ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ
ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ
Τμήμα Ηλεκτρολόγων & Ηλεκτρονικών Μηχανικών
www. eee. uniwa. gr
Θηβών 250, Αθήνα-Αιγάλεω 12244
Τηλ. +30 210 538-1225, Fax. +30 210 538-1226

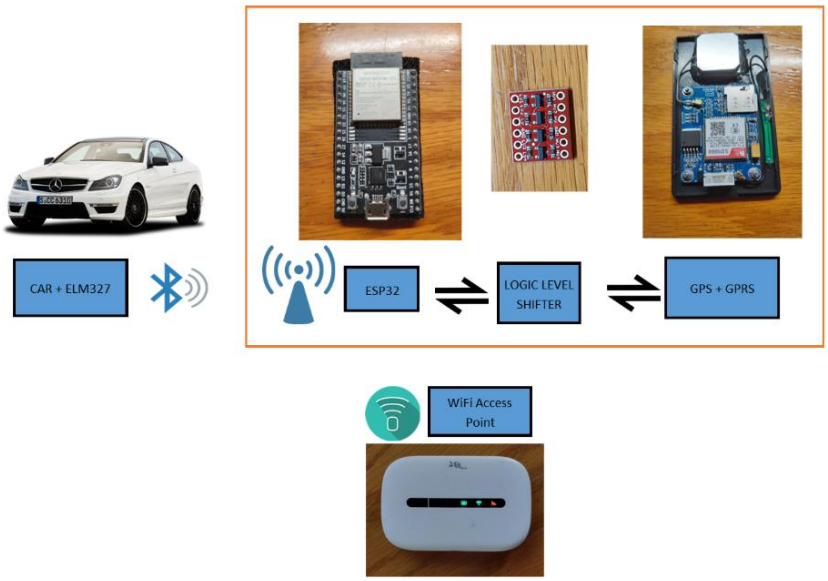
UNIVERSITY OF WEST ATTICA
FACULTY OF ENGINEERING
Department of Electrical & Electronics Engineering
www. eee. uniwa. gr
250, Thivon Str. , Athens, GR-12244, Greece
Tel:+30 210 538-1225, Fax:+30 210 538-1226

Πρόγραμμα Μεταπτυχιακών Σπουδών
Επικοινωνίες και Δίκτυα Δεδομένων

Master of Science in
Data Communications and Networking

ΜΕΤΑΠΤΥΧΙΑΚΗ ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Σύστημα Τηλεμετρίας Οχημάτων



Μεταπτυχιακός Φοιτητής: Αθανάσιος Βλάχος, **ΑΜ:** dcom-01

Επιβλέπων: Δρ. Γρηγόριος Κουλούρας, Αναπληρωτής Καθηγητής

ΑΘΗΝΑ, ΜΑΡΤΙΟΣ 2022

ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ
ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ
Τμήμα Ηλεκτρολόγων & Ηλεκτρονικών Μηχανικών

www. eee. uniwa. gr
Θηβών 250, Αθήνα-Αιγάλεω 12244
Τηλ. +30 210 538-1225, Fax. +30 210 538-1226

UNIVERSITY OF WEST ATTICA
FACULTY OF ENGINEERING
Department of Electrical & Electronics Engineering

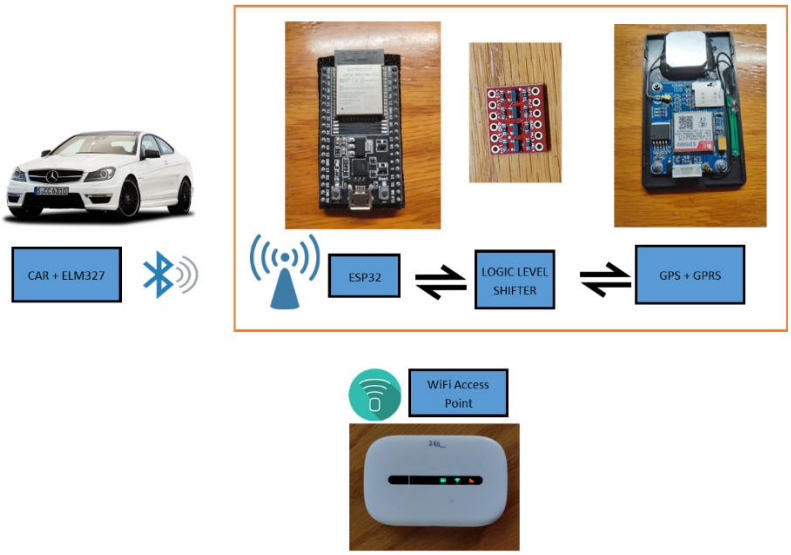
www. eee. uniwa. gr
250, Thivon Str. , Athens, GR-12244, Greece
Tel:+30 210 538-1225, Fax:+30 210 538-1226

Πρόγραμμα Μεταπτυχιακών Σπουδών
Επικοινωνίες και Δίκτυα Δεδομένων

Master of Science in
Data Communications and Networking

MSc Thesis

A Vehicle Telemetry System



MSc Student: Athanasios Vlachos, **RN:** dcom-01

Supervisor: Dr. Grigorios Koulouras, Associate Professor

ATHENS, MARCH 2021

ΔΗΛΩΣΗ ΠΕΡΙ ΠΝΕΥΜΑΤΙΚΩΝ ΔΙΚΑΙΩΜΑΤΩΝ ΚΑΙ ΛΟΓΟΚΛΟΠΗΣ

Με πλήρη επίγνωση των συνεπειών του νόμου περί πνευματικών δικαιωμάτων, δηλώνω ενυπόγραφα ότι η παρούσα εργασία προετοιμάστηκε και ολοκληρώθηκε από εμένα αποκλειστικά και ότι είμαι ο αποκλειστικός συγγραφέας του κειμένου της.

Η εργασία μου δεν προσβάλλει οποιασδήποτε μορφής δικαιώματα πνευματικής ιδιοκτησίας, προσωπικότητας ή προσωπικών δεδομένων τρίτων, δεν περιέχει έργα/εισφορές τρίτων για τα οποία απαιτείται άδεια των δημιουργών/δικαιούχων και δεν είναι προϊόν μερικής ή ολικής αντιγραφής ή λογοκλοπής.

Κάθε βοήθεια που έλαβα για την ολοκλήρωση της εργασίας είναι αναγνωρισμένη και αναφέρεται λεπτομερώς στο κείμενό της. Ειδικότερα, έχω αναφέρει ευδιάκριτα μέσα στο κείμενο και με την κατάλληλη παραπομπή όλες τις πηγές δεδομένων, κώδικα προγραμματισμού Η/Υ, απόψεων, θέσεων και προτάσεων, ιδεών και λεκτικών αναφορών που χρησιμοποιήθηκαν, είτε κατά κυριολεξία είτε βάσει επιστημονικής παράφρασης, και η σχετική αναφορά περιλαμβάνεται στο τμήμα των βιβλιογραφικών αναφορών με πλήρη περιγραφή. Επιπλέον, όλες οι πηγές που χρησιμοποιήθηκαν περιορίζονται στις βιβλιογραφικές αναφορές και μόνον και πληρούν τους κανόνες της επιστημονικής παράθεσης κατά τα διεθνή πρότυπα.

Τέλος δηλώνω ενυπόγραφα ότι αναλαμβάνω πλήρως, ατομικά και προσωπικά, όλες τις νομικές και διοικητικές συνέπειες στην περίπτωση κατά την οποία αποδειχθεί, διαχρονικά, ότι η εργασία αυτή ή τμήμα της είναι προϊόν λογοκλοπής.

Ημερομηνία, 28/03/2022
Digitally signed by ATHANASIOS VLACHOS
Date: 2022.03.28 18:59:17 +03'00'
Αθανάσιος Βλάχος

Η Διπλωματική Εργασία έγινε αποδεκτή και βαθμολογήθηκε από την εξής τριμελή επιτροπή:

Γρηγόριος Κουλούρας, Αναπληρωτής Καθηγητής	Ευάγγελος Ζέρβας, Καθηγητής	Σωτήριος Καραμπέτσος, Αναπληρωτής Καθηγητής
Digitally signed by Grigorios Koulouras DN: c=GR, o=University of West Attica, ou=Class A - Private Key created and stored in hardware CSP, sn=Koulouras, givenName=Grigorios, serialNumber=5281068459, cn=Grigorios Koulouras, email=gregkoul@uniwa.gr Date: 2022.03.28 19:05:19 +03'00'	Evangelos Zervas Digitally signed by Evangelos Zervas Date: 2022.03.28 19:16:23 +03'00'	Sotirios Karampetsos Digitally signed by Sotirios Karampetsos Date: 2022.03.28 23:37:51 +03'00'
(Υπογραφή)	(Υπογραφή)	(Υπογραφή)

Αφιέρωση

Στη μνήμη της μητέρας μου,
Ευγενίας

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον καθηγητή μου κ. Γρηγόριο Κουλούρα, που κατά την διάρκεια της εκπόνηση της διπλωματικής μου εργασίας, προσέφερε την πολύτιμη καθοδήγηση του. Θα ήθελα επίσης να ευχαριστήσω τον συμφοιτητή μου Κωσταντίνο Δ., που προσέφερε το αυτοκίνητο του για της πρακτικές δοκιμές του συστήματος.

Περίληψη

Στην παρούσα Διπλωματική Εργασία έχει αναπτυχθεί ένα σύστημα εποπτείας της λειτουργίας των σύγχρονων οχημάτων που υποστηρίζουν την διαγνωστική υποδοχή On-Board Diagnostics (OBD2). Για την υλοποίηση του πρωτότυπου συστήματος, έχει αναπτυχθεί όχι μόνο κατάλληλο λογισμικό, αλλά και εξειδικευμένο υλικό, που τοποθετείται στο υπό επιτήρηση όχημα. Τα δεδομένα αφού αρχικά υποστούν κατάλληλη προεπεξεργασία τοπικά, αποστέλλονται σε μια πλατφόρμα υπολογιστικού νέφους. Η πλατφόρμα αυτή, δίνει τη δυνατότητα στον εξουσιοδοτημένο χρήστη του συστήματος, να επιτηρεί τόσο το όχημα όσο και την συμπεριφορά του οδηγού του.

Λέξεις – κλειδιά

Τηλεμετρία, Νεφοϋπολογιστική, OBD2, ThingsBoard.io, ESP32

Abstract

In the present dissertation, a system of monitoring the operation of modern vehicles that support the On-Board Diagnostics (OBD2) socket has been developed. For the implementation of the innovative system, not only suitable software has been developed, but also specialized hardware, which is placed in the under-supervision vehicle. The data, after being properly pre-processed locally, are sent to a cloud computing IoT platform. This platform enables the authorized user of the system to monitor both the vehicle and the behavior of its driver.

Keywords

Telemetry, Cloud Computing, OBD2, ThingsBoard.io, ESP32

Περιεχόμενα

Πίνακας Εικόνων

Ακρωνύμια

Acronym	Definition
ABS	Anti-lock Braking System
CAN BUS	Controller Area Network Bus
ECU	Engine Control Unit
GPRS	General Packet Radio Service
GPS	Global Positioning System
HTTP	HyperText Transfer Protocol
IaaS	Infrastructure as a Service
IEEE	Institute of Electrical and Electronics Engineers
IoT	Internet of Things
IT	Information Technology
MQTT	Message Queuing Telemetry Transport
OASIS	Organization for the Advancement of Structured Information Standard
OBD2	On-Board Diagnostics 2
SCADA	Supervisory Control And Data Acquisition
SIM	Subscriber Identity Module
SMS	Short Message Service
WiFi	Wireless Fidelity

Κεφάλαιο 1 - Εισαγωγή

Η καθημερινά αυξανόμενη χρήση του διαδικτύου, αλλά και η εξάπλωσή του 5G, έχει δημιουργήσει μια άμεση εξάρτηση του ανθρώπου από αυτό, σε βαθμό που κατά κάποιο τρόπο, αποτελεί προϊόν πρώτης ανάγκης. Οι περισσότεροι μας πλέον, και ειδικότερα οι νέοι, δεν μπορούν να φανταστούν τον κόσμο χωρίς διαδίκτυο. Καθημερινά όλο και περισσότερες συσκευές συνδέονται είτε με άμεσο είτε με έμμεσο τρόπο, στο διαδίκτυο.

Το λεγόμενο Διαδίκτυο των Πραγμάτων (Internet of Things - IoT) έχει μπει για τα καλά στην καθημερινότητα μας. Το να ανοιγοκλείνουμε μια λάμπα, έναν αφυγραντήρα, ή να εκτυπώνουμε στον εκτυπωτή του γραφείου από οπουδήποτε στον κόσμο, δεν είναι πλέον κάτι ασυνήθιστο. Σε κάποιους ακόμα έχει δημιουργηθεί η ανάγκη να ελέγχουν το φωτισμό, την θέρμανση του σπιτιού τους, ακόμα και να βάζουν σε λειτουργία το πλυντήριο των ρούχων τους από το κινητό, ακόμα και όταν δεν είναι στο σπίτι.

Η δυνατότητες του Διαδικτύου των Πραγμάτων δεν είναι μόνο ο απομακρυσμένος έλεγχος, αλλά και η τηλεμετρία. Μπορεί κάποιος να παρακολουθεί για παράδειγμα τη θερμοκρασία των ψυγείων στο μαγαζί του και να ειδοποιείται όταν η θερμοκρασία ξεπεράσει κάποιο προκαθορισμένο επίπεδο. Μπορεί να παρακολουθεί την υγρασία που έχει ο κήπος ή και το χωράφι του και να ρυθμίζει αντίστοιχα το πότισμα.

1.1 Περιγραφή προβλήματος

Τα ηλεκτρονικά συστήματα στα οχήματα αυξάνονται συνεχώς. Πλέον, ακόμα και τα πιο οικονομικά οχήματα, είναι εφοδιασμένα με δεκάδες αισθητήρες που παρακολουθούν την καλή του λειτουργία. Είτε αυτό είναι ένα οικονομικό ηλεκτρικό πατίνι, είτε είναι ένα πανάκριβο σπορ ηλεκτρικό αυτοκίνητο, υπάρχει η δυνατότητα, μέσω αυτών των αισθητήρων, να αυτοπαρακολουθείται και να προσαρμόζεται δυναμικά σε αλλαγές ανάλογα με τις συνθήκες λειτουργίας.

Αυτό επιτυγχάνεται με την χρήση υπολογιστών. Η πιο συχνή υλοποίηση, είναι με χρήση πολλών μικροϋπολογιστών εντός του οχήματος, με τον κάθε ένα από αυτούς, να έχει και ένα πολύ συγκεκριμένο σκοπό. Παράδειγμα ένας για την μηχανή, ένας για την ασφάλεια του επιβάτη (αερόσακοι), ένας για το σύστημα αντιμπλοκαρίσματος τροχών (Anti-lock Braking System – ABS), το σύστημα αυτοματισμών κ.α. Το σημαντικότερο πλεονέκτημα που έχει αυτή η υλοποίηση, είναι ότι μειώνεται ο όγκος των πληροφοριών που επεξεργάζεται το κάθε υποσύστημα, και κατά συνέπεια αυξάνεται η αξιοπιστία του εκάστοτε μικροϋπολογιστή.

Συνήθως, όλοι αυτοί οι μικροϋπολογιστές, επικοινωνούν μεταξύ τους, μέσω ενός πρωτοκόλλου επικοινωνίας που ονομάζεται Controller Area Network Bus (CAN BUS). Σε όλα σχεδόν τα οχήματα, μετά το 2000, υπάρχει μια διαγνωστική υποδοχή που ονομάζεται

On-Board Diagnostics 2 (OBD2) και επιτρέπει την πρόσβαση σε αυτό το κανάλι επικοινωνίας με χρήση εξειδικευμένου εξοπλισμού.

Με χρήση αυτού του εξειδικευμένου εξοπλισμού, μπορεί κάποιος να έχει πρόσβαση σε πολλούς από αυτούς τους αισθητήρες και να δει από την θερμοκρασία και της στροφές της μηχανής, μέχρι τις μοίρες που είναι γυρισμένο το τιμόνι. Το σημαντικότερο όλων, είναι ότι οι μικροϋπολογιστές έχουν την δυνατότητα να κάνουν διάγνωση μιας βλάβης, με ένα πολύ μεγάλο ποσοστό επιτυχίας.

Υπάρχουν πληθώρα συσκευών, που μπορεί ο καθένας να χρησιμοποιήσει για να έχει πρόσβαση σε πολλούς από αυτούς τους αισθητήρες του οχήματός του. Αυτές οι συσκευές του παρέχουν την δυνατότητα να κάνει ακόμα και ανάγνωση των βλαβών που έχουν συμβεί στο όχημά του. Παρόλα αυτά, η χρήση τους περιορίζεται στην προσωρινή τοποθέτηση της συσκευής, σε αυτή την διαγνωστική υποδοχή OBD2 του οχήματος. Επιπλέον, υπάρχει πρόσβασή στα δεδομένα αυτά μόνο τοπικά και για όσο η συσκευή αυτή είναι τοποθετημένη.

1.2 Σκοπός και στόχοι

Ο σκοπός της παρούσας διπλωματικής είναι ο σχεδιασμός ενός συστήματος που θα μπορεί να συλλέγει αδιάλειπτα τα δεδομένα αυτά και να τα προωθεί σε μια πλατφόρμα IoT μέσω διαδικτύου, καθιστώντας εφικτή την συνεχή εποπτεία ενός οχήματος ή ακόμα και ενός στόλου οχημάτων. Θα παρέχει την δυνατότητα στον εξουσιοδοτημένο χρήστη να βλέπει μέσω τηλεμετρίας, επιλεγμένες παραμέτρους του οχήματος, όπως για παράδειγμα, η ταχύτητά του.

Ο εξουσιοδοτημένος χρήστης θα μπορεί επίσης να παρακολουθεί την οδική συμπεριφορά του οδηγού (π.χ. αν οδηγεί επιθετικά). Επίσης, παρακολουθώντας και συγκρίνοντας τις επιδόσεις του οχήματος με αντίστοιχες από το ιστορικό του οχήματος είναι εφικτός ο εγκυρότερος εντοπισμός μιας βλάβης προτού ακόμα εντοπιστεί από τα συστήματα του αυτοκινήτου.

Τα βασικά υλικά που χρησιμοποιήθηκαν είναι τα ακόλουθα:

1) ELM327
Είναι μια μικρή συσκευή που συνδέεται στην διαγνωστική θύρα του αυτοκινήτου. Έχει ουσιαστικά τον ρόλο ενός διερμηνέα (interpreter), ανάμεσα στο CAN BUS του οχήματος και μιας σειριακής θύρας. Συνήθως είναι εξοπλισμένο με κάποιο ασύρματο Interface, είτε με Bluetooth, είτε με WiFi για ευκολότερη διαχείριση.
2) ESP32
Πρόκειται για ένα μικροελεγκτή εφοδιασμένο με WiFi και Bluetooth. Μπορεί να προγραμματιστεί είτε σε C++ είτε σε Python. Η επεξεργαστική ισχύς του και η μνήμη του το καθιστούν ιδανικό υποψήφιο ακόμα και για περίπλοκες εργασίες.

Επιπλέον, για τη διευκόλυνση της ανάπτυξης και λόγω έλλειψης συμβατού οχήματος, έχει κατασκευαστεί και μια συσκευή προσομοίωσης ενός εγκεφάλου αυτοκινήτου, που σκοπός του είναι να ξεγελάσει το ELM327, έτσι ώστε να μην είναι απαραίτητη η χρήση αυτοκινήτου στο στάδιο των δοκιμών.

1.3 Δομή της εργασίας

Το παρών κεφάλαιο αποτελεί την εισαγωγή της Διπλωματικής Εργασίας. Στο δεύτερο κεφάλαιο γίνεται μια βιβλιογραφική ανασκόπηση για τα υπάρχοντα μέσα εποπτείας οχημάτων, καθώς και μια σύντομη περιγραφή των τεχνολογιών που έχουν χρησιμοποιηθεί. Ακολουθεί το τρίτο κεφάλαιο με την υλοποίηση του ενσωματωμένου συστήματος. Γίνεται περιγραφή του κάθε εξαρτήματος που χρησιμοποιήθηκε, καθώς και σε τι θα ωφελεί η χρήση του. Αναλύεται ο τρόπος διασύνδεσής τους και ο τρόπος λειτουργίας τους. Στο τέταρτο κεφάλαιο αναλύεται η υλοποίηση της πλατφόρμας του Διαδικτύου των Πραγμάτων στην υπολογιστική υποδομή νέφους. Ακολουθεί το πέμπτο κεφάλαιο που παρουσιάζει το σύστημα σε λειτουργία και αναλύονται τα τελικά συμπεράσματα. Τέλος, στο έκτο κεφάλαιο συζητούνται μελλοντικές εργασίες που θα μπορούσαν να βελτιώσουν το σύστημα.

Κεφάλαιο 2 - Βιβλιογραφική μελέτη

2.1 Υπάρχουσες υλοποιήσεις

Ο πιο απλός τρόπος παρακολούθησης οχημάτων είναι με μια απλή συσκευή με κάρτα Subscriber Identity Module (SIM) οπού όταν λάβει αίτημα μέσο (Short Message Service) SMS στέλνει την κατά προσέγγιση θέση του με βάση την θέση τις κεραίας κινητής τηλεφωνίας οπού είναι συνδεμένη. Τα ακριβότερα μοντέλα έχουν επιπλέον πραγματικό δέκτη GPS και επιτρέπουν την παρακολούθηση της θέσης του οχήματος σε πραγματικό χρόνο.

2.2 Παρούσα προσέγγιση

Στην συγκεκριμένη υλοποίηση, γίνεται χρήση ενός μικροελεγκτή, πιο συγκεκριμένα ενός ESP32, που αποτελεί και την ¨καρδιά¨ ολόκληρου του συστήματος. Ο μικροελεγκτής φέρει ενσωματωμένο WiFi και Bluetooth όπου χρησιμοποιούνται για επικοινωνία με την πλατφόρμα IoT στο υπολογιστικό νέφος, καθώς και έμμεσα, μέσο της διεπαφής του ELM327, με το CAN BUS του αυτοκινήτου. Για λόγους ευκολίας ως προς τον προγραμματισμό και την ευκολότερη αποσφαλμάτωση και δοκιμή του, έχει επίσης υλοποιηθεί ένας προσομοιωτής εγκεφάλου αυτοκινήτου (Engine Control Unit – ECU), όπου καθιστά εφικτή την ανάπτυξη του κώδικα, χωρίς να απαιτείται η χρήση και η δοκιμή του σε πραγματικό αυτοκίνητο.

2.3 Τεχνολογίες - Πρωτόκολλα που χρησιμοποιηθήκαν

2.3.1 WiFi

Το WiFi είναι ένα σύνολο από ασύρματα διαδικτυακά πρωτόκολλα βασισμένα στην οικογένεια του IEEE 802.11 και χρησιμοποιείται από συσκευές που απαιτούν πρόσβαση στο διαδίκτυο. Πρόκειται για το πιο ευρέως διαδεδομένο πρωτόκολλο ασύρματης δικτύωσης το οποίο χρησιμοποιείται παγκόσμια από οικιακές εφαρμογές, μέχρι δίκτυα πολλών χιλιομέτρων.

Η κάθε συσκευή, όπως κινητά, φορητοί υπολογιστές, εκτυπωτές, ακόμα και έξυπνες λάμπες φωτισμού, μπορούν να συνδεθούν σε ένα κοινό σημείο πρόσβασης (access point), αποκτώντας έτσι πρόσβαση στο διαδίκτυο. (Wi-Fi Alliance, 2021)

Το WiFi είναι το εμπορικό λογότυπο του μη κερδοσκοπικού οργανισμού WiFi Alliance, ο οποίος περιορίζει την χρήση του σε προϊόντα τα οποία έχουν περάσει την διαδικασία πιστοποίησης συμβατότητας με τα Standards, καθώς και τους περιορισμούς που έχει θεσπίσει η εκάστοτε χώρα (Regulations).

Η σχεδίασή του χρησιμοποιεί πολλαπλά μέρη από την οικογένεια των πρωτοκόλλων IEEE 802 και είναι φτιαγμένο, με τρόπο ώστε να λειτουργεί απρόσκοπτα με το ενσύρματο μέρος του δικτύου, το Ethernet. Οι συμβατές συσκευές μπορούν να περάσουν από το ασύρματο μέρος του δικτυού στο ενσύρματο καθώς και σε άλλα ασύρματα δίκτυα, αλλά και στο Internet με σχετικά μεγάλη ευκολία.

Εικόνα 1: Το λογότυπο του WiFi

Οι διαφορετικές εκδόσεις του WiFi, αναγράφονται ως διάφορα 802.11 πρωτόκολλα, με διαφορετικές ραδιοτεχνικές μετάδοσης, όπου επηρεάζουν την ταχύτητα, την λειτουργική απόσταση και την συχνότητα λειτουργίας. Το εκάστοτε κανάλι μπορεί να διαμοιράζεται μεταξύ δυο ή περισσοτέρων δικτύων αλλά μόνο μια συσκευή μπορεί να μεταδίδει την εκάστοτε χρονική στιγμή.

Οι συχνότητες που χρησιμοποιούνται από το WiFi έχουν τυπικά μεγάλη απορρόφηση και δουλεύουν καλύτερα όταν υπάρχει οπτική επαφή. Πολλά κοινά αντικείμενα όπως οι τοίχοι και οι οικιακές συσκευές μπορούν να μειώσουν δραστικά την εμβέλεια λειτουργίας, αλλά σε περιπτώσεις με πολλαπλά σημεία προσβάσεις τα εμπόδια αυτά μπορούν να βοηθήσουν στην μείωση των παρεμβολών. Η τυπική εμβέλεια ενός σημείου πρόσβαση είναι τα 20 μέτρα σε εσωτερικούς χώρους και τα 150 μέτρα με οπτική επαφή. Η νέες τεχνολογίες του WiFi μπορούν να φτάσουν θεωρητικές ταχύτητές τα 9.6Gbps σε αντίθεση με τον προκάτοχό του στα 3.5Gbps. (Wi-Fi Alliance, 2021)

2.3.2 Bluetooth

Το Bluetooth είναι μια ασύρματη τεχνολογία κοντινής εμβέλειας ανταλλαγής δεδομένων μεταξύ σταθερών και φορητών συσκευών στην UHF μπάντα. Πιο συγκεκριμένα λειτουργεί στις συχνότητες από 2402MHz έως 2480MHz. Η αρχική του χρήση ήταν να μετατρέψει το ενσύρματο σειριακό πρωτόκολλο RS-232 σε ασύρματο. Τώρα χρησιμοποιείται ευρύτατα σαν εναλλακτική σύνδεση, λόγω της καλύτερης σύνδεσης και ποιότητας, για την ανταλλαγή μικρών αρχείων μεταξύ κοντινών φορητών συσκευών και την ασύρματη σύνδεση περιφερειακών όπως ακουστικά με κινητό τηλέφωνο. Στις πιο συνηθισμένες χρήσεις του η

ισχύς εκπομπής είναι περιορισμένη στα 2.5 mW, δίνοντάς του έτσι πολύ κοντινή απόσταση λειτουργίας (10μ).

Εικόνα 2: Το λογότυπο του Bluetooth

Η τεχνολογία αυτή επιβλέπεται από τον οργανισμό τυποποίησης Bluetooth Special Interest Group, στην οποία υπάρχουν πάνω από 36.000 εταιρίες από τον χώρο τον τηλεπικοινωνιών, υπολογιστών, δικτύων, καθώς και καταναλωτικών ηλεκτρονικών (consumer electronics) Στην IEEE το Bluetooth αναφέρετε ως 802.15.1. Η Bluetooth SIG παρακολουθεί την ανάπτυξη των προδιαγραφών, διαχειρίζεται της απαίτησης του και προστατεύει το λογότυπο του. Ο εκάστοτε κατασκευαστής πρέπει να πληροί όλες τις προϋποθέσεις που έχουν οριστεί προκειμένου μια συσκευή να θεωρηθεί συμβατή και να λάβει το λογότυπο. (SIG, 2021)

2.3.3 MQTT

Το MQTT (Message Queuing Telemetry Transport) είναι ένα ελαφρύ στην χρήση διαδικτυακό πρωτόκολλο αποστολής μηνυμάτων μεταξύ δικτυακών συσκευών και συσκευών IoT. Είναι σχεδιασμένο ώστε να είναι εξαιρετικά ελαφρύ στην χρήση και να απαιτεί ελάχιστο κώδικα για να είναι εφικτή η χρήση του ακόμα και σε μικροελεγκτές.

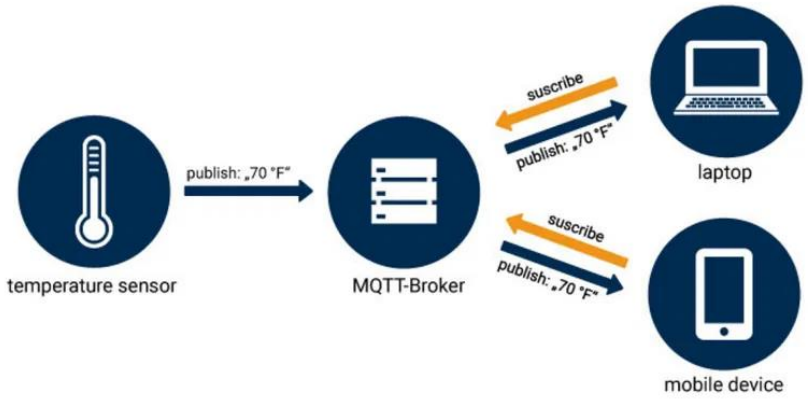
Εικόνα 3: Το λογότυπο του MQTT

Είναι εφικτό να λειτουργήσει πάνω από οποιαδήποτε πρωτόκολλο επικοινωνίας που δεν έχει απώλειες αν και το πιο διαδεδομένο είναι το TCP/IP. Ο τρόπος κατασκευής του επιτρέπει να επιτυγχάνει καλή λειτουργία ακόμα και σε κακές συνθήκες δικτύου κινητής τηλεφωνίας. Διαθέτει πολλαπλά επίπεδα λειτουργίας που διασφαλίζουν την επικοινωνία. Είναι πλέον ένα πρωτόκολλο ανοιχτού κώδικα του οργανισμού OASIS.

Το MQTT χρησιμοποιήθηκε πρώτη φορά το 1999 για την παρακολούθηση εγκαταστάσεων αγρού πετρελαίου ενός δικτύου Supervisory Control And Data Acquisition (SCADA), ενός συστήματος παρακολούθησης, ελέγχου και συλλογής δεδομένων. Ο στόχος ήταν να έχουν ένα χαμηλής ενεργειακής κατανάλωσης και αποδοτικό από πλευράς δεδομένων πρωτόκολλο,

λόγω του ότι οι αισθητήρες τροφοδοτούνταν από μπαταρία και τα δεδομένα στέλνονταν μέσω εξαιρετικά ακριβών δορυφορικών ζεύξεων.

Το πρωτόκολλο απαιτεί 3 οντότητες για να λειτουργήσει. Έναν broker όπου έχει τον ρόλο του server, και τουλάχιστον έναν publisher και έναν subscriber. Ο Publisher ¨δημοσιεύει¨ τα δεδομένα που συλλέγονται τοπικά (message) και τα στέλνει στο server μαζί με ένα θέμα ενδιαφέροντος (topic). Από την άλλη οι Subscriber που ενδιαφέρονται για ένα συγκεκριμένο θέμα, κάνουν εγγραφή στο συγκεκριμένο κανάλι (topic) του Broker, προκειμένου να το παρακολουθούν. Ο Broker προωθεί τα μηνύματα σε όλους τους Subscribers που ενδιαφέρονται για το εκάστοτε θέμα. Επισημαίνεται ότι οι συσκευές μπορούν να λειτουργούν διπλά, και σαν Publisher αλλά και σαν Subscriber.



Εικόνα 4: Τυπικό σχήμα λειτουργίας του MQTT

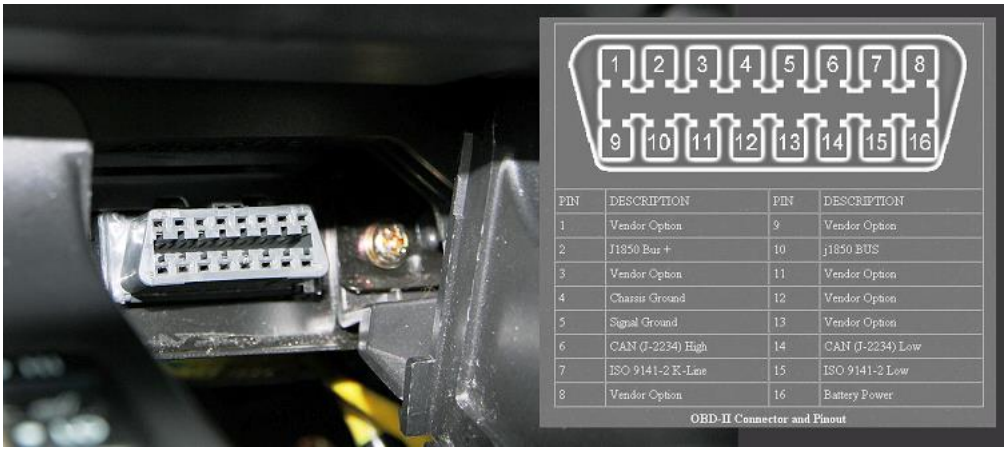
2.3.4 OBD2

Το OBD2 είναι ακρωνύμιο των λέξεων On-Board Diagnostics 2 (Διάγνωση εντός οχήματος). Όπως προκύπτει από και από το όνομά του, είναι η δεύτερη έκδοση που έγινε standard και άρχισε να εφαρμόζεται σε όλα τα οχήματα από της αρχές του 2000. Μέχρι τότε ο κάθε κατασκευαστής έφτιαχνε την δική του υποδοχή (interface) και πρωτόκολλο και απαιτούσε ειδικευμένο εξοπλισμό, μερικές φορές, διαφορετικό ανάμεσα σε οχήματα ακόμα και του ίδιου κατασκευαστή.

Το OBD2 προσφέρει την δυνατότητα συνδεσιμότητας με τον εσωτερικό δίαυλο επικοινωνίας όλων των ενσωματωμένων υποσυστημάτων ενός αυτοκινήτου ή φορτηγού. Ο εσωτερικός αυτός δίαυλος επικοινωνίας στα περισσότερα αυτοκίνητα είναι το CAN BUS και επιτρέπει την ανάγνωση πληροφοριών για την λειτουργία του, όπως είναι για παράδειγμα η ταχύτητά του και η θερμοκρασία του κινητήρα. Οι δυνατότητές του δεν περιορίζονται μόνο στην ανάγνωση πληροφοριακών δεδομένων.

Η κυρίως χρήση του είναι η ευκολότερη και εγκυρότερη αναγνώριση μια βλάβης σε ένα όχημα, καθώς και η ενημέρωση των προγραμμάτων και παραμέτρων λειτουργίας του κάθε υποσυστήματος.

Ένα καλό παράδειγμα είναι αν κάποιος αλλάξει ελαστικά ή και τις ζάντες με άλλες διαφορετικής διατομής από τις εργοστασιακές, να μπορεί να ενημερωθεί το ABS για την αποδοτικότερη και ασφαλέστερη λειτουργία του, καθώς και να προβάλει στο ταχύμετρο την πραγματική ταχύτητα, και κατά συνέπεια να μετρά σωστά τα χιλιόμετρα που έχει διανύσει το αυτοκίνητο.

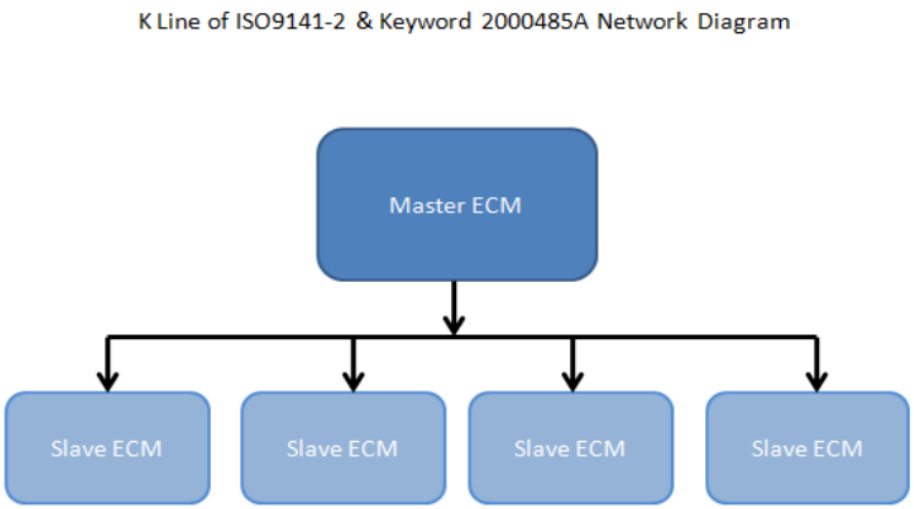


Εικόνα 5: Η θύρα OBD2 σε IX

2.3.5 K-Line (ISO-9141-2)

Το πρωτόκολλο K-Line είναι ένα αργό σειριακό πρωτόκολλο επικοινωνίας ενός καλωδίου που χρησιμοποιούταν από την αυτοκινητοβιομηχανία σε παντός τύπου οχήματα. Η χρήση του συνήθως είναι μεταξύ των ηλεκτρονικών μονάδων ελέγχου (ECUs) και των διαγνωστικών υποδοχών εντός των οχημάτων. Το K-Line είναι δικτυακό πρωτόκολλο βασισμένο στης προδιαγραφές του ISO9141 της California Air Resources Board.

Διαφέρει πολύ από το πρωτόκολλο του CAN BUS, και γενικά από τα περισσότερα επικοινωνιακά δίκτυα παρόμοιας χρήσης. Στο CAN BUS για παράδειγμα δεν υπάρχει κεντρική ή πρωτεύουσα μονάδα ελέγχου. Όλες οι μονάδες είναι ισάξιες και είναι ικανές τόσο να λάβουν όσο και να στείλουν μηνύματα.



Εικόνα 6: Διάγραμμα του δικτύου K-Line

Σε αντίθεση, το K-Line ή οποιοδήποτε δίκτυο συμβατό με της προδιαγραφές του ISO9141, η κατεύθυνση των μηνυμάτων είναι ζωτικής σημασίας. Ο χειρισμός του δικτύου γίνεται αποκλειστικά από την πρωτεύουσα μονάδα ελέγχου (Master ECU). Η κατεύθυνση των μηνυμάτων, ο χρονισμός τους καθώς και ποια μονάδα ελέγχου μεταδίδει και ποια λαμβάνει ελέγχεται από την πρωτεύουσα μονάδα. Αυτό συνεπάγει ότι μόνο μια μονάδα ελέγχου μπορεί να μεταδίδει ανά πάσα χρονική στιγμή, με την διαφορά του ότι θα πρέπει πρώτα να της επιτραπεί να μεταδώσει. (Ashraf Tahat, 2012).

Πλέον το K-Line δεν έχει πια βασικό ρόλο σε νέα οχήματα λόγω τις αντικατάστασής του από άλλα πρωτόκολλά όπως το CAN BUS και το Ethernet όπου έχουν αναλάβει πλήρως τα ινία στον τομέα της διάγνωσης των αυτοκινήτων. Παρόλα αυτά οι κατασκευαστές των οχημάτων καθώς και τα συνεργεία παγκοσμίως είναι υποχρεωμένα να διαθέτουν συμβατό εξοπλισμό λόγω του ότι είναι πολύ συχνό τα οχήματα που κυκλοφορούν να είναι εφοδιασμένα με αυτό το πλέον απαρχαιωμένο πρωτόκολλο.

Ακόμα και σήμερα, παρά την παλαιότητα και των περιορισμών του, παράγονται νέα οχήματα, κυρίως μηχανές, που φέρουν ακόμα το παλιό αυτό πρωτόκολλο. (Decker Peter, 2015)

2.3.6 Python

Η Python είναι μια υψηλού επιπέδου και γενικής χρήσης γλώσσα προγραμματισμού. Η βασική της φιλοσοφία είναι εστιάζει στον ευανάγνωστο κώδικα και στην ευκολία χρήσης. Λειτουργεί διαμέσου ενός διερμηνέα (interpreter) που σημαίνει ότι ένα πρόγραμμα διαβάζει και εκτελεί απευθείας τις εντολές από την γλώσσα προγραμματισμού, χωρίς να είναι απαιτούμενο να έχει γίνει μεταγλώττιση το αρχείο σε γλώσσα μηχανής.

Παρόλα αυτά είναι πιο αργή γλώσσα σε σύγκριση με μεταγλωττιζόμενες (compiled) όπως η C++, γεγονός που την καθιστά ακατάλληλη για χρήση σε αργούς microcontrollers και λειτουργικά συστήματα. Ο Guido van Rossum άρχισε την ανάπτυξή της στα τέλη τις δεκαετίας του 80 σαν διάδοχος τις γλώσσας ABC, και η πρώτη της έκδοση ήταν το 1991 ως Python 0.9.0. Η βασικότερες εκδόσεις της, είναι η έκδοση 2 και η 3, το 2000 και το 2008 αντίστοιχα.

Αναπτύσσεται από την Python Software Foundation ως μια γλώσσα ανοιχτού κώδικα. Το όνομα τις προέρχεται από την κωμική σειρά *Monty Python's Flying Circus* και δεν έχει σχέση με το φίδι παρότι το λογότυπο τις μοιάζει με αυτό. (Python Software Foundation FAQ, 2021)

Εικόνα 7: Το λογότυπο τις Python

2.3.7 C++

Η C++ είναι μια γλώσσα γενικού προγραμματισμού από τον Bjarne Stroustrup ως προέκταση τις C το 1985. Από την αρχική της έκδοση, έχουν προστεθεί νέες δυνατότητες όπως η αντικειμενοστρέφια. Σχεδόν πάντα είναι απαραίτητη η χρήση κάποιου μεταγλωττιστή και πολλοί προμηθευτές λογισμικού, όπως η Microsoft και η Oracle, καθώς και πολλές τρίτες εταιρίες παρέχουν ειδικά εργαλεία αυτού του σκοπού.

Η γλώσσα αυτή είναι εστιασμένη στον προγραμματισμό ενσωματωμένων συστημάτων, καθώς και συστημάτων με περιορισμένους πόρους. Έχει εφαρμογές σε πολλούς κλάδους όπου η απόδοση και η ευελιξία τις εφαρμογής αποτελούν βασικό παράγοντα. Μπορεί να χρησιμοποιηθεί από το να κάνει ένα λαμπάκι να αναβοσβήνει, μέχρι και την δημιουργία ολόκληρων βιντεοπαιχνιδιών.

Εικόνα 8: Το λογότυπο της C++

2.4 Υπηρεσίες υπολογιστικού Νέφους

Ο «Ωκεανός» είναι μια υπηρεσία από το Εθνικό Δίκτυο Υποδομών Τεχνολογίας και Έρευνας, ή GRNET. Η κυρίως υπηρεσία που παρέχει είναι υποδομή IaaS (Infrastructure as a Service, Υποδομή ως Υπηρεσία) για την δημιουργία και την φιλοξενία εικονικών υπολογιστών (Virtual Machines). Αυτοί οι υπολογιστές παραμένουν 24/7 συνδεδεμένοι στο διαδίκτυο, χωρίς ο χρήστης να ανησυχεί για το υλικό ή το λογισμικό, παρέχοντας απρόσκοπτα τις υπηρεσίες τους. Επίσης παρέχει την δυνατότητα φιλοξενίας αρχείων (υπηρεσία pithos+) δίνοντας στον χρήστη τη δυνατότητα να αποθηκεύει στο cloud μεγάλο όγκο αρχείων.

2.5 Συνοπτικά

Συνοψίζοντας, το WiFi δίνει την δυνατότητα στο σύστημα να αποστέλλει δεδομένα στο διαδίκτυο καθιστώντας εφικτή την επικοινωνία με το ThingsBoard. Το Bluetooth εξασφαλίζει την συνδεσιμότητα με το ELM327 και συνεπώς με το όχημα.

Το πρωτόκολλο επικοινωνίας του MQTT προσφέρει την εύκολη άλλα και σταθερή αποστολή των δεδομένων στην εφαρμογή. Η διαγνωστική υποδοχή OBD2 που υπάρχει σε όλα τα σύγχρονα οχήματα προσφέρει την δυνατότητα της επικοινωνίας με τους εκάστοτε μικροϋπολογιστές του οχήματος.

Στην γλώσσα Python έχει αναπτυχτεί κώδικας για τον έλεγχο της ορθής λειτουργίας του ThingsBoard, καθώς μιμείται τον τρόπο αποστολής των δεδομένων που μπορεί να κάνει το ESP32. Τέλος η C++ είναι η γλώσσα που έχει χρησιμοποιηθεί για το ESP32.

Κεφάλαιο 3 – Υλοποίηση ενσωματωμένου συστήματος

3.1 Εισαγωγή

Στο παρόν κεφάλαιο θα γίνει μια σύνοψη του υλικού που έχει χρησιμοποιηθεί, και το πώς αυτό είναι διασυνδεμένο με το υπόλοιπο υλικό. Θα ακολουθήσει μια συνοπτική περιγραφή του software που χρησιμοποιήθηκε για τον προγραμματισμό του καθώς και παραδείγματα κώδικα και δοκιμών του συστήματος.

3.2 Hardware που χρησιμοποιήθηκε

3.2.1 EML327

Είναι ένας προ-προγραμματισμένος microcontroller από την ELM Electronics. Το module αυτό μεταφράζει τα πρωτόκολλα που τρέχουν στην διαγνωστική υποδοχή του οχήματος σε σειριακά μηνύματα μέσω ενός σετ εντολών που παρέχει, η οποία λειτουργία είναι και η πιο διαδεδομένη. Η πιο κοινή του έκδοση είναι με Serial over Bluetooth, ενώ υπάρχει και σε παραλλαγές με RS232,WiFi και USB.

Η αρχική υλοποίηση του ELM327 βασιζόταν σε ένα microcontroller της Microchip Technology, τον PIC18F2480, όπου και σε αυτήν την έκδοση δεν ήταν ενεργοποιημένη η προστασία αντιγραφής στον κώδικα, σαν αποτέλεσμα σήμερα είναι εξαιρετικά δύσκολο να βρεθεί γνήσιος, λόγω τις εξαιρετικά μεγάλης πειρατείας του λογισμικού. Η χρήση του είναι απαραίτητη για την επικοινωνία με το τον CAN BUS του οχήματος.

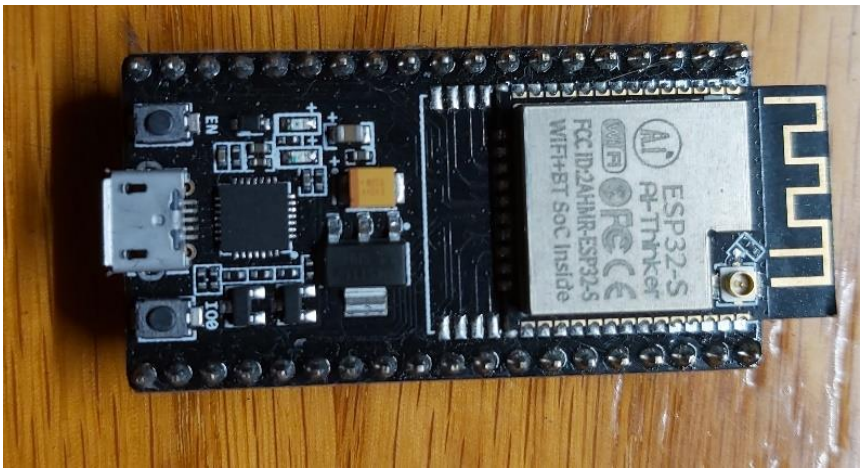
Εικόνα 9: ELM327 interface

3.2.2 ESP32

Ο ESP32 είναι μια μικρή αλλά πολύ ισχυρή αναπτυξιακή πλατφόρμα. Ενσωματώνει δύο δυνατούς, για τον σκοπό που προορίζεται, επεξεργαστές και φέρει τεχνολογίες ασύρματης δικτύωσης WiFi και Bluetooth, με αρκετά μεγάλη μνήμη Flash.

Σε συνδυασμό με το μικρό του κόστος, είναι ιδανικό για IoT εφαρμογές λόγω του ότι δεν απαιτείται ούτε η χρήση εξωτερικής μνήμης, αλλά ούτε και η παρουσία εξειδικευμένου module για την δικτύωσή του. Μπορεί να συνδεθεί άμεσα σε οποιοδήποτε οικιακό ασύρματο δικτύο. Διαθέτει ακόμα 4 κρυπτογραφικές μονάδες που επιταχύνουν τις διαδικασίες ασφαλούς επικοινωνίας είτε με το υπόλοιπο hardware είτε προς το διαδίκτυο.

Χρησιμοποιείται ως η βασική μονάδα επεξεργασίας των δεδομένων. Σε αυτήν τρέχει το πρόγραμμα που έχει αναπτυχθεί και γενικά αποτελεί την καρδιά του συστήματος.



Εικόνα 10: ESP32

3.2.3 Mobile WiFi Access Point

Είναι μια φορητή και μικρή συσκευή που δημιουργεί ένα μικτό ασύρματο δίκτυο WiFi. Περιέχει μια κάρτα SIM, και μέσω του δικτύου κινητής τηλεφωνίας παρέχει πρόσβαση στο διαδίκτυο. Είναι απαραίτητη για την αποστολή των δεδομένων στο cloud.

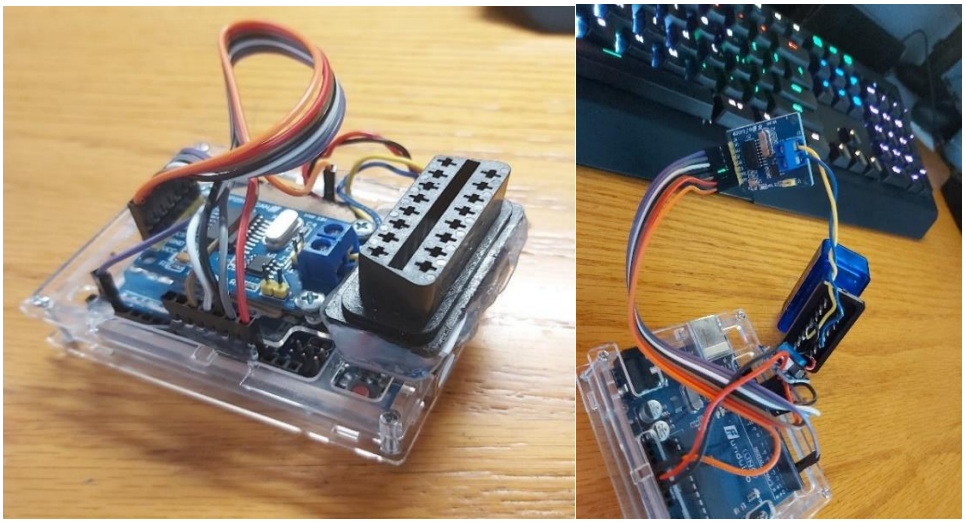
Εικόνα 11: WiFi Access Point

3.2.4 ECU simulator

Λόγω έλλειψης συμβατού αυτοκινήτου, αλλά και για την απλοποίηση της ανάπτυξης του κώδικα, κατασκευάστηκε συσκευή που μπορεί να μιμείται ορισμένα από τα μηνύματα που ανταλλάσσονται μεταξύ του αυτοκινήτου και του ELM327, όπως για παράδειγμα η ταχύτητα του και η στροφές του κινητήρα.

Το Simulator αποτελείται από το Arduino, ένα MCP2515_SPItoCAN interface, και ένα boost converter που παίρνει 5V από το USB και το μετατρέπει σε 13.8V, για την τροφοδοσία του OBD2 port.

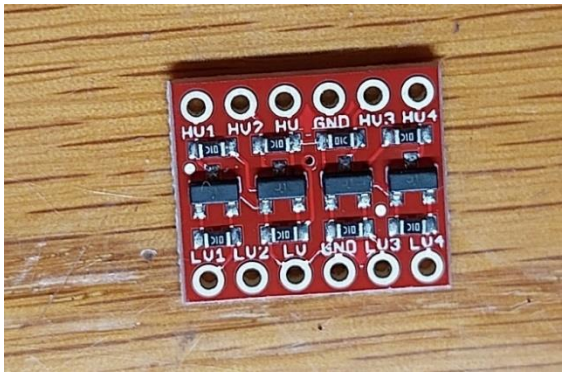
Η κατασκευή και η χρήση του είναι αποκλειστικά για την ευκολότερη και ασφαλέστερη ανάπτυξη του κώδικα, καθώς δεν γίνεται χρήση πραγματικού αυτοκινήτου.



Εικόνα 12: ECU simulator

3.2.6 Logic Level Shifter

Το GPS module και το ESP32 δεν είναι συμβατά αναμεταξύ τους. Ο λόγος είναι η διαφορετική λογική στάθμη τους (logic level), στα 5V το GPS και στα 3,3V το ESP32. Για να επιτευχθεί η επικοινωνία ανάμεσα τους, χρησιμοποιήθηκε το παρακάτω module.

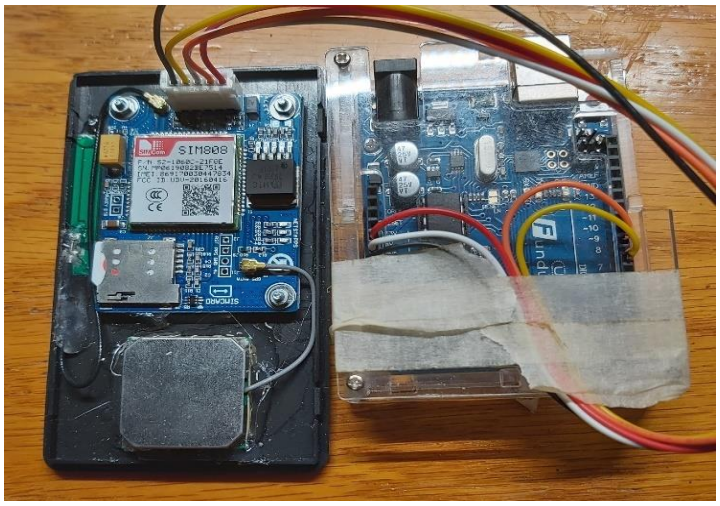


Εικόνα 13: Logic level shifter

3.2.5 GPS + GPRS module (SIM808)

Το SIM808 είναι μια συσκευή που ενσωματώνει ένα δέκτη GPS και GPRS. Μπορεί να λαμβάνει και να στέλνει δεδομένα μέσω του δικτύου κινητής τηλεφωνίας και να ενημερώνει την συνδεμένη συσκευή για την ακριβή του τοποθεσία. Στην φωτογραφία βλέπουμε το module τοποθετημένο στην βάση του κατά την διάρκεια δοκιμαστικής λειτουργείας με το Arduino.

Στην παρούσα φάση της διπλωματικής χρησιμοποιείται μόνο σαν GPS, λόγω του ότι η βιβλιοθήκη που παρέχει για το GPRS παρουσιάζει ασυμβατότητα με το συγκεκριμένο μοντέλο του EPS32.



Εικόνα 14: GPS + GPRS module συνδεμένο με Arduino

3.3 Software που χρησιμοποιήθηκε

Η αρχική ανάπτυξη του κώδικα έγινε στο Arduino 1.8.1. Λόγω κακής συμβατότητας με τα module της ESP, έγινε μετάβαση στο Visual Studio με το πρόσθετο PlatformIO IDE.

3.3.1 Arduino IDE 1.8.1

Το Arduino IDE είναι μια εφαρμογή που σου επιτρέπει να γράψεις και να ¨ανεβάσεις¨ κώδικα σε συμβατές με αυτό συσκευές. Με την βοήθεια από πρόσθετα του προγράμματος η συμβατότητα του δεν σταματά σε συμβατές Arduino συσκευές αλλά και σε άλλες από διάφορους κατασκευαστές. Η εφαρμογή είναι ανοιχτού κώδικα και υπάγεται στην όρους της GPL 2.

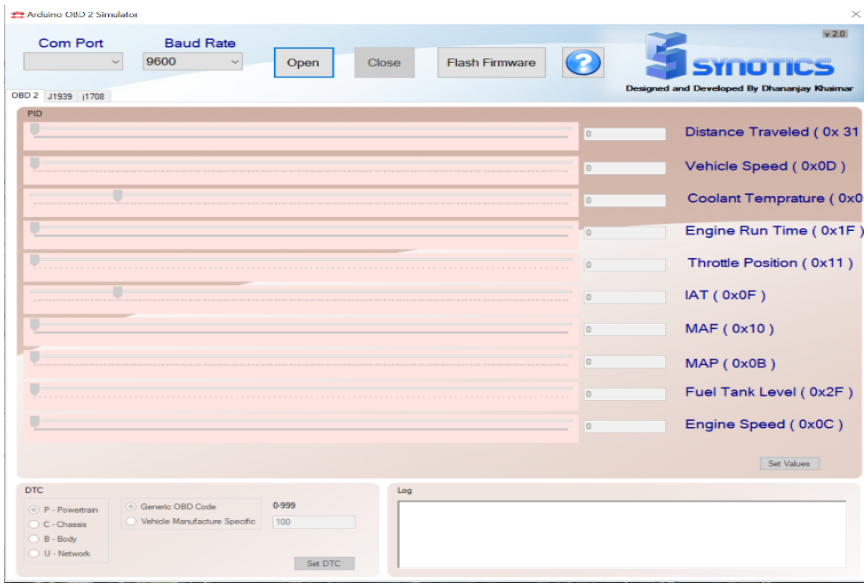
3.3.2 Visual Studio Code με PlatformIO

Το Visual Studio Code είναι ένα εργαλείο επεξεργασίας πηγαίου κώδικα της Microsoft που μπορεί να χρησιμοποιηθεί σε πολλές γλώσσες προγραμματισμού, όπως η java, η Python και η C++. Οι δυνατότητες του περιλαμβάνουν: αποσφαλμάτωση , υποδείξεις συντακτικού, χρωματισμός του κώδικα με βάση τη λειτουργία του, αλλά και δυνατότητες συγχρονισμού με Git Repositories, όπως για παράδειγμα το GitHub.

Το PlatformIO είναι ένα πρόσθετο εργαλείο για την ανάπτυξη κώδικα σε ενσωματωμένα συστήματα. Παρέχει όλα τα απαραίτητα εργαλεία που για την ανάπτυξη, αποσφαλμάτωση και ανέβασμα του κώδικα σε αναπτυξιακές πλακέτες και ενσωματωμένα συστήματα.

3.4 Περίγραμμα υλοποίησης

Μέσω του προγράμματος του simulator καταχωρούνται οι παράμετροι ενός οχήματος. Τοποθετώντας πάνω το ELM327, η συνδεμένη σε αυτό συσκευή 'πιστεύει' ότι έχει συνδεθεί σε πραγματικό αυτοκίνητο. Σε πραγματικό σενάριο, ο χρήστης θα τοποθετεί το ELM327 στην διαγνωστική υποδοχή του οχήματος. Στην εικόνα φαίνεται το παράθυρο του προγράμματος προτού περαστούν οι παράμετροι στην συσκευή του Arduino. Το πρόγραμμα αυτό είναι από τον χρήστη του διαδικτύου με το ψευδώνυμο ZenElectro (ZenElectro, 2019)



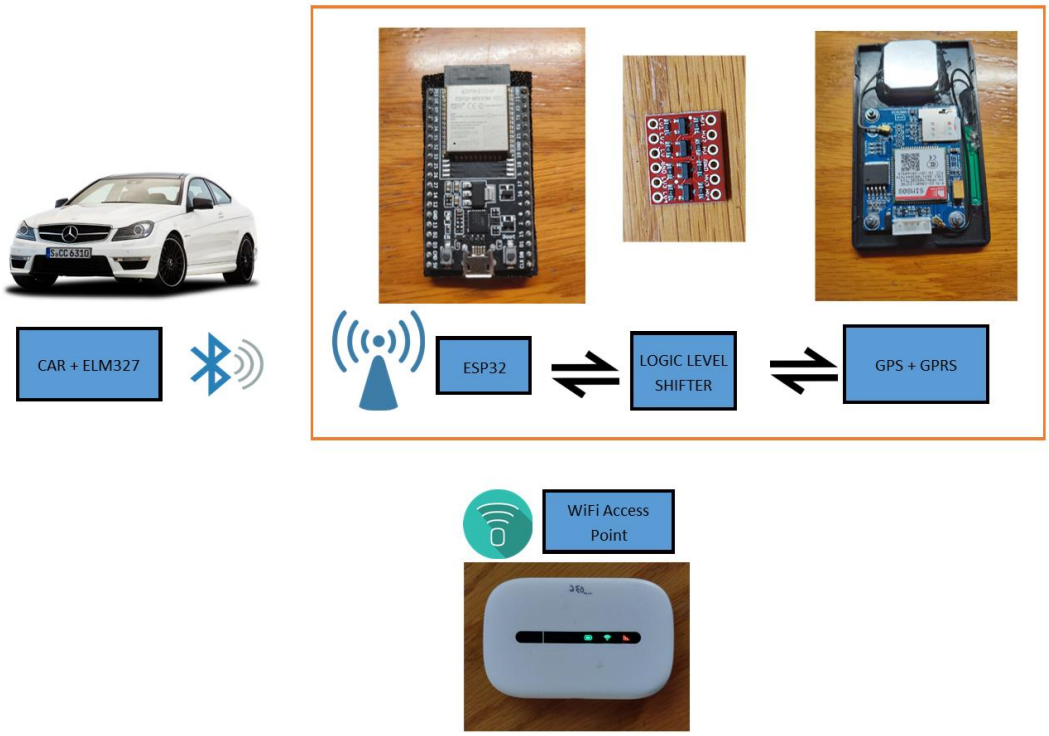
Εικόνα 15: Το πρόγραμμα του simulator

Το ESP32 όταν ανοίξει ο διακόπτης του οχήματος, συνδέεται μέσω Bluetooth στο ELM327 και ξεκινά την αποστολή τον δεδομένων στο υπολογιστικό νέφος, είτε με την βοήθεια του Access Point, είτε μέσο κάρτας SIM που τοποθετείτε στο SIM808 module.

Η τροφοδοσία πρέπει να προέρχεται από το Accessory Bus του οχήματος έτσι ώστε να σταματά η λειτουργία της συσκευής, όταν το όχημα δεν είναι σε λειτουργία. Τα 12V μετατρέπονται σε 5V από ένα step-down buck converter και τροφοδοτούν το ESP32 και το SIM808.

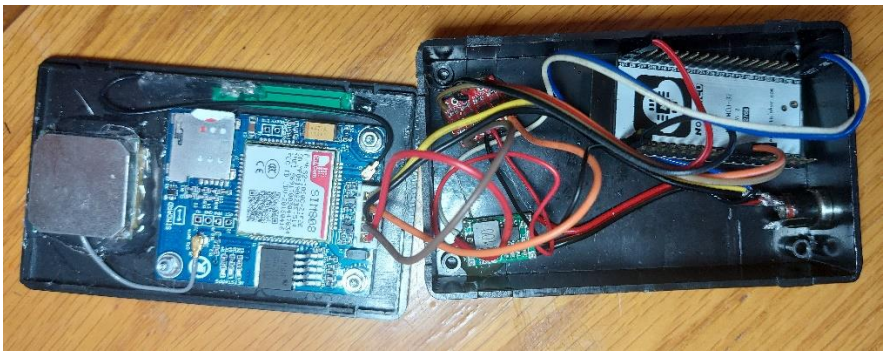
Η επικοινωνία μεταξύ του οχήματος και του ESP32 επιτυγχάνεται μέσω Bluetooth, ενώ η επικοινωνία ανάμεσα στα δυο module γίνεται σειριακά. Λόγω του ότι τα Logic Levels ανάμεσα στις 2 συσκευές είναι διαφορετικά, είναι απαραίτητή η χρήση ενός module, ενός logic level shifter που επιτρέπει την σύνδεση 2 συσκευών διαφορετικού επιπέδου.

Για το ανέβασμα των δεδομένων στο cloud μπορεί να χρησιμοποιηθεί είτε κάποιο WiFi hotspot ή το GPRS που περιλαμβάνετε στο SIM808.



Εικόνα 16: Περίγραμμα υλοποίησης hardware

Το τελικό πρωτότυπο, πριν κλειστεί στο κουτί του, είναι το ακόλουθο. Είναι διακριτά όλα του τα κομμάτια. Στο καπάκι έχει τοποθετηθεί το GPS module, ενώ μέσα στο κουτί το ESP32 με το logic level shifter, καθώς και ο μετατροπέας τάσης και ο ακροδέκτης τροφοδοσίας.



Εικόνα 17: Το ολοκληρωμένο project

3.5 Η Ολοκλήρωση του συστήματος

Ο κώδικας έχει αναπτυχθεί με την πλατφόρμα PlatformIO που βασίζεται στο Visual Studio Code. Ακολουθεί μια συνοπτική επεξήγηση του πως δουλεύει ο κώδικας, και των βιβλιοθηκών, που είναι τέσσερις στο σύνολο.

Η βιβλιοθήκη 'BluetoothSerial' είναι υπεύθυνη για την επικοινωνία μεταξύ του ESP32 και του ELM327. Διαχειρίζεται την ασύρματη επικοινωνία μεταξύ των δύο συσκευών, ενώ η 'HardwareSerial' διαχειρίζεται την ενσύρματη επικοινωνία ανάμεσα στο GPS και το ESP32.

Η βιβλιοθήκη 'WiFi' είναι αυτή που συνδέει το ESP32 με το διαδίκτυο, ενώ ή 'ThingsBoard' είναι υπεύθυνη για την αποστολή των δεδομένων στο Server με συμβατό για αυτόν τρόπο.

Για την διαχείριση της συσκευής ELM327 δεν έχει χρησιμοποιηθεί κάποια βιβλιοθήκη, αντιθέτως όλες οι εντολές για την εκκίνηση και την λειτουργία της στέλνονται διαδοχικά μια προς μια. Κάθε φορά που πραγματοποιείται επικοινωνία, γράφεται στην Bluetooth σειριακή το ερώτημα και η απάντησή της αποκωδικοποιείται με βάση το τι έχει ερωτηθεί. Αυτό θα μπορούσε να είναι πρόβλημα αν το ELM327 έστελνε από μόνο του δεδομένα, αλλά στην περίπτωσή μας, η συσκευή αποστέλλει μηνύματα μόνο όταν λάβει κάποιο αίτημα.

Με την έναρξη τις λειτουργίας το ESP32 ξεκινά την αρχικοποίηση του GPS, στέλνοντας απλά τις απαραίτητες εντολές στην σειριακή του θύρα. Έπειτα ξεκινά το ELM327 με ακριβώς τον ίδιο τρόπο, αλλά χρησιμοποιώντας το Bluetooth. Τελευταία ξεκινά η σύνδεση με το WiFi, όπου για απροσδιόριστους λόγους το συγκεκριμένο ESP αρνείται την λειτουργία του Bluetooth και του WiFi ταυτόχρονα, αν ξεκινήσει πρώτα το WiFi.

Στο βασικό βρόγχο του προγράμματος γίνονται περιοδικά ερωτήματα προς το όχημα σχετικά με την κατάσταση του. Τα ερωτήματα αυτά είναι η ταχύτητά του, η στροφές της μηχανής, θερμοκρασία κινητήρα, το επίπεδο καυσίμων και την τάση τις μπαταρίας. Μετά από κάθε κύκλο ερωτημάτων προς το όχημα, αποστέλλονται και οι συντεταγμένες από το GPS.

3.6 Συνοπτικά

Το σύστημα αποτελείται από 6 εξαρτήματα. Το ELM327 είναι αυτό που επικοινωνεί με το όχημα και τραβά τα δεδομένα που θέλουμε από αυτό. Τα δεδομένα αυτά αποστέλλονται στο ESP32 που κάνει και την βασική τους επεξεργασία. Εναλλακτικά το ELM327 μπορεί να είναι συνδεμένο σε ένα emulator που μιμείται το όχημα. Διαμέσου ενός Logic level Shifter επιτυγχάνεται η επικοινωνία με ένα σύστημα GPS που αποστέλλει της συντεταγμένες του στο ESP. Το ESP συνδέεται σε ένα φορητό Access Point για να έχει πρόσβασή στο διαδίκτυο και να ανεβάσει τα συλλεχθέντα δεδομένα στο ThingsBoard.

Στο πρόγραμμα Arduino IDE ξεκίνησε η αρχική ανάπτυξη και δοκιμές του κώδικα. Λόγω της κακής προσαρμογής του με το ESP, εγκαταλείφθηκε και αντικαταστάθηκε από το Visual Studio Code.

Κεφάλαιο 4 – Υλοποίηση πλατφόρμας IoT στο Cloud

4.1 Εισαγωγή

Ο όρος νέφος ή cloud έχει εξελιχθεί τα τελευταία χρόνια σαν ένα από τα πιο μεγάλα υπολογιστικά πρότυπα στον τομέα του IT (Information Technology). Αυτό έγινε σαν αποτέλεσμα προόδου σε υπάρχοντα υπολογιστικά πρότυπα όπως Παράλληλη Υπολογιστική (Parallel Computing), Υπολογιστική Πλέγματος (Grid Computing) και Υπολογιστική Κατανεμημένων Συστημάτων (Distributed Computing). Αυτές οι τεχνολογικές στρατηγικές επιτρέπουν την πρόσβαση με ομαλό τρόπο σε συστήματα με υπολογιστικούς πόρους, όπου ο χρήστης μπορεί με ελάχιστη προσπάθεια να αυξήσει ή να μειώσει κατά απαίτηση, τους πόρους αυτούς. Ο τρόπος αυτός προσφέρει στους καταναλωτές τρία βασικά μοντέλα υπηρεσιών. Λογισμικό ως υπηρεσία (Software as a Service – SaaS), πλατφόρμα ως υπηρεσία (Platform as a Service – PaaS) και υποδομή ως υπηρεσία (Infrastructure as a Service – IaaS).

Το Λογισμικό ως Υπηρεσία προορίζεται για τον τελικό χρηστή, οπού χρησιμοποιεί ένα λογισμικό σαν μέρος της καθημερινότητάς του, όπως για παράδειγμα το Google Workspace και το Dropbox. Η Πλατφόρμα ως Υπηρεσία προορίζεται κυρίως σε προγραμματιστές εφαρμογών, και παρέχει ένα σύνολο από εργαλεία προγραμματισμού, όπως για παράδειγμα το Windows Azure και το Heroku. Τέλος η Υποδομή ως Υπηρεσία προορίζεται σε αυτούς που χρειάζονται αναπτυξιακά εργαλεία και υπολογιστικούς πόρους σαν το Amazon Web Services (Alam, 2020).

Εικόνα 18: Cloud Computing

4.2 Περίγραμμα υλοποίησης

Υπάρχουν αρκετοί τρόποι να ολοκληρωθεί το cloud κομμάτι. Στην παρούσα προσέγγιση χρησιμοποιήθηκε το Docker. Η χρήση του Docker είναι καθαρά προαιρετική, αλλά με την χρήση των containers διευκολύνεται πολύ η εγκατάσταση, αυξάνεται η ασφάλεια και μπορεί να απομονώσει την μια εφαρμογή από την άλλη.

Το σύστημα αποτελείται από έναν Linux server στον οποίο έχει γίνει εγκατάσταση του Docker Engine. Μέσω ενός docker container έχει εγκατασταθεί το ThingsBoard. Εκεί έχουν δημιουργηθεί δυο πίνακες που προβάλουν τα δεδομένα που λαμβάνονται από το hardware είτε σε πραγματικό χρόνο είτε σε παλαιότερες καταγραφές.

4.3 Υποδομές που χρησιμοποιήθηκαν

Αρχικά έγινε χρήση ενός μικρού NAS server (sinology DS218) αλλά δεν υπήρχαν αρκετοί διαθέσιμοι πόροι καθώς σε αυτό τρέχουν και αρκετές άλλες υπηρεσίες. Για την καλύτερη λειτουργία της εφαρμογής εγκαταστάθηκε εκ νέου στις υποδομές του OKEANOS, καθώς μπορεί να παρέχει πολύ μεγαλύτερη διαθεσιμότητα και σταθερότητα.

Στο OKEANOS αρχικά έγινε χρήση του Docker προκειμένου να υπάρχει η υποδομή για την εγκατάσταση του ThingsBoard. Το Docker είναι μια τεχνολογία που επιτρέπει δημιουργία και διαχείριση Container, κάτι σαν εικονική μηχανή (Virtual Machine), αλλά με πολύ πιο αποδοτικό και εύκολο τρόπο.

Εικόνα 19: Docker logo

Απομένει η χρήση μιας εφαρμογής Dashboard. Υπάρχουν και εκεί πολλές εναλλακτικές επιλογές για αυτό. Μερικά από αυτές είναι FreeBoard, το Node-Red. Παρόλα αυτά έχει γίνει χρήση του ThingsBoard. Δεν υπάρχει κάποιος ιδιαίτερος λόγος για την απόφαση αυτή παρά το γεγονός ότι είναι έτοιμο για χρήση από το Docker. Το κάθε ένα από αυτά έχει τα δικά του πλεονεκτήματα και μειονεκτήματα.

Εικόνα 20: ThingsBoard logo

4.4 Integration του συστήματος

4.4.1 Εγκατάσταση Docker και ThingsBoard

Η εγκατάσταση του Docker και μετέπειτα του Thingsboard είναι αρκετά εύκολη. Σε Ubuntu server αρκεί να εκτελέσουμε την εντολή εγκατάστασης:

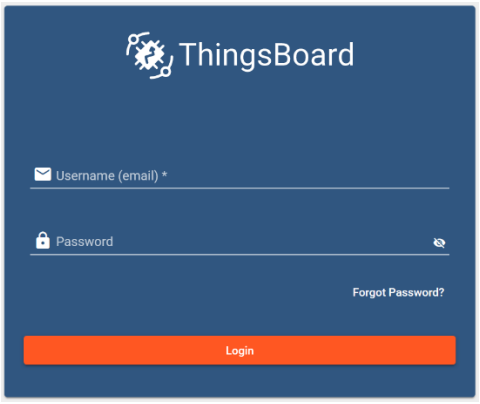
```
apt-get install docker
apt-get install docker-compose
```

Αυτό δεν διασφαλίζει ότι τα πακέτα που θα κατέβουν θα είναι στην πιο πρόσφατη έκδοση τους. Εναλλακτικά αν θέλουμε την πιο πρόσφατη έκδοση αυτό μπορεί να επιτευχθεί με της εντολές που υπάρχουν στο παράρτημα Α.

Αφότου έχει γίνει η εγκατάσταση του docker και docker-compose, η εγκατάσταση του ThingsBoard δεν είναι το ίδιο απλή. Η προτεινόμενη μέθοδος είναι μέσο του docker-compose και την δημιουργία αρχείου YML. Η αναλυτική διαδικασία εγκατάστασης περιγράφεται και αυτή στο παράρτημα Α

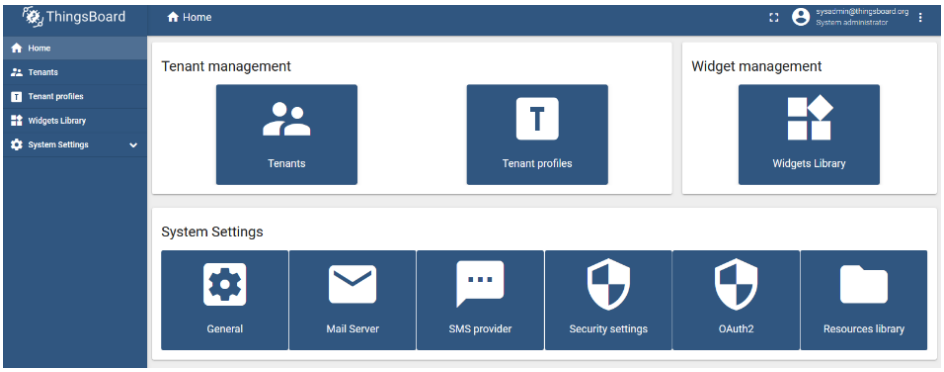
4.4.2 Παραμετροποίηση του ThingsBoard

Όπως και σε κάθε εφαρμογή που διαθέτει χρήστες, ειδικά όταν είναι προσβάσιμη από το διαδίκτυο, είναι ζωτικής σημασίας να γίνει αλλαγή των προεπιλεγμένων κωδικών. Το ThingsBoard έχει τρία ειδή χρηστών με πέντε συνολικά λογαριασμούς που θα πρέπει να γίνει αλλαγή των κωδικών πρόσβασης. Οι κατηγορίες των χρηστών είναι οι διαχειριστές συστήματος (sysadmin), οι ‘απλοί διαχειριστές’ (tenants) και οι πελάτες (customers).



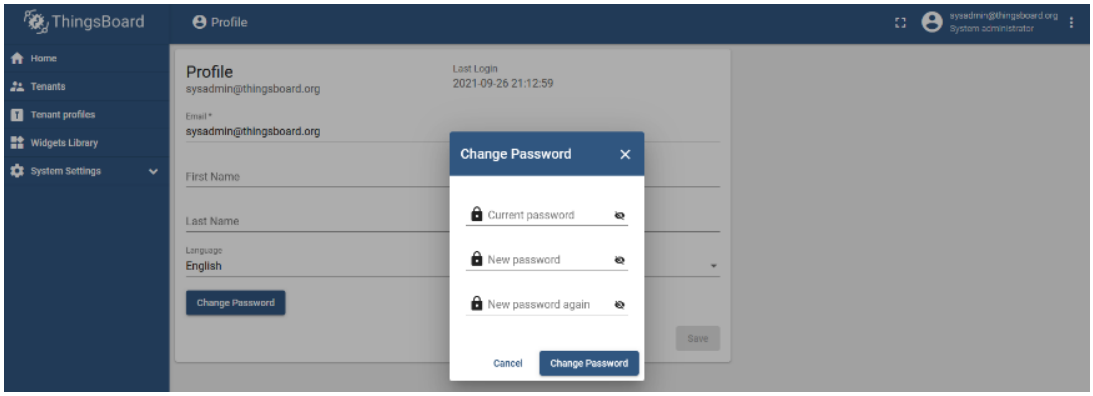
Εικόνα 21: ThingsBoard login screen

Αρχικά πρέπει να γίνει είσοδος στο σύστημα με τον λογαριασμό sysadmin και με τα προεπιλεγμένα στοιχεία πρόσβασης (όνομα χρήστη: sysadmin@thingsboard.org και κωδικός sysadmin). Η αλλαγή των στοιχείων πρόσβασής γίνεται ανοίγοντας το προφίλ του συνδεμένου χρήστη πατώντας πάνω δεξιά στης τρεις κουκίδες και επιλέγοντας προφίλ.



Εικόνα 22: Αρχική σελίδα sysadmin

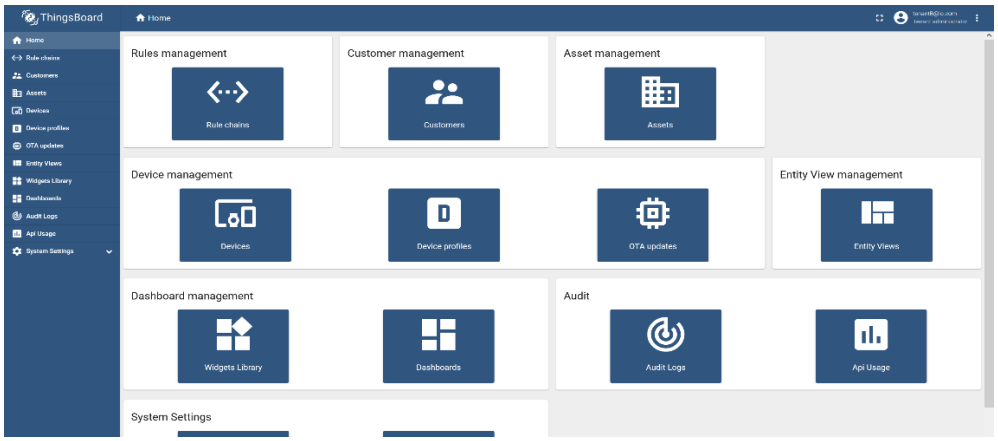
Θα ανοίξει το προφίλ του χρήστη. Από εκεί μπορούν να ενημερωθούν τα προσωπικά στοιχεία του και να γίνει και η αλλαγή κωδικού. Με ακριβώς τον ίδιο τρόπο μπορεί να αλλάξει τα στοιχεία του ο οποιοσδήποτε χρήστης.



Εικόνα 23: Προφίλ χρήστη και αλλαγή κωδικού

Συνοπτικά ο χρήστης αυτός είναι μοναδικός και προορίζεται για τον διαχειριστή του ίδιου του server. Έχει την δυνατότητα να αλλάξει βασικές ρυθμίσεις του ίδιου του συστήματος, όπως η αποστολή μηνυμάτων ηλεκτρονικού ταχυδρομείου και μηνυμάτων SMS. Η βασική του χρήση είναι να φτιάξει τους «απλούς διαχειριστές» και τα προφίλ τους. Επιπλέον έχει την δυνατότητα να επεξεργαστεί την βιβλιοθήκη με τα Widgets.

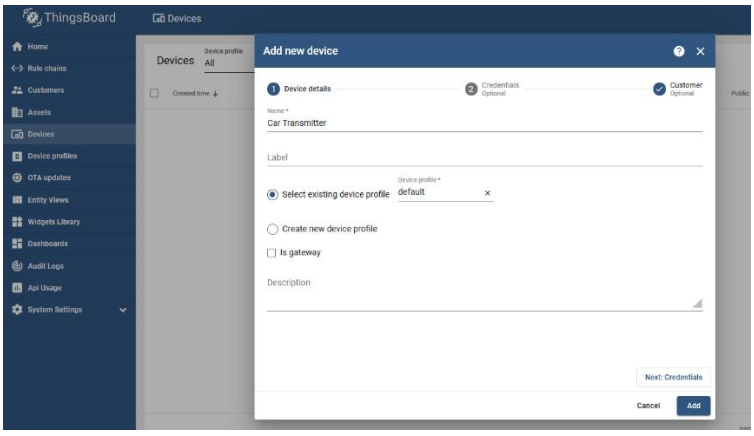
Οι απλοί διαχειριστές είναι ξεχωριστές οντότητες ανεξάρτητες η μια από την άλλη. Αυτοί οι λογαριασμοί προορίζονται για της εκάστοτε εταιρίες ή οργανισμούς. Ο κάθε ένας από τους λογαριασμούς αυτούς έχει της δικές του συσκευές και πελάτες (costumers) και δεν μπορούν να επηρεάσουν τους άλλους χρήστες. Μπορούν μεταξύ άλλων να προσθέσουν της συσκευές τους, τους χρήστες τους, και να φτιάξουν τα Dashboard τους.



Εικόνα 24: Αρχική οθόνη απλού διαχειριστή

4.4.3 Προσθήκη νέας συσκευής

Για να περάσουμε μια καινούρια συσκευή στην εφαρμογή πατάμε στην επιλογή Devices (Συσκευές) έπειτα στην πάνω δεξιά γωνιά στο Add Device. Υπάρχει η δυνατότητα είτε να προσθέσουμε νέα συσκευή είτε να την εισάγουμε από αρχείο. Σε αυτήν την περίπτωση γίνεται προσθήκη νέας συσκευής. Το πρώτο πράγμα που ζητάει είναι να δώσουμε ένα όνομα στην συσκευή, ετικέτα και μια σύντομη περιγραφή. Αν δεν δωθούν διαπιστευτήρια πατώντας στο κουμπί επόμενο, τότε η συσκευή έχει ένα μοναδικό διακριτικό πρόσβασης που χρησιμοποιείται αντί για όνομα χρήστη. Η διαδικασία ολοκληρώνεται πατώντας το κουμπί προσθήκη.



Εικόνα 25: Προσθήκη νέα συσκευής στο ThingsBoard

Αν δεν έχουν δωθεί κωδικοί πρόσβασης για την συσκευή, θα χρειαστεί να σημειωθεί το μοναδικό διακριτικό πρόσβασης. Είναι απαραίτητο για να σταλούν δεδομένα στην συσκευή αυτή. Πατώντας πάνω σε μια από της συσκευές που έχουν προστεθεί ανοίγει ένα νέο

παράθυρο. Πατώντας πάνω στο Copy access token αντιγράφεται το διακριτικό πρόσβασης στο πρόχειρο. Θα χρησιμοποιηθεί στην συνέχεια σαν κωδικός πρόσβασης για την συσκευή.

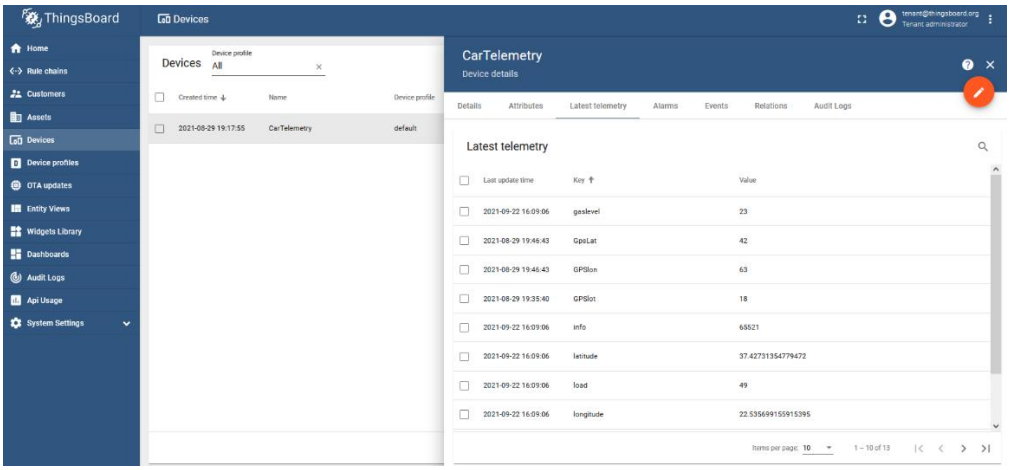
4.4.4 Δοκιμή της νέας συσκευής

Για την δοκιμή της συσκευής έχει αναπτυχθεί ένα πρόγραμμα σε Python όπου προσπαθεί να συνδεθεί στην νέα αυτή συσκευή και να αποστείλει δοκιμαστικά δεδομένα. Το ThingsBoard περιμένει να λάβει τα δεδομένα αυτά, είτε μέσω HTTP, είτε μέσω MQTT. Έχει γίνει χρήσει του paho οπού αποτελεί μια πολύ ευέλικτη βιβλιοθήκη για το πρωτόκολλο MQTT.

Το πρόγραμμα αυτό παράγει τυχαίους αριθμούς και τους αποστέλλει στο ThingsBoard. Το παρακάτω είναι μέρος του προγράμματος με την επεξήγηση του. Το πλήρες πρόγραμμα δοκιμών υπάρχει στο παράρτημα Β

```
(…)
username="wjAb7G5c6ljYTdn2Aruz" # το διακριτικό της συσκευής
(…)
data["car-speed"]=random.randint(0,120) # Τυχαίος αριθμός από το 0 έως το
120 με ετικέτα car-speed
data["engine-rpm"]=random.randint(541,7800) # Τυχαίος αριθμός μεταξύ 541-
7800 για στροφές μηχανής
data_out=json.dumps(data) # Δημιουργεί ένα πακέτο με τους αριθμούς και τις
ετικέτες τους
ret=client.publish(topic,data_out,0) # Αποστολή δεδομένων στο ThingsBoard
client.loop() # Επανάληψη του βρόγχου
(…)
```

Μετά την πρώτη επιτυχημένη εκτέλεση του προγράμματος αυτού, θα πρέπει να εμφανιστούν οι τυχαίοι αυτοί αριθμοί στην συσκευή στο ThingsBoard. Για να το ελέγξουμε αυτό πατάμε πάνω στην συσκευή και μετά στη καρτέλα Latest telemetry. Είναι πολύ σημαντικό να επιβεβαιωθεί η ορθή λείψει των δεδομένων καθώς σε κάθε άλλη περίπτωση μπορεί να υπάρχει πρόβλημα, και υποθέτοντας ότι το πρόγραμμα δοκιμής δουλεύει κανονικά θα πρέπει να εξεταστούν οι ρυθμίσεις του server και το δίκτυο. (Router/Firewall, και port binding στο Docker)



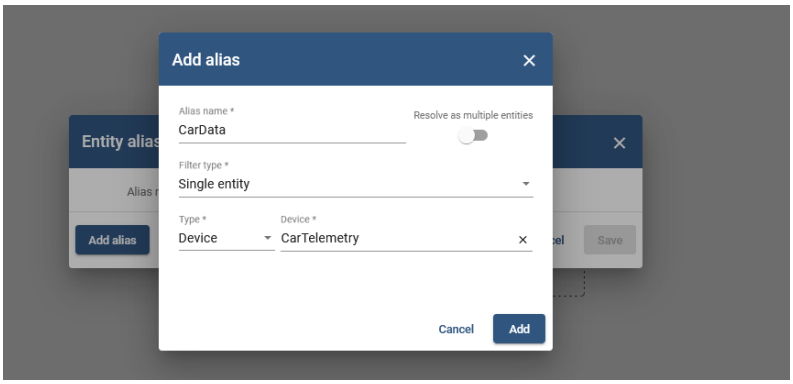
Εικόνα 26: Τα Δοκιμαστικά δεδομένα στην καρτέλα τηλεμετρίας

4.4.6 Τα Dashboards

Τα Dashboards είναι ο τρόπος που θα απεικονίζονται τα λαμβανόμενα από το όχημα δεδομένα. Από την αρχική οθόνη του ThingsBoard και πατώντας στην καρτέλα Dashboards προβάλλονται όλα όσα είναι διαθέσιμα. Από την πάνω δεξιά γωνία, όπως και με την προσθήκη συσκευής, έτσι και με τα dashboard, υπάρχει η επιλογή για δημιουργία νέου ή ανέβασμα από αρχείο. Πατώντας νέο, δίνουμε ένα όνομα και πατάμε προσθήκη. Θα δημιουργηθεί ένα νέο dashboard.

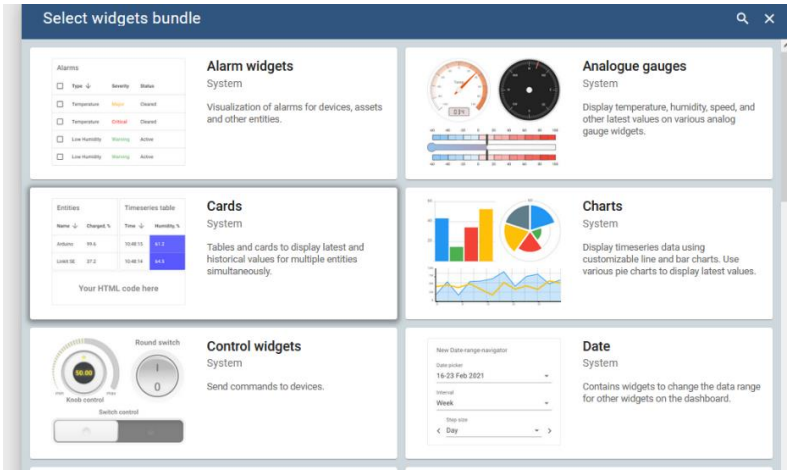
Ανοίγοντας το, στο κάτω δεξιά σημείο τις οθόνης υπάρχει ένα εικονίδιο σαν μολυβί. Πατώντας ενεργοποιεί την επεξεργασία του (Enter edit mode). Εμφανίζονται τρία νέα εικονίδια, για ακύρωση, αποθήκευση και προσθήκη νέου Widget, όπου επιτρέπει την προσθήκη νέου από αρχείο ή το άνοιγμα της υπάρχουσας βιβλιοθήκης με αυτά.

Προτού προστεθούν Widgets πρέπει να γίνει μια γενική ρύθμιση στο Dashboard. Στο edit mode στην οριζόντια μπάρα στο πάνω δεξιά μέρος τις οθόνης πατάμε στο Entity alias και μετά στο Add alias. Στο νέο παράθυρο (Εικόνα 27) δίνουμε ένα νέο όνομα για την πηγή δεδομένων, στο πεδίο Filter type επιλέγουμε Single entity, Type Device, και επιλέγουμε την συσκευή που δημιουργήσαμε παραπάνω. Τέλος πατάμε προσθήκη και μετά αποθήκευση.



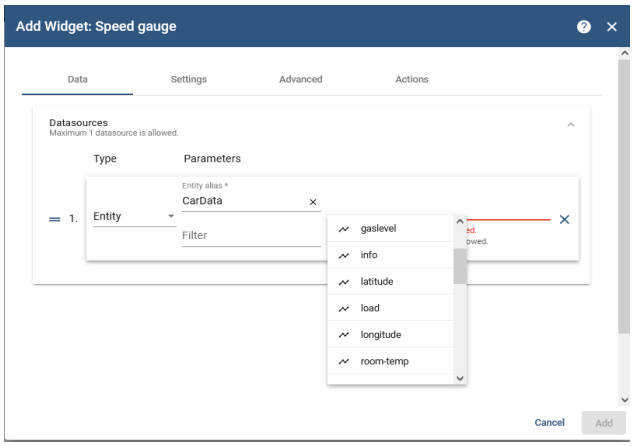
Εικόνα 27: Αρχική ρύθμιση του Dashboard

Το Widget δεν είναι τίποτα άλλο, από ένα όργανο απεικόνισης πληροφοριών με την μορφή αναλογικού ή ψηφιακού ρολογιού, είτε με διαγράμματα και πολλά άλλα σχέδια. Η επιλογή του είναι γίνεται με βάση τι θέλουμε να απεικονίσουμε.



Εικόνα 28: Widget βιβλιοθήκη

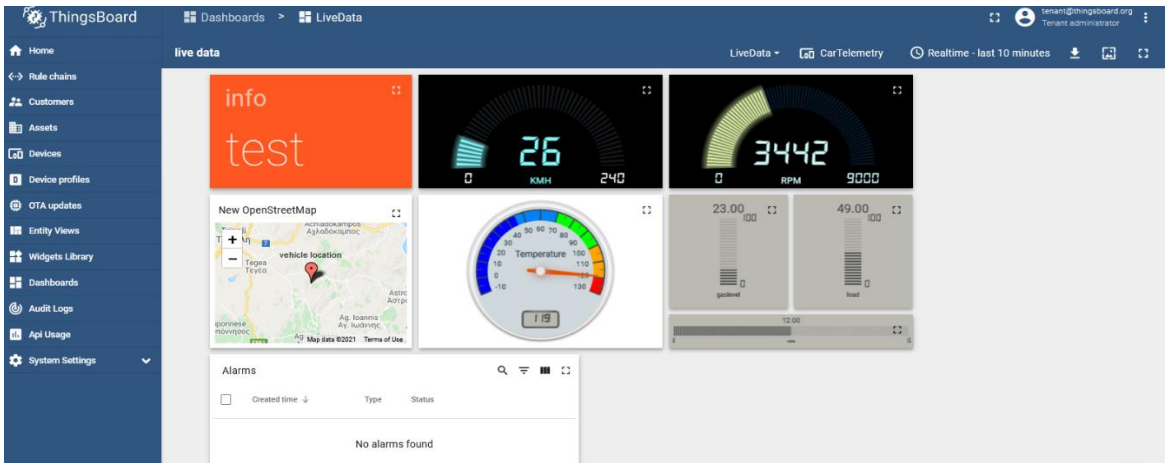
Αφού έχει επιλεγεί το κατάλληλο, στην συγκεκριμένη περίπτωση για την ταχύτητα του οχήματος, εμφανίζει ένα νέο παράθυρο για προσθήκη τον παραμέτρων του οργάνου.



Εικόνα 29: Δεδομένα απεικόνισης

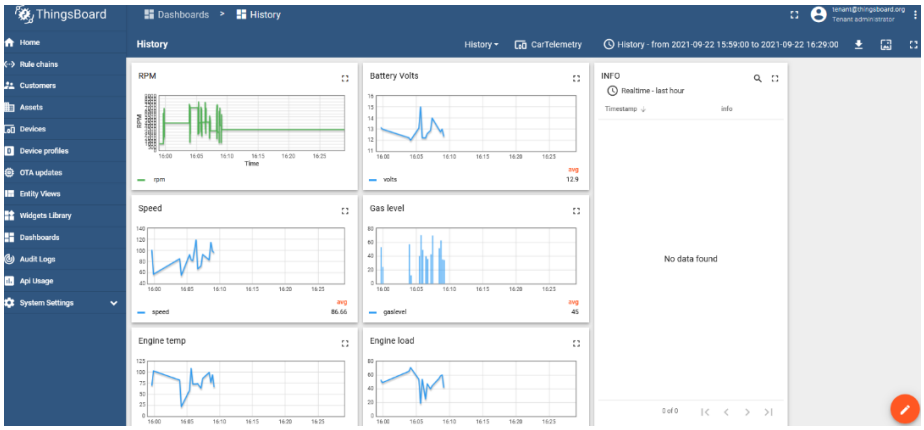
Από εδώ γίνεται η επιλογή για το τι θα απεικονίζει το όργανο. Στο πεδίο Type επιλέγουμε Entity και στο Parameters. Επιλέγουμε την πυγή δεδομένων που ρυθμίσαμε νωρίτερα (CarData). Στο επόμενο πεδίο αν πατήσουμε πάνω θα φορτώσει αυτόματα διάφορες παραμέτρους από την συσκευή τηλεμετρίας. Θα πρέπει να έχουν ληφθεί δεδομένα έστω και μια φορά για να μπορέσουμε να τα απεικονίσουμε στο όργανο. Τα ονόματα τον παραμέτρων είναι ίδια με αυτά που έχει λάβει η συσκευή οπότε είναι και εύκολη η αναγνώριση της απαιτούμενης παραμέτρου στο κάθε όργανο.

Αξίζει να σημειωθεί ότι για τον χάρτη απαιτούνται οι γεωγραφικές συντεταγμένες latitude και longitude και να αποστέλλεται σε δυο ξεχωριστές παραμέτρους από την συσκευή τηλεμετρίας και να επιλεγούν και τα δυο στο χάρτη. Αν τα ονόματα δεν είναι ακριβώς έτσι (latitude και longitude) προκαλεί ασυμβατότητα με της προεπιλεγμένες ρυθμίσεις του οργάνου και απαιτείτε η αλλαγή των ονομάτων σε πολλά σημεία. Η ολοκλήρωση του dashboard με όλα τα όργανα είναι η ακόλουθη:



Εικόνα 30: Το ολοκληρωμένο Dashboard

Με παρόμοιο τρόπο έχει φτιαχτεί και το dashboard που κρατά τα αρχειοθετημένα δεδομένα, αλλά επιλέγοντας διαγράμματα αντί για όργανα. Το σύστημα θα φορτώσει αυτόματα στο διάγραμμα τα όποια δεδομένα διαθέτει για την εκάστοτε χρονική στιγμή.

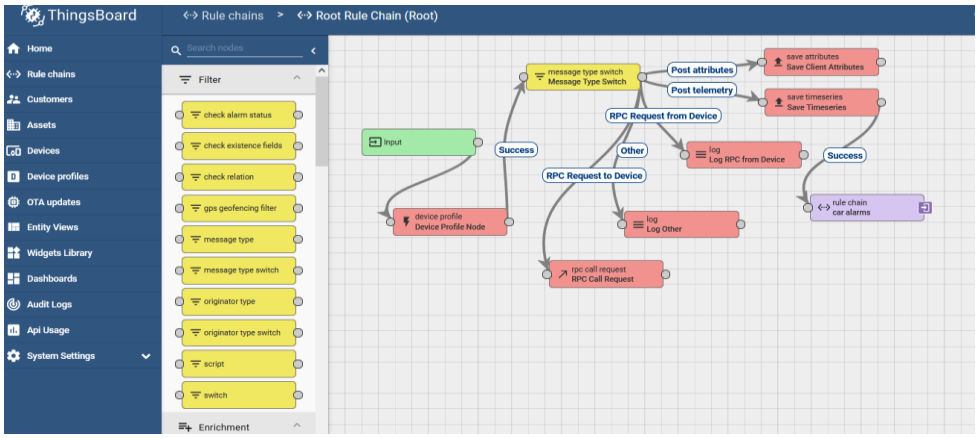


Εικόνα 31:Αρχειοθετημένα δεδομένα

4.4.7 Rule Chains και Alarms

Τα Rule Chains δίνουν επιπλέον δυνατότητες. Είναι εφικτό να αναπτυχθούν ροές δεδομένων ή και ενεργειών με παρόμοιο τρόπο σαν αυτό του Node-Red. Το εργαλείο αυτό έχει χρησιμοποιηθεί για την δημιουργία Alarms, όπου θα ειδοποιούν σε περίπτωση που ανιχνευθεί επιθετική οδήγηση κατά την διάρκεια λειτουργίας του προγράμματος.

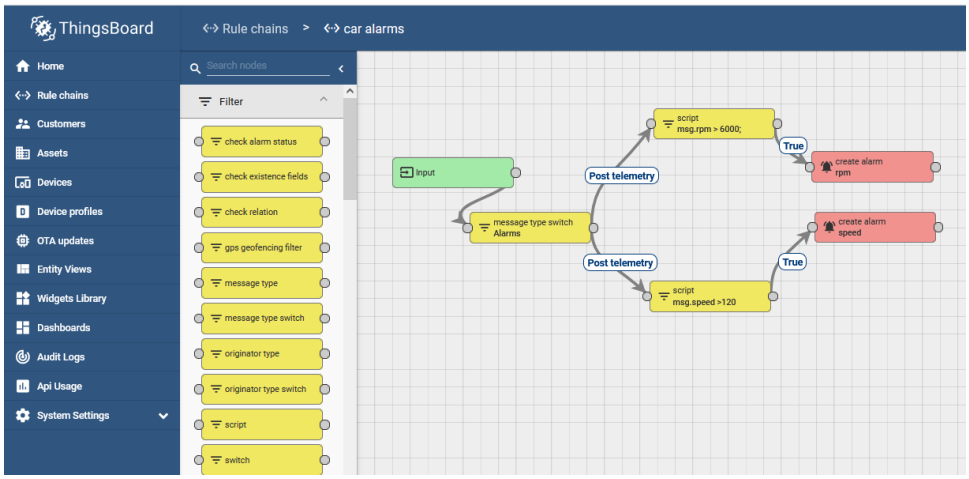
Το ThingsBoard έχει προ εγκατεστημένο ένα rule-chain για την καλύτερη λειτουργία του ελέγχοντας τα εισερχόμενα μηνύματα χωρίζοντας τα σε κάποιες υποκατηγορίες. Για να στείλουμε τα δεδομένα που θέλουμε σε νέο rule chain θα πρέπει τα δεδομένα τηλεμετρίας να προωθηθούν σε νέο. Αυτό μπορεί να γίνει φτιάχνοντας πρώτα ένα νέο. Πατώντας στην καρτέλα Rule chains και στην συνέχεια προσθήκη νέου από την πάνω δεξιά γωνία, και δίνοντας ένα όνομα, γίνεται η αποθήκευση του.



Εικόνα 32: Το αρχικό Rule chain

Από το αρχικό rule chain είναι πλέον εφικτό να προωθήσουμε τα επιτυχώς λαμβανόμενα δεδομένα σε νέο. Συνδέοντας με το νέο rule chain(το στοιχείο για αυτό είναι τελευταίο στην λίστα)την έξοδο του post telemetry, όπως αυτό φαίνεται στην παραπάνω εικόνα.

Πλέον τα δεδομένα προωθούνται στο προς το παρών άδειο διάγραμμα. Ανοίγοντας το πρώτα τοποθετείτε το στοιχείο message type switch με την έξοδο αυτού σε δυο νέα script. Το script τρέχει ένα κώδικα και αλλάζει την έξοδό του ανάμεσα σε αληθής ή ψευδής (true or false). Αυτήν η έξοδος χρησιμοποιείται για την δημιουργία Alarms.

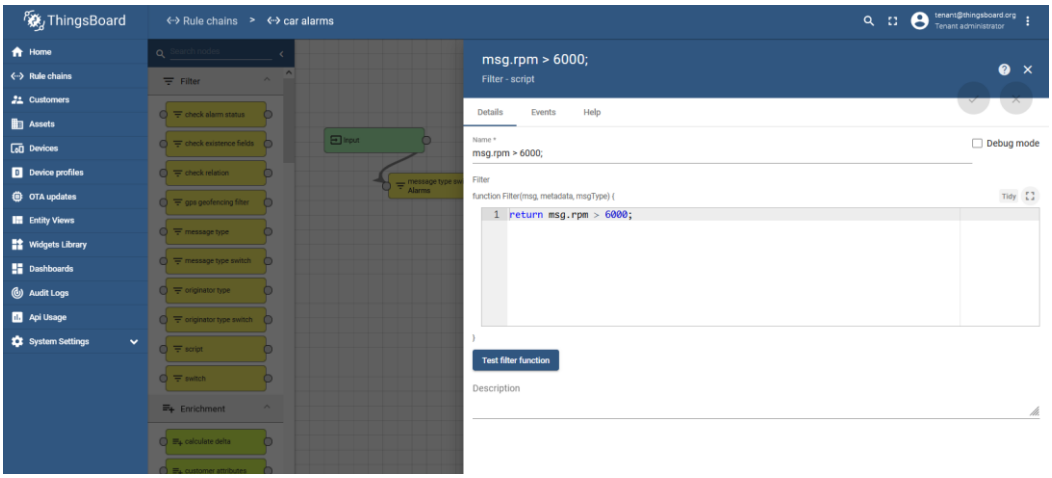


Εικόνα 33: Νέο Rule chain

Το κάθε script τρέχει ένα κώδικα που ελέγχει αν η ταχύτητα και η στροφές του κινητήρα είναι αυξημένα πάνω από ένα προκαθορισμένο όριο. Στην περίπτωση αυτή η έξοδός θα αλλάξει και το σύστημα θα βγάλει ένα νέο Alarm.

```
return msg.rpm > 6000;
```

Ο παραπάνω κώδικας ελέγχει αν η στροφές (rpm) που έχει λάβει από το τελευταίο μήνυμα τηλεμετρίας (msg) και στην περίπτωση αυτή αλλάζει την έξοδο του σε αληθής, δημιουργώντας έτσι το Alarm.



Εικόνα 34: Επεξεργασία του script

4.5 Συνοπτικά

Στο κεφάλαιο αυτό έγινε αναφορά στον τρόπο εγκατάστασης του Docker και του ThingsBoard, καθώς και πως γίνεται η αρχική ρύθμισή του. Έγινε επίδειξη του πως προσθέτουμε μια νέα συσκευή καθώς και η δοκιμή της. Ακολούθως και για το πως φτιάχτηκε το Dashboard και το πως γίνεται η δοκιμή καλής λειτουργίας του. Τέλος έγινε επίδειξη του πως δημιουργούνται τα Alarms καθώς και πως μπορεί να δοκιμαστεί η λειτουργία τους.

Κεφάλαιο 5 – Ολοκληρωμένο σύστημα

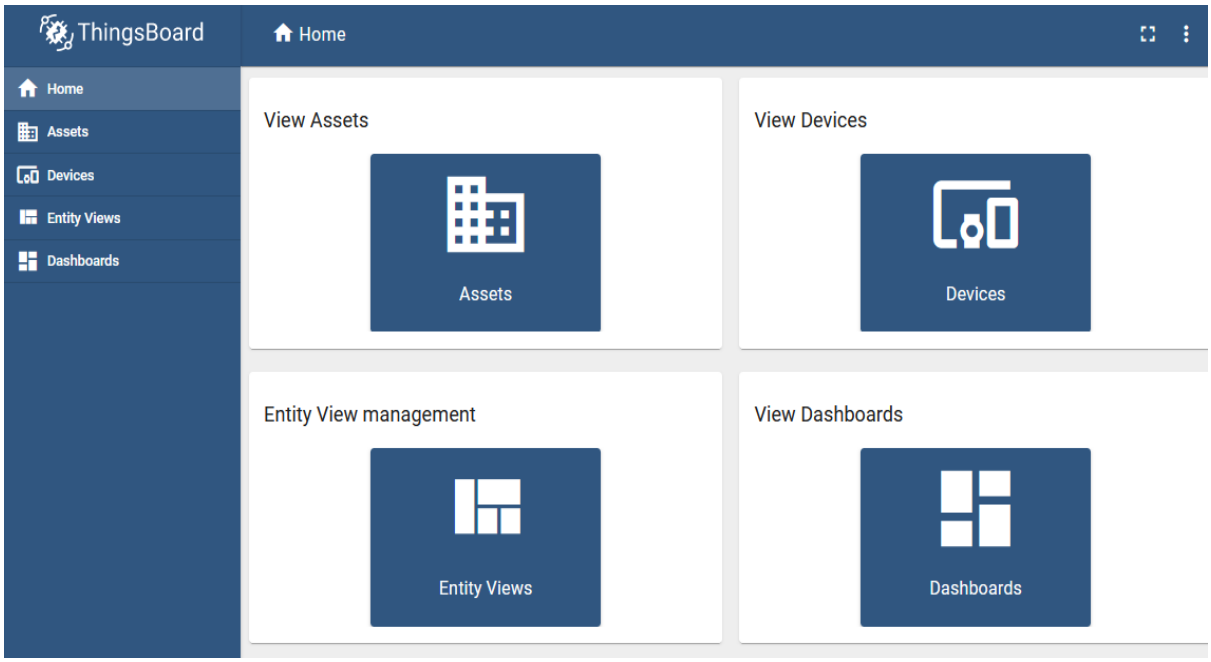
5.1 Εισαγωγή

Στο κεφάλαιο αυτό θα γίνει μια ολοκληρωμένη παρουσίαση τις εργασίας. Θα γίνει επίδειξη των βασικών λειτουργιών του συστήματος καθώς και επίδειξη κάποιων ειδοποιήσεων που παράγει το σύστημα.

5.2 Επίδειξη λειτουργίας

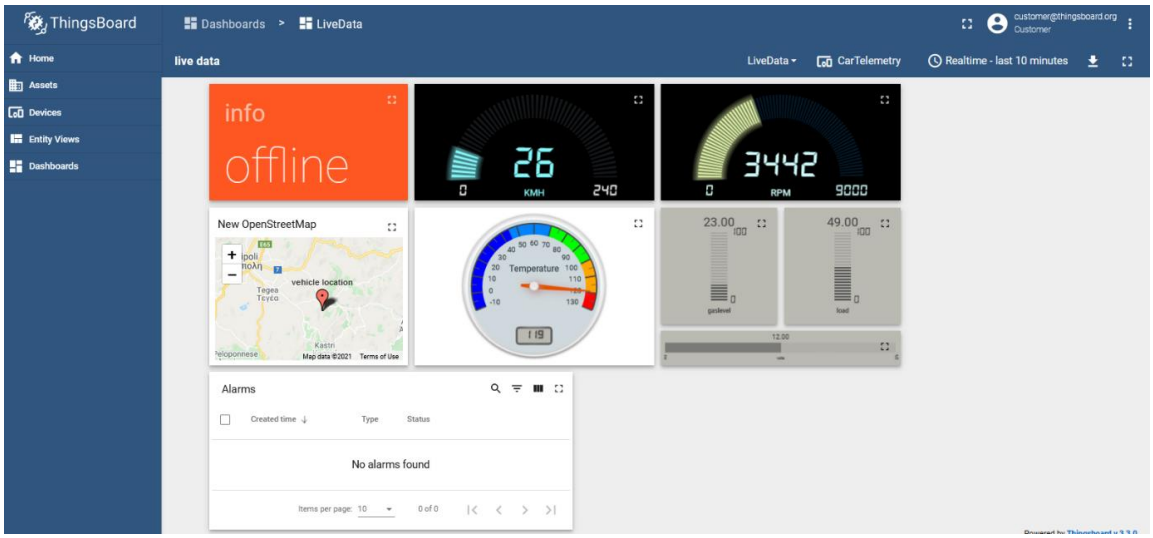
Ο τελικός χριστής έχει περιορισμένη πρόσβαση στην εφαρμογή. Πιο συγκεκριμένα μπορεί μόνο να δει τους τελικούς πίνακες με τα Live δεδομένα, καθώς και τις καταγραφές.

Ανοίγοντας το σύνδεσμο για την πρόσβαση του τελικού χρήστη στο σύστημα, το σύστημα τον προτρέπει να κάνει σύνδεση στην πλατφόρμα. Αμέσως μετά την είσοδο του, του εμφανίζεται η αρχική οθόνη πελάτη.



Εικόνα 35: Αρχική οθόνη πελάτη

Οι βασική λειτουργία είναι η πρόσβαση στα Dashboards που του έχουν ανατεθεί. Πατώντας πάνω στο αντίστοιχο μενού, του παρέχεται η δυνατότητα να φορτώσει τα δυο αυτά Dashboards. Υπάρχει και η δυνατότητα να μπορεί κάποιος έχοντας μόνο ένα μοναδικό σύνδεσμο να φορτώσει αυτόματα το dashboard χωρίς να του ζητείται όνομα χρήστη και κωδικός πρόσβασης.



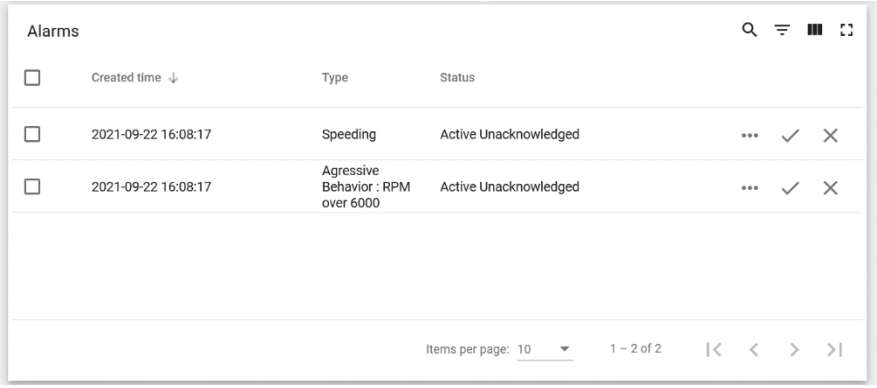
Εικόνα 36: Live Dashboard

5.3 Alarms

Έχουν δημιουργηθεί κάποιες ειδοποιήσεις που το σύστημα προβάλει κάτω από προκαθορισμένες συνθήκες. Η βασική χρήση του συστήματος αυτού είναι η παρακολούθηση του αυτοκινήτου για το αν κινείται μέσα σε προκαθορισμένα όρια. Αν κάποιες από της παραμέτρους του οχήματος υπερβεί τα όρια αυτά τότε ενεργοποιείται ένα από αυτά τα Alarms και καταγράφεται στην βάση. Ο χρηστής έχει την δυνατότητα να τα παρακολουθεί και μπορεί να δει που και πότε έγινε η ενεργοποίησή του.

Παρακάτω είναι ο πίνακας των ειδοποιήσεων αυτών. Την δεδομένη χρονική στιγμή είναι προγραμματισμένες στο σύστημα δυο ειδοποιήσεις σχετικά με την οδηγική συμπεριφορά και πιο συγκεκριμένα για υπέρβαση ενός ορίου ταχύτητας και επιθετική οδήγηση.

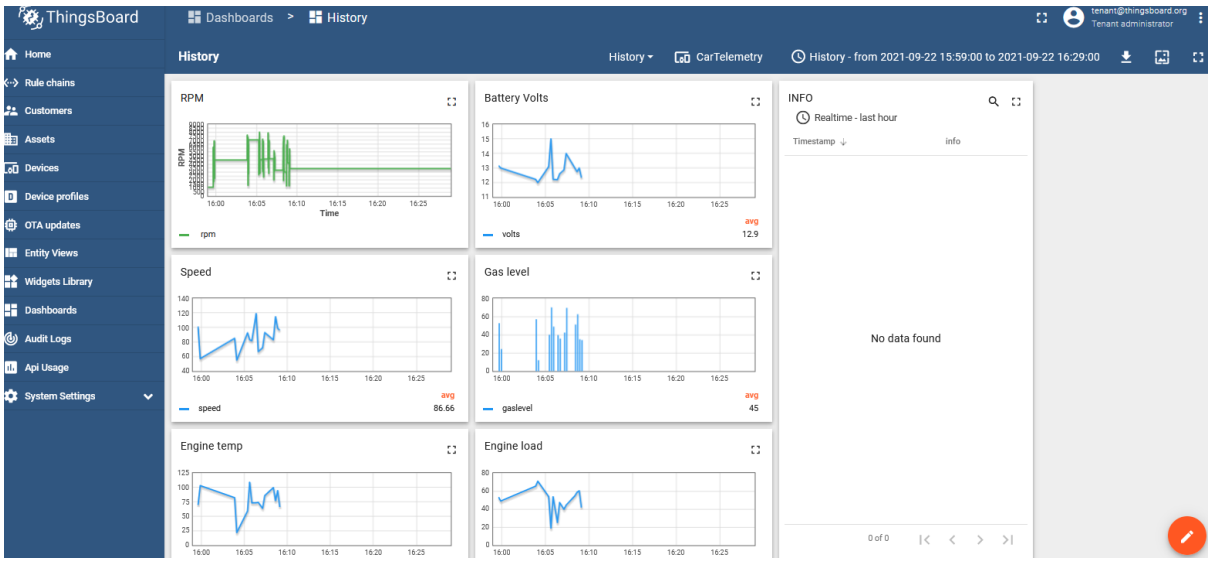
Αν ο οδηγός του οχήματος περάσει μια προκαθορισμένη ταχύτητα ή οι στροφές του κινητήρα περάσουν ένα προκαθορισμένο όριο, τότε δημιουργείται ένα από τα παρακάτω Alarm.



Εικόνα 37: Alarm panel

5.4 Καταγραφές

Οι καταγραφές έχουν ως σκοπό να καταγράφονται όλες οι τιμές που αποστέλλονται από το όχημα. Τα αποθηκευμένα δεδομένα είναι διαθέσιμα στον χρήστη με την μορφή διαγράμματος ώστε να είναι ευκολότερη η παρουσίαση και εξερεύνησή τους.



Εικόνα 38: Καταγραφές

Ο χρήστης έχει την δυνατότητα να επιλέξει συγκεκριμένη ημερομηνία ή προσαρμοσμένο εύρος χρόνου οπού μπορεί να δει στα γραφήματα. Εδώ επίσης εμφανίζονται και το ιστορικό τον Alarms όπου έχουν ενεργοποιηθεί.

Κεφάλαιο 6 – Συμπεράσματα και συζήτηση

6.1 Ιστορικό ανάπτυξής

Στόχος τις διπλωματικής είναι η ανάπτυξη ενός ολοκληρωμένου συστήματος επιτήρησης οχημάτων. Η βασική εφαρμογή του είναι να εφαρμοστεί σε περιπτώσεις που η τοποθεσία και η συμπεριφορά του οδηγού έχουν άμεσες επιπτώσεις σε μια εταιρία ή σε τρίτους, καθώς μπορεί να μεταφέρουν εμπόρευμα ή (και) να τους έχει δοθεί εταιρικό όχημα. Οποίος και να είναι ο λόγος, είναι σημαντικό να τηρείται ο κώδικας οδικής κυκλοφορίας, και το παρών σύστημα παρέχει στην εταιρία ακριβώς αυτό. Τα υλικά που έχουν χρησιμοποιηθεί είναι ευρέως διαθέσιμα στο διαδίκτυο, όπου καθιστά την περεταίρω ανάπτυξη ευκολότερη.

Ξεκινώντας από το υλικό, υλοποιήθηκε αρχικά η συσκευή προσομοίωσης του εγκεφάλου αυτοκινήτου. Ο λόγος που έγινε αυτό, ήταν για να υπάρχει ένας εύκολος τρόπος δοκιμών του κώδικα που αναπτύχθηκε, δεδομένου ότι στην κατοχή του συγγραφέα, δεν υπήρχε συμβατό όχημα για δοκιμές και επιπλέον ήταν κατά πολύ πιο εύκολο και το σημαντικότερο, ασφαλέστερο, το να δοκιμάζεται ο κώδικας από το γραφείο μας.

Στο πρώτο στάδιο ανάπτυξης εφαρμόστηκε η επικοινωνία μεταξύ ESP32 και ELM327 μέσω Bluetooth. Στη συνέχεια υλοποιήθηκε η σειριακή επικοινωνία ESP32 με το GPS. Κρίθηκε επιτυχημένο αυτό το στάδιο όταν οι πληροφορίες από τον προσομοιωτή και το GPS κατέληξαν στην σειριακή.

Για να προχωρήσει περεταίρω ή ανάπτυξη του κώδικα ήταν απαραίτητο να δημιουργηθεί ένα δοκιμαστικό περιβάλλον, πλατφόρμα IoT, στο Cloud. Χρησιμοποιώντας την πλατφόρμα του ThingsBoard αναπτύχθηκε ένα πολύ πρόχειρο περιβάλλον ικανό να προβάλει τις τιμές που λαμβάνει, ενώ παράλληλα φτιάχτηκε και ένα δοκιμαστικό πρόγραμμα για την αξιολόγηση του ThingsBoard και την επιβεβαίωση ότι είναι σε θέση να προβάλει σωστά τα δεδομένα.

Έχοντας πλέον ένα λειτουργικό περιβάλλον IoT στο Cloud, προχώρησε η ανάπτυξη της εφαρμογής WiFi στο ESP32 και στην προώθηση των μηνυμάτων του στο ThingsBoard. Στο αμέσως επόμενο στάδιο εξελίχθηκε περεταίρω ο κώδικας στο ThingsBoard αφού προστέθηκαν όργανα και γραφήματα στα dashboard του χρήστη.

6.2 Επισκόπηση εργασίας

Σε μια σύγχρονη κοινωνία όπου ο αριθμός των μηχανοκίνητων οχημάτων αυξάνεται συνεχώς, παρουσιάζεται η ανάγκη για την άμεση ενημέρωση άλλα και παρακολούθηση ενός οχήματος. Διατυπώνονται οι στόχοι που πρέπει να μπουν, προκειμένου να ολοκληρωθεί η εργασία και το πώς το υλικό που θα χρησιμοποιηθεί βοηθάει στην επίτευξή των στόχων αυτών.

Αρχικά γίνεται μια μικρή αναφορά σχετικά με υφιστάμενα παρεμφερή συστήματα και διατυπώνεται μια προσέγγιση για το πώς θα μπορούσε να πραγματοποιηθεί ένα σύστημα που να καλύπτει τις ανάγκες της εργασίας. Παρουσιάζονται επίσης όλες οι τεχνολογίες και τα απαιτούμενα πρωτόκολλα που έχουν χρησιμοποιηθεί.

Στην συνέχεια γίνεται μια μελέτη σχετικά με τα υλικά που έχουν χρησιμοποιηθεί καθώς είναι απαραίτητη η καλή γνώση του κάθε κομματιού και το τι δυνατότητες παρέχει. Τόσο στο υλικό αλλά και στο λογισμικό επιδεικνύονται οι βασικές δυνατότητες των προγραμμάτων που αξιοποιήθηκαν. Στο σημείο αυτό διατυπώνεται το περίγραμμα της υλοποίησης και εξηγούνται κάποια σημαντικά κομμάτια από την κατασκευή και προγραμματισμό του υλικού.

Ακολουθεί το διαδικτυακό κομμάτι της εργασίας, όπου εξετάστηκε ο τρόπος εγκατάστασης της πλατφόρμας φιλοξενίας, του Docker, του ThingsBoard, και αναλύθηκε η παραμετροποίησή τους για την συγκεκριμένη εφαρμογή.

6.3 Μελλοντικές εργασίες

6.3.1 SIM808 GPRS module

Την δεδομένη χρονική στιγμή το σύστημα βασίζεται σε ένα φορητό σημείο πρόσβασης στο διαδίκτυο (WiFi Access Point). Ένα άμεσο βήμα για την απλοποίηση (από άποψη χρηστικότητας) του συστήματος είναι η αξιοποίηση του GPRS module που διαθέτει η πλακέτα του GPS. Τοποθετώντας μια κάρτα SIM θα παρέχεται η δυνατότητα το σύστημα να συνδέεται στο διαδίκτυο χωρίς να εξαρτάται από ένα ασύρματο δίκτυο WiFi. Επίσης κάτι τέτοιο θα αυξήσει την συνολική ασφάλεια στο σύστημα, μίας και ο χρήστης δεν θα έχει πρόσβαση στα δεδομένα του συστήματος.

6.3.2 Accelerometer sensor

Μια επίσης καλή προσθήκη στο σύστημα θα ήταν να προστεθεί ένα επιταχυνσιόμετρο. Αυτό θα βοηθούσε πολύ για την real-time καταγραφή της οδηγικής συμπεριφοράς του οδηγού, καθώς θα μπορούσαν ευκολότερα και ακριβέστερα να ανιχνευθούν απότομες επιτάχυνσεις ή φρεναρίσματα. Τα δεδομένα αυτά θα μπορούν να αξιοποιηθούν ακόμα και για την ανίχνευση σύγκρουσης του οχήματος.

6.3.3 Περισσότερα Alarms

Την δεδομένη χρονική στιγμή το σύστημα δημιουργεί Alarms για υπερβολική ταχύτητα και για επιθετική οδήγηση, βασισμένη στης στροφές του κινητήρα. Το σύστημα των Alarms όμως είναι πολύ πιο ευέλικτο. Θα μπορούσε να στέλνει ειδοποίησης σε πολλά σενάρια

ακόμα, όπως η χαμηλή στάθμη καυσίμου και η χαμηλή μπαταρία. Επίσης θα μπορούσε περιοδικά να ελέγχεται η ECU του οχήματος και αν εντοπιστούν σφάλματα να αναφέρονται πίσω στον χρήστη.

Βιβλιογραφία

Alam, T. (2020, Aprill). Cloud Computing and its role in the Information Technology. *IAIC Transactions on Sustainable Digital Innovation (ITSDI) 2nd Edition*, pp. 108-114.

Ashraf Tahat, A. S. (2012). Android-Based Universal Vehicle Diagnostic and Tracking System. *EEE 16th International Symposium*.

Decker Peter, P. M. (2015). *Vector*. Ανάκτηση από K-Line: Flexible Solutions for a Classic protocol: https://assets.vector.com/cms/content/know-how/_technical-articles/K_Line_AutomotiveEETimesEurope_201505_PressArticle_EN.pdf

Official docker documentation. (2021). *https://docs.docker.com/engine/install/ubuntu/*.

Python Software Foundation FAQ. (2021, 05). Ανάκτηση από https://docs.python.org/3/faq/general.html.

SIG. (2021, 05 12). *Bluetooth® Technology Website*. Ανάκτηση από Bluetooth Special Interest Group, (SIG): https://www.bluetooth.com/

ThingsBoard IO Comunity edition . (2021). Ανάκτηση από https://thingsboard.io/docs/user-guide/install/docker/.

Wi-Fi Alliance. (2021, 05 14). *Wi-Fi Alliance*. Ανάκτηση από Wi-Fi Alliance: https://www.wi-fi.org/

ZenElectro. (2019). *https://zenelectro.blogspot.com/*. Ανάκτηση από https://www.youtube.com/watch?v=kG1J2tNc_tg&t=0s

Παράρτημα Α Αναλυτικές εντολές εγκατάστασης

Α1 Εγκατάσταση του Docker από το επίσημο Repository

Η παρακάτω οδηγίες αφορούν την εγκατάσταση του docker σε Ubuntu Server 18.04(LTS) Πριν ξεκινήσει ή διαδικασία της εγκατάστασης θα πρέπει να είναι βέβαιο ότι δεν είναι καμία έκδοση του Docker είδη εγκατεστημένη. Αυτό μπορεί να επιτευχθεί με την παρακάτω εντολή: Θα πρέπει η εντολή αυτή να επιστέψει μήνυμα ότι κανένα από τα πακέτα δεν είναι εγκατεστημένο. (Official docker documentation, 2021)

```
sudo apt-get remove docker docker-engine docker.io
```

Για να είναι εφικτή η λήψη του docker από το επίσημο Repository πρέπει να εγκατασταθούν πρώτα τα προ απαιτούμενα πακέτα.

```
sudo apt-get update
```

```
sudo apt-get install apt-transport-https ca-certificates curl software-properties-common
```

Στην συνέχεια είναι απαραίτητο να εγκαταστήσουμε το δημόσιο GPL κλειδί του Docker.

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
```

Τώρα μπορούμε να προβούμε στην εγκατάσταση του επίσημου Repository:

```
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"
```

```
sudo apt-get update
```

Τώρα με την παρακάτω εντολή θα κατέβει η πιο πρόσφατη έκδοση του Docker

```
sudo apt-get install docker-ce docker-compose
```

A2 Εγκατάσταση του ThingsBoard

Η εγκατάσταση ξεκινά με την δημιουργία ενός φακέλου για το ThingsBoard. Μέσα από τον φάκελο αυτό δίνουμε την εντολή:

```
nano docker-compose.yml
```

Θα ανοίξει ένα πρόγραμμα επεξεργασίας κειμένου που μέσα σε αυτό δίνουμε τον παρακάτω κώδικα:

```
  version: '2.2'
services:
 mytb:
  restart: always
  image: "thingsboard/tb-postgres"
  ports:
   - "8080:9090"
   - "1883:1883"
   - "7070:7070"
   - "5683-5688:5683-5688/udp"
  environment:
   TB_QUEUE_TYPE: in-memory
  volumes:
   - ~/.mytb-data:/data
   - ~/.mytb-logs:/var/log/thingsboard
```

Οι εξωτερικές πόρτες μπορούν να αλλαχθούν κατά απαίτηση. Αποθηκεύουμε και κλείνουμε το πρόγραμμα επεξεργασίας. Προτού γίνει εκτέλεση του πρέπει να φτιαχτούν μερικοί ακόμα φάκελοι, και να παραχωρηθούν τα αντίστοιχα δικαιώματα.

```
mkdir -p ~/.mytb-data && sudo chown -R 799:799 ~/.mytb-data
mkdir -p ~/.mytb-logs && sudo chown -R 799:799 ~/.mytb-logs
```

Με της παρακάτω εντολές τώρα είναι εφικτό να τρέξει το ThingsBoard:

```
docker-compose pull
docker-compose up -d
```

Οι λογαριασμοί χριστών και οι κωδικοί πρόσβασης τους είναι ως εξής:

- *System Administrator: sysadmin@thingsboard.org / sysadmin*
- *Tenant Administrator: tenant@thingsboard.org / tenant*
- *Customer User: customer@thingsboard.org / customer*

(ThingsBoard IO Comunity edition , 2021)

Παράρτημα Β – Δοκιμαστικό πρόγραμμα

Ο σκοπός του προγράμματος αυτού είναι ο έλεγχος καλής λειτουργίας του ThingsBoard. Είναι γραμμένο σε Python και αποτελεί ένα πολύ καλό εργαλείο ελέγχου, καθώς αποστέλλει δεδομένα στο ThingsBoard ακριβώς με τον ίδιο τρόπο που θα έκανε και το ESP32. Αυτό βοηθάει στον έλεγχο του δικτύου αλλά και στην επίδειξη λειτουργίας του ThingsBoard.

Ο κώδικας δεν είναι εξ ολοκλήρου δικός μου καθώς έχουν χρησιμοποιηθεί και τροποποιηθεί κατάλληλα διάφορα παραδείγματα κώδικα από το διαδίκτυο προκειμένου να επιτευχθεί το επιθυμητό αποτέλεσμα.

```
#! python3.4

import paho.mqtt.client as mqtt
import time,json
import random
def on_log(client, userdata, level, buf):
 print(buf)
def on_connect(client, userdata, flags, rc):
  if rc==0:
    client.connected_flag=True #set flag
    print("connected OK")
  else:
    print("Bad connection Returned code=",rc)
    client.loop_stop()
def on_disconnect(client, userdata, rc):
 print("client disconnected ok")
def on_publish(client, userdata, mid):
  print("In on_pub callback mid=" ,mid)
count=0
mqtt.Client.connected_flag=False#create flag in class
mqtt.Client.suppress_puback_flag=False
client = mqtt.Client("python1")      #create new instance
#client.on_log=on_log
client.on_connect = on_connect
client.on_disconnect = on_disconnect
client.on_publish = on_publish

broker="IP_address"
port =32769
topic="v1/devices/me/telemetry"
username="Access_Token"
password=""
```

```
if username !="":
 pass
client.username_pw_set(username, password)
client.connect(broker,port)     #establish connection
while not client.connected_flag:  #wait in loop
 client.loop()
 time.sleep(1)
time.sleep(3)
data=dict()
for i in range(10):
  data["speed"]=random.randint(10,180)
  data["rpm"]=random.randint(800,8180)
  data["volts"]=random.randint(11,15)
  data["gaslevel"]=random.randint(0,100)
  data["temp"]=random.randint(20,130)
  data["info"]=random.randint(999,945130)
  data["load"]=random.randint(1,100)
  data["latitude"]=random.uniform(34.6, 39.9)
  data["longitude"]=random.uniform(19.9,25.5)
  data_out=json.dumps(data) #create JSON object
  print("publish topic",topic, "data out= ",data_out)
  ret=client.publish(topic,data_out,0)  #publish
  time.sleep(1)
  client.loop()

client.disconnect()
```

Παράρτημα Γ: Ο κώδικας του ESP32

```
#include "BluetoothSerial.h"
#include <WiFi.h>
#include <PubSubClient.h>
#include <config.h>
#include "ThingsBoard.h"
#include <HardwareSerial.h>

HardwareSerial sim808(2);
WiFiClient espClient;
BluetoothSerial SerialBT;
ThingsBoard tb(espClient);

String data[5];
String state, timegps, latitude, longitude;

String message = "";
#define DEBUG true

////// thingsboard credentials
#define TOKEN "QTSxLPrur66VFczHMkeS"
#define THINGSBOARD_SERVER "192.168.10.100"

void sendTabData(String command, const int timeout, boolean debug);
String sendData(String command, const int timeout, boolean debug);
unsigned int hexToDec(String hexString);

void setup()
{
 Serial.begin(115200);

 /////////// sim808 serial /////
 sim808.begin(9600, SERIAL_8N1, 16, 17);
 sim808.print("AT+CSMP=17,167,0,0"); // set this parameter if empty SMS
received
 delay(100);
 sim808.print("AT+CMGF=1\r");
 delay(400);

 sendData("AT+CGNSPWR=1", 1000, DEBUG);
 delay(50);
 sendData("AT+CGNSSEQ=RMC", 1000, DEBUG);
 delay(150);
 ////////////////////////// SERIAL BT //////////////////////

 SerialBT.begin("ESP32test", true);
 //SerialBT.setPin(pin);
 Serial.println("The device started!");
```

```
 connected = SerialBT.connect(address);

 if (connected)
 {
  Serial.println("Connected Succesfully!");
 }
 else
 {
  while (!SerialBT.connected(4000))
  {
   Serial.println("Failed to connect.");
  }
 }
 ////////////////////////////////////////////////        START        WiFi
///////////////
 WiFi.begin(ssid, password);
 while (WiFi.status() != WL_CONNECTED)
 {
  delay(500);
  Serial.println("Connecting to WiFi..");
 }

 Serial.println("Connected to the WiFi network");

 Serial.println("setup complite");
 delay(500);
}

void loop()
{

 String cmd = message.substring(8, 10);
 String byteA = message.substring(11, 13);
 String byteB = message.substring(14, 16);
 String volts = message.substring(5, 10);
 String STRspeed;
 String STRtemp;
 String STRrpm;
 String STRgas;

 char CHARspeed[5];
 char CHARrpm[5];
 char CHARgas[5];
 char CHARvolts[5];
 char CHARlongitude[10];
 char CHARlatitude[10];

 if (!tb.connected())
 {
```

```
 Serial.print("Connecting to: ");
 Serial.println(THINGSBOARD_SERVER);

 if (!tb.connect(THINGSBOARD_SERVER, TOKEN))
 {
  Serial.println("Failed to connect");
  return;
 }
}

unsigned long currentMillis = millis();
// Send commands timer
if (currentMillis - previousMillis >= interval)
{
 previousMillis = currentMillis;
 if (r == 0)
 {
  SerialBT.println("010D"); // speed
 }
 else if (r == 1)
 {
  SerialBT.println("010C"); // rpm
 }
 else if (r == 2)
 {
  SerialBT.println("0105"); // temp
 }
 else if (r == 3)
 {
  SerialBT.println("012F"); // gas level
 }
 else if (r == 4)
 {
  SerialBT.println("ATRV"); // gas level
 }
 r++;

 if (r > 5)
 {
  r = 0;
 }
}

// Read received messages
if (SerialBT.available())
{
 char incomingChar = SerialBT.read();
 if (incomingChar != '>')
 {
```

```
 message += String(incomingChar);
}
else
{

 ////////////////////////////  SPEED 0D ////////////
 if (cmd == "0D")
 {
  STRspeed = String(hexToDec(byteA));
  STRspeed.toCharArray(CHARspeed, STRspeed.length() + 1);
  Serial.println("speed is: " + STRspeed);
  tb.sendTelemetryInt("speed", STRspeed.toInt());
 }
 ////////////////////////       RPM  0C  /////////
 //////////////          (256*A+B)/4  ////////////
 else if (cmd == "0C")
 {
  byteA = String(hexToDec(byteA));
  byteB = String(hexToDec(byteB));
  byteA = 256 * byteA.toInt();
  byteA = byteA.toInt() + byteB.toInt();
  STRrpm = byteA.toInt() / 4;
  STRrpm.toCharArray(CHARrpm, STRrpm.length() + 1);
  Serial.println("the rpm is: " + STRrpm);
  tb.sendTelemetryInt("rpm", STRrpm.toInt());
 }

 ////////////////////////       temp 05 /////////
 //////////////          A-40  ////////////
 else if (cmd == "05")
 {
  STRtemp = String(hexToDec(byteA));
  STRtemp = STRtemp.toInt() - 40;
  Serial.println("temp : " + STRtemp);
  tb.sendTelemetryInt("temp", STRtemp.toInt());
 }

 ////////////// ////////   gas level 100/255 * A ///////
 else if (cmd == "2F")
 {
  byteA = String(hexToDec(byteA));
  STRgas = 0.3921 * byteA.toInt();
  Serial.println("gas level: " + STRgas);
  STRgas.toCharArray(CHARgas, STRgas.length() + 1);
  tb.sendTelemetryInt("gaslevel", STRgas.toInt());
 }

 ////////////// ////////   volts ///////
 else if (r == 5)
 {
```

```
    Serial.print("the volts are: ");
    Serial.println(volts);
    volts.toCharArray(CHARvolts, volts.length() + 1);
    tb.sendTelemetryString("volts", CHARvolts);
    Serial.println();
   }

   message = "";

   ////////////// SiM808 coms////
   sendTabData("AT+CGNSINF", 1000, DEBUG);
   if (state != 0)
   {
    Serial.println("State :" + state);
    Serial.println("Time :" + timegps);
    Serial.println("Latitude :" + latitude);
    Serial.println("Longitude :" + longitude);

    sim808.print("AT+CMGS=\"");
    sim808.println("\"");

    delay(300);

    sim808.println((char)26); // End AT command with a ^Z, ASCII code 26
    delay(200);
    sim808.println();
    delay(2000);
    sim808.flush();
   }
   else
   {
    Serial.println("GPS Initialising...");
   }
   latitude.toCharArray(CHARlatitude, latitude.length() + 1);
   tb.sendTelemetryString("latitude", CHARlatitude);
   longitude.toCharArray(CHARlongitude, longitude.length() + 1);
   tb.sendTelemetryString("longitude", CHARlongitude);

   tb.loop();
  }
 }
 delay(20);
}

void sendTabData(String command, const int timeout, boolean debug)
{
 sim808.println(command);
 long int time = millis();
 int i = 0;
```

```
 while ((time + timeout) > millis())
 {
  while (sim808.available())
  {
   char c = sim808.read();
   if (c != ',')
   {
    data[i] += c;
    delay(100);
   }
   else
   {
    i++;
   }
   if (i == 5)
   {
    delay(100);
    goto exitL;
   }
  }
 }
exitL:
 if (debug)
 {
  state = data[1];
  timegps = data[2];
  latitude = data[3];
  longitude = data[4];
 }
}

String sendData(String command, const int timeout, boolean debug)
{
 String response = "";
 sim808.println(command);
 long int time = millis();

 while ((time + timeout) > millis())
 {
  while (sim808.available())
  {
   char c = sim808.read();
   response += c;
  }
 }
 if (debug)
 {
  Serial.print(response);
 }
 return response;
```

```
}

unsigned int hexToDec(String hexString)
{
 unsigned int decValue = 0;
 int nextInt;

 for (int i = 0; i < hexString.length(); i++)
 {

  nextInt = int(hexString.charAt(i));
  if (nextInt >= 48 && nextInt <= 57)
   nextInt = map(nextInt, 48, 57, 0, 9);
  if (nextInt >= 65 && nextInt <= 70)
   nextInt = map(nextInt, 65, 70, 10, 15);
  if (nextInt >= 97 && nextInt <= 102)
  nextInt = map(nextInt, 97, 102, 10, 15);
  nextInt = constrain(nextInt, 0, 15);
  decValue = (decValue * 16) + nextInt;
 }
 return decValue;
}
```

Αρχείο config.h:

```
///////////////// INTERNET SETINGS  ///////

const char* ssid = "Millennium Falcon";
const char* password = "St@r_wAr$-FuN";


////////////////////// BT SETINGS //////////////////

String MACadd = "AA:BB:CC:11:22:33";
uint8_t address[6] = {0x00, 0x00, 0x00, 0x00, 0x00, 0x01};
String name = "obd-2";
char *pin ="1234"; //<- standard pin would be provided by default
bool connected;


// Handle received and sent messages

// Timer: auxiliar variables
unsigned long previousMillis = 0;  // Stores last time data was published
const long interval = 3000;      // interval at which to publish sensor
readings
int r = 0 ; // speed =0 rpm =1
```