Vol. 9, Issue 01, Jan. 2020.

[14] D. Rimpas, A. Papadakis and M. Samarakou, "OBD-II sensor diagnostics for monitoring vehicle operation and consumption", *Energy Reports*, Vol. 6, Supplement 3, Pages 55-63, 2020.

ΠΑΡΑΡΤΗΜΑ Α : Κώδικας συσκευής

```
#include <Wire.h> // Βιβλιοθήκη για επικοινωνία I2C
#include <OBD2UART.h> // Βιβλιοθήκη για επικοινωνία με OBD-II
#include <SH1106.h> // Βιβλιοθήκη για έλεγχο οθονών SH1106

// Ορισμός της διεύθυνσης του πολυπλέκτη
#define MULTIPLEXER_ADDR 0x70

// Δημιουργία αντικειμένων για τις δύο οθόνες
LCD_SH1106 lcd1; // Πρώτη οθόνη
LCD_SH1106 lcd2; // Δεύτερη οθόνη
COBD obd; // Δημιουργία αντικειμένου για τον προσαρμογέα OBD-II

// Συνάρτηση για την επιλογή συγκεκριμένου καναλιού στον πολυπλέκτη
void PCA9548A(uint8_t bus) {
  Wire.beginTransmission(MULTIPLEXER_ADDR); // Έναρξη μετάδοσης στη διεύθυνση του πολυπλέκτη
  Wire.write(1 << bus); // Ενεργοποίηση συγκεκριμένου καναλιού
  Wire.endTransmission(); // Τερματισμός μετάδοσης
}

// Συνάρτηση για επανασύνδεση με τον προσαρμογέα OBD-II και επαναφορά των οθονών
void reconnect() {
  for (uint8_t bus : {0, 2}) { // Επανάληψη για τα κανάλια 0 και 2
    PCA9548A(bus); // Επιλογή καναλιού
    LCD_SH1106 &lcd = (bus == 0) ? lcd1 : lcd2; // Επιλογή σωστής οθόνης
    lcd.clear(); // Εκκαθάριση οθόνης
    lcd.setFontSize(FONT_SIZE_MEDIUM); // Ορισμός μεσαίου μεγέθους γραμματοσειράς
    lcd.print("RECONNECTING..."); // Εμφάνιση μηνύματος επανασύνδεσης
  }

  for (uint16_t i = 0; !obd.init(); i++) { // Προσπάθειες επανασύνδεσης
```

```
    if (i == 5) { // Αν οι προσπάθειες είναι πάνω από 5, εκκαθάριση οθονών
      for (uint8_t bus : {0, 2}) {
        PCA9548A(bus);
        LCD_SH1106 &lcd = (bus == 0) ? lcd1 : lcd2;
        lcd.clear();
      }
    }
    delay(3000); // Αναμονή 3 δευτερολέπτων πριν την επόμενη προσπάθεια
  }
}

// Συνάρτηση για εμφάνιση δεδομένων OBD στις οθόνες
void showData(byte pid1, int value1, byte pid2, int value2, LCD_SH1106 &lcd) {
  lcd.clear(); // Εκκαθάριση οθόνης πριν την εμφάνιση νέων δεδομένων
  switch (pid1) { // Επεξεργασία του πρώτου PID
    case PID_SPEED: // Αν το PID είναι η ταχύτητα
      lcd.setCursor(0, 0); // Ορισμός θέσης κειμένου στην αρχή της οθόνης
      lcd.setFontSize(FONT_SIZE_XLARGE); // Ορισμός πολύ μεγάλης γραμματοσειράς
      lcd.print("SPD: "); // Εμφάνιση ετικέτας "SPD"
      lcd.printInt(value1 % 1000, 3); // Εμφάνιση τιμής ταχύτητας (3 ψηφία)
      break;
    case PID_BATTERY_VOLTAGE: // Αν το PID είναι η τάση μπαταρίας
      lcd.setCursor(0, 0); // Ορισμός θέσης κειμένου στην αρχή της οθόνης
      lcd.setFontSize(FONT_SIZE_XLARGE); // Ορισμός πολύ μεγάλης γραμματοσειράς
      lcd.print("VOLT: "); // Εμφάνιση ετικέτας "VOLT"
      lcd.printInt(value1, 2); // Εμφάνιση τιμής τάσης (2 ψηφία)
      lcd.setFontSize(FONT_SIZE_MEDIUM); // Ορισμός μεσαίας γραμματοσειράς
      lcd.print("V"); // Εμφάνιση μονάδας μέτρησης (Volt)
      break;
  }
  switch (pid2) { // Επεξεργασία του δεύτερου PID
    case PID_RPM: // Αν το PID είναι οι στροφές κινητήρα
      lcd.setCursor(0, 5); // Τοποθέτηση κειμένου κάτω από την ταχύτητα
```

```
      lcd.setFontSize(FONT_SIZE_LARGE); // Ορισμός μεγάλης γραμματοσειράς
      lcd.print("RPM: "); // Εμφάνιση ετικέτας "RPM"
      lcd.printInt(value2 % 10000, 4); // Εμφάνιση τιμής RPM (4 ψηφία)
      break;
    case PID_COOLANT_TEMP: // Αν το PID είναι η θερμοκρασία ψυκτικού
      lcd.setCursor(0, 5); // Τοποθέτηση κειμένου κάτω από την τάση
      lcd.setFontSize(FONT_SIZE_LARGE); // Ορισμός μεγάλης γραμματοσειράς
      lcd.print("TEMP: "); // Εμφάνιση ετικέτας "TEMP"
      lcd.printInt(value2 % 100, 3); // Εμφάνιση θερμοκρασίας (3 ψηφία)
      lcd.setFontSize(FONT_SIZE_MEDIUM); // Ορισμός μεσαίας γραμματοσειράς
      lcd.print("C"); // Εμφάνιση μονάδας μέτρησης (Celsius)
      break;
  }
}

// Συνάρτηση για αρχικοποίηση της οθόνης
void initScreen(LCD_SH1106 &lcd, const char *title) {
  lcd.clear(); // Εκκαθάριση της οθόνης
  lcd.setFontSize(FONT_SIZE_SMALL); // Ορισμός μικρής γραμματοσειράς
  lcd.setCursor(0, 0); // Τοποθέτηση στην αρχή της οθόνης
  lcd.print(title); // Εμφάνιση τίτλου
}

// Συνάρτηση αρχικοποίησης
void setup() {
  delay(500); // Αναμονή πριν την έναρξη
  Serial.begin(115200); // Ενεργοποίηση σειριακής επικοινωνίας

#ifdef ARDUINO_ESP32C3_DEV
  Wire.begin(4, 5); // Ορισμός ακίδων SDA και SCL για ESP32-C3
#else
  Wire.begin(); // Χρήση προεπιλεγμένων ακίδων για άλλα boards
#endif
```

```
  // Αρχικοποίηση οθονών
  for (uint8_t bus : {0, 2}) {
    PCA9548A(bus); // Επιλογή καναλιού
    LCD_SH1106 &lcd = (bus == 0) ? lcd1 : lcd2; // Επιλογή οθόνης
    lcd.begin(); // Έναρξη οθόνης
    lcd.setFontSize(FONT_SIZE_MEDIUM); // Ορισμός μεσαίας γραμματοσειράς
    lcd.println("SEARCHING OBD "); // Μήνυμα αναζήτησης OBD
    lcd.println();
    lcd.println("CONNECTING..."); // Μήνυμα σύνδεσης
  }

  // Αρχικοποίηση σύνδεσης OBD-II
  for (;;) {
    obd.begin(); // Έναρξη επικοινωνίας με OBD-II
    if (obd.init()) break; // Αν επιτύχει, έξοδος από το loop
    obd.end(); // Τερματισμός επικοινωνίας
    Serial.print('.'); // Εκτύπωση "." για debugging
    delay(1000); // Αναμονή πριν την επόμενη προσπάθεια
  }

  // Εμφάνιση τίτλων στις οθόνες
  PCA9548A(0);
  initScreen(lcd1, "Screen 1: SPD & RPM");
  PCA9548A(2);
  initScreen(lcd2, "Screen 2: VOLT & TMP");
}

// Κυρίως loop
void loop() {
  int value1, value2;

  // Ενημέρωση οθόνης 1 (Ταχύτητα και RPM)
```

```
  PCA9548A(0);
  if (obd.readPID(PID_SPEED, value1) && obd.readPID(PID_RPM, value2)) {
    showData(PID_SPEED, value1, PID_RPM, value2, lcd1);
  }

  // Ενημέρωση οθόνης 2 (Τάση και Θερμοκρασία ψυκτικού)
  PCA9548A(2);
  if (obd.readPID(PID_BATTERY_VOLTAGE, value1) && obd.readPID(PID_COOLANT_TEMP,
value2)) {
    showData(PID_BATTERY_VOLTAGE, value1, PID_COOLANT_TEMP, value2, lcd2);
  }

  // Αντιμετώπιση σφαλμάτων OBD-II
  if (obd.errors >= 2) {
    reconnect(); // Επανασύνδεση
    setup(); // Επαναφορά συστήματος
  }

  delay(500); // Καθυστέρηση για αποφυγή υπερφόρτωσης
}
```

ΠΑΡΑΡΤΗΜΑ Β : Εναλλακτικός κώδικας συσκευής

```
#include <Wire.h>                  // Εισαγωγή βιβλιοθήκης για επικοινωνία I2C
#include <OBD2UART.h>              // Εισαγωγή βιβλιοθήκης για επικοινωνία OBD-II μέσω UART
#include <SH1106.h>                // Εισαγωγή βιβλιοθήκης για την οθόνη SH1106

LCD_SH1106 lcd;                    // Δημιουργία αντικειμένου LCD για την οθόνη
COBD obd;                          // Δημιουργία αντικειμένου OBD για τη μονάδα OBD-II

void reconnect()
{
  lcd.clear();                     // Καθαρισμός της οθόνης
  lcd.setFontSize(FONT_SIZE_MEDIUM); // Ορισμός μεσαίου μεγέθους γραμματοσειράς
  lcd.print("RECONNECT");          // Εκτύπωση μηνύματος "RECONNECT"

  for (uint16_t i = 0; !obd.init(); i++) { // Προσπάθεια επανασύνδεσης μέχρι να επιτύχει
    if (i == 5) {
      lcd.clear();                 // Καθαρισμός της οθόνης μετά από 5 αποτυχημένες προσπάθειες
    }
    delay(3000);                   // Αναμονή 3 δευτερολέπτων πριν την επόμενη προσπάθεια
  }
}

void showData(byte pid, float value)
{
  switch (pid) {
  case PID_RPM:                    // Αν η τιμή αφορά τις στροφές κινητήρα (RPM)
    lcd.setCursor(64, 0);          // Ορισμός θέσης του κέρσορα
    lcd.setFontSize(FONT_SIZE_XLARGE); // Ορισμός μεγέθους γραμματοσειράς XL
    lcd.printInt((unsigned float)value % 10000, 4); // Εκτύπωση στροφών, μορφοποιημένο σε 4 ψηφία
    break;
  case PID_SPEED:                  // Αν η τιμή αφορά την ταχύτητα
    lcd.setCursor(0, 0);           // Ορισμός θέσης κέρσορα στην πρώτη στήλη
    lcd.setFontSize(FONT_SIZE_XLARGE); // Ορισμός μεγέθους γραμματοσειράς XL
    lcd.printInt((unsigned float)value % 1000, 3); // Εκτύπωση ταχύτητας, μορφοποιημένο σε 3 ψηφία
    break;
```

```
  case PID_COOLANT_TEMP:            // Αν η τιμή αφορά τη θερμοκρασία ψυκτικού υγρού
    lcd.setCursor(88, 5);           // Ορισμός θέσης κέρσορα
    lcd.setFontSize(FONT_SIZE_MEDIUM); // Ορισμός μεσαίου μεγέθους γραμματοσειράς
    lcd.printInt(value % 100, 2);    // Εκτύπωση τελευταίων 2 ψηφίων της θερμοκρασίας
    lcd.setFontSize(FONT_SIZE_MEDIUM); // Επαναφορά μεγέθους γραμματοσειράς
    lcd.print("C");                 // Εκτύπωση της μονάδας "C" (Κελσίου)
    break;
  case PID_BATTERY_VOLTAGE:         // Αν η τιμή αφορά την τάση μπαταρίας
    lcd.setCursor(12, 5);           // Ορισμός θέσης κέρσορα
    lcd.setFontSize(FONT_SIZE_MEDIUM); // Ορισμός μεσαίου μεγέθους γραμματοσειράς
    lcd.printInt(value / 100.0, 3);  // Εκτύπωση τάσης με 3 ψηφία
    lcd.setFontSize(FONT_SIZE_MEDIUM); // Επαναφορά μεγέθους γραμματοσειράς
    lcd.print("V");                 // Εκτύπωση της μονάδας "V" (Volt)
    break;
  }
}

void initScreen()
{
  lcd.clear();                      // Καθαρισμός της οθόνης
  lcd.setFontSize(FONT_SIZE_SMALL);  // Ορισμός μικρού μεγέθους γραμματοσειράς
  lcd.setCursor(24, 3);             // Θέση κέρσορα για την ταχύτητα
  lcd.print("km/h");                // Εκτύπωση "km/h"
  lcd.setCursor(110, 3);            // Θέση κέρσορα για τις στροφές κινητήρα
  lcd.print("rpm");                 // Εκτύπωση "rpm"
  lcd.setCursor(0, 7);              // Θέση κέρσορα για την τάση
  lcd.print("VOLTAGE");             // Εκτύπωση "VOLTAGE"
  lcd.setCursor(80, 7);             // Θέση κέρσορα για τη θερμοκρασία
  lcd.print("TEMP");                // Εκτύπωση "TEMP"
}

void setup()
{
  delay(500);                       // Αναμονή 500 ms κατά την εκκίνηση
  Serial.begin(115200);             // Εκκίνηση σειριακής επικοινωνίας στα 115200 bps

#ifdef ARDUINO_ESP32C3_DEV
```

```
  Wire.begin(4, 5);                // Αρχικοποίηση I2C για ESP32C3
#else
  Wire.begin();                    // Αρχικοποίηση I2C για άλλες πλατφόρμες
#endif

  lcd.begin();                     // Εκκίνηση της οθόνης
  lcd.setFontSize(FONT_SIZE_MEDIUM); // Ορισμός μεσαίου μεγέθους γραμματοσειράς
  lcd.println("SEARCHING OBD");    // Εκτύπωση μηνύματος "SEARCHING OBD"
  lcd.println();
  lcd.println("CONNECTING...");    // Εκτύπωση μηνύματος "CONNECTING..."
  for (;;) {                       // Άπειρη επανάληψη μέχρι να συνδεθεί η OBD
    obd.begin();                   // Έναρξη επικοινωνίας OBD
    if (obd.init()) break;         // Αν η σύνδεση επιτύχει, έξοδος από τη λούπα
    obd.end();                     // Τερματισμός επικοινωνίας αν αποτύχει
    Serial.print('.');             // Εκτύπωση τελείας για ένδειξη προσπάθειας
    delay(1000);                   // Αναμονή 1 δευτερολέπτου πριν την επόμενη προσπάθεια
  }
  initScreen();                    // Αρχικοποίηση της οθόνης με βασικές ενδείξεις
}
void loop()
{
  static byte pids[]= {PID_RPM, PID_SPEED, PID_BATTERY_VOLTAGE,
PID_COOLANT_TEMP}; // Πίνακας PID για ανάγνωση δεδομένων
  static byte index = 0;           // Δείκτης για τον πίνακα PID
  byte pid = pids[index];          // Επιλογή του τρέχοντος PID
  int value;
  // Αποστολή ερώτησης στο OBD adapter για το συγκεκριμένο PID
  if (obd.readPID(pid, value)) {
     showData(pid, value);         // Εμφάνιση των δεδομένων στην οθόνη
  }
  index = (index + 1) % sizeof(pids); // Εναλλαγή στον επόμενο PID κυκλικά

  if (obd.errors >= 2) {           // Έλεγχος για σφάλματα επικοινωνίας
     reconnect();                  // Επανασύνδεση σε περίπτωση σφαλμάτων
     setup();                      // Επανεκκίνηση του setup για επανασύνδεση
  }
}
```

ΠΑΡΑΡΤΗΜΑ C : Κώδικας scanner

```
#include <Wire.h>  // Εισαγωγή βιβλιοθήκης για επικοινωνία I2C

void setup() {
  Serial.begin(115200); // Εκκίνηση σειριακής επικοινωνίας με baud rate 115200
  Wire.begin(4, 5); // Αρχικοποίηση I2C με SDA στη θύρα 4 και SCL στη θύρα 5
  Serial.println("I2C Scanner"); // Εκτύπωση μηνύματος εκκίνησης
}

void loop() {
  Serial.println("Scanning..."); // Εκτύπωση μηνύματος σάρωσης
  for (byte address = 1; address < 127; address++) {// Επαναληπτικός έλεγχος για όλες τις πιθανές διευθύνσεις I2C
    Wire.beginTransmission(address); // Έναρξη επικοινωνίας με τη συγκεκριμένη διεύθυνση
    if (Wire.endTransmission() == 0) {// Αν η επικοινωνία ήταν επιτυχής (endTransmission επιστρέφει 0)
      Serial.print("I2C device found at address 0x"); // Εκτύπωση μηνύματος εύρεσης συσκευής
      Serial.println(address, HEX); // Εκτύπωση της διεύθυνσης σε μορφή δεκαεξαδική
    }
  }
  delay(3000);  // Αναμονή 3 δευτερολέπτων πριν την επόμενη σάρωση
}



#include <Wire.h>  // Εισαγωγή βιβλιοθήκης για επικοινωνία I2C

#define PCA_ADDRESS 0x70 // Η βασική διεύθυνση του πολυπλέκτη PCA9548A

void PcaSelect(uint8_t i) {  // Συνάρτηση για επιλογή καναλιού στον πολυπλέκτη
  if (i > 7) return; // Έλεγχος ότι το κανάλι είναι εντός ορίων (0-7)
  Wire.beginTransmission(PCA_ADDRESS); // Έναρξη επικοινωνίας με τον πολυπλέκτη
  Wire.write(1 << i);  // Ενεργοποίηση του συγκεκριμένου καναλιού
```

```
  Wire.endTransmission(); // Τερματισμός επικοινωνίας
}

void setup() {
  Serial.begin(115200);  // Έναρξη σειριακής επικοινωνίας στα 115200 bps
  Wire.begin(4, 5);  // Έναρξη I2C με SDA στη θύρα 4 και SCL στη θύρα 5
  Serial.println("PCA9548A Scanner"); // Μήνυμα εκκίνησης για τον χρήστη
}

void loop() {
  for (uint8_t channel = 0; channel < 8; channel++) {// Επανάληψη για κάθε κανάλι του πολυπλέκτη
    PcaSelect(channel); // Επιλογή του καναλιού
    Serial.print("Scanning channel: "); // Ενημέρωση χρήστη για το τρέχον κανάλι
    Serial.println(channel);

    for (uint8_t address = 1; address < 127; address++) {// Σάρωση όλων των πιθανών διευθύνσεων I2C
      Wire.beginTransmission(address); // Έναρξη επικοινωνίας με τη συγκεκριμένη διεύθυνση
      if (Wire.endTransmission() == 0) {// Έλεγχος αν υπάρχει συσκευή στη διεύθυνση
        Serial.print("I2C device found at address 0x"); // Μήνυμα αν βρεθεί συσκευή
        Serial.print(address, HEX); // Εκτύπωση της διεύθυνσης σε δεκαεξαδική μορφή
        Serial.print(" on channel ");
        Serial.println(channel); // Εκτύπωση του καναλιού όπου βρέθηκε η συσκευή
      }
    }
  }
  delay(3000);  // Αναμονή 5 δευτερολέπτων πριν την επόμενη σάρωση
}
```

ΠΑΡΑΡΤΗΜΑ D : Κώδικας για OBD UART

```
#include <Arduino.h> // Εισαγωγή της βασικής βιβλιοθήκης για Arduino

// Χρονικά όρια για την επικοινωνία μέσω OBD-II
#define OBD_TIMEOUT_SHORT 1000 /* ms */ // Σύντομο χρονικό όριο (π.χ., για μικρές λειτουργίες)
#define OBD_TIMEOUT_LONG 10000 /* ms */ // Μεγάλο χρονικό όριο (π.χ., για πιο αργές εντολές)

#ifndef OBDUART
// Ελέγχει αν το OBDUART έχει οριστεί ήδη και κάνει αντιστοίχιση της σειριακής θύρας
#if defined(__AVR_ATmega328P__) || defined(__AVR_ATmega168P__)
#define OBDUART Serial // Χρήση Serial για AVR-based boards όπως Arduino Uno
#else
#define OBDUART Serial1 // Χρήση Serial1 για πλατφόρμες με περισσότερες θύρες UART
#endif
#endif

#ifdef ESP32
extern HardwareSerial Serial1; // Για ESP32, χρήση της Serial1
#endif

// Mode 1 PIDs (Προσδιορισμοί αναγνωριστικών παραμέτρων OBD-II)
#define PID_ENGINE_LOAD 0x04 // Φορτίο κινητήρα
#define PID_COOLANT_TEMP 0x05 // Θερμοκρασία ψυκτικού υγρού
#define PID_SHORT_TERM_FUEL_TRIM_1 0x06 // Σύντομη ρύθμιση καυσίμου (τράπεζα 1)
// Παρόμοιες τιμές PID για άλλες παραμέτρους, όπως στροφές κινητήρα, ταχύτητα, θερμοκρασίες, κ.λπ.
// ...

// Ορισμοί μη OBD PID για ειδικές χρήσεις
#define PID_ACC 0x20 // Επιτάχυνση
#define PID_GYRO 0x21 // Γυροσκόπιο
#define PID_COMPASS 0x22 // Πυξίδα
#define PID_MEMS_TEMP 0x23 // Θερμοκρασία αισθητήρα MEMS
#define PID_BATTERY_VOLTAGE 0x24 // Τάση μπαταρίας
```

```
// Πρωτόκολλα OBD-II
typedef enum {
   PROTO_AUTO = 0, // Αυτόματη ανίχνευση πρωτοκόλλου
   PROTO_ISO_9141_2 = 3, // Πρωτόκολλο ISO 9141-2
   PROTO_CAN_11B_500K = 6, // Πρωτόκολλο CAN με 11-bit ID και 500 kbps ταχύτητα
   // Άλλα πρωτόκολλα OBD-II
} OBD_PROTOCOLS;

// Καταστάσεις σύνδεσης OBD
typedef enum {
   OBD_DISCONNECTED = 0, // Αποσυνδεδεμένο
   OBD_CONNECTING = 1, // Σύνδεση σε εξέλιξη
   OBD_CONNECTED = 2, // Συνδεδεμένο
   OBD_FAILED = 3 // Αποτυχία σύνδεσης
} OBD_STATES;

// Υποστηρικτικές συναρτήσεις για μετατροπή τιμών από δεκαεξαδικό σε ακέραιο
uint16_t hex2uint16(const char *p); // Μετατροπή 16-bit από δεκαεξαδικό
uint8_t hex2uint8(const char *p); // Μετατροπή 8-bit από δεκαεξαδικό

// Κλάση COBD: Υλοποιεί τις λειτουργίες επικοινωνίας OBD-II
class COBD
{
public:
   virtual byte begin(); // Έναρξη της UART επικοινωνίας
   virtual void end(); // Τερματισμός της επικοινωνίας
   virtual bool init(OBD_PROTOCOLS protocol = PROTO_AUTO); // Εγκατάσταση σύνδεσης OBD-
II
   virtual void reset(); // Επαναφορά σύνδεσης
   virtual void uninit(); // Απεγκατάσταση σύνδεσης
   virtual bool setBaudRate(unsigned long baudrate); // Ρύθμιση baud rate
   virtual OBD_STATES getState() { return m_state; } // Επιστροφή κατάστασης σύνδεσης
   virtual bool readPID(byte pid, int& result); // Ανάγνωση τιμής για συγκεκριμένο PID
   virtual byte readPID(const byte pid[], byte count, int result[]); // Ανάγνωση πολλών PID
   virtual void enterLowPowerMode(); // Ενεργοποίηση λειτουργίας χαμηλής ισχύος
   virtual void leaveLowPowerMode(); // Απενεργοποίηση λειτουργίας χαμηλής ισχύος
```

```
    virtual byte sendCommand(const char* cmd, char* buf, byte bufsize, int timeout =
OBD_TIMEOUT_LONG); // Αποστολή εντολής AT
    virtual byte readDTC(uint16_t codes[], byte maxCodes = 1); // Ανάγνωση διαγνωστικών κωδικών
βλαβών (DTC)
    virtual void clearDTC(); // Εκκαθάριση διαγνωστικών κωδικών
    virtual float getVoltage(); // Λήψη τάσης μπαταρίας
    virtual bool getVIN(char* buffer, byte bufsize); // Λήψη VIN οχήματος
    virtual bool memsInit(bool fusion = false); // Αρχικοποίηση MEMS αισθητήρων
    virtual bool memsRead(int16_t* acc, int16_t* gyr = 0, int16_t* mag = 0, int16_t* temp = 0); //
Ανάγνωση δεδομένων MEMS
    virtual bool memsOrientation(float& yaw, float& pitch, float& roll); // Ανάγνωση
προσανατολισμού
    virtual bool getResult(byte& pid, int& result); // Ανάγνωση και ανάλυση δεδομένων PID
    virtual bool isValidPID(byte pid); // Έλεγχος αν το PID είναι έγκυρο
    virtual byte getVersion(); // Έκδοση firmware
    byte dataMode = 1; // Τρέχουσα λειτουργία δεδομένων
    byte errors = 0; // Καταγραφή σφαλμάτων
protected:
    virtual char* getResponse(byte& pid, char* buffer, byte bufsize); // Ανάκτηση απάντησης
    virtual int receive(char* buffer, int bufsize, unsigned int timeout = OBD_TIMEOUT_SHORT); //
Λήψη δεδομένων
    OBD_STATES m_state = OBD_DISCONNECTED; // Αρχική κατάσταση
};
```

ΠΑΡΑΡΤΗΜΑ Ε : Κώδικας για SH1106

```
#include <Arduino.h> // Εισαγωγή της βασικής βιβλιοθήκης του Arduino

//#define MEMORY_SAVING
// Αν ενεργοποιηθεί, μπορεί να μειώσει την κατανάλωση μνήμης με κόστος λιγότερες δυνατότητες.

typedef enum {
   FONT_SIZE_SMALL = 0, // Μικρό μέγεθος γραμματοσειράς
   FONT_SIZE_MEDIUM,    // Μεσαίο μέγεθος γραμματοσειράς
   FONT_SIZE_LARGE,     // Μεγάλο μέγεθος γραμματοσειράς
   FONT_SIZE_XLARGE     // Πολύ μεγάλο μέγεθος γραμματοσειράς
} FONT_SIZE;

#define FLAG_PAD_ZERO 1 // Σημαία για συμπλήρωση με μηδενικά
#define FLAG_PIXEL_DOUBLE_H 2 // Σημαία για διπλασιασμό οριζόντιων pixel
#define FLAG_PIXEL_DOUBLE_V 4 // Σημαία για διπλασιασμό κάθετων pixel
#define FLAG_PIXEL_DOUBLE (FLAG_PIXEL_DOUBLE_H | FLAG_PIXEL_DOUBLE_V) //
Διπλασιασμός pixel και στους δύο άξονες

// Εξωτερικά δεδομένα γραμματοσειρών και αριθμητικών ψηφίων
extern const unsigned char font5x8[][5]; // 5x8 γραμματοσειρά
extern const unsigned char digits8x8[][8]; // 8x8 ψηφία
extern const unsigned char digits16x16[][32]; // 16x16 ψηφία
extern const unsigned char digits16x24[][48]; // 16x24 ψηφία
extern const unsigned char font8x16_doslike[][16]; // DOS-like 8x16 γραμματοσειρά
extern const unsigned char font8x16_terminal[][16]; // Terminal-like 8x16 γραμματοσειρά

// Κοινή κλάση για LCD (κοινά χαρακτηριστικά και λειτουργίες)
class LCD_Common
{
public:
   LCD_Common():m_font(FONT_SIZE_SMALL),m_flags(0) {} // Αρχικοποίηση με μικρή
γραμματοσειρά και χωρίς σημαίες
   void setFontSize(FONT_SIZE size) { m_font = size; } // Ρύθμιση μεγέθους γραμματοσειράς
   void setFlags(byte flags) { m_flags = flags; } // Ρύθμιση σημαιών για την οθόνη
```

```
    virtual void backlight(bool on) {} // Εικονική συνάρτηση για ενεργοποίηση/απενεργοποίηση οπίσθιου φωτισμού
    virtual void draw(const byte* buffer, byte width, byte height) {} // Εικονική συνάρτηση για σχεδίαση εικόνας
    void printInt(uint16_t value, int8_t padding = -1); // Εκτύπωση ακέραιου αριθμού με πιθανή συμπλήρωση
    void printLong(uint32_t value, int8_t padding = -1); // Εκτύπωση μεγάλου ακέραιου αριθμού
protected:
    virtual void writeDigit(byte n) {} // Εικονική συνάρτηση για εκτύπωση ψηφίου
    byte m_font; // Τρέχον μέγεθος γραμματοσειράς
    byte m_flags; // Σημαίες για την οθόνη
};

// Κλάση για την οθόνη SH1106
class LCD_SH1106 : public LCD_Common, public Print
{
public:
    void begin(); // Έναρξη της οθόνης
    void setCursor(byte column, byte line); // Ορισμός θέσης κέρσορα
    void draw(const byte* buffer, byte width, byte height); // Σχεδίαση εικόνας
    size_t write(uint8_t c); // Γράψιμο χαρακτήρα
    void clear(byte x = 0, byte y = 0, byte width = 128, byte height = 64); // Καθαρισμός μέρους ή όλης της οθόνης
    void clearLine(byte line); // Καθαρισμός γραμμής
    byte getLines() { return 21; } // Επιστροφή αριθμού γραμμών (υποθετική ανάλυση 21 γραμμών)
    byte getCols() { return 8; } // Επιστροφή αριθμού στηλών (υποθετική ανάλυση 8 στηλών)
private:
    void WriteCommand(unsigned char ins); // Αποστολή εντολής στην οθόνη
    void WriteData(unsigned char dat); // Αποστολή δεδομένων στην οθόνη
    void writeDigit(byte n); // Γράψιμο αριθμητικού ψηφίου
    byte m_col; // Τρέχουσα στήλη
    byte m_row; // Τρέχουσα γραμμή
};
```