Τεχνολογικό Πανεπιστήμιο Κύπρου
Σχολή Μηχανικής και Τεχνολογίας

Μεταπτυχιακή εργασία
MSc Επιστήμη και Μηχανική Δεδομένων

ΟΛΟΚΛΗΡΩΜΕΝΟ ΣΥΣΤΗΜΑ ΑΥΤΟΜΑΤΗΣ ΚΑΤΑΓΡΑΦΗΣ ΚΑΙ ΑΝΑΛΥΣΗΣ ΑΝΩΜΑΛΙΩΝ ΟΔΙΚΟΥ ΔΥΚΤΙΟΥ

Παναγιώτης Ιακώβου

Επιβλέπων Καθηγητής
Δρ. Ευθύβουλος Κυριάκου

Λεμεσός, Μάιος 2023

Πνευματικά δικαιώματα

ΕΥΧΑΡΙΣΤΙΕΣ

Αρχικά, θα θέλαμε να ευχαριστήσουμε τον επιβλέπων καθηγητή μας Δρ. Ευθύβουλο Κυριάκου για τις συμβουλές και την καθοδήγηση που μας πρόσφερε σε όλη την διάρκεια της παρούσας πτυχιακής εργασίας.

Επίσης, θα θέλαμε να ευχαριστήσουμε τον Δρ. Τάσο Κουνούδη και ολόκληρη την οικογένεια της SignalGeneriX για την βοήθεια και την υποστήριξη καθ' όλη την διάρκεια εκτέλεσης του έργου.

Στη συνέχεια, θα θέλαμε να ευχαριστήσουμε την οικογένεια και τους φίλους μας για την συμπαράσταση που μας παρείχαν σε όλη την διάρκεια της παρούσας πτυχιακής εργασίας.

1 Περίληψη

Έχοντας εις γνώσιν την αναγκαιότητα της χρήσης του οδικού δικτύου και ειδικότερα σε χώρες που το χρησιμοποιούν ως το μοναδικό δίκτυο για την εγχωρία μετακίνηση πληθυσμού, η ανάγκη για εμπλούτηση και βελτίωση των συνθηκών γύρω από αυτό χρίζει μεγάλης σημασίας και για τον πολίτη αλλά και για το κράτος. Το οδικό δίκτυο και γενικά οι ασφαλτόδρομοι είναι ψηλά στη λίστα δαπανών ενός κράτους με ενδεικτική τιμή κόστους στα 1.5 με 2 εκατομμύρια ευρώ ανά χιλιόμετρο, και απασχολούν καθημερινά πληθώρα εργαζομένων στη κατασκευή αλλά και στη συντήρηση τους.

Η παρούσα εργασία στοχεύει στον τομέα της συντήρησης χρησιμοποιώντας μία πρωτότυπη συσκευή εξοπλισμένη με αισθητήρες(accelerometer, gyroscope, gnss) η οποία συνδέετε ασύρματα με μία κάμερα δράσης(GoPro) και ανάλογα με το οδόστρωμα δίνει εντολή έναρξης καταγραφής βίντεο στην κάμερα η οποία είναι προσκολλημένη στο πίσω μέρος του οχήματος. Με το τέλος της διαδρομής, ακολουθώντας μία αυτοματοποιημένη διαδικασία η συλλογή βίντεο αποστέλλετε στην συσκευή. Ακολούθως τα βίντεο και τα δεδομένα που συλλέγηκαν κατά την διαδρομή αποστέλνονται σε ένα κεντρικό διακομιστή όπου γίνεται ανάλυση και επεξεργασία των δεδομένων για να καταλήξουν τελικά σε μία βάση δεδομένων μέσω ενός REST API. Τα βίντεο ανακατασκευάζονται στον διακομιστή και τους προστίθεται χρονική σήμανση και συντεταγμένες(χαρτογράφηση) για να γίνεται εύκολη η ταύτιση του βίντεο με την ακριβή τοποθεσία.

Επιπρόσθετα, παρουσιάζεται περεταίρω επεξεργασία στα βίντεο, με μεθοδολογία φιλτραρίσματος και αφαίρεση σκιών για αποφυγή σφαλμάτων. Ως αποτέλεσμα έχουμε ένα καινούργιο βίντεο με σημάνσεις που υποδεικνύουν τις φθορές στο οδόστρωμα.
Αξίζει να σημειωθεί ότι η συσκευή μπορεί να συνδεθεί με όλων των τύπων οχημάτων και η όλη διαδικασία είναι εντελώς αυτοματοποιημένη για να μπορεί να χρησιμοποιηθεί από τον οποιοδήποτε.

Η εργασία αυτή θα βοηθήσει κυρίως τις υπηρεσίες επισκευής του οδικού δικτύου στον καλύτερο προγραμματισμό, αξιολόγηση και αντιμετώπιση των φθορών του οδοστρώματος και θα επωφεληθεί ολόκληρη η κοινωνία χρησιμοποιώντας δρόμους σύγχρονους και ασφαλείς.

Λέξεις κλειδιά: οδικό δίκτυο, accelerometer, χαρτογράφηση, λακκούβες, GoPro, Rest API, filtering

Περιεχόμενα

Κατάλογος Διαγραμμάτων

Κατάλογος Πινάκων

Κατάλογος Εξισώσεων

2 Εισαγωγή

Αναγνωρίζοντας τη ζωτική σημασία που έχει ο τομέας των μεταφορών μιας χώρας όσον αφορά τις οικονομικές επιπτώσεις αλλά κυρίως όσον αφορά την ποιότητα ζωής των πολιτών της, σε αυτό το κεφάλαιο αναλύεται η σημαντικότητα του οδικού δικτύου σε ένα κράτος και η ανάγκη για ανάπτυξη υπηρεσιών και έργων γύρω από αυτό. Παρουσιάζεται το οδικό δίκτυο και πώς επηρεάζετε, τον τρόπο που επηρεαζει τα συγχρονα κράτη, η φθορές που προκύπτουν και οι τρόποι αξιολόγησης του οδοστρώματος με βάση τις φθορες.

2.1 Αντικειμενικός σκοπός

Σκοπός αυτής της διπλωματικής εργασίας είναι ο σχεδιασμός και η υλοποίηση της αρχιτεκτονικής ενός ολοκληρωμένου χαμηλού κόστους αυτοματοποιημένου υπολογιστικού συστήματος το οποίο θα χωρίζεται σε επιμέρους στοιχεία για αυτόματη καταγραφή βίντεο, ανάλυση και επεξεργασία βίντεο, ανάλυση, μεταφορά, επεξεργασία και αποθήκευση δεδομένων.

Η συγκεκριμένη συσκευή-πλατφόρμα έχει ως στόχο να εξομαλύνει την διαδικασία ανίχνευσης, και καταγραφής φθορών στο οδόστρωμα απαιτώντας ελάχιστη παρέμβαση από τον χρήστη λόγο αυτοματοποιήσεων.

Επιπρόσθετα είναι μια πολύ οικονομική λύση σε σχέση με αυτά που κυκλοφορούν. Η προτεινόμενη πλατφόρμα-συσκευή στοιχίζει περίπου 1000 ευρώ.

Το προτεινόμενο αυτοματοποιημένο σύστημα αποτελείται από τα ακόλουθα υποσυστήματα:

Ι)Το πρώτο και κυριότερο υποσύστημα είναι ουσιαστικά η συσκευή αυτοματοποιημένης καταγραφής βίντεο το οποίο αποτελείτε από διάφορα μικρότερα υποσυστήματα:

- Ισχυρός μικροελεγκτής με 4 GB eMMC που υποστηρίζει Debian Linux OS (Beagle Bone Black)
- Wi-Fi, Bluetooth modules
- Αισθητήρες επιτάχυνσης
- Go Pro Κάμερα

Σκοπός της πιο πάνω συσκευής είναι η αυτόματη καταγραφή βίντεο μέσο της Go Pro Κάμερας εφόσον δοθεί το έναυσμα εκκίνησης από το accelerometer. Η συσκευή είναι υπεύθυνη για την διαχείριση και καθαρισμό των βίντεο, δεδομένων που συλλέγονται όπως επίσης και για την μεταφορά τους σε ένα κεντρικό διακομιστή(υποσύστημα ΙΙ) που βρίσκετε σε εσωτερικό δίκτυο.

Στόχος της συσκευής είναι να εξυπηρετεί τον χρήστη παρέχοντας αρκετές αυτοματοποιημένες διαδικασίες έτσι ώστε να μπορεί να χρησιμοποιηθεί και από άτομα που δεν είναι καταρτισμένα σε τεχνολογικούς τομείς.

II)Το δεύτερο υποσύστημα είναι ένας κεντρικός διακομιστής σε εσωτερικό δίκτυο που είναι υπεύθυνος για την εξαγωγή δεδομένων από τα βίντεο, τα οποία τροποποιούνται. Μετά την επεξεργασία των βίντεο τα δεδομένα συλλέγονται και αποστέλλονται μέσω microservice(Rest Api) σε έναν άλλο διακομιστή(υποσύστημα III) όπου αποθηκεύονται σε μία βάση δεδομένων. Ο κεντρικός διακομιστής που βρίσκεται στο ίδιο δίκτυο με την συσκευή είναι ένα Raspberry Pi 4.

III)Το τρίτο υποσύστημα είναι ένας VPS(virtual private server) ο οποίος φιλοξενεί ένα microservice(Rest Api), το οποίο λαμβάνει τα δεδομένα από το Raspberry Pi και τα επεξεργάζεται έτσι ώστε να μπορούν να αποθηκευτούν σε μια τύπου SQL βάση δεδομένων που επίσης φιλοξενείτε σε αυτόν. Σκοπός αυτής της οντότητας είναι η συλλογή των δεδομένων με απώτερο σκοπό μετέπειτα επεξεργασία για χαρτογράφηση, και αξιολόγηση της κατάστασης των φθορών του οδοστρώματος.

IX) Το τέταρτο υποσύστημα είναι ένα λογισμικό το οποίο παίρνει το βίντεο πρίν να περάσει απο επεξεργασία για τροποποίηση και εξαγωγή δεδομένων, και το αναλύει σε frames για να εξαλήψει πιθανές σκιές που μπορεί να παρουσιαζονται. Τα φιλτράρει και με την χρήση διαφορων τεχνικών επεξεργασίας εικόνας δημιουργεί τετράγωνα τα οποία υποδικνείουν τις φθορές. Στη συνέχεια μέσω αλγορίθμου βαζει προταιρεότητα για επισκευεί του οδοστρώματος με βάση τισ φθορες που παρουσιάζονται.

2.2 Βιβλιογραφική Ανασκόπηση

Πρόσφατα, έχουν μελετηθεί αυτόματα συστήματα ανίχνευσης λακκούβων που χρησιμοποιούν διάφορους αισθητήρες. Οι υπάρχουσες προτάσεις μπορούν να κατηγοριοποιηθούν σε μεθόδους που βασίζονται σε κραδασμούς [1] [2] [3] [4], σε μεθόδους σάρωσης με λέιζερ [5] [6] και σε μεθόδους που βασίζονται στην όραση [7] [8] [4].

Οι μέθοδοι που βασίζονται σε κραδασμούς χρησιμοποιούν γενικά διακύμανση κλίσης από δεδομένα επιταχυνσιόμετρου. Τα επιταχυνσιόμετρα έχουν χρησιμοποιηθεί για την ανίχνευση λακκούβων, λόγω του χαμηλού κόστους και των σχετικά απλών αλγορίθμων ανίχνευσης. Ωστόσο, η ακρίβεια ανίχνευσης είναι χαμηλότερη από αυτή που επιτυγχάνεται με άλλους αισθητήρες, όπως κάμερες και λέιζερ, επειδή οι λακκούβες εντοπίζονται μόνο όταν οι τροχοί ενός οχήματος διασχίζουν μια λακκούβα. Επιπλέον, μπορεί να προκύψουν ψευδείς ανιχνεύσεις όταν τα οχήματα περνούν πάνω από καλύμματα φρεατίων και προσκρούσεις ταχύτητας. Ωστόσο, η ανίχνευση λακκούβων με βάση κραδασμούς είναι πλεονεκτική λόγω του χαμηλού κόστους και της απλής μεθοδολογίας παρά τους περιορισμούς της. Πολλές μελέτες έχουν διεξαχθεί σε μια προσπάθεια να αυξηθεί η ακρίβεια της ανίχνευσης με βάση τους κραδασμούς, σχεδιάζοντας προηγμένους αλγόριθμους και συνδυάζοντας άλλα δεδομένα αισθητήρων [1] [2] [3].

Πρόσφατα, τα smartphones έχουν προταθεί για την υποστήριξη κινητής ανίχνευσης [2], αλλά αυτές οι μέθοδοι έχουν τα ίδια προβλήματα με τις μεθόδους που βασίζονται σε κραδασμούς.

Η σάρωση με λέιζερ προσφέρει εξαιρετική απόδοση ανίχνευσης, σε σύγκριση με άλλες μεθόδους. Αυτή η προσέγγιση είναι σε θέση να συλλέγει εξαιρετικά λεπτομερείς πληροφορίες για την επιφάνεια του δρόμου χρησιμοποιώντας μια τεχνική που χρησιμοποιεί ανακλώμενους παλμούς λέιζερ για τη δημιουργία ακριβών ψηφιακών μοντέλων [5] [6]. Ωστόσο, ενώ η σάρωση με λέιζερ είναι πολύ ακριβής, ο εξοπλισμός που απαιτείται είναι εξίσου ακριβός. Επιπλέον, αυτή η μέθοδος δεν μπορεί να εφαρμοστεί σε μεγάλη περιοχή για γρήγορη ανίχνευση λακκούβων.

Ωστόσο, οι μέθοδοι που βασίζονται στην όραση είναι κατάλληλες για την ακριβή ανίχνευση λακκούβων σε μια ευρεία περιοχή με χαμηλό κόστος. Πολλές προσεγγίσεις που χρησιμοποιούν εικόνες 2D και δεδομένα βίντεο έχουν μελετηθεί. Η ανίχνευση λακκούβων χρησιμοποιώντας 2D εικόνες εισήχθη αρχικά από τους Koch και Brilakis [9]. Η μέθοδός τους περιελάμβανε την αναζήτηση συγκεκριμένων χαρακτηριστικών λακκούβων και τον προσδιορισμό των περιοχών των λακκούβων. Χρησιμοποίησαν ένα τηλεκατευθυνόμενο πρωτότυπο οχήματος ρομπότ εξοπλισμένο με κάμερα web (μια αυτόματη εστίαση HP Elite) εγκατεστημένη σε περίπου 60 cm πάνω από το έδαφος.

Οι Buza et al. εισήγαγε μια νέα μέθοδο βασισμένη στην όραση χωρίς επίβλεψη που δεν απαιτεί ακριβό εξοπλισμό, πρόσθετους αλγόριθμους φιλτραρίσματος ή μια φάση εκπαίδευσης [10].

Οι Jog et al. παρουσίασαν μια νέα προσέγγιση βασισμένη στη δισδιάστατη αναγνώριση και την τρισδιάστατη ανακατασκευή για την ανίχνευση και τη μέτρηση του πλάτους, της ποσότητας και του βάθους των λακκούβων χρησιμοποιώντας μια μονόφθαλμη κάμερα τοποθετημένη στο πίσω μέρος ενός οχήματος [11].

Οι περισσότεροι υπάρχοντες αλγόριθμοι ανίχνευσης λακκούβων που βασίζονται στην όραση χρησιμοποιούν εικόνες 2D που συλλέγονται από το Διαδίκτυο και συγκεκριμένες κάμερες υψηλής ανάλυσης. Επιπλέον, αυτοί οι αλγόριθμοι πρέπει να υλοποιούνται σε ισχυρούς επιτραπέζιους υπολογιστές.

Το 2019, οι Ali Anaissi et al πρότειναν ένα virtual road network inspector (VRNI) ο οποίος ήταν υπεύθυνος χρησιμοποιώντας μόνο μετρήσεις επιταχυνσιόμετρου παρατηρούσε την κατάσταση του οδοστρώματος. Αυτό γινόταν εφικτό χρησιμοποιώντας Support Vector Machines πάνω στα δεδομένα που συλλέγονταν με ακρίβεια 97.5% και false alarm rate 4%. Οι συγκεκριμένη αισθητήρες δοκιμάστηκαν και εφαρμόστηκαν για αυτό το πρωτότυπο έργο σε σχολικά λεωφορεία στην Αυστραλία.

Οι Christian Koch και Ioannis Brilakis [9] χρησιμοποίησαν 120 εικόνες οδοστρώματος (training and test set) σε αλγόριθμο κατασκευασμένο σε MATLAB ο οποίος με βάση το histogram shape-based thresholding ανίχνευε τις λακκούβες στο οδόστρωμα. Τα αποτελέσματα της όλης διαδικασίας φαίνεται να έχουν μεγάλο ποσοστό ακρίβειας, αφού ο αλγόριθμος είναι σε θέση να μπορεί να σχεδιάσει στην υφιστάμενη εικόνα το περίγραμμα της φθοράς.

Το 2020 οι Ashwini KS et al [13], πρότειναν μία οικονομική μέθοδο η οποία χρησιμοποιεί ένα smartphone και μια μονάδα OBD-II για τον εντοπισμό και τον εντοπισμό λακκούβων στους δρόμους. Χρησιμοποιεί δεδομένα όρασης, δεδομένα αισθητήρων και δεδομένα OBD για τη

δημιουργία και την επικύρωση εναυσμάτων που προκαλούνται από λακκούβες. Οι προτεινόμενες μέθοδοι ενεργοποίησης εικόνας και ενεργοποίησης δεδομένων αναγνωρίζουν τις λακκούβες στους δρόμους και δημιουργούν ένα έναυσμα μέσω της επεξεργασίας εικόνας/βίντεο και επεξεργασίας δεδομένων αντίστοιχα. Η προτεινόμενη εργασία είχε τοποθετημένο ένα smartphone τοποθετημένο στο αυτοκίνητο για να καταγραφεί βίντεο σε συνεχόμενη χρήση και να επισημαίνει τα εναύσματα.

Το 2020 οι Braian Varona et al [1], προτείναν μια προσέγγιση deep learning που τους επίτρεπε (α) να προσδιορίζουν αυτόματα τα διαφορετικά είδη οδοστρώματος και (β) να διακρίνουν αυτόματα τις λακκούβες από τις αποσταθεροποιήσεις που προκαλούνται από ανωμαλίες ταχύτητας ή ενέργειες του οδηγού στο πλαίσιο εφαρμογής που βασίζεται στο crowdsensing . Συγκεκριμένα, αναλύουν και εφαρμόζουν διαφορετικά μοντέλα deep learning: συνελικτικά νευρωνικά δίκτυα, δίκτυα LSTM και reservoir computing models. Τα πειράματα πραγματοποιήθηκαν με πληροφορίες πραγματικού κόσμου και τα αποτελέσματα έδειξαν μια πολλά υποσχόμενη ακρίβεια (98%) στην επίλυση και των δύο προβλημάτων.

Οι Kelvin C.P. Wang et al [10], περιγράφουν την ανάπτυξη ενός πολυλειτουργικού συστήματος σε πραγματικό χρόνο για την απόκτηση και ανάλυση δεδομένων οδοστρώματος, ιδιαίτερα για την έρευνα κινδύνου της επιφάνειας του οδοστρώματος και τη διαχείριση περιουσιακών στοιχείων στην άκρη του δρόμου. Αυτό το σύστημα, Digital Highway Data Vehicle (DHDV), συνδυάζει τις τεχνολογίες της ψηφιακής απεικόνισης με βάση τον φωτισμό λέιζερ, του αδρανειακού προφίλ και της χαρτογράφησης GPS σε ένα ολοκληρωμένο σύστημα για την ολοκλήρωση των πολλαπλών εργασιών έρευνας και διαχείρισης δεδομένων οδοστρώματος. Η ανάλυση των εικόνων του οδοστρώματος είναι 1 mm τόσο σε διαμήκη όσο και σε αντίστροφες κατευθύνσεις με τη χρήση καμερών γραμμής υψηλής ανάλυσης που διασχίζουν ολόκληρο το πεζοδρόμιο μιας λωρίδας. Η ανάλυση του οδοστρώματος και των ανωμαλιών του γίνονται σε πραγματικό χρόνο. Αξίζει να σημειωθεί ότι το πιο πάνω σύστημα κοστίζει γύρω στα 1 εκ. δολάρια και είναι ένα ολοκληρωμένο όχημα.

Φυσικά, οι υπάρχουσες μέθοδοι δεν μπορούν να υποστηρίξουν τον εντοπισμό λακκούβων χρησιμοποιώντας ένα ολοκληρωμένο οικονομικό και αυτόνομο σύστημα. Έτσι, σε αυτή την εργασία προτείνουμε ένα σύστημα ανίχνευσης λακκούβων χρησιμοποιώντας μια εμπορική κάμερα Go Pro8. Το σύστημα ανίχνευσης λακκούβων που διαθέτουμε μπορεί να χρησιμοποιηθεί ως ένα αποτελεσματικό και χαμηλού κόστους σύστημα συντήρησης λακκούβων και μπορεί να βοηθήσει στην ανάπτυξη καλύτερων πολιτικών για τη συντήρηση του δρόμου.

2.3 Οδικό Δίκτυο και Πως Επηρεάζει

Το οδικό δίκτυο μίας χώρας κατέχει σημαντική θέση στην κλίμακα ποιότητας ζωής του πολίτη και για πολλά κράτη μέλη το έχει θεσπιστεί ως εθνική προτεραιότητα. Σε μικρές σε έκταση χώρες όπως είναι η Κύπρος που δεν τίθεται ανάγκη για χρήση μαζικών μέσον μεταφοράς όπως τρένα, τραμ και μετρό το οδικό δίκτυο και πιο συγκεκριμένα οι ασφαλτόδρομοι πρέπει

να είναι σε κατάσταση που να θεωρούνται πολυτέλεια, και να παρέχουν ασφάλεια σε όποιον τους χρησιμοποιούν.

Αυτός ο τομέας έχει μεγάλες προοπτικές για ανάπτυξη έρευνας και καινοτομίας (E&K), καθώς συνδέεται άμεσα με την οικονομική ανάπτυξη, τις κοινωνικές υπηρεσίες, την προσβασιμότητα, τη ρύπανση του περιβάλλοντος και τελικά την ποιότητα της καθημερινής ζωής των ανθρώπων. Οι μεταφορές αποτελούν θεμέλιο για την ευρωπαϊκή ολοκλήρωση και συνδέονται άμεσα με τη δημιουργία της ενιαίας αγοράς, η οποία προωθεί τη δημιουργία θέσεων εργασίας, την οικονομική ανάπτυξη και την κοινωνική συνοχή.

Ο τομέας των μεταφορών το 2015 αντιπροσώπευε περίπου το 9% της συνολικής Ακαθάριστης Προστιθέμενης Αξίας της οικονομίας της ΕΕ και το 9% της συνολικής απασχόλησης στην ΕΕ.

Ωστόσο, οι μεταφορές προκαλούν σημαντική κατανάλωση πόρων, ενώ παράγουν αρνητικές κοινωνικές επιπτώσεις όπως ατυχήματα και αρνητικές περιβαλλοντικές επιπτώσεις όπως εκπομπές αερίων θερμοκηπίου, θόρυβο και ατμοσφαιρική ρύπανση. Το άθροισμα αυτών των επιπτώσεων οδήγησε σε ένα εκτιμώμενο συνολικό εξωτερικό κόστος περίπου 4% του Ακαθάριστου Εγχώριου Προϊόντος της ΕΕ το 2011.

Το οδικό δίκτυο επηρεάζει άμεσα τις μεταφορές και αποτελεί βασικό παράγοντα στην ανταγωνιστικότητα των εταιριών. Οι εταιρείες έχουν ως κύριο μέλημα να προμηθεύσουν το συντομότερο δυνατό τους πελάτες τους, και μακροχρόνια κτίζουν εμπιστοσύνη στην παράδοση εξασφαλίζοντας μίας μορφής διαφήμιση.

Εικόνα 1: Λακκούβα στο οδόστρωμα(traction.gr)

2.4 Οδικό Δίκτυο, Φθορές

Η κατάσταση των ασφαλτόδρομων αφορά άμεσα τους οδηγούς και γενικότερα τους χρήστες δεδομένου ότι συνδέεται πρωτίστως με την ασφάλεια των εμπλεκομένων, την

κυκλοφοριακή συμφόρηση, το κόστος μετάβασης, την ταχύτητα στις μετακινήσεις και στην κυκλοφοριακή άνεση.

Με το πέρασμα του χρόνου είναι αδύνατον να διατηρηθεί η κατάσταση του οδοστρώματος στο επίπεδο που είχε όταν προκατασκευάστηκε. Σημαντικοί παράγοντας φθοράς και αλλοίωσης των ασφαλτόδρομων είναι, η αυξημένη κυκλοφορία, τα καιρικά φαινόμενα και η ηλικιακή αλλοίωση των υλικών κατασκευής.

Οι γερασμένοι δρόμοι και τα κακά συστήματα οδικής συντήρησης έχουν ως αποτέλεσμα μεγάλο αριθμό λακκούβων, ο αριθμός των οποίων αυξάνεται με την πάροδο του χρόνου.
Χωρίς τακτική συντήρηση των δρόμων, το ασφαλτόστρωτο των δρόμων φθείρεται γρήγορα. Οι ακατάλληλα συντηρημένοι δρόμοι περιορίζουν την κινητικότητα των οχημάτων, αυξάνουν σημαντικά το λειτουργικό κόστος του οχήματος, τον αριθμό των ατυχημάτων και το σχετικό κόστος ανθρώπινων και περιουσιακών στοιχείων.

Η αναβολή της εκτέλεσης της συντήρησης του δρόμου αυξάνει το άμεσο και έμμεσο κόστος. Εάν τα οδικά προβλήματα διορθωθούν αμέσως, το κόστος είναι γενικά χαμηλό.
Εάν τα ελαττώματα και οι φθορές είναι μακροχρόνιες και μεγάλες, τότε πρέπει να γίνει η ανακατασκευή και η τιμή του απαιτεί ακόμη μεγαλύτερα έξοδα έως και τριπλάσια από την τακτική τεχνική συντήρηση. Με πρωτοβουλία της Παγκόσμιας Τράπεζας, η μελέτη, που διεξήχθη στη Νοτιοαφρικανική Οδική Υπηρεσία SANRAL το 2004, έδειξε ότι το κόστος αυξάνεται κατά 6 φορές χωρίς την απαραίτητη συντήρηση και επισκευή του δρόμου σε 3 χρόνια από τα κεφάλαια που θα δίνονταν. Το κόστος αυξάνεται κατά 18 φορές χωρίς να γίνει συντήρηση του δρόμου για 5 χρόνια. Για να μην αυξηθεί το κόστος, συνιστάται η χρήση των οικονομικών πόρων με την εξής σειρά: συντήρηση οδών, ανακατασκευή, νέα κατασκευή.

Μελέτες στη Σκωτία (National Roads Maintenance Review by Transport Scotland το 2011) έδειξαν ότι μετά τη μείωση του κόστους συντήρησης των δρόμων, τα σημάδια της αρνητικής επίδρασης για τους χρήστες του δρόμου και το περιβάλλον είναι ορατά. Υπολογίζεται ότι σε 10 χρόνια η κοινωνία της Σκωτίας θα έχανε 1,2 € δισεκατομμύρια καθώς τα κονδύλια που διατίθενται για τη συντήρηση των δρόμων μειώνονται κατά 40%. 1,14 € η ασθενέστερη επένδυση πολλών περιοδικών συντηρήσεων αποτελεί το 1,71 € ακόμη υψηλότερο κόστος για την κοινωνία και την οικονομία.

Σημειώθηκε επίσης ότι το χρηματικό ποσό, που διατέθηκε για τη συντήρηση των δρόμων κάθε χρόνο, δεν μπορεί να είναι το ίδιο, γιατί η οικονομία επηρεάζεται από τον πληθωρισμό και λίγα χρόνια αργότερα τα ίδια χρήματα, που επενδύονται στη συντήρηση των δρόμων, είναι ανεπαρκή. Επί του παρόντος, πολλές χώρες στον τομέα των οδικών μεταφορών αντιμετωπίζουν μια δύσκολη κατάσταση, όταν οι προϋπολογισμοί συντήρησης και επισκευής των δρόμων μειώνονται κάθε χρόνο (ή τουλάχιστον δεν αυξάνονται, αν και το κόστος συντήρησης ακινήτων αυξάνεται), αλλά ταυτόχρονα απαιτείται διατηρούν τουλάχιστον ισοδύναμη με την τρέχουσα κατάσταση του δρόμου και, σε ορισμένες περιπτώσεις, ακόμη καλύτερη [9].

2.5 Αξιολόγηση Οδικού δικτύου

Για την ασφαλτόστρωση ή την ανακατασκευή του δρόμου και την επιλογή της συντήρησης, είναι απαραίτητο να αξιολογηθούν και να αξιολογηθούν πολλές επιλογές και εναλλακτικές λύσεις. Το ποσό τους καθορίζεται από τις φυσικές συνθήκες, τα πρότυπα σχεδιασμού και κατασκευής δρόμων, υπάρχοντα υλικά κατασκευής και συντήρησης, τεχνολογίες και την τιμή τους, απαιτήσεις για αυτοκίνητα και άλλους χρήστες του δρόμου, στην οργάνωση κυκλοφορίας, πρότυπα συντήρησης οδών, κανονιστικά και νομικά έγγραφα και διάφοροι άλλοι παράγοντες.

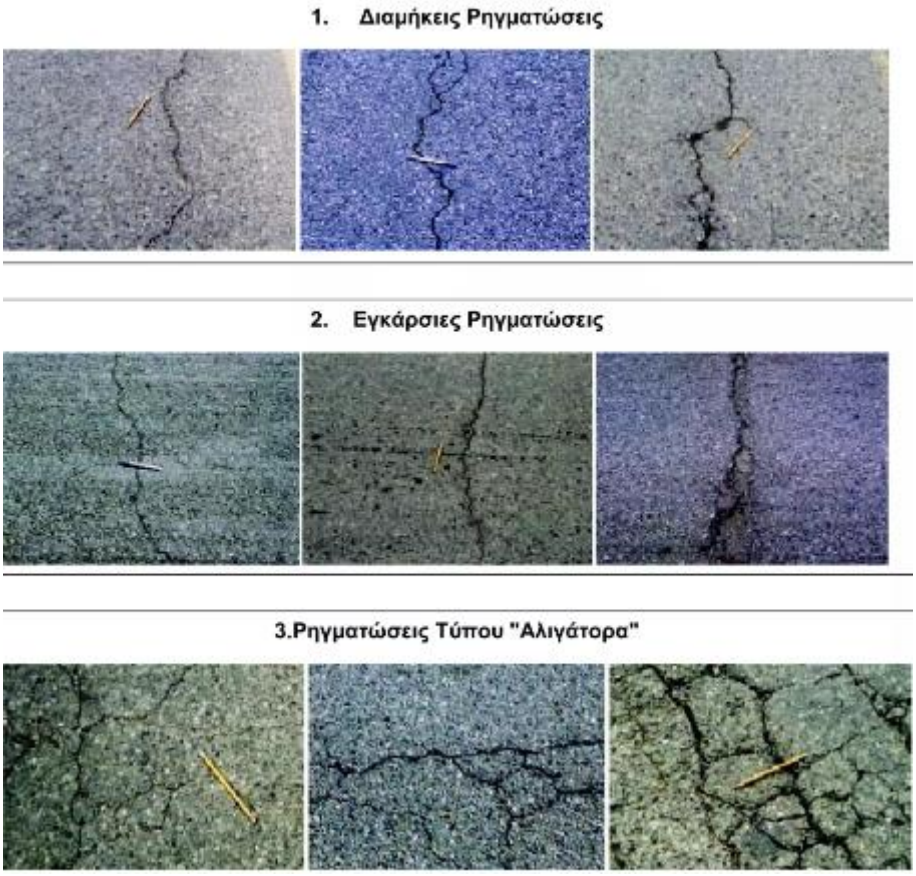
Ο καλός σχεδιασμός και το αποτελεσματικό σύστημα συντήρησης βοηθούν στην αντιμετώπιση απρόβλεπτων περιστάσεων και στη διατήρηση της δομικής ακεραιότητας του συστήματος [9]
Η τιμή κατασκευής δρόμων, ανακατασκευής ή επισκευής εξαρτάται από την επιλογή των προκαθορισμένων λύσεων. Εξαρτάται επίσης από το πόσο κοστίζει η συντήρηση και οι επισκευές του δρόμου και πόσο συχνά θα εκτελούνται.

Για μια σωστή αξιολόγηση της κατάστασης του οδοστρώματος πρέπει να εξεταστούν τα επιφανειακά ποιοτικά χαρακτηριστικά του οδοστρώματος, και πώς αυτά επηρεάζουν την ποιότητα διέλευσης και κύλισης των οχημάτων. Άμεσα χαρακτηριστικά του οδοστρώματος θεωρούνται η ομαλότητα, οι επιφανειακές φθορές και η ολισθηρότητα .

Για να επιτευχθεί η πραγματική αξιολόγηση της δομικής κατάστασης του οδοστρώματος πρέπει να εξεταστεί μέρος του οδοστρώματος και η αντοχές των επιμέρους στρώσεων με μη καταστροφικές μεθόδους που επιτυγχάνονται με εξειδικευμένες συσκευές ή με καταστροφικές μεθόδους όπου η αξιολόγηση γίνεται σε δείγμα από το οδόστρωμα .

Τύπος φθοράς
Διαμήκεις ανωμαλίες
Εγκάρσιες ανωμαλίες
Ολισθηρότητα
Απόσπαση αδρανών
Ανάδυση ασφάλτου
Διαμήκεις ρηγματώσεις
Εγκάρσιες ρηγματώσεις
Λακκούβες
Αστοχίες/καθιζήσεις

Πίνακας 1:Διάφοροι τύποι φθορών [12]



Εικόνα 2:Διαφορετικοί τύποι φθορών οδοστρώματος [12]

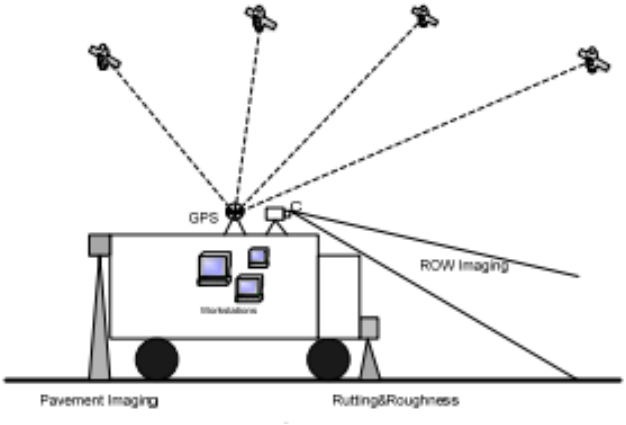
2.6 Οδικό Δίκτυο, Μέθοδοι Καταγραφής φθορών

Οι τρόποι καταγραφής φθορών αποτελούνται συνήθως από οπτική καταγραφή της κατάστασης του οδοστρώματος στο σημείο φθοράς ή με λήψη ψηφιακών εικόνων είτε με αυτόματες μεθόδους είτε με την επίδραση ατόμων. Η αυτόματη καταγραφή γίνετε με οχήματα τα οποία κινούνται με μικρές ταχύτητες στην άκρη του δρόμου και καταγράφουν τις πιθανές φθορές χρησιμοποιώντας εξιδεικευμένες κάμερες ή φωτογραφικές μηχανές και η χειροκίνητη επιτυγχάνεται με έμπειρους, ειδικευμένους τεχνικούς οι οποίοι κινούνται στην άκρη του δρόμου και καταγράφουν τις φθορές, ανωμαλίες του οδοστρώματος.
Συγκρίνοντας τις δύο μεθόδους είναι προφανές ότι η ψηφιακή καταγραφή φθορών είναι πιο αποτελεσματική και προσδίδει μια πιο λεπτομερή εικόνα για το μέγεθος του προβλήματος.

Αφού έχουν αναπτυχθεί συστήματα [12] συλλογής δεδομένων μεγάλης κλίμακας τα οποία δίνουν δυνατότητα σάρωσης οδοστρώματος 28.000 γραμμών/δευτερόλεπτο και λήψη εικόνας με 4.094 pixels, με αποτέλεσμα να μπορούν να απεικονίσουν ρήγματα με ακρίβεια 1 mm. Το πιο πάνω σύστημα(Digital Highway Data Vehicle – DHDV) έχει την δυνατότητα να συλλέξει δεδομένα αφού κινείται με ταχύτητα μέχρι 100 km/h.
Αυτό το σύστημα, Digital Highway Data Vehicle (DHDV), συνδυάζει τις τεχνολογίες της ψηφιακής απεικόνισης βάσει φωτισμού λέιζερ, του αδρανειακού προφίλ(αισθητήρες υψηλής απόδοσης, αισθητήρες υπολογισμού απόστασης, κτλ.) και της χαρτογράφησης του

Global Positioning System (GPS) σε ένα ολοκληρωμένο σύστημα για την ολοκλήρωση των πολλαπλών εργασιών έρευνας και διαχείρισης δεδομένων οδοστρώματος [10]
Αξίζει να σημειωθεί ότι το πιο πάνω σύστημα έχει δημιουργηθεί αρχικά για σκοπούς έρευνας, και στην συνέχεια αφού μελετήθηκε και διορθώθηκε σε τομείς όπως η αυτοματοποίηση και η ακρίβεια, τέθηκε σε πραγματική λειτουργία. Το πιο πάνω σύστημα προκατασκευάστηκε γύρω στο 1990 αλλά για πολλά χρόνια δεχόταν περαιτέρω ανάπτυξη.

Ένα τέτοιο σύστημα σήμερα κοστίζει περίπου 1 εκατομμύριο ευρώ και για χώρες που επενδύουν στην ανάπτυξη και συντήρηση του οδικού δικτύου, μπορούν να δαπανήσουν χρήματα για αγορά ενός τέτοιου συστήματος.



Εικόνα 3: Το DHDV σύστημα [10]

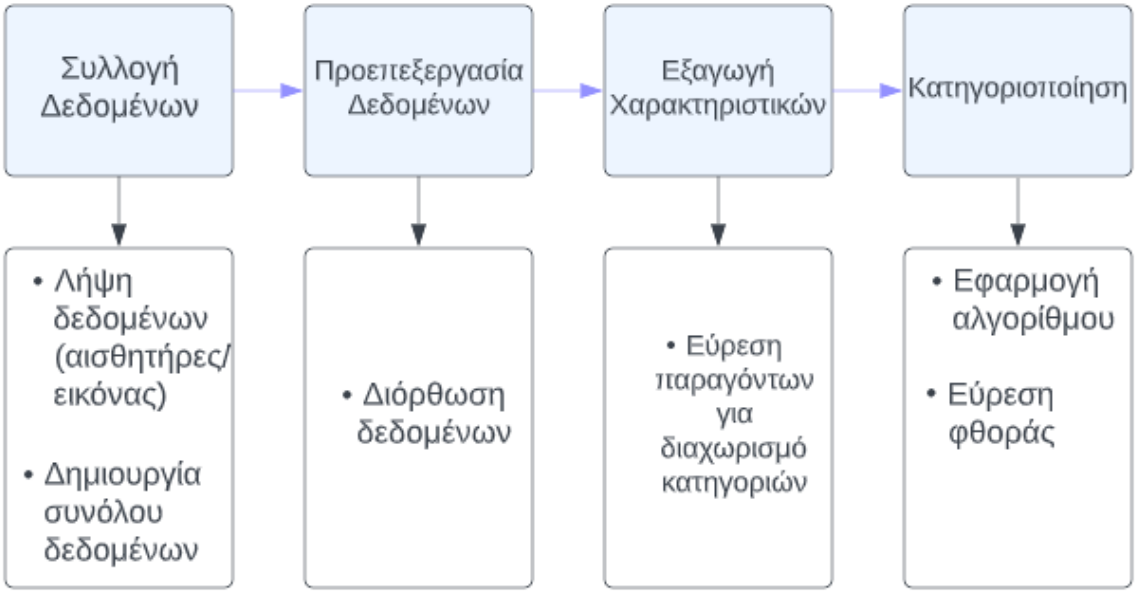
Στην αντίθετη περίπτωση η καταγραφή φθορών χειροκίνητα εμπεριέχει κινδύνους κατά την περίοδο δειγματοληψίας αφού η συλλογή δεδομένων εξαρτάται από τις γνώσεις και την εμπειρία του εκάστοτε μηχανικού, και δεν μπορούν να ελεγχτούν μεγάλα μήκη δρόμου σε μικρό χρονικό διάστημα. Επίσης για την χειροκίνητη καταγραφή απαιτείτε περιορισμός της οδού διέλευσης οχημάτων ή ακόμα και αποκλεισμός ολόκληρης της οδού, για συλλογή δεδομένων που θα βοηθήσουν στην αξιολόγηση της κατάστασης.

Επιπρόσθετα η συλλογή δεδομένων, η αποθήκευση και η επεξεργασία αποτελεί ένα αρκετά δαπανηρό κομμάτι του συστήματος διαχείρισης οδικού δικτύου. Φυσικά το κόστος εξαρτάται άμεσα από την μέθοδο μου τίθεται σε εφαρμογή για την καταγραφή. Σύμφωνα με έρευνες που έγιναν στις ΗΠΑ φαίνεται ότι η φθηνότερη μέθοδος για συλλογή δεδομένων είναι η αυτοματοποιημένη, αφού του κόστος ανεβαίνει στην περίπτωση της χειροκίνητης λόγο της αργής δειγματοληψίας και της κακής ποιότητας [12].

2.7 Οδικό Δίκτυο, Ανάλυση Δεδομένων

Καταλήγοντας στην ψηφιακή μέθοδο δειγματοληψίας φτάνουμε στο αποτέλεσμα ότι κατέχουμε ως δείγμα είτε ένα βίντεο που περιγράφει την φθορά είτε μια φωτογραφία είτε

δεδομένα από αισθητήρες. Για την ανάλυση εικονικού, αισθητήρων περιεχομένου υπάρχει πληθώρα από λογισμικά και αλγορίθμους που αναλαμβάνουν να εξάγουν την χρήσιμη πληροφορία από το δείγμα, με σχετικά ψηλούς δείκτες επιτυχίας.



Εικόνα 4: Η γενική διαδικασία εύρεσης φθορών

Για την εξαγωγή συμπερασμάτων και εύρεσης της φθοράς χρησιμοποιούνται διαφορετικοί τρόποι προσέγγισης ανάλογα με το είδος των δεδομένων. Με την χρήση οπτικών δεδομένων χρησιμοποιούνται συνήθως μέθοδοι επεξεργασίας εικόνας και τεχνολογίες μηχανικής μάθησης για να εντοπιστεί η πιθανή φθορά. Όμως παράλληλα υπάρχει το μειονέκτημα στον προσδιορισμό του βάθους της τρέχουσας λακκούβας και ο επηρεασμός του φωτός και της σκίασης προκαλώντας ανακρίβειες στους αλγορίθμους. Για την εύρεση του βάθους μίας λακκούβας συνήθως χρησιμοποιούνται τα δεδομένα από τα accelerometers τα οποία υστερούν στον προσδιορισμό του σχήματος της λακκούβας και εξαρτώνται από την ακρίβεια του ίδιου του αισθητήρα.

Για καλυτέρα αποτελέσματα η ανάλυση των πιο πάνω δεδομένων μπορεί να συνδυαστεί και να αποτελέσει 3-D απεικόνιση έχοντας όλη την πληροφορία για τον προσδιορισμό της φθοράς.

3 Μεθοδολογία και Υλοποίηση Συσκευής Καταγραφής Βίντεο

Σε αυτό το κεφάλαιο αναλύεται η μεθοδολογία που πραγματοποιήθηκε για την υλοποίηση της συσκευής αυτόματης καταγραφής βίντεο. Θα επεξηγηθούν η ανάγκες για την επιλογή των hardware modules και πώς αυτά μας προσφέρουν αυτοματοποιημένο έλεγχο της διαδικασίας.

Θα γίνει μία αναδρομή σε ολόκληρη την αρχιτεκτονική της συσκευής, και πώς μπορεί να χρησιμοποιηθεί.

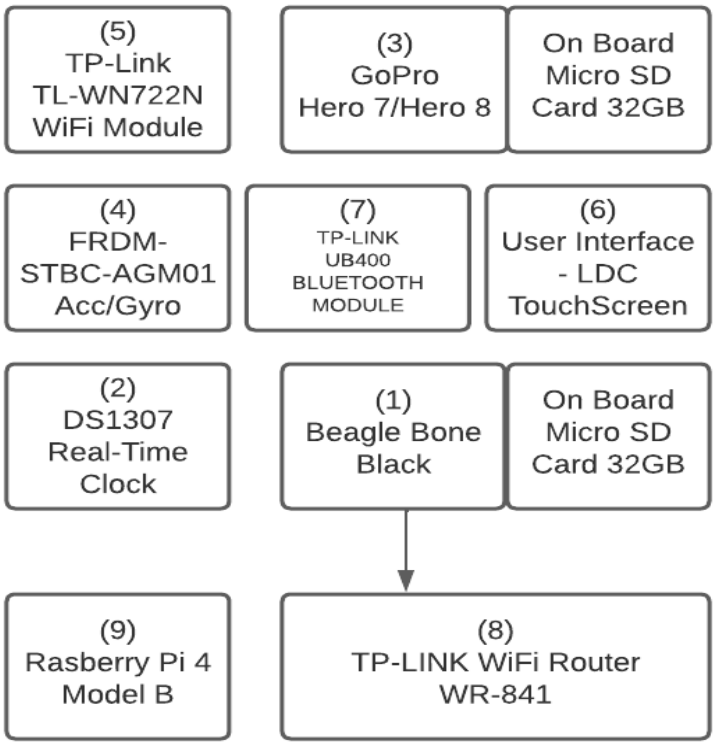
3.1 Επιλογή Υλικού Συσκευής

Η συσκευή αυτόματης καταγραφής βίντεο αναπτύχθηκε για να ανιχνεύει ανωμαλίες στους δρόμους. Αυτό επιτυγχάνεται μέσω αισθητήρων που έχουν ενσωματωθεί στη συσκευή παρακολούθησης, οι οποίοι ενεργοποιούν την κάμερα GoPro όταν υπερβούν μια τιμή κατωφλίου που έχει ορίσει ο χρήστης. Η κάμερα GoPro καταγράφει ένα βίντεο της ανωμαλίας που εντοπίστηκε και το κρατάει για ένα χρονικό διάστημα που ορίζεται προγραμματιστικά και στην συνέχεια το μεταφέρει στη συσκευή. Το βίντεο και τα γεωγραφικά δεδομένα που επανακωδικοποιούνται αργότερα μεταφέρονται και υποβάλλονται σε επεξεργασία σε μια βάση δεδομένων PostgreSQL. Μια οθόνη αφής LCD είναι ενσωματωμένη στη συσκευή για πρόσβαση σε διάφορες παραμέτρους της συσκευής.

Για την υλοποίηση της αρχιτεκτονικής υλικού που προέβλεπαν οι λειτουργίες της συγκεκριμένης εργασίας, χρησιμοποιήθηκαν και ενσωματώθηκαν τα πιο κάτω hardware modules:

Sub Module	Description
Microprocessor module	Beagle Bone Black (BBB)
Operating System	Linux
Bluetooth module	Bluetooth 4.0
Camera	Go Pro HERO 8 Black
USB hub	Powered USB Hub USB3.0 5Gbps
On-Board memory	SD memory card with 32GB capacity
WiFi module	Wireless 802.11 bgn
Accelerometer/Gyro Sensor	FRDM-STBC-AGM01 module with enabled for sensor fusion with FXAS21002C 3-axis gyroscope and FXOS8700C 6-axis integrated e-compass
RTC module	Real-time clock module (clock synchronization) DS1307
LCD Touch screen	SK-gen4-32PT with I2C, SPI and USB
3D Printed Enclosure	Designed 3D printer enclosure

Πίνακας 2: Υλικά που χρησιμοποιήθηκαν



Εικόνα 5: Διάγραμμα απεικόνισης της συσκευής καταγραφής

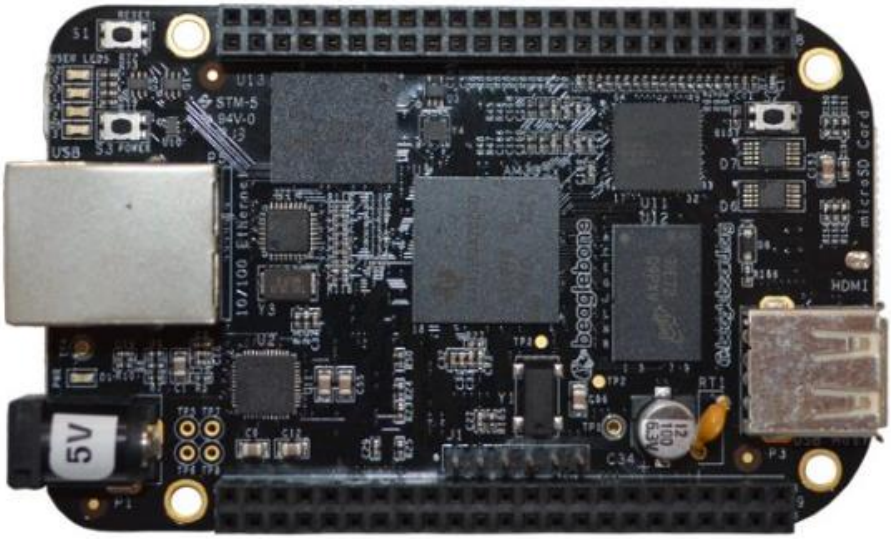
3.1.1 Microprocessor module (1): Beagle Bone Black (BBB)

Για την επιλογή του υλικού έγινε έρευνα έτσι ώστε να ικανοποιούνται όλες οι ανάγκες της εν λόγο συσκευής. Αρχικά έπρεπε να επιλεγεί το development board το οποίο έπρεπε να είναι ανοικτού κώδικα και να χρησιμοποιεί λειτουργικό σύστημα το οποίο να καθιστά εύκολη την ανάπτυξη λογισμικού. Για τον σκοπό αυτό ο μικρός υπολογιστής που χρησιμοποιήθηκε είναι το Beagle Bone Black(BBB).

Το BeagleBone Black Rev C διαθέτει επεξεργαστή TI AM335x ARM Cortex A8 που τρέχει στο 1Ghz. Διαθέτει 512 MB μνήμης RAM και 4 GB flash για την κατασκευή εξελιγμένου λογισμικού με υποδοχή κάρτας microSD για επιπλέον χώρο αποθήκευσης. Άλλα χαρακτηριστικά περιλαμβάνουν εξόδους βίντεο και ήχου HDMI, USB και θύρα δικτύου Ethernet.

Αυτή η πλακέτα διαθέτει προ εγκατεστημένο το Debian Linux OS το οποίο είναι ευρέως γνωστό λειτουργικό σύστημα. Το BeagleBone Black τροφοδοτείτε από καλώδιο Mini USB σε USB Type A είτε από Barrel Jack Male σε USB Type A και η τάση εισόδου είναι 5V.

Για την διαδικασία αποθήκευσης βίντεο από την Go Pro HERO 8 Black στο BBB προστέθηκε microSD κάρτα 32GB στην οποία μένουν τα βίντεο για κάποιό χρονικό διάστημα έως ότου ανακτηθεί σύνδεση με τον τοπικό διακομιστή για να αποσταλούν.



Εικόνα 6: Beagle Bone Black(beagleboard.org)

3.1.2 Bluetooth Module (7): TP-Link UB400

Εφαρμόζει το Bluetooth 4.0 με τεχνολογία χαμηλής ενέργειας (BLE) και έχει εμβέλεια 10 μέτρα εξυπηρετώντας τις ανάγκες της συσκευής. Το μέγεθος του είναι αρκετά μικρό έτσι δεν επιβαίνει το τελικό σχεδιασμό της συσκευής σε θέμα όγκου.

Το Bluetooth module εντάχθηκε στην αρχιτεκτονική της συσκευής αφού για την αρχική ενεργοποίηση του Wi-Fi της κάμερας χρειαζόταν σύνδεση με Bluetooth. Αυτό έφερε ως αποτέλεσμα την ανάγκη για επιπλέον USB θύρες αφού ήδη το σύστημα χρειαζόταν το Wi-FI module το οποίο ήταν απαραίτητο για μεταφορά δεδομένων και βίντεο. Έτσι προστέθηκε στο σύστημα το USB Hub με εξωτερική τροφοδοσία έτσι ώστε να προσφέρει επιπλέον συσκευές με σύνδεση USB αλλά να μπορεί να τες τροφοδοτεί και εξωτερικά για να μην επιβαρύνει την ολική ισχύ του συστήματος.

3.1.3 Camera (3): GO Pro Hero 8

Για την επιλογή της κάμερες έπρεπε να πληρούνται κάποια κριτήρια. Τα σημαντικότερα κριτήρια είναι:

- Να υποστηρίζονται πολλά διαφορετικά Resolutions έτσι ώστε να γίνουν δοκιμές για το πιο Resolution μας δίνει ικανοποιητική πληροφορία για την κατάσταση του οδοστρώματος αλλά να μην καταναλώνει μεγάλο όγκο δεδομένων.
- Να παρέχει απομακρυσμένη πρόσβαση μέσω Bluetooth ή WIFI και να έχει Api ανοιχτού κώδικα, έτσι ώστε να χειρίζεται απομακρυσμένα.
- Να παρέχει βάσεις τοποθέτησης με δυνατότητες εφαρμογής σε οποιαδήποτε επιφάνεια (για να μην εξαρτάται το είδος του αυτοκινήτου που θα την χρησιμοποιεί).
- Να μπορεί να συλλέγει δεδομένα όπως συντεταγμένες, δεδομένα επιταχυνσιόμετρου και ώρας.

- Να έχει τιμή κάτω από 500 ευρώ ώστε να μην είναι απαγορευτική η αγορά της και να μπορεί κάποιος να την αγοράσει εύκολα.

Για τους πιο πάνω λόγους επιλέχτηκε η GO Pro 8.

Εικόνα 7: Go Pro8 Camera(gopro.com)

3.1.4 Wi-Fi Module (5): TP-LINK(TL-WN722N)

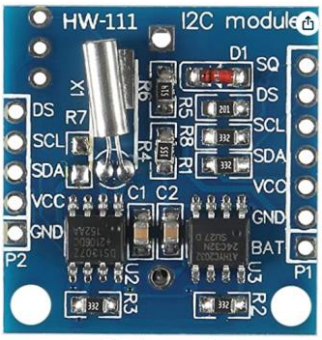
Η χρήση του Wi-Fi module έχει μεγάλη σημασία στην ολοκλήρωση της συσκευής, αφού μέσω αυτού γίνεται σχεδόν όλη η αυτοματοποίηση. Σκοπός του module είναι η σύνδεση της συσκευής με τα προ-αποθηκευμένα WIFI δίκτυα αλλά και η σύνδεση με την κάμερά.

Το Wi-Fi module TP-LINK(TL-WN722N) επιλέχτηκε μετά από δοκιμές με άλλα πιο μικρά χωρίς εξωτερική αντένα και παρουσιάστηκε πρόβλημα στην συνδεσιμότητα ειδικότερα όταν η συσκευή βρισκόταν μπροστά στον συνοδηγό και η κάμερά προσκολλημένη στον πίσω προφυλακτήρα του αυτοκινήτου.

3.1.5 Real Time Clock (2)

Το RTC module χρησιμοποιείται για να κρατά τη συσκευή συνεχώς ενημερωμένη με τον πραγματικό χρόνο. Με το ξεκίνημα της συσκευής η Go Pro λαμβάνει την ώρα από τη συσκευή.

Ο λόγος που κρίθηκε αυτό αναγκαίο είναι επειδή με την καταγραφή ενός βίντεο όταν η ώρα ήταν λάθος υπήρχε σύγχυση μεταξύ του βίντεο και της ώρας που καταγράφτηκε.
Είναι ένα ρολόι/ημερολόγιο με κωδικοποίηση BCD (binary-coded decimal) χαμηλής κατανάλωσης με 56-byte NV SRAM. Η διεύθυνση και τα δεδομένα μεταφέρονται σειριακά μέσω ενός I2C, αμφίδρομου διαύλου. Το ρολόι/ημερολόγιο παρέχει πληροφορίες για δευτερόλεπτα, λεπτά, ώρες, ημέρα, ημερομηνία, μήνα και έτος.



Εικόνα 8: RTC module(hub360.com)

3.1.6 Accelerometer/Gyro Sensor (4)

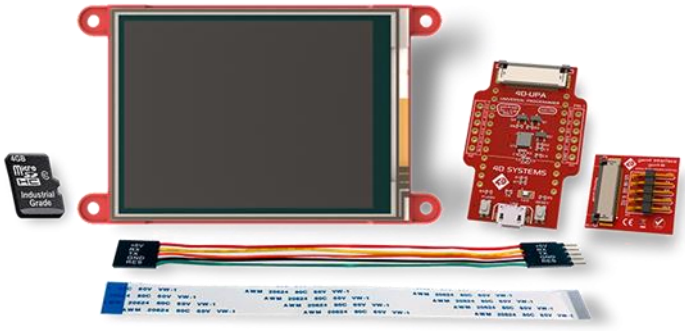
Η χρήση του αισθητήρα επιτάχυνσης ήταν αναγκαία, αφού αυτός δίνει το έναυσμα για την έναρξη καταγραφής βίντεο. Ο συγκεκριμένος αισθητήρας παρέχει δυνατότητες ρύθμισης της ευαισθησίας, την οποία μπορεί να ρυθμίσει ο χρήστης ανάλογα με την αποδοτικότητα του συστήματος. Είναι ένα μικρό, χαμηλής ισχύος, γραμμικό επιταχυνσιόμετρο 3 αξόνων και 3 αξόνων, μαγνητόμετρο συνδυασμένο σε μια ενιαία συσκευασία. Η συσκευή διαθέτει I2C ή SPI με επιταχυνσιόμετρο 14 bit και μαγνητόμετρο ADC 16 bit ανάλυση.

Εικόνα 9: Accelerometer/ Gyroscope Sensor(nxp.com)

3.1.7 LCD Touch screen (6)

Η χρήση της οθόνης κρίθηκε αναγκαία αφού υπάρχουν αρκετά στοιχεία τα οποία ο χρήστης θα πρέπει να γνωρίζει. Για παράδειγμα όπως προαναφέρθηκε, ο χρήστης έχει την δυνατότητα να ρυθμίσει πόσο ευαίσθητο θα είναι το σύστημα. Επίσης ο χρήστης πρέπει να ενεργοποιήσει για πρώτη φορά την σύνδεση μεταξύ της συσκευής και της κάμερας. Επιπρόσθετα η οθόνη ειδοποιεί για σφάλματα κατά την διάρκεια χρήσης της και δίνει την δυνατότητα απενεργοποίησης της.

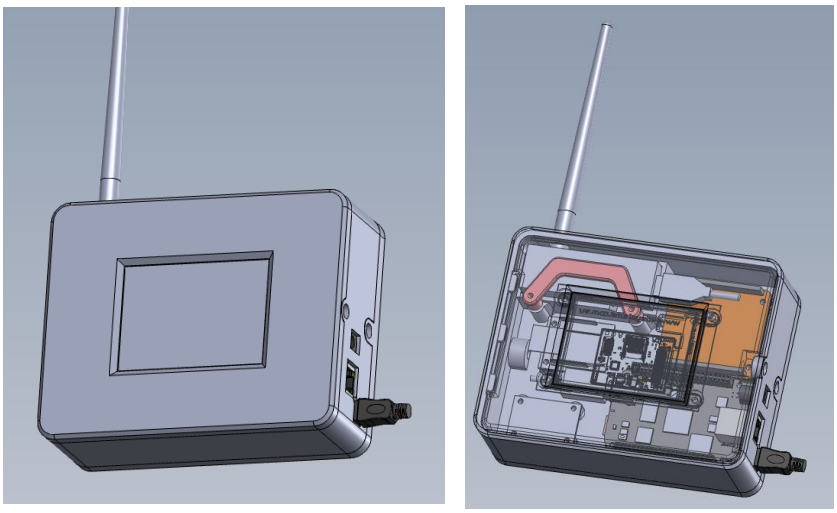
Η συγκεκριμένη οθόνη επιλέχθηκε επειδή είναι ανοιχτού κώδικα και μπορεί να προγραμματιστεί για τις ανάγκες του συστήματος, και είναι μικρού μεγέθους, έγχρωμη με γρήγορη ανταπόκριση στις αλλαγές.



Εικόνα 10: LCD touch display(4dsystems.com)

3.1.8 3D Printed Enclosure

Για την συγχώνευση και ομαδοποίησή όλων τον προαναφερθέντων υποσυστημάτων σχεδιάστηκε και τυπώθηκε στα εργαστήρια της SignalGeneriX enclosure το οποίο δίνει την αντίληψη ότι πρόκειται για προϊόν. Όλα τα υποσυστήματα τοποθετήθηκαν με άρτιο τρόπο έτσι επιτυγχάνεται μικρό μέγεθος της συσκευής που το καθιστά εργονομικό.



Εικόνα 11: 3D Enclosure Design

Εικόνα 12: Η συσκευή καταγραφής μετά την συναρμολόγηση και την τοποθέτηση στο 3D enclosure

3.2 Αρχή λειτουργίας της συσκευής για καταγραφή βίντεο

Σε αυτό το σημείο θα επεξηγηθεί ο τρόπος λειτουργίας της συσκευής. Αρχικά έχουμε την συσκευή που αποτελείτε από όλα τα επιμέρους στοιχεία που αναλύθηκαν πιο πάνω και την κάμερα. Ως πρώτο βήμα ορίζουμε την τοποθέτηση της κάμερας σε ένα επίπεδο σημείο στο πίσω μέρος του αυτοκινήτου. Αυτό το σημείο είναι πολύ σημαντικό για την λήψη των βίντεο αφού η γωνία και το ύψος επηρεάζουν άμεσα την εικόνα και τα χαρακτηριστικά μίας καταγραφής.

Για λόγους προστασίας ιδιωτικής ιδιοκτησίας η κάμερα πρέπει να έχει κλίση σχετικά κάθετη προς το οδόστρωμα για να αποφεύγονται οι λήψεις βίντεο με πινακίδες αυτοκίνητων και το resolution της κάμερα πρέπει να είναι 720p x 1080p και πάνω έτσι ώστε να είναι ευκρινείς τα χαρακτηριστικά της οδού.

Εικόνα 13: Mounting camera on the car(dall.e)

Το επόμενο βήμα είναι η ενεργοποίηση της συσκευής η οποία ενεργοποιείται συνδέοντας την στην υποδοχή που εμπεριέχεται σε όλα τα αυτοκίνητα(12V) αφού πρώτα χρησιμοποιηθεί μετατροπέας 5V. Η συσκευή συνδέεται με καλώδιο USB type A to Jack Barrel Male και ξεκινά μόλις συνδεθεί το καλώδιο. Η συσκευή χρειάζεται περίπου 40 δευτερόλεπτα για να εκτελέσει τα αρχικά βήματα και να μπει σε λειτουργία το λογισμικό που βρίσκεται σε μορφή service.

Αφού ξεκινήσει το λογισμικό ο χρήστης ενεργοποιεί την κάμερα και την τοποθετεί στην βάση πάνω στο αυτοκίνητο. Στη συνέχεια μέσω της οθόνης της συσκευής πατάει το κουμπί για σύζευξη της συσκευής με την κάμερα και αναμένει περίπου 15 δευτερόλεπτα για να επιτευχθεί η σύνδεση. Ακολούθως το σύστημα είναι έτοιμο για να ξεκινήσει την καταγραφή ανωμαλιών στο οδόστρωμα.

Το σύστημα μέσο της έξυπνης αρχιτεκτονικής του μπορεί σε οποιανδήποτε στιγμή να ενημερώσει τον χρήστη αν υπάρχει πρόβλημα κατά την διαδικασία καταγραφής βίντεο, σύνδεσης ή να το ενημερώσει για το αν δέχτηκε κραδασμό και είναι έτοιμο να ξεκινήσει την καταγραφή.

Ο χρήστης έχει την δυνατότητα να αλλάξει τον δείκτη ευαισθησίας της συσκευής ανάλογα με τα αποτελέσματα που βλέπει και αισθάνεται στο οδόστρωμα. Αν δηλαδή διαπιστώσει ότι πέρασε πάνω από μια μεγάλη λακκούβα αλλά η συσκευή δεν το κατέγραψε, τότε μπορεί με έναν σύρτη να δώσει περισσότερη ευαισθησία.
Τελειώνοντας την διαδρομή, ο χρήστης είναι ιδανικότερο να απενεργοποιήσει μέσω της οθόνης την συσκευή πατώντας το κουμπί για απενεργοποίηση και όχι να αποσυνδέσει το καλώδιο όπου αυτό θα μπορούσε να προκαλέσει ζημιά στο σύστημα.

3.3 Εισαγωγή ανάπτυξης λογισμικού

Αρχικά για να ξεκινήσει η διαδικασία ανάπτυξης λογισμικού χρειάστηκε να γραφτεί το λειτουργικό σύστημα Debian Linux σε microSD card, που είναι συμβατό με το BBB. Για την ανάπτυξη λογισμικού χρησιμοποιήθηκε η γλώσσα προγραμματισμού Python στην οποία προσφέρονται πληθώρα από βιβλιοθήκες για ανάκτηση δεδομένων από αισθητήρες αλλά και για αυτοματοποιήσεις. Μετά το τέλος της εργασίας θα βρείτε προσαρτημένο και τον κώδικα που απαρτίζει την λειτουργία της συσκευής.

Το σκεπτικό πίσω από της λειτουργίες που εκτελούνται στην συσκευή καταγραφής έγινε γύρω από κάποιες βασικές αρχές ανάπτυξης λογισμικού που δίνουν στο σύστημα:

- Ευκολία στη διαχείριση
- Ευκολία στη συντήρηση
- Προσιτότητα
- Επεκτασιμότητα

- Ευκολία στην επαναχρησιμοποίηση

Στην συνέχεια θα γίνει η αναφορά για τα πρωτόκολλα που χρησιμοποιήθηκαν για την επικοινωνία μεταξύ του BBB και των περιφερειακών μονάδων/αισθητήρων.

Hardware Module	Protocol
LCD Touch screen	Uart
Accelerometer/Gyro Sensor	I2C
GO Pro Hero 8	Bluetooth (Only for Pairing)
Real Time Clock	I2C

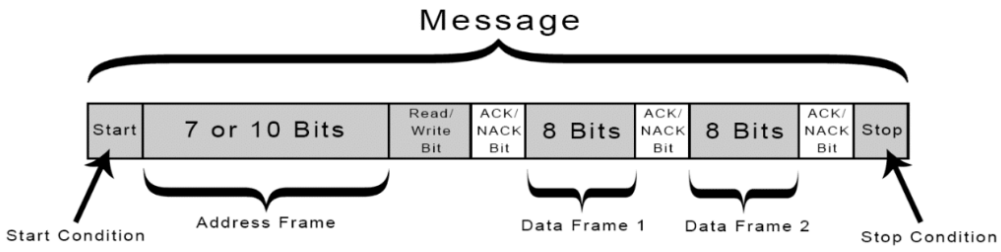
Πίνακας 3: Πρωτόκολλα επικοινωνίας επιμέρους συσκευών

3.3.1 Θεωρητικό υπόβαθρο πρωτοκόλλων

Uart: Το UART είναι ένα πρωτόκολλο επικοινωνίας υλικού που χρησιμοποιεί ασύγχρονη σειριακή επικοινωνία με ρυθμιζόμενη ταχύτητα. Ο όρος ασύχρονη σημαίνει ότι δεν υπάρχει σήμα ρολογιού για συγχρονισμό των bit εξόδου από τη συσκευή εκπομπής που πηγαίνει στο άκρο λήψης. Δεν είναι ένα πρωτόκολλο επικοινωνίας όπως το SPI και το I2C, αλλά ένα φυσικό κύκλωμα, το οποίο χρησιμοποιεί μόνο δύο καλώδια για τη μετάδοση δεδομένων μεταξύ συσκευών.

I2C: Το I2C είναι ένα σειριακό πρωτόκολλο επικοινωνίας, επομένως τα δεδομένα μεταφέρονται κομμάτι προς κομμάτι κατά μήκος ενός μόνο καλωδίου (SDA). Το I2C είναι σύγχρονο, επομένως η έξοδος των bit συγχρονίζεται με τη δειγματοληψία των bit από ένα σήμα ρολογιού που μοιράζεται μεταξύ του master και του slave. Το κάθε IC έχει δική του διεύθυνση για να μπορεί να κατονομάζετε από τον master και να ανταλλάζει πληροφορία.

Με το I2C, τα δεδομένα μεταφέρονται σε μηνύματα. Τα μηνύματα χωρίζονται σε πλαίσια δεδομένων. Κάθε μήνυμα έχει ένα πλαίσιο διεύθυνσης που περιέχει τη δυαδική διεύθυνση του IC και ένα ή περισσότερα πλαίσια δεδομένων που περιέχουν τα δεδομένα που μεταδίδονται. Το μήνυμα περιλαμβάνει επίσης συνθήκες έναρξης και διακοπής, bit ανάγνωσης/εγγραφής και bit ACK/NACK μεταξύ κάθε πλαισίου δεδομένων:



Εικόνα 14: Μήνυμα στο πρωτόκολλο I2C[circuit basics]

3.3.2 Επεξήγηση λογισμικού με διαγράμματα ροής

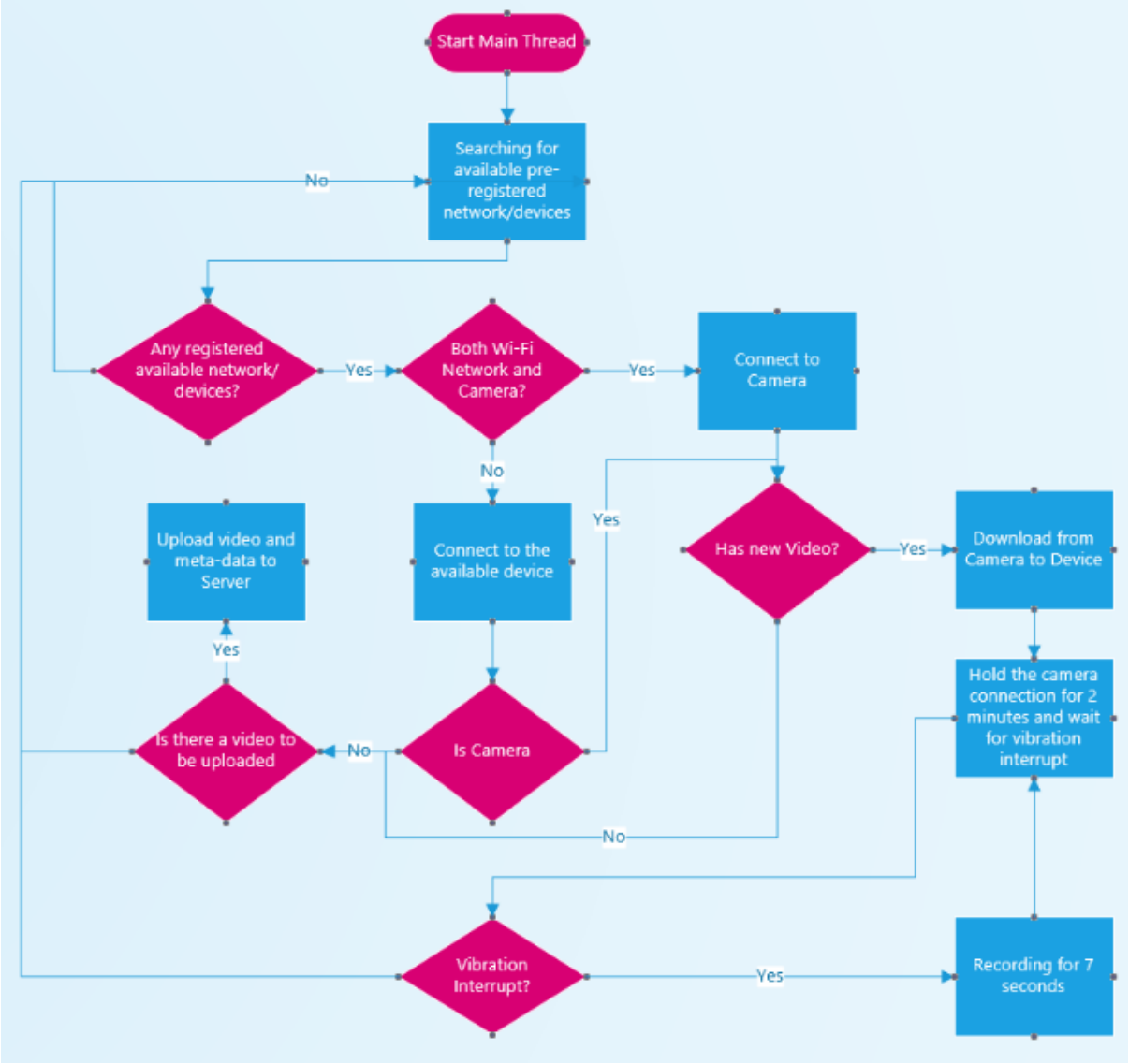
Το λογισμικό της συσκευής αποτελείτε από 5 κύριες διαδικασίες η οποίες είναι υπεύθυνες για την σωστή λειτουργία της συσκευής και την αυτοματοποίηση. Αυτές οι διαδικασίες εκτελούνται ταυτόχρονα χρησιμοποιώντας Python Daemon Threads τα οποία μοιράζονται μεταξύ τους την υπολογιστική ισχύ, και εξυπηρετούν από το παρασκήνιο την κύρια λειτουργία του κώδικα. Το λογισμικό που αναπτύχθηκε εκτελείτε από το BBB ως service το οποίο ξεκινάει με το που ξεκινήσει το λειτουργικό σύστημα, και τερματίζεται μόνο όταν απενεργοποιηθεί η συσκευή.

Μία από τις κύριες λειτουργίες του συστήματος είναι η εναλλαγή μεταξύ δικτύων Wi-Fi. Αυτό εξυπηρετεί το σύστημα στο να χειρίζεται και να καταγράφει βίντεο από την κάμερα αλλά και να ανεβάζει τα βίντεο από την συσκευή στον τοπικό διακομιστή. Για αυτόν τον λόγο για αποφυγή καθυστερήσεων τα δίκτυα που χρησιμοποιούνται από την συσκευή αποθηκεύονται μέσο του διαχειριστή δικτύων στο λειτουργικό σύστημα. Άρα κατά την διαδικασία ανίχνευσης δικτύων τα μόνα δίκτυα που μας ενδιαφέρουν είναι τα ήδη καταχωρημένα. Η πιο πάνω διαδικασία μπορεί να εκτελεστεί μέσω bash commands ή προγραμματιστικά μέσω λογισμικού που φτιάχτηκε για τον σκοπό αυτό.
Πιο κάτω φαίνονται τα 5 κύρια Threads που αποτελούν το λογισμικό του συστήματος.

Thread	Λεπτομέρειες
Main	Κύρια λειτουργία
Network Scanning	Έλεγχος Δικτύου
Accelerometer	Έλεγχος για δονήσεις από το οδόστρωμά
Screen	Έλεγχος για ανανέωση οθόνης
Failures	Έλεγχος για πιθανά σφάλματα

Πίνακας 4: Main threads of the system

- **Main Thread**



Εικόνα 15: Flowchart of Main Thread

Το πιο πάνω διάγραμμα εξηγεί την λειτουργία της κύριας λειτουργίας του λογισμικού το οποίο εκτελείτε συστηματικά. Ουσιαστικά με την βοήθεια του Network thread το Main thread ενημερώνεται συνεχώς για τα διαθέσιμα δίκτυα. Στο πρώτο βήμα ψάχνει για να βρει κάποιο από τα διαθέσιμα δίκτυα το οποίο βρίσκεται καταχωρημένο στην λίστα του διαχειριστή δικτύου. Όταν αναγνωρίσει έστω και ένα δίκτυο τότε συνδέετε μαζί του.

Στην περίπτωση της κάμερας, για να ενεργοποιηθεί το δίκτυο της πρέπει ο χρήστης να πατήσει το κουμπί «Pair GoPro 8». Πατώντας το κουμπί αυτό στέλνεται μέσω Bluetooth εντολή για ενεργοποίηση του WiFi δικτύου της κάμερας(αυτός είναι ένας περιορισμός που προστέθηκε μετά την Pair GoPro 7 στον οποίο δεν επιτρέπεται η σύνδεση από εξωτερικές συσκευές στην κάμερα εκτός αν η ενεργοποίηση γίνει με Bluetooth).

Αν είναι η κάμερα τότε συνδέετε μαζί της(για να γίνει αυτό σημαίνει ότι κατά πάσα πιθανότητα ο χρήστης βρίσκεται στο αυτοκίνητο), αν υπάρχει νέο βίντεο το κατεβάζει στη συσκευή. Με το που κατεβάσει τα βίντεο, κρατάει την σύνδεση και περιμένει για vibration interrupt για τα επόμενα δύο λεπτά. Όταν γίνει αυτό τότε καταγραφή βίντεο για 7 δευτερόλεπτα και ανανεώνετε ο χρόνος των 2 λεπτών.

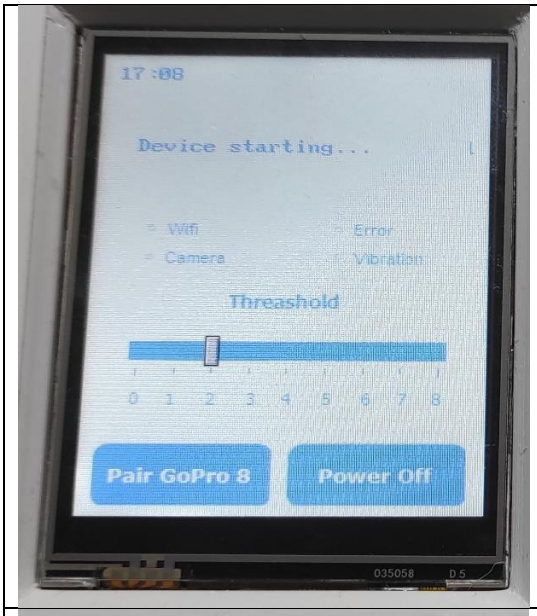
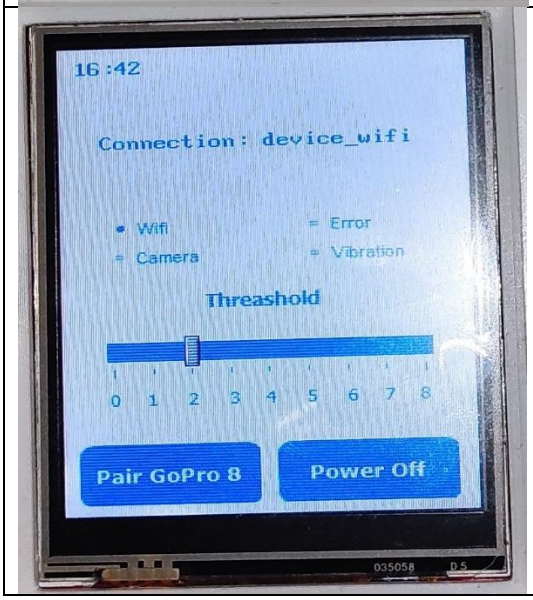
Κατά την διάρκεια τον 2 λεπτών η συσκευή είναι έτοιμη για καταγραφή βίντεο. Μόλις λήξει η προθεσμία τότε κατεβάζει τα καινούργια βίντεο στη συσκευή, αν σε οποιανδήποτε στιγμή κατά την διάρκεια της μεταφοράς των βίντεο υπάρξει vibration interrupt τότε επαναρχίζει ο χρόνος των δύο λεπτών και σταματάει η μεταφορά βίντεο.

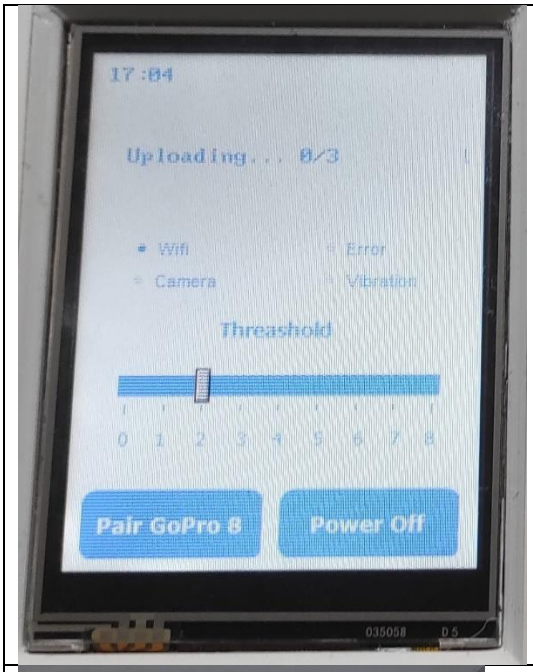
Ο χρόνος αναμονής για την καταγραφή καινούργιων βίντεο επιλέχθηκε να είναι 2 λεπτά έτσι ώστε να αποφευχθεί υπερφόρτωση στην κάρτα μνήμης της κάμερας σε περίπτωση που υπάρξουν πολλές ανωμαλίες στο οδόστρωμα και να ξεκινήσει να γίνετε επανεγγραφή των βίντεο, και χαθούν οι αρχικές λήψεις. Η κάμερα χρησιμοποιεί μία κάρτα 32 GB.

Επίσης ο χρόνος καταγραφής βίντεο είναι αρκετά μικρός και ορίστηκε στα 7 δευτερόλεπτα για δύο λόγους. Το μέγεθος των βίντεο παραμένει σχετικά χαμηλό της τάξης των 40-45 MB. Και ο δεύτερος είναι ότι η χρήσιμη πληροφορία και η ουσιαστική φθορά του οδοστρώματος βρίσκεται στα πρώτα 3 δευτερόλεπτα.

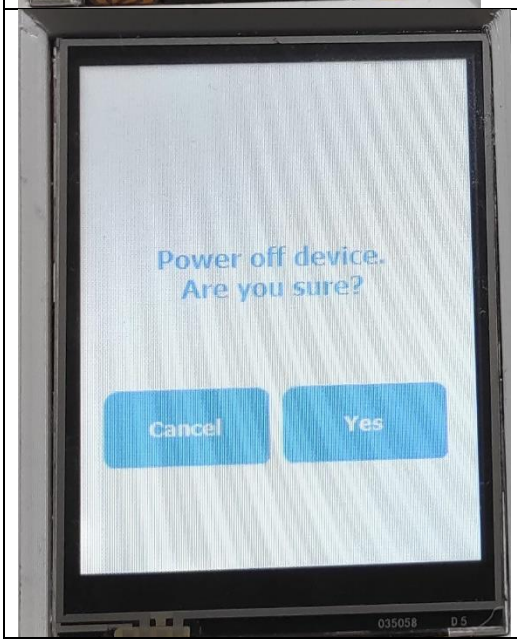
Αν το πρώτο δίκτυο που βρει η συσκευή είναι αυτό του τοπικού δικτύου τότε συνδέετε σε αυτό (για να γίνει αυτό σημαίνει ότι κατά πάσα πιθανότητα ο χρήστης βρίσκεται στο γραφείο), και αρχίζει την μεταφορά των βίντεο(αν υπάρχουν) στον κεντρικό διακομιστή (Raspberry Pi) για περεταίρω επεξεργασία.

Αν για κάποιο λόγο βρεθούν και τα δύο δίκτυα στην ίδια χρονική στιγμή τότε η συσκευή συνδέετε πρώτα στην πρώτη συσκευή όποια και να είναι και ανάλογα κατεβάζει ή ανεβάζει βίντεο. Λόγω της πειραματικής διαδικασίας, μπορούμε να πούμε ότι αυτό μπορεί να συμβεί πολλές φορές επειδή είναι πολύ πιθανόν όταν τελειώσει ο χρήστης με μία διαδρομή να μην περιμένει την κάμερα να μεταφέρει τα βίντεο στην συσκευή. Άρα είναι αναγκαίο να ενεργοποιήσει ξανά την κάμερα στο γραφείο για να πάρει τα βίντεο.
Επίσης υπήρξε η σκέψη ότι μελλοντικά η συσκευή θα μπορούσε να λειτουργήσει και με 4G router, άρα θα έμενε αποκλειστικά στο αυτοκίνητο. Για αυτό τον λόγο, όταν η συσκευή είναι συνδεδεμένη σε οποιοδήποτε δίκτυο και εκτελεί οποιανδήποτε λειτουργία και υπάρξει vibration interrupt, σταματά ότι κάνει και συνδέετε με την κάμερα (αν δεν είναι ήδη συνδεδεμένη) και μπαίνει σε mode καταγραφής βίντεο.

	Αρχική συνθήκη στην συσκευή. Σε αυτό το σημείο η συσκευή λαμβάνει την σωστή ώρα που αναγράφεται στο RTC και είναι το αρχικό σημείο ξεκινούν όλες οι διαδικασίες που αποτελούν το σύστημα.
	Σε αυτό το σημείο η συσκευή είναι σε μια φάση αναζήτησης δικτύων. Συνήθως αυτό συμβαίνει μόλις τεθεί σε λειτουργία η συσκευή μέχρι να βρει το πρώτο δίκτυο.
	Σε αυτό το σημείο η συσκευή ανίχνευσε το τοπικό δίκτυο που βρίσκετε ήδη καταχωρημένο στη συσκευή και συνδέθηκε σε αυτό.

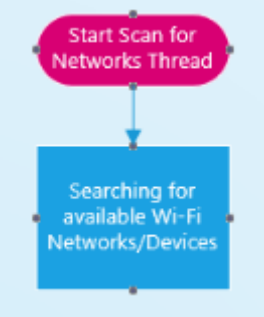
	Αφού συνδεθεί η συσκευή στο τοπικό δίκτυο τότε προσπαθεί να αποστείλει τα βίντεο που είναι αποθηκευμένα στην SD card στον κεντρικό διακομιστή μέσω του δρομολογητή.
	Ο χρήστης πάτησε το κουμπί «Pair GoPro 8» για ενεργοποίηση του WiFi.

	Σε αυτό το σημείο η συσκευή ανίχνευσε την κάμερα και ακολούθως συνδέθηκε μαζί της. Αυτό μπορεί να γίνει όταν ο χρήστης πατήσει το κουμπί «Pair GoPro 8». Έτσι ενεργοποιείτε το WiFi της κάμερας(μετά από 20 δευτερόλεπτα) και η συσκευή μπορεί να συνδεθεί μαζί της.
	Σε αυτή την φωτογραφία βλέπουμε την διαδικασία της ηχογράφησης. Η συσκευή δέχτηκε δόνηση και αμέσως άναψε η ένδειξη στην οθόνη (vibration led) και στάλθηκε εντολή στην κάμερα μέσω WiFi για καταγραφή βίντεο 7 δευτερολέπτων.

	Η διαδικασία της αποστολής των βίντεο στην συσκευή μπορεί να εκτελεστεί σε 2 χρονικές στιγμές. Η πρώτη είναι κατά την αρχική σύνδεση της συσκευής με την κάμερα. Και η δεύτερη είναι όταν από το πέρας των δύο λεπτών καταγραφής βίντεο. Μετά από το τέλος της αποστολής η συσκευή ψάχνει αυτόματα το τοπικό δίκτυο για να συνδεθεί. Αλλά αν δεν το βρει παραμένει συνδεδεμένη στην κάμερα.
	Πατώντας το κουμπί «Power Off» η συσκευή θέλει να επιβεβαιώσει ότι ο χρήστης είναι σίγουρος για την απενεργοποίηση. Αυτό γίνεται για λόγους ασφαλείας επειδή η συσκευή μπορεί να εκτελεί κάποια άλλη λειτουργία (πχ. Αποστολή βίντεο).

Πίνακας 5: Οι λειτουργίες της συσκευής καταγραφής βίντεο με αναφορά σε φωτογραφίες

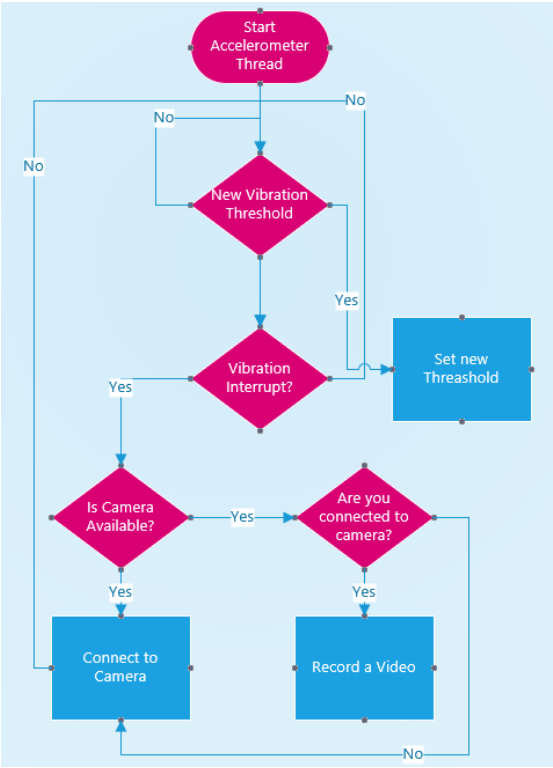
- **Network Scanning Thread**



Εικόνα 16: Network Scanning Flowchart Thread

Αυτή η διαδικασία έχει την λιγότερη προγραμματιστική υλοποίηση όμως έχει μεγάλο βάρος στην αυτοματοποίηση της συσκευής. Ουσιαστικά αυτό το Thread είναι υπεύθυνο για να ελέγχει συνεχώς για τα δίκτυα που βρίσκονται διαθέσιμα στην περιοχή. Ο έλεγχος γίνεται κατ' επανάληψη κάθε 9 δευτερόλεπτα.

- **Accelerometer Thread**



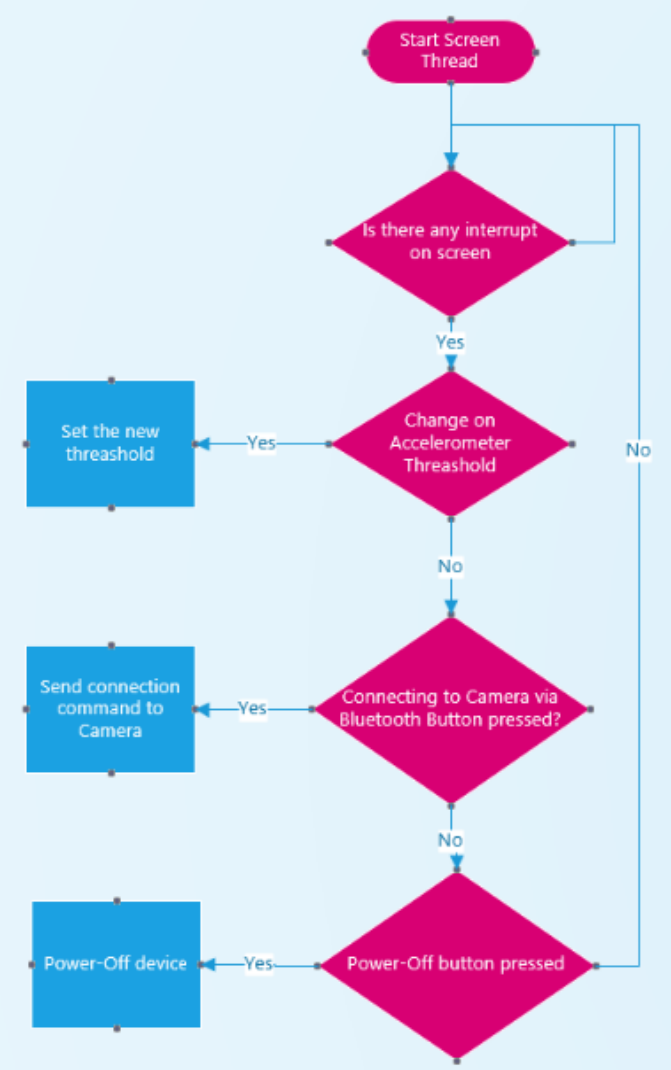
Εικόνα 17: Accelerometer Thread Flowchart

Αυτή η διαδικασία έχει ως κύριο στόχο την ενημέρωση του συστήματος για τα vibration interrupts. Μέσω αυτής ο χρήστης μπορεί να επιλέξει τον δείκτη ευαισθησίας μέσω της οθόνης ανάλογα με τις προτιμήσεις του.

Η συσκευή ενημερώνεται για τις δονήσεις που λαμβάνει το accelerometer και δίνει το σινιάλο για να ξεκινήσει η καταγραφή βίντεο.

Σε περίπτωση που υπάρξει νέα δόνηση προτού τελειώσει η καταγραφή του προηγούμενου βίντεο, τότε δεν δίνεται πληροφορία προς τη συσκευή για καινούργια καταγραφή. Αφού όπως είναι λογικό η φθορά του δρόμου καταγράφετε στο τρέχων βίντεο.

- **Screen Thread**



Εικόνα 18: Screen Flowchart Thread

Αυτή η διαδικασία είναι υπεύθυνη για να ενημερώνει για της αλλαγές που πρόκειται να συμβούν στην οθόνη αλλά και για της αλλαγές που συμβαίνουν από την αλληλεπίδραση του χρήστη.

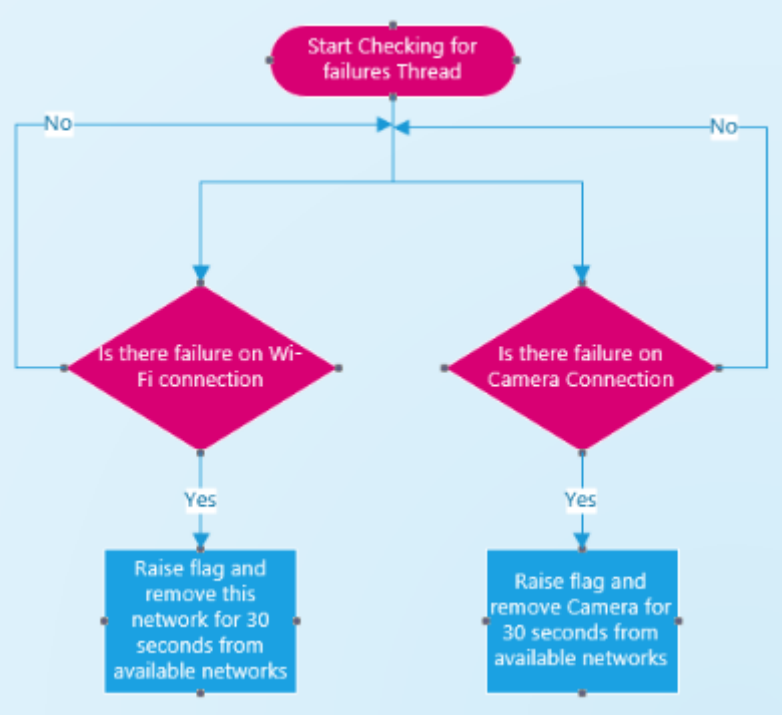
Η επικοινωνία μεταξύ της συσκευής και της οθόνης είναι αμφίδρομη, δηλαδή η συσκευή ανακτά πληροφορίες που δίνονται από την οθόνη και αφορούν την συσκευή, όπως επίσης

και η συσκευή παρουσιάζει την κατάσταση της μέσω της οθόνης για να ενημερώνει τον χρήστη.

Από την οθόνη ο χρήστης μπορεί να ενημερωθεί για πιθανά σφάλματα της συσκευής όπως: σφάλμα δικτύου, σφάλμα κάμερας, σφάλμα κατά την διαδικασία μεταφοράς βίντεο στην συσκευή, σφάλμα μεταφοράς βίντεο από την συσκευή στον διακομιστή, και ενδείξεις όπως το ξεκίνημα του προγράμματος, σύνδεσης κάμερας ή δικτύου ή ακόμα και ένδειξη ότι παρουσιάστηκε vibration interrupt.

Μέσω της οθόνης ο χρήστης μπορεί να εξασφαλίσει την αρχική σύνδεση με την GoPro 8. Μπορεί να μεταβάλει το threshold για την ευαισθησία του vibration interrupt έτσι ώστε να μπορεί να κάνει διάφορα πειράματα μέσω συστηματικής δειγματοληψίας. Και μπορεί επίσης να απενεργοποιήσει σωστά την συσκευή χωρίς να πρέπει να αποσυνδέσει το καλώδιο τροφοδοσίας με κίνδυνο καταστροφής στην συσκευή, είτε σε οποιοδήποτε επιμέρους στοιχείο.

- **Failures Thread**



Εικόνα 19: Failures Flowchart Thread

Αυτή η διαδικασία έχει σημαντικό αντίκτυπο στην σωστή λειτουργία της συσκευής, επειδή υπάρχουν προβλήματα που μπορεί να προκύψουν στον τρόπο χρήσης της από τον χρήστη. Ένα μικρό παράδειγμα θα ήταν αν ο χρήστης απομακρινέ την κάμερα από την συσκευή κατά την διαδικασία μεταφοράς των βίντεο. Αυτό θα σήμαινε αλλεπάλληλες προσπάθειες από την συσκευή για να ανακτήσει την σύνδεση. Σε μία τέτοια περίπτωση έχει οριστεί στην τρίτη προσπάθεια που θα γίνει για επαναφορά της σύνδεσης, αυτόματα αποκλείει την κάμερα

από τα διαθέσιμα δίκτυα έτσι ώστε να γλιτώνετε χρόνος αναμονής από τον χρήστη και να μπορεί η συσκευή να ανατρέξει μια άλλη λειτουργία που δεν έχει σχέση με την κάμερα. Ακολούθως όταν ανιχνευτεί ξανά η κάμερα θα συνεχίσει την μεταφορά των βίντεο από εκεί που είχε μείνει προηγούμενος.

Ένα άλλο πρόβλημα που είχε παρουσιαστεί κατά την διάρκεια δοκιμών της συσκευής ήταν να ξεκινήσει η καταγραφή βίντεο 2-3 δευτερόλεπτα προτού ο χρήσης απενεργοποιήσει τη συσκευή. Αυτό είχε αποτέλεσμα τον μη τερματισμό καταγραφής βίντεο. Έτσι για αποφυγή συλλογής τέτοιου είδους μεγάλου βίντεο(με άχρηστη πληροφορία) εφαρμόστηκαν δύο σχετιζόμενες λύσεις. Η πρώτη είναι ο έλεγχος της κάμερας κάθε φορά που υπάρχει σύνδεση με την συσκευή, αν γίνεται καταγραφή βίντεο και αν αυτό συμβαίνει τότε το σταματάει. Και η δεύτερη είναι ο έλεγχος του μεγέθους των βίντεο κατά την μεταφορά. Αν ένα βίντεο είναι μεγαλύτερο του προκαθορισμένου χρόνου(7 δευτερόλεπτα) τότε διαγράφετε.

Μέσω κάποιων έξυπνων λειτουργιών που έγιναν προγραμματιστικά, η κάθε λειτουργία που τίθεται σε εφαρμογή έχει έναν χρονικό περιορισμό. Με αυτό τον τρόπο αποφεύγονται μεγάλες χρονοτριβές και πηγές μπερδέματος της συσκευής. Τέτοιου είδους λειτουργίες εφαρμόστηκαν κατά την διαδικασία μεταφοράς των βίντεο είτε από την κάμερα στη συσκευή είτε από την συσκευή στον κεντρικό διακομιστή. Σε κάθε περίπτωση έχει οριστεί ένας χρόνος 30 δευτερολέπτων μεταφοράς του κάθε βίντεο. Αν η μεταφορά υπερβεί τον χρόνο αυτό τότε το βίντεο αυτό θεωρείτε προβληματικό και ξανά προσπαθεί να το ανεβάσει. Αν για δεύτερη φορά συμβεί αυτό τότε το βίντεο δεν αποστέλλετε στον τελικό παραλήπτη και μπαίνει σε καραντίνα έως ότου ο διαχειριστής επέμβει χειροκίνητα και αποφασίσει για την καταλληλόλητα του βίντεο.

Επιπρόσθετα όπως δείχνει και το πιο πάνω σχεδιάγραμμα, αν σε οποιανδήποτε στιγμή παρουσιαστεί πρόβλημα στη σύνδεση της κάμερας ή στο δίκτυο Wi-Fi τότε αποκλείετε το συγκεκριμένο δίκτυο για 30 δευτερόλεπτα και η συσκευή εκτελεί μια άλλη λειτουργία.

3.3.3 Επιπρόσθετες διαδικασίες που εκτελούνται στην συσκευή

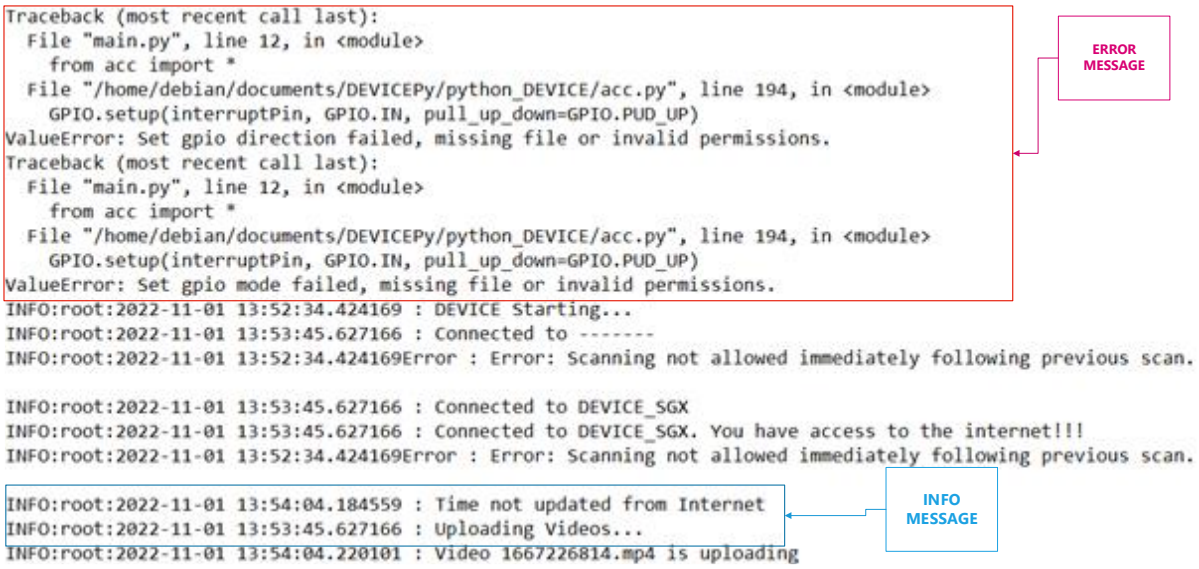
Σε αυτό το σημείο θα επεξηγηθούν επιπρόσθετες διαδικασίες που παίρνουν μέρος για την σωστή λειτουργία του συστήματος.

Όπως αναφέρθηκε πιο πριν η συσκευή είναι υπεύθυνη να ενημερώνει την κάμερα στο πρώτο βήμα κατά την σύνδεση για την ώρα της συσκευής για αποφυγή λανθασμένης ώρας των βίντεο και της ώρας εκτέλεσης της καταγραφής των βίντεο. Το BBB από μόνο του δεν παρέχει μία αυτοματοποιημένη μορφή ενημέρωσης για την τρέχων ώρα.

Έτσι θεωρήθηκε σωστό να υπάρξει μία λειτουργία που θα τρέχει σαν service και να ενημερώνεται μέσω δικτύου για την σωστή ώρα. Ουσιαστικά κάθε φορά που ενεργοποιείτε η συσκευή και συνδέετε στο Wi-Fi και έχει πρόσβαση στο διαδίκτυο ενημερώνεται για την ώρα και ρυθμίζει την ώρα στο RTC Module. Στη συνέχεια το η ώρα που κρατάει το RTC θεωρείτε η επίσημη ώρα της συσκευής και δίνετε στο main service που χειρίζεται την συσκευή. Έτσι παρέχουμε πλήρη συγχρονισμό στο δίκτυο. Σε περίπτωση που δεν γίνει σύνδεση με το Wi-Fi τότε θεωρείτε επίσημη ώρα αυτή που έχει ήδη το RTC Module αφού

αυτό δεν επηρεάζετε από τα ανοιγο-κλεισήματα της συσκευής λόγω ενσωματωμένης μπαταρίας που διαθέτει.
Για σκοπούς ανίχνευσης σφάλματος κατά την διάρκεια χρήσης της συσκευής το λογισμικό κάνει καταγραφή log έτσι ώστε να μπορεί να γίνει αποσφαλμάτωση. Αυτό ήταν αναγκαίο ειδικότερα στην διαδικασία ανάπτυξης του λογισμικού, άσχετα με το αν η οθόνη σου δίνει την πληροφορία για τα σφάλματα που παρουσιάστηκαν. Σε αυτό το log file υπάρχουν πιο ξεκάθαρες πληροφορίες που αφορούν το σφάλμα όπως επίσης και χρονικές ενδείξεις για το πότε έγινε. Εκτός από τα σφάλματα το log file περιέχει και ενδείξεις για το πότε έγινε καταγραφή βίντεο, μεταφορά βίντεο στην συσκευή ή στον διακομιστή, σύνδεση με την κάμερα ή άλλο δίκτυο.

Αυτό το αρχείο μεγαλώνει ανάλογα με τις ώρες χρήσης της συσκευής λόγω της μεγάλης συλλογής πληροφοριών. Λόγο του ότι δεν θέλουμε να υπερφορτώσουμε την συσκευή με μεγάλου όγκου log files ακολουθείτε μία προσέγγιση εκκαθάρισης που δεν επηρεάζει την χρήση του log file. Ουσιαστικά όταν το log file φτάσει το μέγεθος των 50MB δημιουργείτε ένα αντίγραφο του αρχείου και διαγράφετε όλο το εσωτερικό του αυθεντικού log file. Με αυτό τον τρόπο η διαδικασία logging μπορεί να δεσμεύσει το μέγιστο 100 MB. Μία προσεγγιστική τιμή μεγέθους 100MB αντιστοιχεί περίπου σε 25 ώρες χρήσης της συσκευής. Η διαδικασία που εξηγήθηκε πιο πάνω λειτουργεί υπό μορφή service ελέγχοντας σε κάθε ενεργοποίηση της συσκευής το μέγεθος του log file.

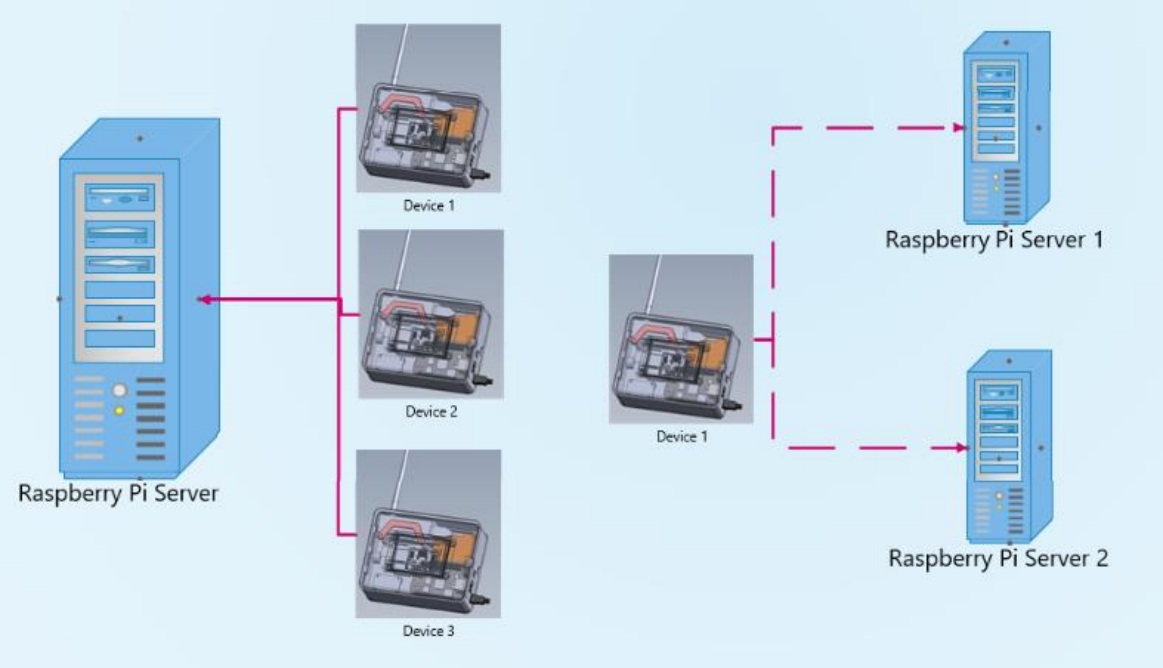


Εικόνα 20: Log File presenting errors and info messages

3.3.4 Επεκτασιμότητα του συστήματος

Κατά την ανάπτυξη λογισμικού, ειδικότερα στο λογισμικό της συσκευής λαμβανόταν υπόψη η πιθανότητα ένας διακομιστή να μπορεί να επεξεργαστεί και να διαχειριστεί δεδομένα/βίντεο από περισσότερες από μία συσκευές, αλλά και μία συσκευή να μπορεί να

εξυπηρετηθεί από περισσότερους από έναν διακομιστή.



Εικόνα 21: Επεκτασιμότητα του συστήματος. Βλέπουμε ένα τοπικό διακομιστή να εξυπηρετεί πολλές συσκευές καταγραφής. Και μια συσκευή καταγραφής να εξυπηρετείτε από πολλούς διακομιστές.

Για τον λόγο αυτό η κάθε συσκευή είναι ενημερωμένη για την στατική διεύθυνση IP η οποία παραμένει η ίδια σε όλους τους διακομιστές (ο κάθε διακομιστής βρίσκεται σε διαφορετική τοποθεσία άρα δεν υπάρχει λόγος ο καθένας να κρατάει διαφορετική διεύθυνση IP). Επίσης όλοι οι δρομολογητές που συνοδεύουν τους διακομιστές είναι ρυθμισμένοι με κοινά SSID και Password έτσι ώστε να συνδέονται εύκολα και με τη συσκευή αλλά και με τον διακομιστή.

Ο τρόπος με τον οποίο μία συσκευή αναγνωρίζεται από τον χρήστη είναι από την ονοματολογία των βίντεο που λαμβάνονται από την εκάστοτε συσκευή. π.χ. device_01_timestamp, device_02_timestamp. Η ονοματοποίηση των βίντεο γίνεται αφότου τα βίντεο μεταφερθούν από την κάμερα στην συσκευή προτού σταλθούν στον κεντρικό διακομιστή.

Επίσης για σκοπούς αποφυγής συγχύσεων η συσκευή θεωρήθηκε ότι θα ήταν καλό να συνοδεύετε πάντοτε με την ίδια κάμερα, για τον λόγο ότι ο αλγόριθμος που είναι υπεύθυνος για την σύνδεση με την κάμερα θεωρεί ως κάμερα συστήματος αυτήν που θα συνδεθεί πρώτα μαζί της. Στη συνέχεια αναπτύχθηκε λογισμικό για αποφυγή τέτοιου είδους λάθη από τους χρήστες, και έτσι όλες οι κάμερες πρέπει να περαστούν πάνω στο BBB έτσι ώστε να μπορεί ο χρήστης/ διαχειριστής ενός σταθμού να μην εξαρτάτε από την συμβατότητα συσκευής-κάμερας, αλλά να μπορεί να πάρει τυχαία μία κάμερά και να κάνει καταγραφή.

4 Μεθοδολογία και Υλοποίηση Τοπικού Διακομιστή

Σε αυτό το κεφάλαιο θα αναλυθεί και θα επεξηγηθεί η όλη διαδικασία υλοποίησης του τοπικού διακομιστή. Θα γίνει αναφορά στα καθήκοντα του διακομιστή και τι προσφέρει στο ολοκληρωμένο σύστημα. Επίσης Θα παρουσιαστούν τα επιμέρους στοιχεία που φέρουν εις πέρας τις λειτουργίες του διακομιστή.

4.1 Διακομιστής και τοπικό δίκτυο

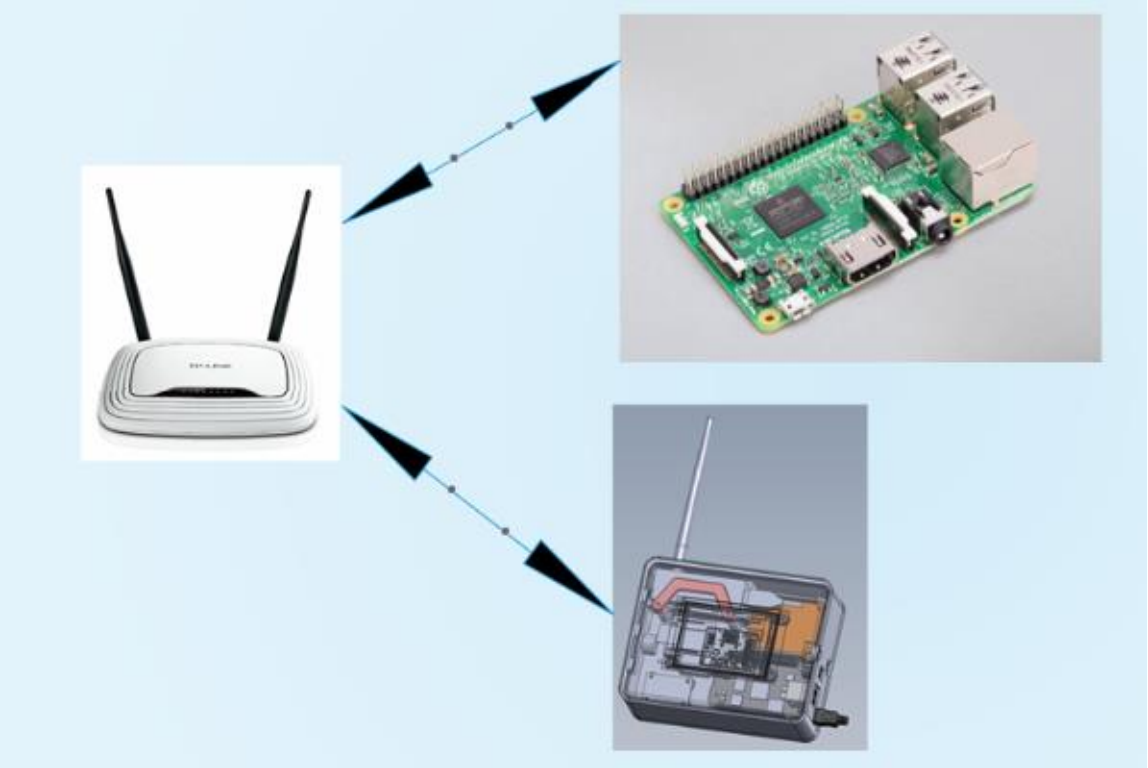
Για το κομμάτι της επεξεργασίας και μεταφοράς των βίντεο ήταν αναγκαίο να ενσωματωθεί στο σύστημα ένας δρομολογητής ο οποίος δίνει την οντότητα του τοπικού δικτύου, έτσι ώστε να μπορούν να συνδεθούν τοπικά ο διακομιστής με την συσκευή. Με αυτό τον τρόπο αποφεύγουμε την χειροκίνητη ρύθμιση δικτύου κάθε φορά που αλλάζουμε τοποθεσία στον διακομιστή και στην συσκευή. Υπό κανονικές συνθήκες κάθε φορά που αλλάζουμε τοποθεσία που κάνουμε τα πειράματα θα έπρεπε να γνωρίζουμε τα στοιχεία του τοπικού δικτύου WiFi για να έχουμε σύνδεση στο διαδίκτυο έτσι ώστε να μπορούμε να στέλνουμε τα βίντεο σε έναν κεντρικό διακομιστή για περεταίρω επεξεργασία. Έχοντας εντάξει τον δρομολογητή έχουμε καταχωρημένα στατικά IP για τον διακομιστή και για την συσκευή(BBB) και το μόνο που χεριάζετε είναι να συνδέσουμε στον δρομολογητή καλώδιο με παροχή διαδικτύου.

Υλοποιώντας το πιο πάνω μικραίνουν οι αποστάσεις μεταφοράς των βίντεο και έτσι η μεταφορά γίνετε με μεγάλες ταχύτητες χωρίς να πρέπει τα βίντεο να ταξιδέψουν μέσω διαδικτύου για να φτάσουν στον προορισμό τους. Έτσι επιτυγχάνετε και η ασφάλεια κατά την μεταφορά των βίντεο, χωρίς να διατρέχουμε κινδύνους υποκλοπής ή παρέμβασης στα δεδομένα.

Αξίζει να σημειωθεί ότι κατά την περίοδο ανάπτυξης λογισμικού στο διακομιστή τοπικού δικτύου, λήφθηκε υπόψη η περίπτωση ένταξης περισσοτέρων από μίας συσκευών καταγραφής βίντεο. Για αυτόν το λόγο κατά τον προγραμματισμό εφαρμόστηκαν τεχνικές όπου ο διακομιστής θα μπορούσε να χειρίζεται δεδομένα/βίντεο από περισσότερες συσκευές, κατανέμοντας τα δεδομένα στην βάση δεδομένων προσθέτοντας μία στήλη για να ταυτοποιείται η συσκευή από την οποία λήφθηκαν.

4.2 Αρθρωτότητα του συστήματος

Με τον τρόπο που επεξηγήθηκε πιο πάνω επιτυγχάνετε η ολική μεταφορά του συστήματος μεταφέροντας τον διακομιστή (Raspberry Pi) και τον δρομολογητή(TP Link Wireless Router) σε έναν άλλο σταθμό. Αυτό θα βοηθήσει αρκετά κατά την περίοδο χαρτογράφησης του οδοστρώματος. Πχ Αν θέλουμε να χαρτογραφήσουμε τον δήμο Μέσα Γειτονίας μπορούμε να εγκαταστήσουμε τον διακομιστή και τον δρομολογητή στο Δημαρχείο της Μέσα Γειτονίας και να θεωρούμε ως σταθμό μεταφοράς των βίντεο, το δημαρχείο. Και αντίστοιχα να μεταφέρουμε τον σταθμό μεταφοράς ανάλογα με την περιοχή που θέλουμε να χαρτογραφήσουμε.



Εικόνα 22: Ο τοπικός δρομολογητής ουσιαστικά συνδέει τον τοπικό διακομιστή και την συσκευή καταγραφής σε ένα εσωτερικό δίκτυο

Σε περίπτωση που ένας σταθμός (Raspberry Pi + Router) χρειαστεί να εξυπηρετεί περισσότερες από μία συσκευές καταγραφής, τότε το μόνο που χρειάζεται είναι να καταγραφούν οι συσκευές μέσω στατικής διεύθυνσης IP στον δρομολογητή για να μπορεί να γίνεται αποσφαλμάτωση πιο εύκολα από τον διαχειριστή. Η πιο πάνω διαδικασία προσφέρει επεκτασιμότητα στο σύστημα (μπορούν να προστεθούν καινούργιές συσκευές), ευελιξία (ευκολία στην μεταφορά/αλλαγή σταθμού) και ευκολία χρήσης.

4.3 Υλοποίηση και Μεθοδολογία Λογισμικού Διακομιστή

Σε αυτό το σημείο θα επεξηγηθούν οι διάφορες λειτουργίες που εκτελούνται στην πλευρά του κεντρικού διακομιστή και τί προσφέρει στο όλο σύστημα.

Στον κεντρικό διακομιστή (Raspberry Pi) εκτελούνται ουσιαστικά δύο μεγάλες διαδικασίες. Η πρώτη διαδικασία είναι ένα API το οποίο είναι υπεύθυνο να λαμβάνει τα βίντεο που στέλνονται από την συσκευή και η δεύτερη διαδικασία είναι υπεύθυνη για να επεξεργάζεται τα βίντεο και να προσθέτει χρονική σήμανση και συντεταγμένες, αλλά και να αποστέλλει μετά τα δεδομένα που εξάγονται από τα meta-data των βίντεο.

Το σκεπτικό πίσω από της λειτουργίες που εκτελούνται στον τοπικό διακομιστή έγινε γύρω από κάποιες βασικές αρχές ανάπτυξης λογισμικού που δίνουν στο σύστημα:

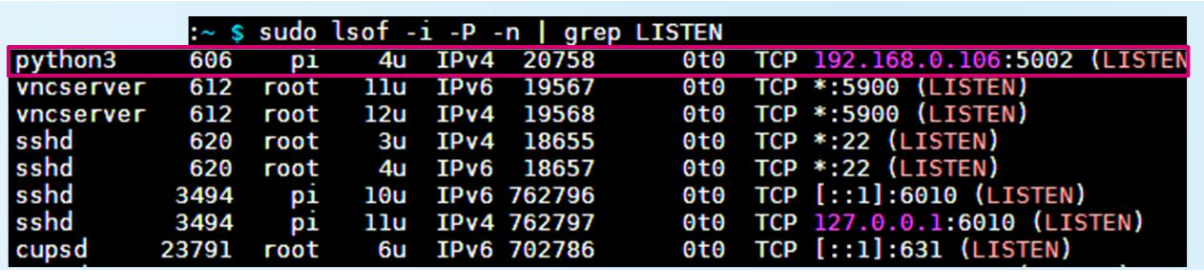
- Ευκολία στη διαχείριση
- Ευκολία στη συντήρηση
- Προσιτότητα
- Επεκτασιμότητα
- Ευκολία στην επαναχρησιμοποίηση

4.3.1 API για Αποστολή Βίντεο

Για αυτή την διαδικασία εξετάστηκαν δύο τρόποι προσέγγισης για την λήψη των βίντεο. Ο πρώτος τρόπος ήταν μέσω καναλιού σύνδεσης FTP το οποίο πρόσθετε επιπλέον φόρτο στην διαδικασία αποστολής αφού για κάθε χρήση του θα έπρεπε η συσκευή να ξεκινήσει την επικοινωνία με την χρήση συνθηματικών.

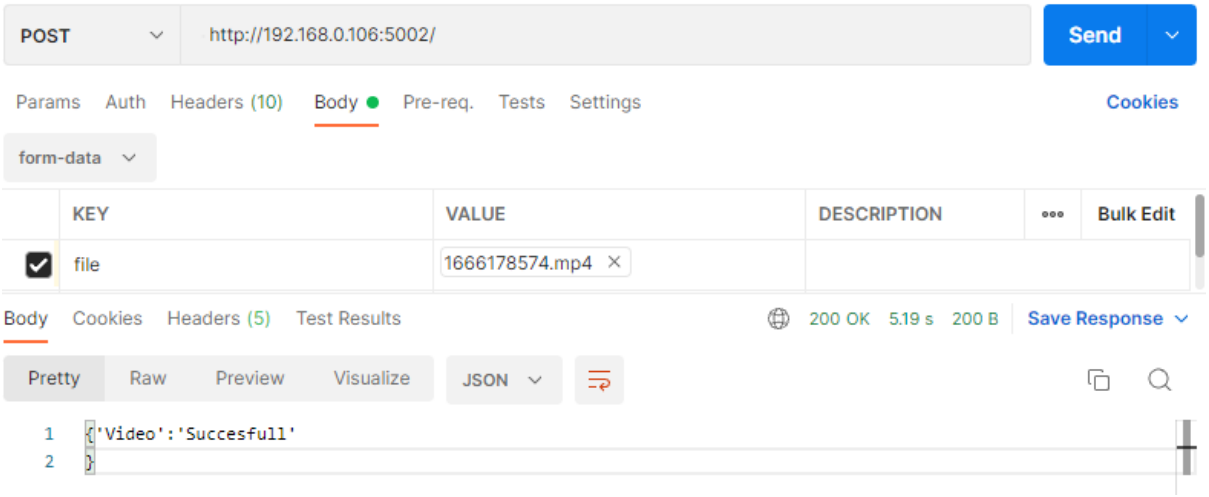
Επίσης με αυτό τον τρόπο εμπεριέχεται ρίσκο κατά την αποστολή, αφού έπρεπε να υλοποιηθούν τεχνικές έτσι ώστε να καταστεί σίγουρη η μεταφορά των βίντεο, επειδή ενδέχεται να υπάρξουν προβλήματα σύνδεσης ή προβλήματα υλικού κτλ. Ο δεύτερος τρόπος ήταν η υλοποίηση ενός restful Api το οποίο θα τρέχει υπό την μορφή service και θα εξυπηρετεί την συσκευή μέσω HTTP requests.

Το service που είναι υπεύθυνο για την λήψη των βίντεο έχει γραφτεί κάτω από το Flask Framework χρησιμοποιώντας την γλώσσα προγραμματισμού Python και εξυπηρετεί την συσκευή καταγραφής βίντεο μέσο του URL του διακομιστή και το port 5002 (192.168.0.106: 5002).



Εικόνα 23: Βλέπουμε το Rest Api να εκτελείτε και να περιμένει requests στο port 5002

Για να γίνει μία λήψη βίντεο από την συσκευή, ουσιαστικά γίνεται ένα POST request στο URL που αναφέρεται πιο πάνω στο οποίο περιέχεται το βίντεο το οποίο αποστέλλεται σε μορφή byte.



Εικόνα 24: Παράδειγμα Post Request σε περιβάλλον Postman, για αποστολή βίντεο

Γενικότερα για να γίνει ένα request στο Api για αποθήκευση ενός βίντεο το request πρέπει να είναι τύπου POST και να υποδεικνύει μέσο του URL ποιος είναι ο τελικός παραλήπτης (σε αυτή την περίπτωση το Raspberry Pi 192.168.0.106). Επίσης πρέπει να εμπεριέχει ως Key την λέξη “file” και ως Value το βίντεο που θέλουμε να αποστείλουμε.

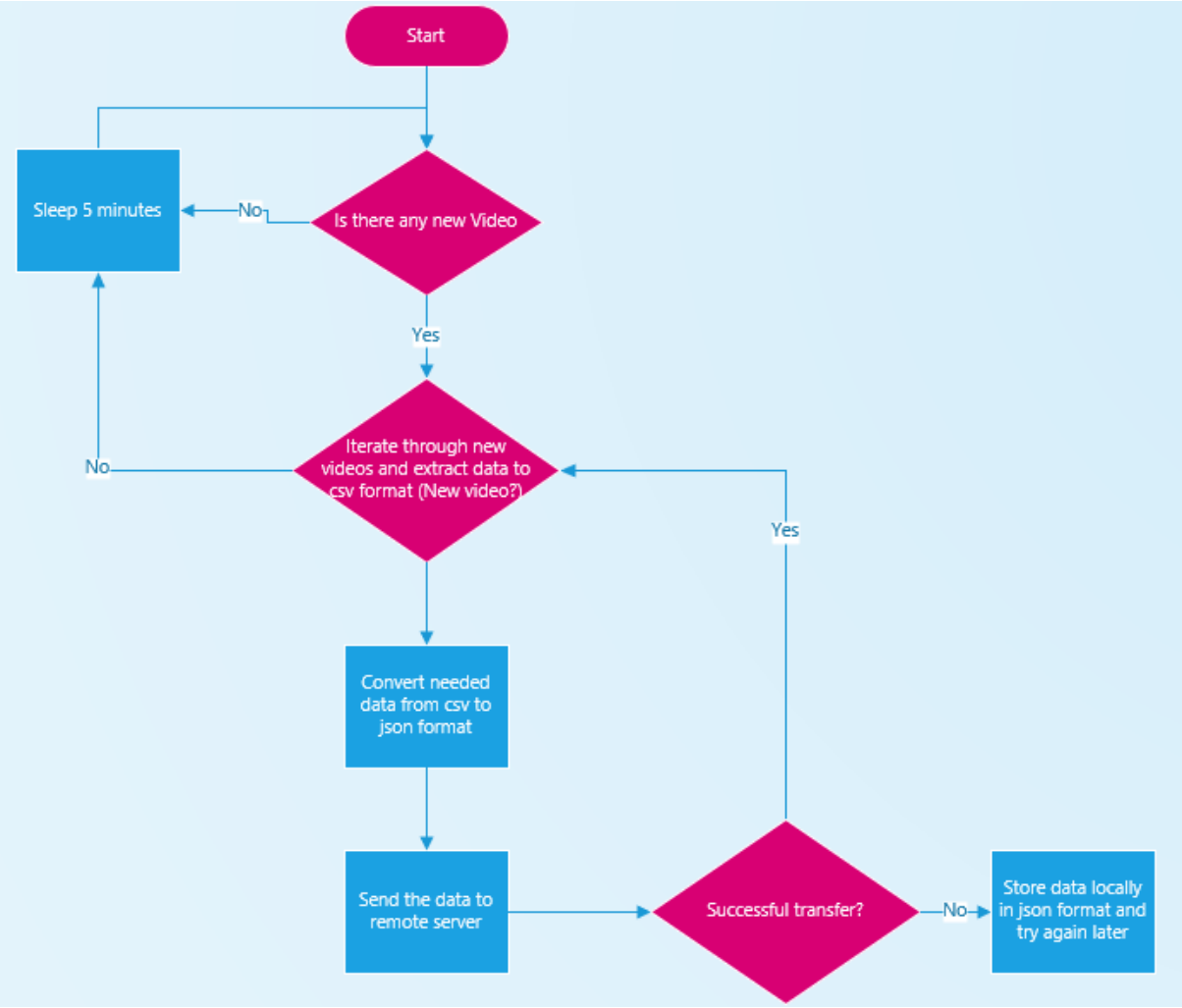
Θεωρούμε επιτυχγημένη την αποστολή όταν πάρουμε ως response code τον αριθμό 200 και ως απάντηση 'Video':'Succesfull'. Σε περίπτωση όπου λαμβάνουμε οποιανδήποτε άλλη απάντηση εκτός από 200 τότε θεωρούμε ότι η αποστολή ήταν ανεπιτυχής και επαναλαμβάνεται από την μεριά της συσκευής η διαδικασία για 3 φορές. Αν το πρόβλημα παραμένει τότε θα γίνει προσπάθεια αποστολής την επόμενη φορά που θα γίνει επιτυχής σύνδεση στον δρομολογητή.

Όταν η αποστολή ενός βίντεο γίνει με επιτυχία τότε το βίντεο αποθηκεύεται στο directory /static/video όπου και παραμένει για κάποιο χρονικό διάστημα έως ότου το επόμενο service κοιτάζει στο συγκεκριμένο directory για να βρει τα βίντεο και να τα επεξεργαστεί.

4.3.2 Επεξεργασία Βίντεο και Εξαγωγή Δεδομένων

Σε αυτή την διαδικασία γίνεται η επεξεργασία και η εξαγωγή δεδομένων που βρίσκονται στα meta-data που υπάρχουν στα βίντεο. Η γλώσσα προγραμματισμού που χρησιμοποιήθηκε για την επίτευξη τής πιο πάνω διαδικασίας είναι η Python όπου από την οποία για το κομμάτι τις εξαγωγής των δεδομένων από τα meta-data εκτελείτε κώδικας σε γλώσσα προγραμματισμού GO.

Μεθοδολογία για τη εξαγωγή δεδομένων



Εικόνα 25: Μεθοδολογία για την εξαγωγή των δεδομένων από τα βίντεο και αποστολή στον απομακρυσμένο διακομιστή σε διάγραμμα ροής

Η πιο πάνω διαδικασία εκτελείτε κάθε 5 λεπτά και είναι υπό την μορφή service στο περιβάλλον Linux.

Επεξήγηση:

Αναλυτικότερα όταν το service τεθεί σε λειτουργία το πρώτο πράγμα που κάνει είναι να κοιτάξει στο φάκελο που αναφέραμε πιο πάνω για να βρει τα καινούργια βίντεο που λήφθηκαν από τις συσκευές καταγραφής.

Όταν βρεθούν καινούργια βίντεο τότε για το κάθε ένα βίντεο γίνεται εξαγωγή τον δεδομένων μέσο μίας third party βιβλιοθήκης η οποία μας γραμμένης σε γλώσσα προγραμματισμού GO η οποία δημιουργεί csv files τα οποία περιέχουν όλη την χρήσιμη πληροφορία που

εμπεριέχεται σε μία καταγραφή βίντεο. Από την εξαγωγή των δεδομένων προκύπτουν 4 διαφορετικά csv files: accl, gps, gyro και temp

- Accl.csv: περιέχει δεδομένα που αφορούν το επιταχυνσιόμετρο της κάμερας
- Gps.csv: περιέχει δεδομένα που αφορούν τη χαρτογράφηση
- Gyro.csv: περιέχει δεδομένα που αφορούν το γυροσκόπιο της κάμερας
- Temp.csv: περιέχει δεδομένα που αφορούν την θερμοκρασία της κάμερας

	A	B	C	D	E	F	G	H	I
1	Milliseconds	Latitude	Longitude	Altitude	Speed	Speed3D	TS	GpsAccuracy	GpsFix
2	0	34.7008902	33.0479415	98.709	7.331	7.25	1.6534E+15	198	3
3	54	34.7008935	33.0479393	98.661	7.364	7.34	1.6534E+15	198	3
4	109	34.7008965	33.0479372	98.63	7.215	7.37	1.6534E+15	198	3
5	164	34.7008997	33.047935	98.593	7.266	7.22	1.6534E+15	198	3
6	219	34.7009027	33.0479328	98.544	7.128	7.27	1.6534E+15	198	3
7	274	34.7009059	33.0479306	98.469	7.124	7.13	1.6534E+15	198	3
8	329	34.700909	33.0479284	98.451	7.146	7.13	1.6534E+15	198	3
9	384	34.7009121	33.047926	98.432	7.178	7.15	1.6534E+15	198	3
10	439	34.7009153	33.0479235	98.401	7.254	7.18	1.6534E+15	198	3
11	494	34.7009181	33.0479209	98.418	7.239	7.26	1.6534E+15	198	3
12	549	34.7009209	33.0479182	98.42	7.266	7.25	1.6534E+15	198	3
13	605	34.7009241	33.0479158	98.417	7.309	7.27	1.6534E+15	198	3
14	659	34.7009272	33.0479133	98.416	7.371	7.32	1.6534E+15	198	3
15	714	34.7009303	33.0479107	98.377	7.388	7.38	1.6534E+15	198	3

Εικόνα 26: Παράδειγμα από το csv που εξάγεται με βάση τις συντεταγμένες

	A	B	C	D
1	Milliseconds	AcclX	AcclY	AcclZ
2	0	4.055023923	0.303827751	9.088516746
3	4.830917874	2.820574163	0.102870813	7.997607656
4	9.661835749	2.562200957	0.342105263	7.605263158
5	14.49275362	4.198564593	0.581339713	9.203349282
6	19.3236715	4.188995215	0.227272727	8.811004785
7	24.15458937	2.983253589	0.543062201	8.342105263
8	28.98550725	4.016746411	0.600478469	9.011961722
9	33.81642512	3.83492823	0.122009569	9.203349282
10	38.647343	2.744019139	0.045454545	8.495215311
11	43.47826087	3.586124402	0.418660287	8.906698565
12	48.30917874	4.476076555	0.236842105	9.193779904
13	53.14009662	3.490430622	0.141148325	8.428229665
14	57.97101449	4.179425837	0.523923445	9.327751196
15	62.80193237	5.002392344	0.409090909	10.22727273

Εικόνα 27: Παράδειγμα από το csv που εξάγεται με βάση τα δεδομένα του επιταχυνσιόμετρου

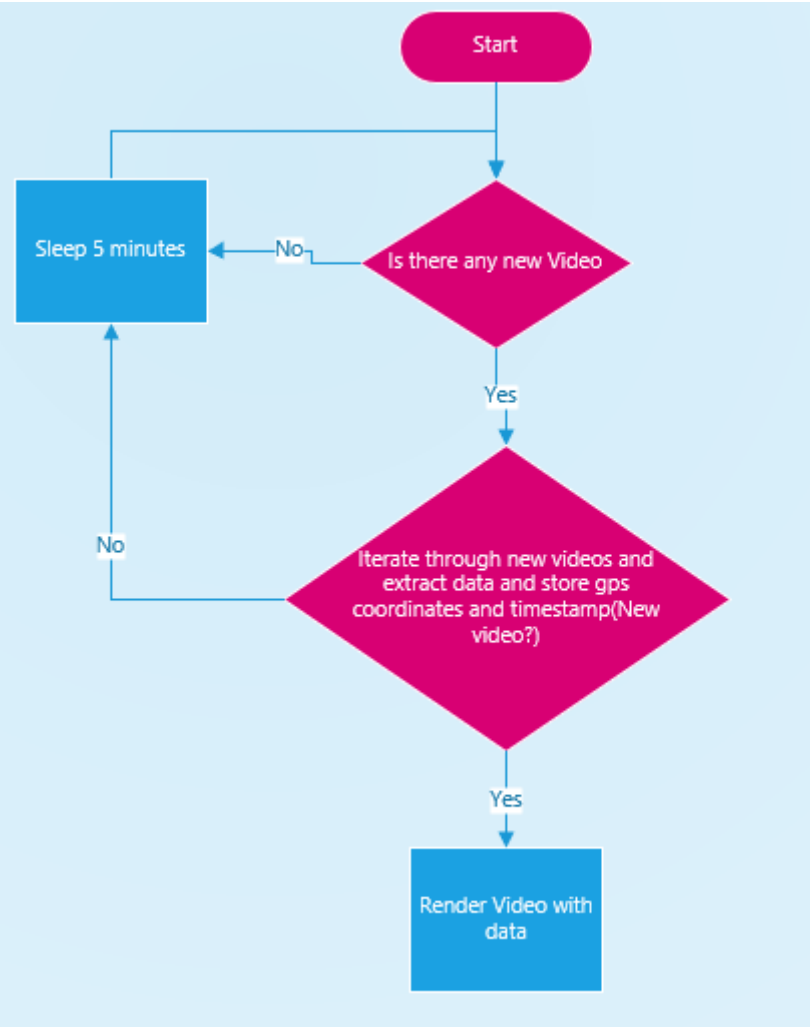
	A	B	C	D
1	Milliseconds	GyroX	GyroY	GyroZ
2	0	0.024494143	0.021831736	0.048988285
3	4.830917874	0.040468584	-0.015974441	0.033546326
4	9.661835749	0.04313099	-0.071352503	0.035676251
5	14.49275362	0.009052183	0.012247071	0.033546326
6	19.3236715	0.069755059	0.025026624	0.025026624
7	24.15458937	0.036208733	-0.030883919	0.035676251
8	28.98550725	0.004259851	0.024494143	0.041533546
9	33.81642512	0.062300319	0.028221512	0.03887114
10	38.647343	0.03887114	-0.064430245	0.033013845
11	43.47826087	0.023961661	-0.067625133	0.037806177
12	48.30917874	0.043663472	0.022896699	0.03514377
13	53.14009662	0.054313099	-0.048988285	0.0314164
14	57.97101449	0.034611289	-0.039403621	0.046325879
15	62.80193237	0.02342918	0.05857295	0.030351438

Εικόνα 28: Παράδειγμα από το csv που εξάγεται με βάση τα δεδομένα του γυροσκοπίου

Από τα πάρα πάνω csv files ο αλγόριθμος χρησιμοποιεί κάποιες από τις στήλες που εμπεριέχονται και ετοιμάζει ένα json file με αυτά που κρίθηκαν σημαντικά. Αυτό το json file ετοιμάζεται με βάση το τι περιμένει η βάση δεδομένων να αποθηκεύσει στο απομακρυσμένο διακομιστή.

Σε περίπτωση όπου το Raspberry Pi έχει πρόσβαση στο διαδίκτυο τότε η αποστολή των δεδομένων ξεκινά και μέσω ενός Rest Api αποθηκεύονται τα δεδομένα στον απομακρυσμένο διακομιστή, στην αντίθετη περίπτωση τα δεδομένα αποθηκεύονται σε αυτό υπό τη μορφή json file και όταν ανακτηθεί η σύνδεση τότε αποστέλλονται.

Μεθοδολογία για την επεξεργασία των βίντεο

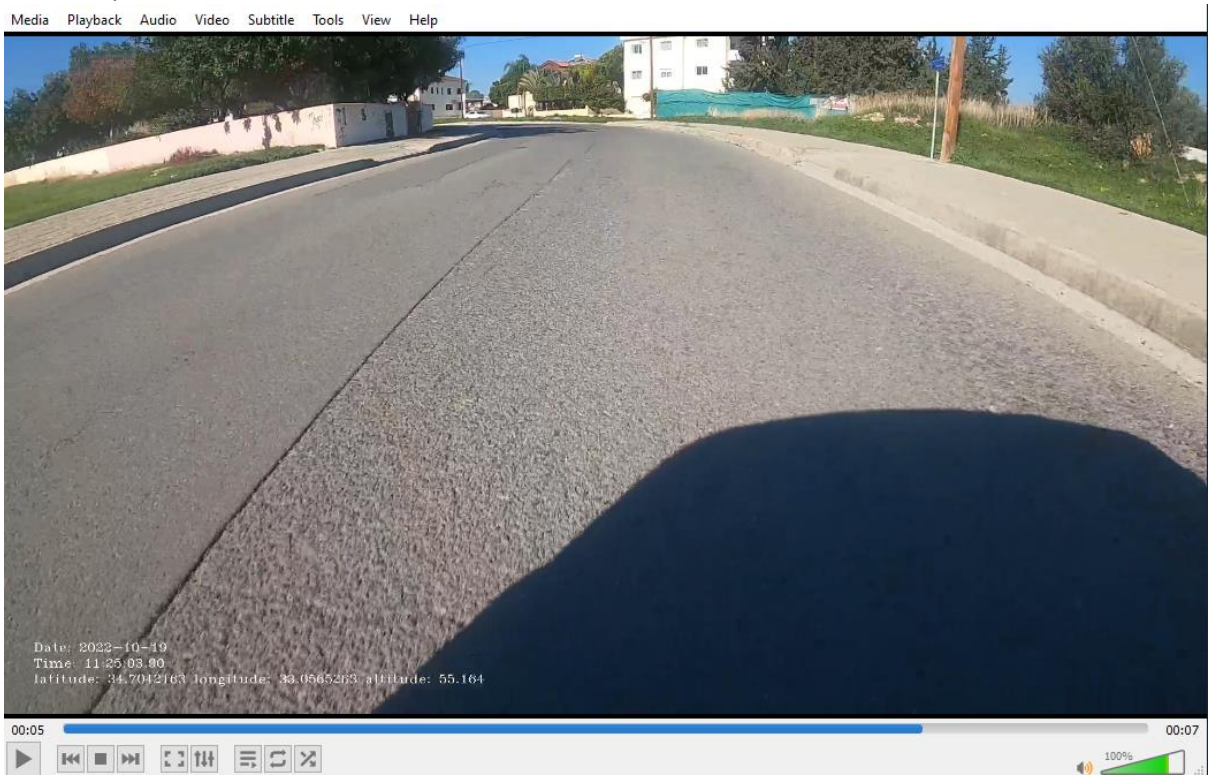


Εικόνα 29: Μεθοδολογία για την επεξεργασία των βίντεο και την επαναδημιουργία τους

Εκτός από την εξαγωγή των δεδομένων γίνεται και επεξεργασία των βίντεο έτσι ώστε ο χρήστης να μπορεί να καταλάβει το χρονικό σημείο που λήφθηκε ένα βίντεο αλλά και την τοποθεσία που λήφθηκε.

Όταν υπάρξουν καινούργια βίντεο στο φάκελο που αναφέραμε προηγουμένως τότε μέσω της διαδικασίας εξαγωγής δεδομένων αποθηκεύονται ξεχωριστά οι συντεταγμένες και η χρονική σήμανση. Ακολούθως μέσω της βιβλιοθήκης FFMPEG τα βίντεο χωρίζονται σε Frames και τοποθετείται σε αυτά χρονική σήμανση και συντεταγμένες. Όταν τελειώσει η όλη διαδικασία το βίντεο επαναδομείται με τα καινούργια Frames και αποθηκεύονται στη

συσκευή.



Εικόνα 30: Παράδειγμα βίντεο μετά την πρόσθεση συντεταγμένων και χρονικής σήμανσης

4.3.3 Βιβλιοθήκες που χρησιμοποιήθηκαν

Gopro-utils

Για την εξαγωγή των δεδομένων από τα βίντεο χρησιμοποιήθηκε μία βιβλιοθήκη η οποία εξυπηρετεί στην εξαγωγή δεδομένων από τα meta-data που πάρθηκαν συγκεκριμένα από βίντεο καταγραμμένα από κάμερες GoPro. Η βιβλιοθήκη λέγετε gopro-utils και μπορεί να βρεθεί μέσο GitHub.

Ουσιαστικά ένα βίντεο καταγραμμένο από GoPro κάμερα περιέχει Metadata σε μορφή GPMDF όταν είναι ενεργοποιημένο το GPS, με αλγόριθμους reverse-engineering η συγκεκριμένη βιβλιοθήκη μετατρέπει αυτού του είδους αρχεία στο γνωστό Format CSV αφού πρώτα το μετατρέψει σε BIN file.

```
ffmpeg -y -i GOPR0001.MP4 -codec copy -map 0 :m :handler_name:" GoPro MET" -f
rawvideo GOPR0001.bin

gpmd2csv -i GOPR0001.bin -o GOPR0001.csv
```

Χρησιμοποιώντας τις 2 πιο πάνω εντολές οι οποίες εκτελούνται μέσο python επιτυγχάνουμε την εξαγωγή τον δεδομένων σε CSV όπου χρησιμοποιούνται για την αποστολή στον VPS όπου αποθηκεύονται σε μια βάση δεδομένων PostgreSQL.

FFmpeg

Το ffmpeg είναι ένα open-source project που αποτελείται από μια σουίτα βιβλιοθηκών και προγραμμάτων για το χειρισμό βίντεο, ήχου και άλλων αρχείων και πολυμέσων. Στον πυρήνα του βρίσκεται το ίδιο το εργαλείο γραμμής εντολών ffmpeg, σχεδιασμένο για την επεξεργασία αρχείων βίντεο και ήχου.

Η βιβλιοθήκη ffmpeg χρησιμοποιείται στο πρόγραμμα μας για την επεξεργασία αλλά και για την εξαγωγή δεδομένων. Για την επεξεργασία των βίντεο η βιβλιοθήκη είναι υπεύθυνη για να εξάγει το κάθε ξεχωριστό frame και να προσθέτει σε αυτό συντεταγμένες και χρονική σήμανση. Με το τέλος της επεξεργασίας των frames το βίντεο ξανακτίζετε.
Επίσης με το ffmpeg γίνεται η εξαγωγή δεδομένων αναλύοντας τα Metadata που περιέχονται στο βίντεο.

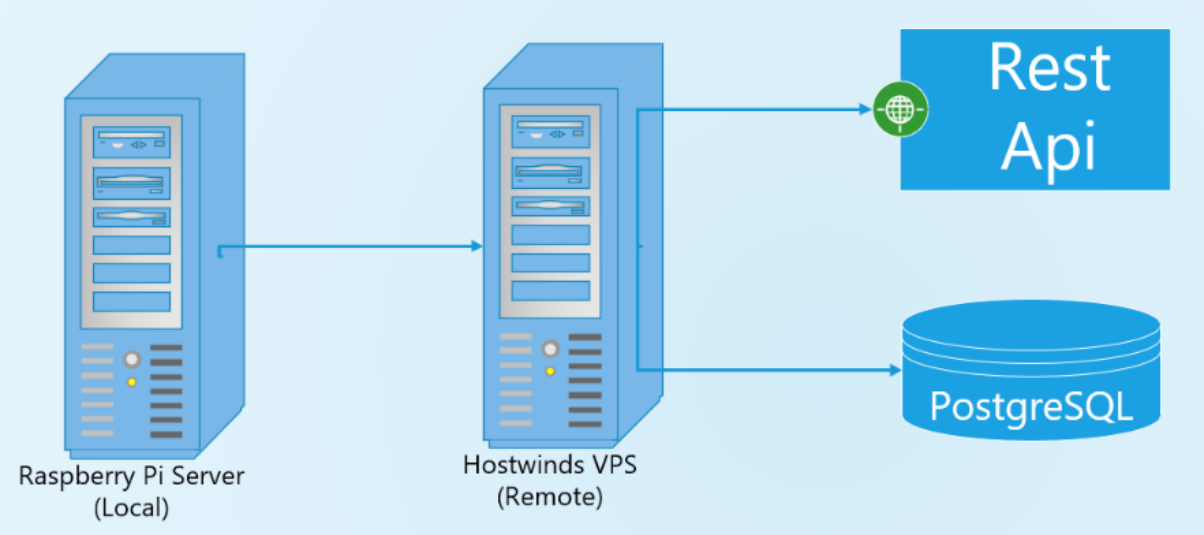
```
#Command to send via the command prompt which specifies the pipe parameters
command = ['ffmpeg',
        '-y', # (optional) overwrite output file if it exists
        '-f', 'rawvideo', #Input is raw video
        '-vcodec', 'rawvideo',
        '-pix_fmt', 'bgr24', #Raw video format
        '-s', str(int(width)) + 'x' + str(int(height)), # size of one frame
        '-r', str(FPS), # frames per second
        '-i', '-', # The input comes from a pipe
         '-i' ,videoWithDir,
        '-vcodec', 'h264',
        '-b:v', '10M', #Sets a maximum bit rate
        Output_name]
```

Εικόνα 31: Οι παράμετροι που παίρνει ως όρισμα η βιβλιοθήκη FFmpeg για την επεξεργασία των βίντεο

5 Μεθοδολογία και Υλοποίηση VPS

Σε αυτό το κεφάλαιο θα επεξηγηθεί η όλη διαδικασία υλοποίησης και εφαρμογής διαδικασιών σε έναν εξωτερικό διακομιστή ο οποίος είναι υπεύθυνος για την λήψη των δεδομένων που συλλέχτηκαν από τα βίντεο, όπως επίσης και για την αποθήκευση τους σε μία βάση δεδομένων PostgreSQL.

Ουσιαστικά στον Virtual Private Server έχει εγκατασταθεί και δημιουργηθεί μία βάση δεδομένων PostgreSQL η οποία τροφοδοτείται από ένα Restful Api που επίσης είναι εγκατεστημένο στον Virtual Private Server.



Εικόνα 32: Απεικονίζεται η διαδικασία που ακολουθούν τα δεδομένα φεύγοντας από τον τοπικό διακομιστή

5.1 Τι είναι το VPS και τι διαφορά έχει με το Cloud Server

Ένα VPS, ή Virtual Private Server, είναι μια μορφή multi-tenant cloud hosting στην οποία οι εικονικοί πόροι διακομιστή διατίθενται σε έναν τελικό χρήστη μέσω του Διαδικτύου μέσω ενός cloud ή ενός παρόχου υπηρεσιών.

Κάθε VPS είναι εγκατεστημένο σε ένα φυσικό μηχάνημα, το οποίο λειτουργεί από το cloud ή τον πάροχο υπηρεσιών, που φιλοξενεί πολλαπλά VPS. Ωστόσο, ενώ τα VPS μοιράζονται έναν hypervisor και ένα υποκείμενο υλικό, κάθε VPS εκτελεί το δικό του λειτουργικό σύστημα (OS) και εφαρμογές και διατηρεί το δικό του τμήμα των πόρων του μηχανήματος (μνήμη, CPU, κ.λπ.).

Ένα VPS προσφέρει επίπεδα απόδοσης, ευελιξίας και ελέγχου κάπου ανάμεσα σε αυτά που προσφέρονται από την κοινή φιλοξενία πολλαπλών ενοικιαστών και την αποκλειστική φιλοξενία ενός μισθωτή. Αν και μπορεί να φαίνεται αδιανόητο ότι η συμφωνία VPS

πολλαπλών ενοικιαστών θα ονομαζόταν «ιδιωτική» ειδικά όταν υπάρχουν διαθέσιμες επιλογές μεμονωμένου μισθωτή ο όρος «VPS» χρησιμοποιείται πιο συχνά από τους παραδοσιακούς παρόχους υπηρεσιών για να τη διακρίνει από την κοινή φιλοξενία, ένα μοντέλο φιλοξενίας όπου όλοι οι πόροι υλικού και λογισμικού μιας φυσικής μηχανής μοιράζονται εξίσου σε πολλούς χρήστες.

Οι διαφορές μεταξύ των παρόχων μπορεί να είναι πραγματικά σημαντικές. Για τους παραδοσιακούς παρόχους υπηρεσιών, ένα VPS αντιπροσωπεύει μια καλή ισορροπία κόστους, ευελιξίας, επεκτασιμότητας και ελέγχου μεταξύ κοινόχρηστης και αποκλειστικής φιλοξενίας και το καθιστά κατάλληλο για ηλεκτρονικό εμπόριο, εφαρμογές που έχουν μέτρια ή έντονη επισκεψιμότητα, διακομιστές email, CRM κ.λπ.

Σε μια ρύθμιση VPS, ο πάροχος υπηρεσιών εγκαθιστά πολλές εικονικές μηχανές σε έναν μόνο φυσικό διακομιστή και στη συνέχεια τις νοικιάζει σε μεμονωμένους πελάτες. Σε ένα πρόγραμμα υπηρεσιών cloud, υπάρχει ένας λογαριασμό που μπορεί να αξιοποιήσει μια δεξαμενή πόρων που παρέχονται από ένα σύμπλεγμα διασυνδεδεμένων φυσικών διακομιστών.

5.2 PostgreSQL

Αρχικά εγκαταστάθηκε και δημιουργήθηκε η βάση δεδομένων στον VPS και στη συνέχεια δημιουργήθηκε ένα Πίνακας με το όνομα vibration που περιέχει τις εξής παραμέτρους :

- Id: indexing
- Result_time: χρόνος δειγματοληψίας καταγραφής
- Device_id: ταυτότητα της συσκευής που κατέγραψε το βίντεο
- Latitude: γεωγραφικό πλάτος
- Longitude: γεωγραφικό μήκος
- Altitude: γεωγραφικό ύψος
- Direction: κατεύθυνση κίνησης
- Ax: επιτάχυνση στον άξονα X
- Ay: επιτάχυνση στον άξονα Y
- Az: επιτάχυνση στον άξονα Z
- Wx: γωνιακή επιτάχυνση στον άξονα X
- Wy: γωνιακή επιτάχυνση στον άξονα Y
- Wz: γωνιακή επιτάχυνση στον άξονα Z
- Roll: περιστροφή γύρω από τον άξονα εμπρός-πίσω
- Pitch: περιστροφή γύρω από τον άξονα από πλευρά σε πλευρά
- Yaw: περιστροφή γύρω από τον κατακόρυφο άξονα

Όλες αυτές οι παράμετροι επιλέχθηκαν κυρίως από τα δεδομένα που μπορούμε να εξάγουμε από την επεξεργασία των βίντεο.

Κάθε ένα δευτερόλεπτο καταγραφής βίντεο μπορεί να συλλεγεί πληροφορία έως και 350 γραμμών στη βάση δεδομένων. Άρα με τα βίντεο που καταγράφουμε στην συσκευή συλλέγουμε περίπου 2500 γραμμές ανά βίντεο.

Query Editor Query History

```
1  SELECT * FROM public.vibration
2  ORDER BY id DESC LIMIT 1000
3
```

Data Output Explain Messages Notifications

	id [PK] bigint	result_time timestamp without time zone	device_id text	latitude double precision	longitude double precision	altitude double precision	direction real	ax real	ay real	az real
1	2293191	2022-12-15 15:27:16.997	device_01	35.11415	33.3359642	201.416	0	-9.82254	-0.470024	-0.263789
2	2293190	2022-12-15 15:27:16.992	device_01	35.11415	33.3359642	201.416	0	-9.81295	-0.486811	-0.280576
3	2293189	2022-12-15 15:27:16.987	device_01	35.11415	33.3359642	201.416	0	-9.81535	-0.486811	-0.258993
4	2293188	2022-12-15 15:27:16.987	device_01	35.11415	33.3359642	201.416	0	-9.81535	-0.486811	-0.258993
5	2293187	2022-12-15 15:27:16.982	device_01	35.11415	33.3359642	201.416	0	-9.82014	-0.491607	-0.275779
6	2293186	2022-12-15 15:27:16.977	device_01	35.11415	33.3359642	201.416	0	-9.82254	-0.484412	-0.273381
7	2293185	2022-12-15 15:27:16.972	device_01	35.11415	33.3359642	201.416	0	-9.83213	-0.470024	-0.28777
8	2293184	2022-12-15 15:27:16.967	device_01	35.11415	33.3359642	201.416	0	-9.82734	-0.482014	-0.268585

Εικόνα 33: Απεικονίζει την βάση δεδομένων και το πίνακα με τις μισές παραμέτρους

Query Editor Query History

```
1  SELECT * FROM public.vibration
2  ORDER BY id DESC LIMIT 1000
3
```

Data Output Explain Messages Notifications

latitude double precision	longitude double precision	altitude double precision	direction real	ax real	ay real	az real	wx real	wy real	wz real	roll real	pitch real	yaw real
35.11415	33.3359642	201.416	0	-9.82254	-0.470024	-0.263789	0	0	0	0	0	-0.00106496
35.11415	33.3359642	201.416	0	-9.81295	-0.486811	-0.280576	0	0	0	0.00212993	-0.00106496	-0.00212993
35.11415	33.3359642	201.416	0	-9.81535	-0.486811	-0.258993	0	0	0	0.00212993	-0.00106496	-0.00106496
35.11415	33.3359642	201.416	0	-9.81535	-0.486811	-0.258993	0	0	0	0.00212993	-0.00106496	-0.00106496
35.11415	33.3359642	201.416	0	-9.82014	-0.491607	-0.275779	0	0	0	0	0.00319489	-0.00106496
35.11415	33.3359642	201.416	0	-9.82254	-0.484412	-0.273381	0	0	0	0.00212993	0.00212993	-0.00425985
35.11415	33.3359642	201.416	0	-9.83213	-0.470024	-0.28777	0	0	0	0.00212993	0.00106496	-0.00212993
35.11415	33.3359642	201.416	0	-9.82734	-0.482014	-0.268585	0	0	0	0.00106496	0	0

Εικόνα 34: Απεικονίζει την βάση δεδομένων και το πίνακα με τις μισές παραμέτρους

5.3 Restful Api

Η αρχιτεκτονική του middleware βασίζεται σε μια λίστα από services πελάτη-διακομιστή, με δυνατότητα προσωρινής αποθήκευσης (cache) που παρέχει τη μέγιστη απόδοση και ευρωστία ολόκληρου του συστήματος. Χρησιμοποιεί RESTful Web-Services πάνω από το Σύστημα Βάσης Δεδομένων για την έκθεση πόρων της βάσης δεδομένων μέσω του δικτύου έτσι ώστε να επιτρέπει σε άλλες εφαρμογές (π.χ. Διεπαφή χρήστη Ιστού, Εφαρμογή για φορητές συσκευές κ.λπ.) να καταναλώνουν αυτούς τους πόρους. Για την υλοποίηση αυτού του πεδίου, το πλαίσιο Python Flask έχει χρησιμοποιηθεί πάνω από τη βάση δεδομένων PostgreSQL.

Με το rest Api που δημιουργήθηκε εξυπηρετούνται ταυτόχρονα πολλές οντότητες μέσα στο σύστημα.

5.3.1 CRUD Operations

Post Request

Μετά την επεξεργασία των βίντεο και τα δεδομένα που προκύπτουν από την διαδικασία δημιουργείτε ένα json file το οποίο αποστέλλεται μέσω ενός Post Request, χρησιμοποιώντας τα απαραίτητα διαπιστευτήρια για έλεγχο ταυτότητας και πρόσβασης όπου ελέγχονται και προετοιμάζονται για να αποθηκευτούν στη βάση δεδομένων PostgreSQL. Στη συνέχεια μέσω διεπαφής τα γίνεται η σύνδεση μεταξύ του Api και της βάσης δεδομένων και γίνεται η αποθήκευση.

Η όλη διαδικασία που επεξηγήθηκε θεωρείτε ως μέλος της εξυπηρέτησης της συσκευής και γενικότερα του συστήματος αφού μέσω αυτού του κόμβου αποθηκεύονται τα δεδομένα όπου μαζί με τα βίντεο είναι τα μόνα αποδεικτικά στοιχεία που δίνουν νόημα στην δημιουργία και εκτέλεση της τρέχουσας εργασίας.

Get Request

Αφού τα δεδομένα αποθηκευτούν κατάλληλα στη βάση δεδομένων τότε μπορούν να ανακτηθούν μέσω του Get Request. Αυτό το request δημιουργήθηκε με απώτερο σκοπό την εξυπηρέτηση του χρήστη. Ο χρήστης μπορεί να ανακτήσει τα δεδομένα είτε μέσω εργαλείων όπως είναι τα Postman, Advanced REST Client, Apigee είτε να τα ανακτήσει προγραμματιστικά για περαιτέρω επεξεργασία και παρουσίαση προς το κοινό. Μέσω του Get Request τα δεδομένα μπορούν να παρουσιαστούν είτε από ιστοσελίδα είτε από εφαρμογή τηλεφώνου ή ακόμα και να γίνει εφαρμογή υπολογιστή.

Delete Request

Σε περίπτωση που ο χρήστης κρίνει ότι ένα βίντεο το οποίο λήφθηκε από την συσκευή περιείχε εσφαλμένη πληροφορία (πχ false alarm recording) τότε μπορεί να διαγράψει ένα βίντεο. Διαγράφοντας ένα βίντεο αυτόματα παραμένει πληροφορία στην βάση δεδομένων η οποία δεν ταυτίζεται με κανένα βίντεο. Έτσι θεωρήθηκε σωστό να δημιουργηθεί ένα Delete Request στο οποίο ο χρήστης να μπορεί να δώσει τον χρόνο που κατέγραφε το βίντεο

δίνοντας τον χρόνο εκκίνησης και τερματισμού έτσι ώστε να μπορεί να διαγράψει όλα τα δεδομένα που βρίσκονται στη βάση που είναι πλέον άχρηστά. Ξεκάθαρα αυτό το request χρησιμοποιείτε για καθήκοντα που εξυπηρετούν τον χρήστη ή οποιανδήποτε εφαρμογή που θα δημιουργηθεί στο μέλλον για αυτόματο καθάρισμα της βάσης.

Στη συνέχεια περιγράφονται όλες οι λειτουργίες του Web Service API που απαιτούνται.

Parameter	Value	Description
<SERVER BASE PATH>	http://142.11.210.23:5000/	API endpoint
<SERVICE VERSION >	/v0/	Current API version
<SERVICE IDENTIFIER>	vibration/	All available functions described in the following

Πίνακας 6: Rest Api Uri

Request Authorization (Basic Auth)

Parameter	Value	Description
Username	DeviceAdmin	Username needed per request
Password	DeviceV0Pass0	Password needed per request

Πίνακας 7: Rest Api Authentication Credentials

Vibration Services

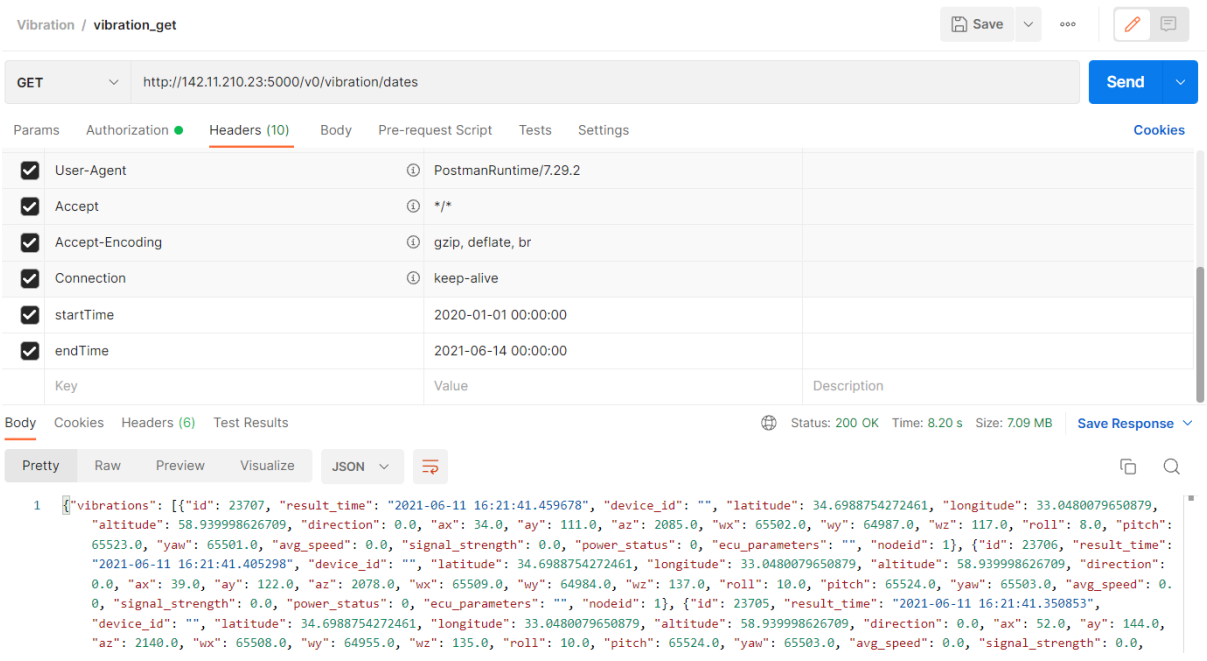
1.Get entries from vibration Πίνακας

IDENTIFICATION		
Resource		**/vibration/<generalInput >**
Type		**GET**
Reply		**Application/json**
PARAMETERS		
Name	**Type**	**Reasoning**
generalInput	String Required	Insert number of entries (i.e., “2”) or “All” (or “all” or “ALL”) to retrieve all entries. You can also use “dates” and set starting date/timestamp and final date/timestamp to fetch entries for the given period.

		(i.e., startTime = yyyy-mm-dd hh:mm:ss (i.e 2020-01-01 00:00:00) endTime = yyyy-mm-dd hh:mm:ss (i.e 2021-06-14 00:00:00)
RESPONSE CODES		
HTTP/1.1 Status	**Description**	
200	OK	
405	Not AccepΠίνακας entry. Number or All must be inserted	
400	Bad Request	
500	Internal server error	

Πίνακας 8: Get Request Parameters

Example with Dates in Postman:



Εικόνα 35: Get Request example in Postman

GET request with date range -> URL : http://142.11.210.23:5000/v0/vibration/dates

Headers: startTime = yyyy-mm-dd hh:mm:ss (i.e 2020-01-01 00:00:00)
endTime = yyyy-mm-dd hh:mm:ss (i.e 2021-06-14 00:00:00)

Authorization Basic: Username = DeviceAdmin
Password =DeviceV0Pass0

Note: Application/Json Output:


```
{
  "vibrations": [
    {
      "id": 711258,
      "result_time": "2022-06-30 15:54:17.065000",
      "device_id": "device_01",
      "latitude": 34.6985431,
      "longitude": 33.0476814,
      "altitude": 75.118,
      "direction": 0.0,
      "ax": 0.284689,
      "ay": 0.188995,
      "az": 10.5048,
      "wx": 0.0,
      "wy": 0.0,
      "wz": 0.0,
      "roll": -0.00319489,
      "pitch": -0.00212993,
      "yaw": -0.00425985,
      "avg_speed": 0.056,
      "signal_strength": 0.0,
      "power_status": 0,
      "ecu_parameters": "",
      "nodeid": 1
    }
,…}
```


2.Post an entry in vibration Πίνακας

IDENTIFICATION		
Resource		**/vibration**
Type		**POST**
Accept		**Application/json**
PARAMETERS		
Name	**Type**	**Reasoning**
vibration	$Vibration Schema	JSON object to be posted containing the vibration record details
RESPONSE CODES		

HTTP/1.1 Status	Description	
200	OK	
400	Bad Request	
500	Internal server error	

Πίνακας 9: Post Request Parameters

Note: Input Application/json:


```
{
        "result_time": "2020-05-25 09:48:09.2" ,
        "device_id":"device_01",
        "latitude":1111,
        "longitude": 2222,
        "altitude":100,
        "direction":213,
        "ax":"[3,3,3,3]",
        "ay":"[3,3,3,3]",
        "az":"[3,3,3,3]",
        "wx":"[3,3,3,3]",
        "wy":"[3,3,3,3]",
        "wz":"[3,3,3,3]",
        "roll":"[3,3,3,3]",
        "pitch":"[3,3,3,3]",
        "yaw":"[3,3,3,3]",
        "avg_speed":30,
        "signal_strength":23,
        "power_status":1,
        "ecu_parameters":"first parameter"
        }
```


3.Delete entries from vibration Πίνακας

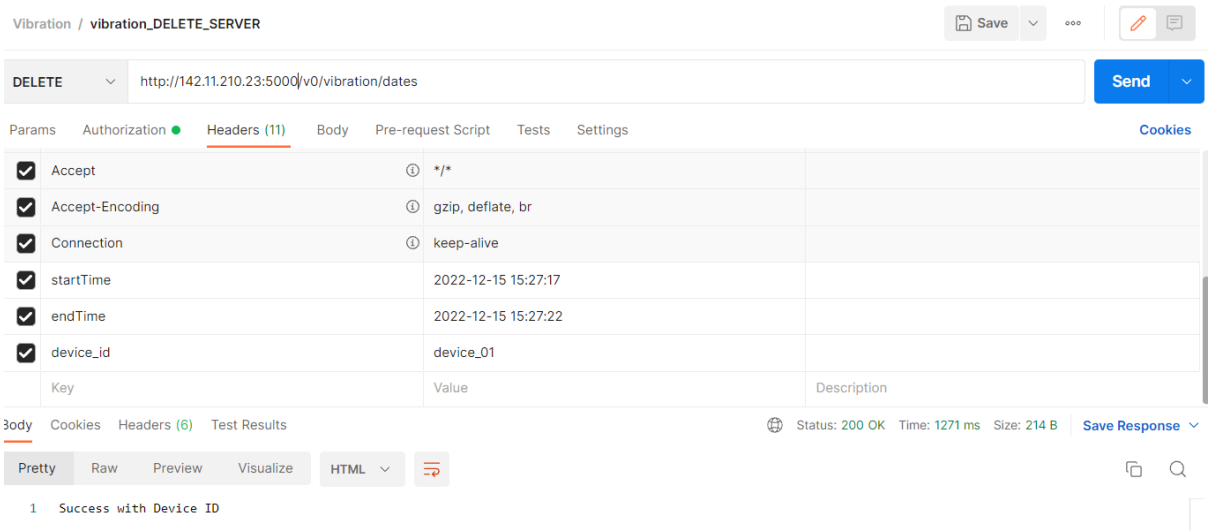
IDENTIFICATION		
Resource		**/vibration/dates**
Type		**Delete**
Reply		**Application/json**
PARAMETERS		
Headers	**Type**	**Reasoning**
startTime endTime device_id	String Required String Required String Required	Using “dates” you have to set start time and end time for the proposed deleted time frame. (i.e., startTime = yyyy-mm-dd hh:mm:ss (i.e 2020-01-01 00:00:00)

		endTime = yyyy-mm-dd hh:mm:ss (i.e 2021-06-14 00:00:00) You have to choose whether you want to include device_id in headers or not. Without using it, the deletion is only specified from timeframe
RESPONSE CODES		
HTTP/1.1 Status	**Description**	
200	Success with No Device ID, Success with Device ID	
405	Not AccepΠίνακας entry. Number or All must be inserted	
400	Bad Request	
500	Internal server error	

Πίνακας 10: Delete Request Parameters

Authorization Basic: Username = DeviceAdmin
Password = DeviceV0Pass0

Example of Deletion in Postman:



Πίνακας 11: Example of Delete Request in Postman

6 Μεθοδολογία και Επεξεργασία βίντεο

Σε αυτό το κεφάλαιο θα επεξηγηθεί η ανάλυση που εφαρμόστηκε στα βίντεο έτσι ώστε να αναγνωρίζονται οι φθορές στο οδόστρωμα.

Για την επεξεργασία των βίντεο πρέπει τα βίντεο να ανακτηθούν μόλις αποθηκευτούν στην συσκευή καταγραφής χειροκίνητα. Δηλαδή πριν η συσκευή συνδεθεί στο τοπικό δίκτυο έτσι ώστε να μεταφερθούν τα βίντεο και ακολούθως να διαγραφούν από τη συσκευή.

Χρησιμοποιώντας την γλώσσα προγραμματισμού Python έγινε υλοποίηση κώδικα ο οποίος προσθέτει φίλτρα έτσι ώστε να μπορεί να ξεχωρίσει τις φθορές. Ακολούθως επαναδημιουργεί το βίντεο με σημαδεμένες τις φθορές προσθέτοντας ένα τετράγωνο για να μπορεί να τις κάνει ευδιάκριτες.

6.1 Προσωπικά δεδομένα

Κατά την διαδικασία καταγραφής των βίντεο παρουσιάστηκε ζήτημα όταν σε ένα βίντεο υπήρχε ευαίσθητη πληροφορία. Κατά την δειγματοληψία, σε αρκετά βίντεο εμπεριέχονταν αυτοκίνητα στα οποία ήταν ευδιάκριτοι οι αριθμοί στις πινακίδες ή υπήρχαν πεζοί οι οποίοι διασταύρωναν το δρόμο πίσω από το αυτοκίνητο.

Στην πραγματικότητα για να θεωρηθεί αδίκημα η καταγραφή βίντεο στο δρόμο πρέπει ένα βίντεο να δημοσιοποιηθεί και να υπάρχουν προσωπικά δεδομένα που ταυτίζονται με άτομο. Σε αυτή την περίπτωση η άμεσα εμπλεκόμενοι μπορούν να κινηθούν νομικά ενάντια σε αυτόν που δημοσιοποίησε το βίντεο και όχι σε αυτόν που το κατέγραψε.

Για αποφυγή οποιονδήποτε πιθανόν παραπτωμάτων που επεξηγήθηκαν πιο πάνω, κατά την επεξεργασία των βίντεο αφαιρέθηκε το ένα τρίτο από τον κάθετο άξονα από την εικόνα των βίντεο, έτσι ώστε να παραμένει για επεξεργασία μόνο ο ασφαλτόδρομος. Έτσι οι διαστάσεις των βίντεο από 1920 x 1080 έγιναν 1920 x 720 pixels.

Εικόνα 36: Frame από βίντεο

Εικόνα 37: Frame από βίντεο αφού πρώτα αποκόπηκε το ένα τρίτο από το πάνω μέρος

6.2 Ορισμοί, βιβλιοθήκες κατά την επεξεργασία των βίντεο

- **OpenCV**

Το OpenCV (Open Source Computer Vision Library) είναι μια βιβλιοθήκη λογισμικού ανοιχτού κώδικα όρασης υπολογιστή και μηχανικής εκμάθησης. Το OpenCV κατασκευάστηκε για να παρέχει μια κοινή υποδομή για εφαρμογές υπολογιστικής όρασης και να επιταχύνει τη χρήση της αντίληψης μηχανών στα εμπορικά προϊόντα.

Η βιβλιοθήκη διαθέτει περισσότερους από 2500 βελτιστοποιημένους αλγόριθμους, οι οποίοι περιλαμβάνουν ένα ολοκληρωμένο σύνολο κλασικών και υπερσύγχρονων αλγορίθμων υπολογιστικής όρασης και μηχανικής μάθησης. Αυτοί οι αλγόριθμοι μπορούν να χρησιμοποιηθούν για την ανίχνευση και αναγνώριση προσώπων, την αναγνώριση αντικειμένων, την ταξινόμηση των ανθρώπινων ενεργειών σε βίντεο, την παρακολούθηση κινήσεων της κάμερας, την παρακολούθηση κινούμενων αντικειμένων, την εξαγωγή τρισδιάστατων μοντέλων αντικειμένων, τη συρραφή εικόνων για παραγωγή υψηλής ανάλυσης εικόνα μιας ολόκληρης σκηνής, εύρεση παρόμοιων εικόνων από βάση δεδομένων εικόνων κ.λπ. Το OpenCV έχει περισσότερους από 47 χιλιάδες χρήστες κοινότητα και ο εκτιμώμενος αριθμός λήψεων υπερβαίνει τα 18 εκατομμύρια. Η βιβλιοθήκη χρησιμοποιείται εκτενώς σε εταιρείες, ερευνητικές ομάδες και από κυβερνητικούς φορείς.

- **Μορφολογικές Λειτουργίες**

Είναι ένα σύνολο λειτουργιών που επεξεργάζονται εικόνες με βάση σχήματα, αντικείμενα. Οι μορφολογικές λειτουργίες εφαρμόζουν ένα δομικό στοιχείο σε μια εικόνα εισόδου και δημιουργούν μια εικόνα εξόδου.

Οι πιο βασικές μορφολογικές επεμβάσεις είναι: Διάβρωση και Διαστολή και έχουν ένα ευρύ φάσμα χρήσεων:

- Αφαίρεση θορύβου
- Απομόνωση μεμονωμένων στοιχείων και ένωση ανόμοιων στοιχείων σε μια εικόνα.
- Εύρεση εξογκωμάτων ή οπών έντασης σε μια εικόνα

Θα εξηγήσουμε εν συντομία τη διαστολή και τη διάβρωση, χρησιμοποιώντας την παρακάτω εικόνα ως παράδειγμα:

Εικόνα 38: Εικόνα χωρίς οποιαδήποτε επεξεργασία

- **Διαστολή (Dilation)**

Αυτή η λειτουργία αποτελείται από τη συνένωση μιας εικόνας A με κάποιο πυρήνα B (μάσκα), που μπορεί να έχει οποιοδήποτε σχήμα ή μέγεθος, συνήθως ένα τετράγωνο ή κύκλο.
Η μάσκα B έχει ένα καθορισμένο σημείο αγκύρωσης, που συνήθως είναι το κέντρο του πυρήνα.

Καθώς η μάσκα Β σαρώνεται πάνω από την εικόνα, υπολογίζουμε τη μέγιστη τιμή pixel που επικαλύπτεται από το Β και αντικαθιστούμε το εικονοστοιχείο εικόνας στη θέση σημείου αγκύρωσης με αυτήν τη μέγιστη τιμή. Όπως μπορείτε να συμπεράνετε, αυτή η λειτουργία μεγιστοποίησης προκαλεί την «μεγέθυνση» των φωτεινών περιοχών μέσα σε μια εικόνα (επομένως το όνομα διαστολή).

Η διαστολή προσφέρει:
- Αυξάνει την περιοχή του αντικειμένου
- Χρησιμοποιείται για να τονίσει χαρακτηριστικά

Η λειτουργία διαστολής είναι:

$$dst(x,y)=max(x',y'):element(x',y')\neq 0 src(x+x',y+y')$$

Εξίσωση 1: Εξίσωση υπολογισμού διαστολής

Εφαρμόζοντας διαστολή μπορούμε να πάρουμε το πιο κάτω αποτέλεσμα:

Εικόνα 39: Εικόνα μετά από επεξεργασία διαστολής

- **Διάβρωση (Erosion)**

Υπολογίζει ένα τοπικό ελάχιστο στην περιοχή της δεδομένης μάσκας.
Καθώς ο πυρήνας Β σαρώνεται πάνω από την εικόνα, υπολογίζουμε την ελάχιστη τιμή pixel που επικαλύπτεται από το Β και αντικαθιστούμε το εικονοστοιχείο εικόνας κάτω από το σημείο αγκύρωσης με αυτήν την ελάχιστη τιμή.

Η διάβρωση προσφέρει:
- Διαβρώνει τα όρια του αντικειμένου του πρώτου πλάνου.
- Χρησιμοποιείται για να μειώσει τα χαρακτηριστικά μιας εικόνας.

Η λειτουργία διάβρωσης είναι:

$$dst(x,y)=min(x',y'):element(x',y')\neq 0 src(x+x',y+y')$$

Εξίσωση 2:Εξίσωση υπολογισμού διάβρωσης

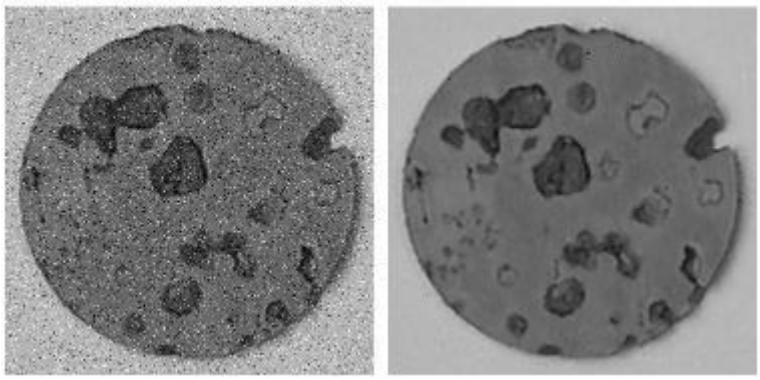
Μπορείτε να δείτε στο παρακάτω αποτέλεσμα ότι οι φωτεινές περιοχές της εικόνας γίνονται πιο λεπτές, ενώ οι σκοτεινές ζώνες γίνονται μεγαλύτερες.

Εικόνα 40: Εικόνα μετά από επεξεργασία διάβρωσης

- **Median filter**

Το διάμεσο φίλτρο είναι η τεχνική φιλτραρίσματος που χρησιμοποιείται για την αφαίρεση θορύβου από εικόνες και σήματα. Το διάμεσο φίλτρο είναι πολύ σημαντικό στον τομέα της επεξεργασίας εικόνας, καθώς είναι γνωστό για τη διατήρηση των άκρων κατά την αφαίρεση θορύβου.

Το διάμεσο φίλτρο είναι ένα από τα γνωστά order-statistic filters λόγω της καλής του απόδοσης για ορισμένους συγκεκριμένους τύπους θορύβου όπως οι θόρυβοι "Gaussian", "random" και "salt and pepper".



Εικόνα 41: Καθαρισμός εικόνας με median filter

- **Mean Max Normalization**

Η κανονικοποίηση Min-Max (συνήθως ονομάζεται κλιμάκωση χαρακτηριστικών) εκτελεί έναν γραμμικό μετασχηματισμό στα αρχικά δεδομένα (pixel). Αυτή η τεχνική λαμβάνει όλα τα δεδομένα κλίμακας στην περιοχή (0, 255). Ο τύπος για να επιτευχθεί αυτό είναι ο ακόλουθος:

$$x_{scaled} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

Εξίσωση 3: Εξίσωση υπολογισμού Mean-Max Normalization

Η κανονικοποίηση Min-Max διατηρεί τις σχέσεις μεταξύ των αρχικών τιμών δεδομένων.

- **Gaussian Filter**

Το φίλτρο Gaussian είναι ένα χαμηλοπερατό φίλτρο που χρησιμοποιείται για τη μείωση του θορύβου (στοιχεία υψηλής συχνότητας) και το θόλωμα των περιοχών μιας εικόνας. Το φίλτρο υλοποιείται ως συμμετρικός πυρήνας περιττού μεγέθους που διέρχεται από κάθε εικονοστοιχείο της Περιοχής Ενδιαφέροντος για να ληφθεί το επιθυμητό αποτέλεσμα. Τα pixel στο φίλτρο Gaussian προς το κέντρο του πυρήνα έχουν μεγαλύτερη βαρύτητα προς την τελική τιμή από την περιφέρεια.

Στη διαδικασία χρήσης του φίλτρου Gauss σε μια εικόνα, πρώτα ορίζουμε το μέγεθος του πυρήνα/μήτρας που θα χρησιμοποιηθεί για την κατάργηση της εικόνας.

Επίσης οι πυρήνες είναι συμμετρικοί και επομένως έχουν τον ίδιο αριθμό σειρών και στηλών. Οι τιμές μέσα στον πυρήνα υπολογίζονται από τη συνάρτηση Gaussian, η οποία είναι η εξής:

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

Εξίσωση 4: Εξίσωση του φίλτρου Gaussian

Εικόνα 42: Εφαρμογή του Gaussian filter

- **Canny Edge Detection**

Το Canny Edge είναι ένας αλγόριθμος πολλαπλών βημάτων για τον εντοπισμό των άκρων για οποιαδήποτε εικόνα εισόδου. Περιλαμβάνει τα παρακάτω βήματα που πρέπει να ακολουθούνται κατά την ανίχνευση άκρων μιας εικόνας:

- Noise reduction
- Gradient calculation
- Non-maximum suppression
- Double threshold
- Edge Tracking by Hysteresis

Εικόνα 43: Εφαρμογή του Canny Edge Detector

6.3 Μεθοδολογία και Υλοποίηση

Βήμα 1

Αφού αποκτηθούν τα βίντεο από την συσκευή καταγραφής περνούν όλα από ένα script το οποίο γράφτηκε σε Python και χρησιμοποιώντας την βιβλιοθήκη OpenCV κάθε βίντεο περνά από επεξεργασία έτσι ώστε να μπορούμε να συλλέξουμε όλα τα frames που απαρτίζουν το εκάστοτε βίντεο. Όταν έχουμε πλέον πρόσβαση σε κάθε frame τότε υπολογίζουμε το ύψος

στο frame και αποκόπτουμε από αυτό το ένα τρίτο από πάνω προς τα κάτω (για λόγους που αφορούν προσωπικά δεδομένα).

Ακολούθως για κάθε βίντεο δημιουργούμε ένα φάκελο με το όνομα του βίντεο όπου φυλάγονται όλα τα αποψιλωμένα frames για περαιτέρω επεξεργασία.

Βήμα 2

Αφού έγινε ο διαχωρισμός σε frame στα βίντεο, έχουμε πλέον όλα τα ανεξάρτητα frame σε φάκελο τα οποία τακτοποιούνται με βάση έναν αριθμό index που αντικατοπτρίζεται στο όνομα του κάθε frame(έγινε στο βήμα 1) στον φάκελο. Ακολούθως το κάθε frame στον φάκελο περνά από ακόμα ένα script το οποίο γράφτηκε σε Python έτσι ώστε να εξαλείψουμε όλες τις πιθανές σκιές που προκαλούν πρόβλημα στην ανάλυση της εικόνας.

Για την επίτευξη του πιο πάνω, το κάθε frame πέρασε από την πιο κάτω διαδικασία :

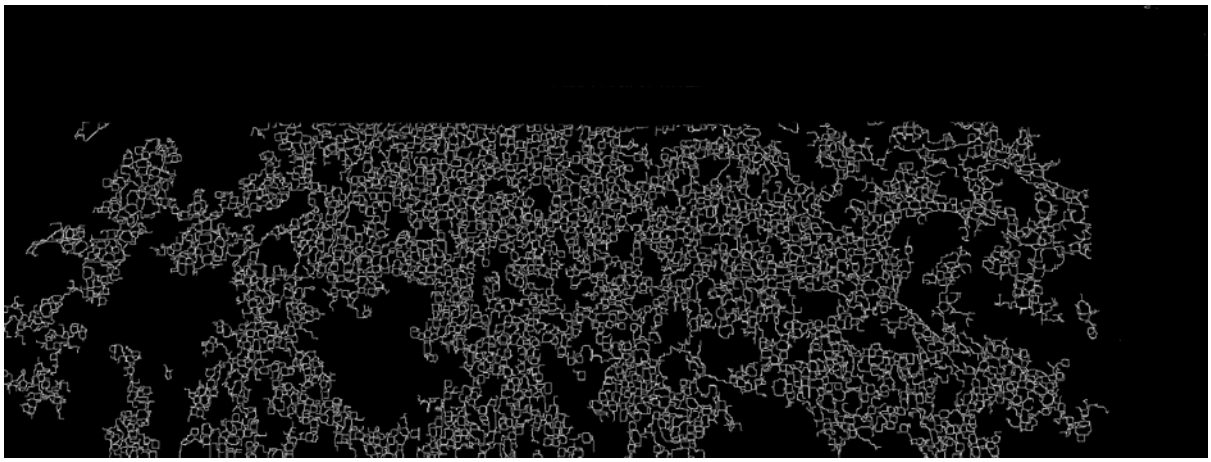
1. Διαχωρισμός στα 3 χρώματα που απαρτίζουν ένα frame.
2. Στο κάθε χρώμα του frame έγινε:
    i. Dilation – έτσι ώστε να μεγαλώσει το σημείο όπου αρχίζει και τελειώνει η πιθανή σκιά.
    ii. Median filter – Για αφαίρεση θορύβου.
    iii. Αντιστροφή των χρωμάτων (grayscale) της εικόνας και κανονικοποίηση έτσι ώστε να μην είναι έντονο το σημείο σκίασης
    iv. Αποθήκευση

Εικόνα 44: Αφαίρεση σκιών από το Frame

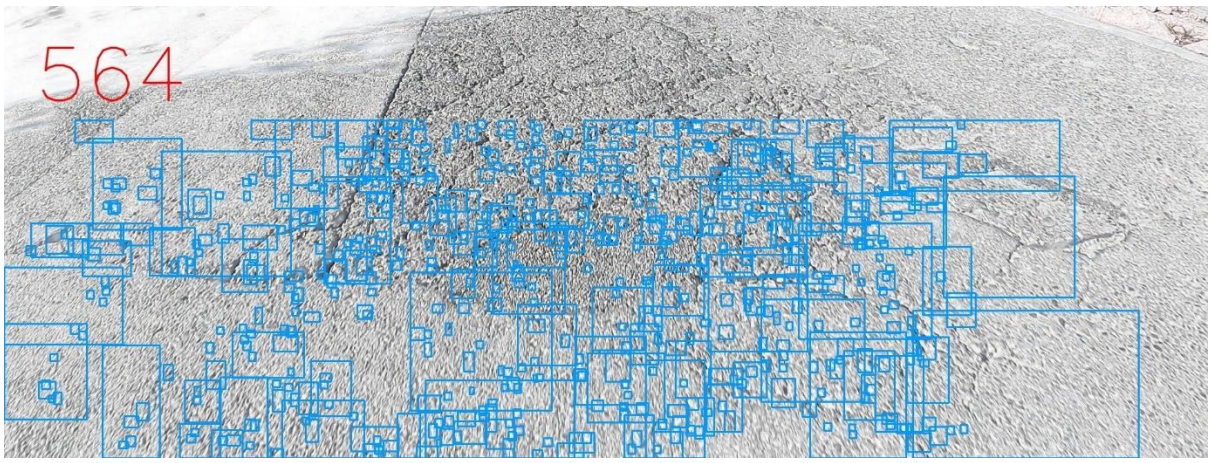
Βήμα 3

Αφού έγινε η επεξεργασία στις εικόνες και αφαιρέθηκαν οι σκιές ήρθε η ώρα να βρεθούν οι φθορές στο οδόστρωμα και να επισημανθούν.
Για κάθε νέο frame έγινε η πιο κάτω διαδικασία:

i. Median filter – Για αφαίρεση θορύβου.
ii. Erosion and Dilation – για ενοποίηση σημείων(pixel) που ήταν πανομοιότυπα και τονισμό των σημείων που ήταν ανόμοια.
iii. Canny Filter – Για εύρεση ακμών στην εικόνα.
iv. Δημιουργία τετραγώνου, περικλείοντας κάθε περιοχή που βρέθηκε ως συνεχόμενη ακμή.
v. Καταμέτρηση τετραγώνων που δημιουργήθηκαν για καθορισμό κρισιμότητας φθοράς.
vi. Αποθήκευση



Εικόνα 45: Εφαρμογή του Canny Edge Detector σε frame από πραγματικό βίντεο



Εικόνα 46: Δημιουργία τετραγώνων ανάλογα με τις συνεχόμενες ακμές που βρέθηκαν και καταμέτρηση

Βήμα 4

Αφού επεξεργαστήκαμε πλήρως όλα τα frames από όλα τα βίντεο, έχουμε πλέον φακέλους οι οποίοι αντιπροσωπεύουν ο καθένας ξεχωριστά ένα επεξεργασμένο βίντεο.
Το μόνο που απομένει είναι να δημιουργήσουμε ξανά τα βίντεο για να έχουμε μια τελική εικόνα για το μέγεθος της φθοράς στην κάθε καταγραφή.

Εικόνα 47: Δημιουργία τελικού βίντεο όπου απεικονίζονται οι φθορές

6.4 Αποτελέσματα

Κατά την επεξεργασία των βίντεο, θεωρήθηκε σωστό να κατασκευάσουμε κάποιου είδους μοντέλο έτσι ώστε να μπορούμε να διακρίνουμε ποια βίντεο περιέχουν εμπεριέχουν φθορές. Με αυτό τον τρόπο μπορούμε να διαχωρίσουμε σε ποιες περιοχές με βάση τα βίντεο πρέπει να επισπεύσουμε τη διαδικασία συντήρησης του οδοστρώματος.

Αναλυτικότερα, αφού παρακολουθήσαμε μερικά (17)βίντεο τα κατηγοριοποιήσαμε με 3 δείκτες High, Medium, Low οι οποίοι μας διευκρινίζουν σε ποια κατηγορία ανήκουν οπτικά οι φθορές που παρουσιάζονται. Δηλαδή αν το οδόστρωμα:

- Μεγάλη ζημία, άμεση ανάγκη για συντήρηση (High)
- Μέτρια ζημία, μέτρια ανάγκη για συντήρηση (Medium)
- Μικρή ζημία, μικρή ανάγκη για συντήρηση (Low)

Στη συνέχεια περάσαμε τα βίντεο από την επεξεργασία που επεξηγήθηκε πιο πάνω και μετρήσαμε των αριθμό των τετραγώνων που εμπεριέχονται συνολικά σε όλα τα frames στο εκάστοτε βίντεο. Στη συνέχεια υπολογίσαμε τον μέσο όρο τετραγώνων per frame και κατηγοριοποιήσαμε ακολούθως την προτεραιότητα συντήρησης:

- 60 rectangles/frame (High)
- 30-60 rectangles/frame (Medium)
- 0-30 rectangles/frame (Low)

Στον πίνακα πιο κάτω παρουσιάζονται τα αποτελέσματα αξιολόγησης με βάση αυτά που εξηγήθηκαν ποιο πάνω.

Video	Sum of Rect	Number of frames	Av.Rect per frame	Ground Truth	False Predictions
1666178698	41363	424	97.55424528	High	✓
1667393562	5920	423	13.99527187	Low	✓
1667393580	3854	423	9.111111111	Low	✓
1667393598	23160	424	54.62264151	Medium	✓
1667393614	76265	426	179.0258216	High	✓
1667393676	8645	423	20.43735225	Low	✓
1667393714	17954	423	42.44444444	Medium	✓
1667393732	22639	426	53.14319249	Low	✗
1667393746	34342	425	80.80470588	High	✓
1667393760	10125	485	20.87628866	Low	✓
1667393776	14894	426	34.96244131	Low	✗
1667393790	18565	423	43.88888889	Medium	✓
1667393806	22982	423	54.33096927	Medium	✓
1667397402	8930	427	20.91334895	Low	✓
1667397418	17755	424	41.875	Medium	✓
1667397430	10257	426	24.07746479	Low	✓
1667397450	37954	427	88.8852459	High	✓

Πίνακας 12: Αποτελέσματα κατηγοριοποίησης της κατάστασης του οδοστρώματος με βάση την μέθοδο επεξεργασίας των βίντεο

Στον πίνακα ποιο πάνω βλέπουμε στην στήλη Ground Truth την εκτίμηση που έγινε εμπειρικά με μόνη βάση το οπτικό περιεχόμενο των βίντεο και στην στήλη Av. Rect per frame βλέπουμε το αποτέλεσμα που μας δίνει ο αλγόριθμος.

Παρατηρούμε ότι ο αλγόριθμος επαληθεύει τα αποτελέσματα που δόθηκαν εμπειρικά στα 15 από τα 17 βίντεο. Αυτό μας δίνει:

Accuracy Rate (%)	88.2352941
Error Rate(%)	11.7647059

Αξίζει να σημειωθεί ότι τα λάθη που προέκυψαν είναι 2 από την κατηγόρια που δόθηκε εμπειρικά ως μικρής κλίμακας ζημία (Low) ενώ ο αλγόριθμος αξιολόγησε ότι τα συγκεκριμένα βίντεο εμπεριέχουν ζημία μεσαίας κλίμακας. Αυτού του είδους σφάλμα δεν θεωρείται σφάλμα μεγάλης επικινδυνότητας αφού τελικά ο αλγόριθμος έκρινε την φθορά ως μεγαλύτερη άρα το αρμόδιο συνεργείο πρέπει να της δώσει την ανάλογη προτεραιότητα. Απαγορευτικό σφάλμα θα ήταν αν ο αλγόριθμος έκρινε έναν βίντεο ότι εμπεριέχει φθορές μικρής κλίμακας αλλά στην πραγματικότητα ο δρόμος έχει φθορές μέτριας ή χειρότερα μεγάλης κλίμακας.

7 Συζήτηση και σύγκριση Αποτελεσμάτων

Σε αυτό το κεφάλαιο θα συζητηθούν τα αποτελέσματα που προέκυψαν και στη συνέχεια θα γίνει σύγκριση των μεθόδων που χρησιμοποιήθηκαν, με σκοπό να ικανοποιήσει τους στόχους αυτής της διπλωματικής εργασίας, με άλλες προηγούμενες παρόμοιες εργασίες. Στόχος αυτού του κεφαλαίου είναι να αναδείξει τα επιτεύγματα της παρούσας διπλωματικής εργασίας σε σχέση με προηγούμενες εργασίες.

7.1 Συζήτηση Αποτελεσμάτων

Στόχος της παρούσας διπλωματικής εργασίας ήταν ο σχεδιασμός και η υλοποίηση της αρχιτεκτονικής ενός ολοκληρωμένου χαμηλού κόστους αυτοματοποιημένου υπολογιστικού συστήματος το οποίο θα χωρίζεται σε επιμέρους στοιχεία για αυτόματη καταγραφή βίντεο, ανάλυση και επεξεργασία βίντεο, ανάλυση, μεταφορά, επεξεργασία και αποθήκευση δεδομένων.

Επιπρόσθετα η συγκεκριμένη συσκευή-πλατφόρμα είχε ως στόχο να εξομαλύνει την διαδικασία ανίχνευσης, και καταγραφής φθορών στο οδόστρωμα απαιτώντας ελάχιστη παρέμβαση από τον χρήστη λόγο αυτοματοποιήσεων. Και να προσφέρει όλα τα πιο πάνω ως μια οικονομική λύση που θα μπορεί εύκολα να αγοραστεί και να χρησιμοποιηθεί από τους αρμόδιους φορείς.

Ο γενικός στόχος έχει επιτευχθεί αφού έχει υλοποιηθεί και δοκιμαστεί το όλο σύστημα και όλα τα επιμέρους χαρακτηριστικά του.

Πιο αναλυτικά, επιλέχθηκαν και συναρμολογήθηκαν όλα τα απαραίτητα ηλεκτρονικά, αφού πρώτα μελετήθηκε το όλο έργο και ποιες λειτουργίες έπρεπε να εκτελούνται. Δημιουργώντας την συσκευή έπρεπε να υλοποιηθεί λογισμικό το οποίο να εξυπηρετεί το υλικό στον μέγιστο δυνατό βαθμό χωρίς να παρουσιάζει προβλήματα κατά την διάρκεια χρήσης αλλά ούτε στον χρήστη του. Το λογισμικό έπρεπε να υλοποιηθεί με γνώμονα την εύκολη αποσφαλμάτωση του, την επαναχρησιμοποίηση, την ευρωστία, την επεκτασιμότητα και την ανοχή σε αλλαγές κατά τις δοκιμές. Τα πλείστα από τα πιο πάνω επιτεύχθηκαν λόγω της αρθρωτότητας που προσφέρεται χρησιμοποιώντας multi-threading κατά την εκτέλεση. Επίσης η τελική συσκευή προσφέρει ευκολία στην χρήση αφού χρειάζεται ελάχιστη επαφή με τον χρήστη για να τεθεί σε λειτουργία. Από την στιγμή της αρχικής ενεργοποίησης από τον χρήστη η συσκευή δουλεύει αυτόνομα, καταγράφοντας και μεταφέροντας βίντεο είτε από την κάμερα στη συσκευή είτε από τη συσκευή στον τοπικό διακομιστή για περαιτέρω επεξεργασία αφού πρώτα σιγουρευτεί ότι το βίντεο κατέχει τις σωστές προδιαγραφές.

Όσο αφορά τον τοπικό διακομιστή έπρεπε να υλοποιηθεί λογισμικό το οποίο να εξυπηρετεί τις ανάγκες του συστήματος. Αυτό επιτεύχθηκε αφού ο τοπικός διακομιστής μέσω λογισμικού που υλοποιήθηκε μπορεί να επεξεργαστεί βίντεο από την συσκευή καταγραφής βίντεο, εξάγοντας τα δεδομένα από αυτά και τοποθετώντας λεζάντες συντεταγμένων και χρονικής ένδειξης πάνω σε αυτά. Υλοποιώντας τον τοπικό διακομιστή καταφέραμε να αποφορτίσουμε την συσκευή από την επεξεργασία των βίντεο σε θέμα επεξεργαστικής ισχύς

αλλά και χώρου αποθήκευσης, έχοντας την συσκευή καταγραφής ελεύθερη για ολική απασχόληση στην καταγραφή.

Επιπρόσθετα προσθέτοντας ένα δρομολογητή στο σύστημα καταφέραμε να απομονώσουμε το σύστημα από εξωτερικούς παράγοντες και να μπορούμε να μεταφέρουμε το σύστημα μας από περιοχή σε περιοχή χωρίς να πρέπει ο χρήστης της συσκευής να πρέπει να γυρίζει κάθε φορά στον σταθμό απλά για να ανεβάσει τα βίντεο στον διακομιστή του. Ακόμα ένα πλεονέκτημα που προσφέρει στο σύστημα ο δρομολογητής είναι ότι μείωσε τις αποστάσεις για την μεταφορά των βίντεο, αφού πλέον η μεταφορά γίνεται εντός δικτύου(τοπικά).

Επιπλέων ο τοπικός διακομιστής έχει κατασκευαστεί σε επίπεδο όπου μπορεί να φιλοξενήσει περισσότερες από μία συσκευές καταγραφής, άρα με αυτόν τον τρόπο μπορεί γλιτώνονται χρήματα στην αγορά των διακομιστών, κάνοντας το σύστημα ακόμα πιο προσιτό κατά την αγορά.

Επιπρόσθετα με την βοήθεια του τοπικού διακομιστή μεταφέρονται τα δεδομένα τα οποία εξάγονται από τη επεξεργασία τον βίντεο σε ένα απομακρυσμένο διακομιστή(VPS) και φυλάγονται σε βάση δεδομένων το οποίο προσθέτει μια έξτρα αφαιρετικότητα στο σύστημα. Τα δεδομένα μπορούν να ληφθούν μέσω ενός Api στον έξω κόσμο και μπορούν να χρησιμοποιηθούν για περαιτέρω επεξεργασία ή απεικόνιση, από πληθώρα εφαρμογών σε όλων τον τύπων υπολογιστικές μηχανές και προγράμματα.

Σαν επιπρόσθετο παράρτημα στην τρέχουσα διπλωματική εργασία θεωρείτε το κομμάτι επεξεργασίας των βίντεο για αναγνώριση των φθορών. Σε αυτό το κομμάτι τα βίντεο υποδιαιρέθηκαν σε frames και επεξεργάστηκαν ανεξάρτητα για να γίνει εντοπισμός των ανωμαλιών στο οδόστρωμα αφού πρώτα πέρασαν από φίλτρα καθαρισμού του θορύβου αλλά και από μέθοδο αφαίρεσης των σκιών. Μετά το τέλος της διαδικασίας ο χρήστης έχει ένα ολοκληρωμένο βίντεο στο οποίο απεικονίζονται ξεκάθαρα οι ανωμαλίες στο οδόστρωμα. Επιπρόσθετα ο αλγόριθμος μπορεί να κατηγοριοποιήσει το μέγεθος των φθορών που εμπεριέχονται σε κάθε βίντεο με High, Medium, Low. Και με αυτό τον τρόπο έχουμε ουσιαστικά ένα σύστημα προτεραιότητας που θα βοηθήσει το αρμόδιο συνεργείο.

Γενικότερα όπως επεξηγήθηκε και πιο πάνω η παρούσα διπλωματική εργασία φαίνεται να καλύπτει διάφορες τεχνολογικές κατευθύνσεις ενώ παρουσιάζει ανάπτυξη λογισμικού και υλικού σε ένα ολοκληρωμένο σύστημα. Ειδικά στο λογισμικό χρησιμοποιήθηκαν τεχνικές οι οποίες προσφέρουν στο σύστημα ευχρηστία, ευρωστία, επεκτασιμότητα, δίνοντας την ώθηση για να μπορεί να θεωρείτε ένα πρωτότυπο που μπορεί να προσφέρει στην κοινωνία και στον κόσμο.

7.2 Σύγκριση Αποτελεσμάτων

Σε αυτό το κεφάλαιο θα συγκριθούν τα αποτελέσματα και οι μέθοδοι που χρησιμοποιήθηκαν σε αυτή την εργασία με άλλες προηγούμενες εργασίες.

Για να μπορέσουμε να συγκρίνουμε τα αποτελέσματα μας θα πρέπει να υποδιαιρέσουμε τα επιμέρους κεφάλαια της παρούσας διπλωματικής εργασίας έτσι ώστε να είναι ξεκάθαρη η σύγκριση.

Όσων αφορά παλαιότερες εργασίες με θέμα την αναγνώριση και καταγραφή φθορών στο οδόστρωμα, μπορούμε να βρούμε πληθώρα από εργασίες οι οποίες βασίζονται κυρίως στο γεγονός ότι υπάρχουν τα δεδομένα από το οδόστρωμα(βίντεο, φωτογραφίες και δεδομένα από αισθητήρες) και χρησιμοποιούν αλγόριθμους μηχανικής μάθησης για να εντοπίσουν αν υπάρχει φθορά σε όλων των ειδών των δεδομένων που κατέχουν. Σε αυτές τις προσεγγίσεις το κομμάτι των δεδομένων θεωρείτε έτοιμο εφαρμόζοντας μόνο τους αλγόριθμους.

Το 2019, οι Ali Anaissi et al πρότειναν ένα virtual road network inspector (VRNI) ο οποίος ήταν υπεύθυνος χρησιμοποιώντας μόνο μετρήσεις επιταχυνσιόμετρου παρατηρούσε την κατάσταση του οδοστρώματος. Αυτό γινόταν εφικτό χρησιμοποιώντας Support Vector Machines πάνω στα δεδομένα που συλλέγονταν με ακρίβεια 97.5% και false alarm rate 4%. Οι συγκεκριμένη αισθητήρες δοκιμάστηκαν και εφαρμόστηκαν για αυτό το πρωτότυπο έργο σε σχολικά λεωφορεία στην Αυστραλία.

Οι Christian Koch και Ioannis Brilakis [9] χρησιμοποίησαν 120 εικόνες οδοστρώματος (training and test set) σε αλγόριθμο κατασκευασμένο σε MATLAB ο οποίος με βάση το histogram shape-based thresholding ανίχνευε τις λακκούβες στο οδόστρωμα. Τα αποτελέσματα της όλης διαδικασίας φαίνεται να έχουν μεγάλο ποσοστό ακρίβειας, αφού ο αλγόριθμος είναι σε θέση να μπορεί να σχεδιάσει στην υφιστάμενη εικόνα το περίγραμμα της φθοράς.

Το 2020 οι Ashwini KS et al [10], πρότειναν μία οικονομική μέθοδο η οποία χρησιμοποιεί ένα smartphone και μια μονάδα OBD-II για τον εντοπισμό και τον εντοπισμό λακκούβων στους δρόμους. Χρησιμοποιεί δεδομένα όρασης, δεδομένα αισθητήρων και δεδομένα OBD για τη δημιουργία και την επικύρωση εναυσμάτων που προκαλούνται από λακκούβες. Οι προτεινόμενες μέθοδοι ενεργοποίησης εικόνας και ενεργοποίησης δεδομένων αναγνωρίζουν τις λακκούβες στους δρόμους και δημιουργούν ένα έναυσμα μέσω της επεξεργασίας εικόνας/βίντεο και επεξεργασίας δεδομένων αντίστοιχα. Η προτεινόμενη εργασία είχε τοποθετημένο ένα smartphone τοποθετημένο στο αυτοκίνητο για να καταγραφεί βίντεο σε συνεχόμενη χρήση και να επισημαίνει τα εναύσματα.

Το 2020 οι Braian Varona et al [1], προτείναν μια προσέγγιση deep learning που τους επίτρεπε (α) να προσδιορίζουν αυτόματα τα διαφορετικά είδη οδοστρώματος και (β) να διακρίνουν αυτόματα τις λακκούβες από τις αποσταθεροποιήσεις που προκαλούνται από ανωμαλίες ταχύτητας ή ενέργειες του οδηγού στο πλαίσιο εφαρμογής που βασίζεται στο crowdsensing . Συγκεκριμένα, αναλύουν και εφαρμόζουν διαφορετικά μοντέλα deep learning: συνελικτικά νευρωνικά δίκτυα, δίκτυα LSTM και reservoir computing models. Τα πειράματα πραγματοποιήθηκαν με πληροφορίες πραγματικού κόσμου και τα αποτελέσματα έδειξαν μια πολλά υποσχόμενη ακρίβεια (98%) στην επίλυση και των δύο προβλημάτων.

Οι Kelvin C.P. Wang et al [10], περιγράφουν την ανάπτυξη ενός πολυλειτουργικού συστήματος σε πραγματικό χρόνο για την απόκτηση και ανάλυση δεδομένων οδοστρώματος, ιδιαίτερα για την έρευνα κινδύνου της επιφάνειας του οδοστρώματος και τη διαχείριση περιουσιακών στοιχείων στην άκρη του δρόμου. Αυτό το σύστημα, Digital Highway Data Vehicle (DHDV), συνδυάζει τις τεχνολογίες της ψηφιακής απεικόνισης με βάση τον φωτισμό λέιζερ, του αδρανειακού προφίλ και της χαρτογράφησης GPS σε ένα ολοκληρωμένο σύστημα για την ολοκλήρωση των πολλαπλών εργασιών έρευνας και διαχείρισης δεδομένων οδοστρώματος. Η ανάλυση των εικόνων του οδοστρώματος είναι 1 mm τόσο σε διαμήκη όσο και σε αντίστροφες κατευθύνσεις με τη χρήση καμερών γραμμής υψηλής ανάλυσης που διασχίζουν ολόκληρο το πεζοδρόμιο μιας λωρίδας. Η ανάλυση του οδοστρώματος και των ανωμαλιών του γίνονται σε πραγματικό χρόνο. Αξίζει να σημειωθεί ότι το πιο πάνω σύστημα κοστίζει γύρω στα 1 εκ. δολάρια και είναι ένα ολοκληρωμένο όχημα.

Με βάση τα πιο πάνω η παρούσα διπλωματική εργασία παρουσιάζει σχεδόν όλα τα μέρη που επεξηγήθηκαν. Πιο αναλυτικά στην παρούσα διπλωματική εργασία έχουμε κατασκευάσει μία συσκευή καταγραφής βίντεο με φτηνά υλικά τα οποία μπορούν να διατεθούν στο κοινό εύκολα. Η συσκευή αυτή μπορεί να κατασκευαστεί με λιγότερα από 1000 ευρώ.

Η συσκευή είναι σε θέση να καταγράψει βίντεο μόνο όταν είναι απαραίτητο (vibration trigger, 7 seconds video) και να ρυθμιστεί η ευαισθησία της για να γίνεται πιο συγκεκριμένο το μέγεθος της φθοράς που θέλουμε να καταγράψουμε. Επιπρόσθετα το όλο σύστημα μπορεί να επεκταθεί εύκολα(περισσότερες από 1 συσκευές καταγραφής) αφού οι μέθοδοι που υλοποιήθηκαν σε συνδυασμό με την ιδέα το τοπικού διακομιστή του προσφέρουν τον τίτλο «μετακινούμενος σταθμός καταγραφής». Επίσης το σύστημα προσφέρει και αποθήκευση των δεδομένων που συλλέχτηκαν, τα οποία είναι προσβάσιμα μέσω Api για περαιτέρω ανάλυση ή παρουσίαση. Τέλος, τα βίντεο μέσω μεθοδολογίας, επεξεργάστηκαν και ανακατασκευάστηκαν, δίνοντας ως τελικό αποτέλεσμα ένα νέο βίντεο το οποίο παρουσιάζει τις φθορές στο οδόστρωμα με μεγάλη ακρίβεια. Αλλά και την κατηγοριοποίηση των φθορών, έτσι ώστε να μπορεί η αρμόδια υπηρεσία να βάζει προτεραιότητα στην διαδικασία επισκευής των φθορών.

Με λίγα λόγια παρουσιάζουμε μια λύση η οποία ανακτά τα βίντεο από το δρόμο, αναλύει επεξεργάζεται, αποθηκεύει δεδομένα και επαναδημιουργεί οπτικά αποτελέσματα για τον χρήστη υποδεικνύοντας το πρόβλημα.

8 Συμπεράσματα και Μελλοντικές Προοπτικές

Σε αυτό κεφάλαιο επεξηγούνται τα συμπεράσματα που προκύπτουν μέσα από την τρέχουσα εργασία. Επισημαίνονται εισηγήσεις για το πως θα μπορούσε η παρούσα εργασία να εξελιχθεί και να προσφέρει καινοτόμα και πιο αξιόπιστα αποτελέσματα όσων αφορά την καταγραφή την επεξεργασία και την αποθήκευση των δεδομένων. Μελλοντικός στόχος μας είναι η εκτέλεση,η ανάπτυξη και η διεύρυνση της ερευνάς σχετικά με νέες καινοτόμες τεχνικές που θα μπορούσαν να προστεθούν στην μεθοδολογία μας για ανάπτυξη του λογισμικού επεξεργασίας των βίντεο για καλύτερα αποτελέσματα.

8.1 Συμπεράσματα

Αρχικά η σκέψη για την πραγματοποίηση της παρούσας μεταπτυχιακής εργασίας ανατέθηκε σε συνεργασία με την εταιρία SignalGeneriX όπου θα έπρεπε να δημιουργηθεί μία συσκευή για αυτόματη καταγραφή βίντεο και δεδομένων που αφορούν την κατάσταση του οδοστρώματος. Σημαντικής ανάγκης κρίνεται η είσοδος ενός τέτοιου εργαλείου σε όλες τις αρμόδιες αρχές που ασχολούνται με την επισκευή/επιδιόρθωση του οδοστρώματος.

Με βάση το στόχο του έργου που αναφέραμε πιο πάνω, το πρώτο συμπέρασμα που εξάγουμε, είναι η επιτυχής εκπλήρωση του στόχου αφού καταφέραμε να υλοποιήσουμε όλα τα στάδια στο μέγιστο του επιθυμητού αποτελέσματος εντός του χρονικού περιορισμού. Επιπρόσθετα μέσω αυτής της διπλωματικής εργασίας η συνεργασία μεταξύ εταιρίας και πανεπιστημιακής κοινότητας, αφού σε πολλές περιπτώσεις χρειάστηκε η άμεση συμβολή και τω δύο. Μετά την ολοκλήρωση του έργου έχουμε φέρει εις πέρας ένα πρωτότυπο σύστημα που με της απαραίτητες μεταρρυθμίσεις θα μπορεί να ενταχτεί στα εργαλεία του κράτους για εκμετάλλευση πόρων και αποφόρτιση προσωπικού που έως τώρα όλες οι απαραίτητες μετρήσεις και αξιολογήσεις γίνονταν εμπειρικά και με την συμβολή πολλών καταρτισμένων ατόμων.

Τέλος, θέλουμε να ελπίζουμε πως η παρούσα διπλωματική εργασία θα γίνει το εναρκτήριο λάκτισμα για περεταίρω έρευνα στους συγκεκριμένους τομείς, αφού μέσω αυτών καθημερινά δαπανιόνται τεράστια ποσά από την τσέπη του πολίτη.

8.2 Μελλοντικές Προοπτικές

Στην τρέχουσα διπλωματική εργασία εφαρμόσαμε την μεθοδολογία μας γύρω από την συσκευή καταγραφής βίντεο και στο τί μπορεί να προσφέρει σε συνεργασία με την κάμερα που επιλέχθηκε. Έχοντας τα δεδομένα που συλλέχθηκαν μέσω της συσκευής το μόνο που κάνουμε είναι να τα προσαρμόσουμε έτσι να μπορούν να αποθηκευτούν σε μία βάση δεδομένων. Θα ήταν βέλτιστο αν αυτά τα δεδομένα γίνονταν είσοδος σε κάποιου είδους αλγόριθμο μηχανικής μάθησης έτσι ώστε τα αποτελέσματα που θα εξάγουμε να μπορούν να συνοδεύσουν τα επεξεργασμένα βίντεο και να έχουμε μία πιο ολοκληρωμένη εικόνα της κατάστασης στο οδόστρωμα.
Επιπρόσθετα, θα ήταν καλό να γίνει πιο αυτοματοποιημένη η διαδικασία επεξεργασίας και επισήμανσης των λακκούβων στα βίντεο αφού στο παρόν στάδιο έχουμε δύο ανεξάρτητες διαδικασίες. Δηλαδή έχουμε επεξεργασία των βίντεο για να προσθέσουμε χρονική σήμανση

και συντεταγμένες, και μία επιπρόσθετη διαδικασία που παίρνει το αρχικό βίντεο και προσθέτει σημάνσεις για τις λακκούβες.

Ακόμα κάτι που θα θεωρείτο ιδανικό για αυτή την εργασία θα ήταν η ανάπτυξη ενός γραφικού περιβάλλοντος το οποίο θα αποτελείτε από ένα χάρτη στον οποίο θα απεικονιζόταν με στίγματα τα σημεία που είχαμε καταγραφή βίντεο και πατώντας σε αυτά θα ξεκινούσε το βίντεο που θα δείχνει την πραγματική φθορά μετά από την επεξεργασία που έγινε.

Τέλος ελπίζουμε ότι μετά το τέλος της εργασίας θα γίνουν οι απαραίτητες ενέργειες για να τεθεί σε λειτουργία ένα τέτοιο σύστημα. Και θα δημιουργηθεί μία ομάδα ατόμων που να μπορούν να συνεχίσουν και να αναπτύξουν αυτό το έργο.

9 Βιβλιογραφία

[1] B. M. A. T. A. Varona, "A deep learning approach to automatic road surface monitoring and pothole detection," *Personal and Ubiquitous Computing*, vol. 24, no. 2020, pp. 519-534, 2020.

[2] A. B. F. ∗. Tedeschi, "A real-time automatic pavement crack and pothole recognition system for mobile Android-based devices," *Computers & Graphics 107*, vol. 32, pp. 11-25, April 2017.

[3] A. D. R. Ramesh, "Cloud-Based Collaborative Road-Damage Monitoring with Deep Learning and Smartphones.," *Sustainability 2022*, vol. 14, no. 14, p. 8682, 20 May 2022.

[4] E. R. A. B. S. C. M. S. I. P. M.-K. Moscoso Thompson, "SHREC 2022: Pothole and crack detection in the road pavement using images and RGB-D data.," *Computers & Graphics 107*, vol. 107, pp. 161-171, Octomber 2022.

[5] E. X. Yu, "Pavement pothole detection and severity measurement using laser imaging," in *IEEE International Conference on Electro-Information Technology*, Mankato, 2011.

[6] A. K. R. Dhiman, "Pothole Detection Using Computer Vision and Learning," *IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS*, vol. 21, no. 8, pp. 3536-3550, August 2020.

[7] E. J. M. C. J. R. Salcedo, "A Novel Road Maintenance Prioritisation System Based on Computer Vision and Crowdsourced Reporting.," *Journal of Sensor & Actuator Networks.*, vol. 11, no. 1, pp. p15-N.PAG. 22p, 14 February 2022.

[8] H. Z. ·. F. M. N. A. Fahimifar, "Image Based Techniques for Crack Detection, Classification and Quantification in Asphalt Pavement: A Review," *Archives of Computational Methods in Engineering*, vol. 24, no. 2017, p. 935–977, September 2016.

[9] C. Koch and I. Brilakis, "Pothole detection in asphalt pavement images," 2011.

[10] Z. H. a. W. G. Kelvin C.P. Wang, "Automation Techniques for Digital Highway Data Vehicle (DHDV)," in *7th International Conference on Managing Pavement Assets*, Calgary Alberta, Canada, 2008.

[11] G. Jog, C. Koch, M. Golparvar-Fard and I. Brilakis, "Pothole properties measurement through visual 2D recognition and 3D reconstruction," in *ASCE International Conference on Computing in Civil Engineering,*, FL, USA, 2012.

[12] Γ. Π. ΠΑΠΑΓΕΩΡΓΙΟΥ, "ΒΕΛΤΙΣΤΟΠΟΙΗΣΗ ΑΝΑΒΑΘΜΙΣΗΣ ΚΑΙ ΣΥΝΤΗΡΗΣΗΣ," Πανεπιστημίου Θεσσαλίας, Πανεπιστημίου Θεσσαλίας, 22 Ιουλίου 2010.

[13] K. B. G. S. T. P. P. Ashwini, "Trigger-Based Pothole Detection Using Smartphone and OBD-II," in *2020 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT)*, Bangalore, India, 2020.

[14] D. C. A. M. A. P. Audrius Vaitkus, "Improvement of Road Pavement Maintenance Models and Technologies," *THE BALTIC JOURNAL OF ROAD AND BRIDGE ENGINEERING*, vol. 11, no. 3, pp. 242-249, September 2016.

[15] E. Buza, S. Omanovic and Huseinnovic, "Pothole detection with image processing and spectral clustering," in *2nd International Conference on Information Technology and Computer Networks*, Antalya, Turkey, 2013.

[16] N. Z. X. G. Y. Chen Dong Chen, "Real-Time Road Pothole Mapping Based on Vibration Analysis in Smart City," *IEEE JOURNAL OF SELECTED TOPICS IN APPLIED EARTH OBSERVATIONS AND REMOTE SENSING*, vol. 15, no. 2022, pp. pp 6972-6984, 19 August 2022.

[17] J. Erikson, L. Girod and B. Hull, "The Pothole Patrol: Using a mobile sensor network for road surface monitoring," in *Proceedings of the 6th International Conference on Mobile Systems, Applications, and Services (MobiSys 2008)*, Breckenridge, CO, USA, 2008.

[18] R. Fan, M. J. Bocus, Y. Zhu, J. Jiao, L. Wang, F. Ma, S. Cheng and M. Liu, "Road Crack Detection Using Deep Convolutional Neural Network and Adaptive Thresholding," in *Intelligent Vehicles Symposium*, Paris, 2019.

[19] Z. S. Hernanda, H. Mahmudah and R. W. Sudibyo, "CNN-Based Hyperparameter Optimization Approach for Road Pothole and Crack Detection Systems," in *AI IoT Congress (AIIoT)*, Jun, 2022.

[20] Y. S. T. S. T. K. &. H. O. Hiroya Maeda, "Road Damage Detection and Classification Using Deep Neural Networks with Smartphone Images," *Comput. Aided Civil Infrastruct. Eng.*, vol. 33, no. 12, pp. 1127-1141, 2018.

[21] A. A. ·. N. L. D. Khoa, "Smart pothole detection system using vehicle-mounted sensors," *Journal of Civil Structural Health Monitoring*, vol. 9, p. 91–102, 10 January 2019.

[22] J. M. C. Manalo, A. S. Alon, Y. D. Austria, N. E. Merencilla, M. A. Misola and R. C. Sandil, "A Transfer Learning-Based System of Pothole Detection in Roads through Deep Convolutional Neural Networks," in *International Conference on Decision Aid Sciences and Applications (DASA)*, Chiangrai, Thailand, 2022.

[23] A. Mednis, G. Strazdins, R. Zviedris, G. Kanonirs and L. Selavo, "Real time pothole detection using Android smartphones with accelerometers," in *2011 International Conference on Distributed Computing in Sensor Systems and Workshops (DCOSS)*, Barcelona, Spain, 2011.

[24] A. Molenaar, "Structural evaluation and strengthening of flexible pavements using deflections measurements and visual condition surveys," in *Sino-Netherlands seminar te Hianan, China*, Delft, 2005.

[25] M. Y. X. Y. X. Q. Li, "A real-time 3D scanning system for pavement distortion inspection," *Measurement Science and Technology*, vol. 21, no. 1, 16 November 2009.

[26] S. K. P. S. P. K. S. Satti, Detecting potholes on Indian roads using Haar feature-based cascade classifier, convolutional neural network, and instance segmentation, vol. 26, Springer Berlin Heidelberg, 2022, p. 9141–9153.

[27] Y.-G. K. ,. S.-Y. S. ,. S.-Y. L. ,. B.-Y. C. a. D.-H. C. Young-Mok Kim 1, "Recent Automated Pothole-Detection Methods," *Applied Sciences,* vol. 12, no. 11, p. 5320, May 2022.

[28] B. X. Yu and a. X. Yu, "Vibration-Based System for Pavement Condition Evaluation," in *Ninth International Conference on Applications of Advanced Technology in Transportation (AATT)*, Chicago, 2006.

[29] F. A. Zhang, "Research on Pothole Detection Method for Intelligent Driving Vehicle," in *3rd International Conference on Pattern Recognition and Machine Learning*, Chengdu, China, 2022.

[30] K. Zoysa, C. Keppitiyagama, G. Seneviratne and W. Shihan, "A public transport system based sensor network for road surface condition monitoring.," *NSDR '07: Proceedings of the 2007 workshop on Networked systems for developing regions,* no. 9, pp. 1-6, 27 August 2007.

[31] Γ. Γκέκας, "Εγχειρίδιο Επιθεώρησης Κατάστασης Οδοστρώματος," 2013.

10 Παράρτημα Α

Σε αυτό το κεφάλαιο θα παρουσιαστεί το λογισμικό που δημιουργήθηκε για την συσκευή καταγραφής βίντεο αλλά και για τον τοπικό διακομιστή. Πιο αναλυτικά θα παρουσιαστούν τα κύρια αρχεία που αποτελούν το service της συσκευής καταγραφής βίντεο αλλά και το τα αρχεία που αποτελούν το Rest Api στον τοπικό διακομιστή αλλά και τα αρχεία επεξεργασίας των βίντεο για προσθήκη λεζάντων σε αυτά.

10.1 Λογισμικό συσκευής αυτόματης καταγραφής βίντεο

Main.py

```
import logging
import datetime
import sys
import time
import subprocess
import screen
import nmcli
from pathsLinux import *
import globals
import os
from threading import Thread
from acc import *
import setTime


'''Main function of the program'''
if __name__ == "__main__":
    screen.setMessage("Device Starting...")
    setTime.current_time_set_linux_time()

    logging.basicConfig(filename=log_url, level=logging.INFO, filemode='a')
    now = str(datetime.datetime.now())
    logging.info(now + " : " + "Device Starting...")

    '''Looking for new scanned network every 9 sec'''
    def check_for_networks():
        while True:
            time.sleep(9)
            if globals.isDownloading:
                pass
            else:
                try:
                    nmcli.device.wifi_rescan()
                except Exception as e:
                    now = str(datetime.datetime.now())
                    logging.info(now + ": Error : " + str(e))
```

```
'''Initialization of network scanning'''
networkDaemon=Thread(target=check_for_networks)
networkDaemon.setDaemon(True)
networkDaemon.start()

'''Accelerometer handler for vibration interrupts'''
def accelerometerHandler():
    while True:
        accLoop()
from connecting_to_network import connecting_to_network

'''Initialization of accelerometer handler thread'''
accelerometerHandlerDaemon=Thread(target=accelerometerHandler,args=())
accelerometerHandlerDaemon.setDaemon(True)
accelerometerHandlerDaemon.start()

network_connection = connecting_to_network(screen)

'''Initialization of screen thread'''
screeThreadDaemon=Thread(target=screenUpdate,args=())
screeThreadDaemon.setDaemon(True)
screeThreadDaemon.start()

if len(network_connection.register_networks_to_connect)==0:
    logging.error(now + " : " + "Create wifi configuration file first!!!")
    sys.exit(0)
while True:

    network_connection.connect()
```

Acc.py

```
import time
# import board
import adafruit_fxos8700
import screen
import datetime
import logging
import Adafruit_BBIO.GPIO as GPIO
import Adafruit_GPIO.I2C as I2C
import signal
import sys
from threading import Thread
import globals
```

```
from pathsLinux import *

'''I2C configuration setup'''
fx = I2C.Device(0x1E, 2)

thres = 19;
screen.setErrorLed(False)
screen.setWifiLed(False)
screen.setCameraLed(False)
logging.basicConfig(filename=log_url, level=logging.INFO, filemode='a')

interruptPin = "P9_15"  # GPIO 48
resetInt = False
accInt = False


# FXOS8700 Drivers for accelerometer

def is_Connected():
    if (fx.readU8(0x0D) == 0xC7):
        screen.setMessage("                    ")
        return True
    else:
        screen.setErrorLed(True)
        screen.setMessage(" FX8700 not connected!")


threshold = 4

# Configure threshold of accelerometer
def conImageThreshold(ths):
    global threshold
    if ths < 0:
        ths = 0
    if ths > 100:
        ths = 100
    threshold = ths

# Configure threshold of accelerometer on X axis
def conImagePulseThsX():
    global threshold
    val = 0
    val = val | threshold
    fx.write8(0x23, val)

# Configure threshold of accelerometer on Y axis
def conImagePulseThsY():
    global threshold
    val = 0
    val = val | threshold
```

```
    fx.write8(0x24, val)

# Configure threshold of accelerometer on Z axis
def conImagePulseThsZ():
    global threshold
    val = 0
    val = val | threshold
    fx.write8(0x25, val)

# Configure register 1 in i2c
def conImageRegister1(active):
    aslp_rate = 0
    dr = 0x01
    lnoise = True
    f_read = False

    val = 0
    val = val | ((aslp_rate & 0x03) << 6) | ((dr & 0x07) << 3) | lnoise << 2 | f_read << 1 | active
    fx.write8(0x2A, val)

# Configure register 2 in i2c
def conImageRegister2():
    st = False
    rst = False
    dr = 0x01
    slpe = False
    mods = 0

    val = 0
    val = val | st << 7 | rst << 6 | ((dr & 0x3) << 3) | slpe << 2 | (mods & 0x03)
    fx.write8(0x2B, val)

# Configure register 3 i2c
def conImageRegister3():
    fifo_gate = False
    wake_trans = False
    wake_lndprt = False
    wake_pulse = False
    wake_ffmt = False
    wake_a_vecm = False
    ipol = False
    pp_od = False

    val = 0
    val = val | fifo_gate << 7 | wake_trans << 6 | wake_lndprt << 5 | wake_pulse << 4 |
wake_ffmt << 3 | wake_a_vecm << 2 | ipol << 1 | pp_od
    fx.write8(0x2C, val)

# Configure register 4 in i2c
def conImageRegister4():
```

```
    en_aslp = False
    en_fifo = False
    en_trans = False
    en_lndprt = False
    en_pulse = True
    en_ffmt = False
    en_a_vecm = False
    en_drdy = False

    val = 0
    val = val | en_aslp << 7 | en_fifo << 6 | en_trans << 5 | en_lndprt << 4 | en_pulse << 3 |
en_ffmt << 2 | en_a_vecm << 1 | en_drdy
    fx.write8(0x2D, val)


def conImageMRegistor1():
    m_acal = False
    m_rst = False
    m_ost = False
    m_os = 0x07
    m_hms = 0x03

    val = 0
    val = val | m_acal << 7 | m_rst << 6 | m_ost << 5 | ((m_os & 0x07) << 2) | ((m_hms &
0x03))
    fx.write8(0x5B, val)


def conImageMRegister2():
    hyb_autoinc_mode = True
    m_maxmin_dis = False
    m_maxmin_dis_ths = False
    m_maxmin_rst = False
    m_rst_cnt = 0

    val = 0
    val = val | hyb_autoinc_mode << 5 | m_maxmin_dis << 4 | m_maxmin_dis_ths << 3 |
m_maxmin_rst << 2 | (
            m_rst_cnt & 0x03)
    fx.write8(0x5C, val)


def conImageRegisterXYZ():
    hpf_out = False
    fs = 0x01

    val = 0
    val = val | hpf_out << 4 | (fs & 0x03)
    fx.write8(0x0E, val)
```

```
def conImagePulseCfg():
    pls_dpa = False
    pls_ele = False
    pls_zdpefe = False
    pls_zspefe = True
    pls_ydpefe = False
    pls_yspefe = True
    pls_xdpefe = False
    pls_xspefe = True

    val = 0
    val = val | pls_dpa << 7 | pls_ele << 6 | pls_zdpefe << 5 | pls_zspefe << 4 | pls_ydpefe << 3
| pls_yspefe << 2 | pls_xdpefe << 1 | pls_xspefe
    fx.write8(0x21, val)
```

'''Initializing all registers of accelerometer'''

```
def initialise(ths):
    conImageRegister1(False)

    conImageThreshold(ths)

    conImageMRegistor1()

    conImageMRegister2()

    conImageRegisterXYZ()

    conImageRegister4()

    conImagePulseCfg()

    conImagePulseThsX()

    conImagePulseThsY()

    conImagePulseThsZ()

    conImageRegister1(True)


def signal_handler(sig, frame):
    GPIO.cleanup()
    sys.exit(0)


def vibrationInterrupt(channel):
```

```
    global accInt
    accInt = True

# Set permissions for accelerometer changes
while True:
    try:
        time.sleep(2)
        GPIO.setup(interruptPin, GPIO.IN, pull_up_down=GPIO.PUD_UP)
        GPIO.add_event_detect(interruptPin, GPIO.FALLING, callback=vibrationInterrupt)
        logging.info("Accelerometer has been initialised")

        count = 0
        break
    except Exception as e:
        logging.info("Issue with permissions for accelerometer")
        logging.info(str(e))

'''Loop for vibration interrupts '''


def accLoop():
    global resetInt
    global accInt
    global count

    if (resetInt == False):
        if accInt == True:
            resetInt = True
            # VIBRATION FROM INTERUPT

            threadd = Thread(target=screen.setVibrationLed, args=([True, 3]))
            threadd.setDaemon(True)
            threadd.start()

            now = str(datetime.datetime.now())
            #Logging when interrupt happens
            logging.info(now + " : " + "Accelerometer Interrupt ")

            if globals.isRecording == False:
                globals.wasVibrated = True

            else:

                globals.wasVibrated = False
    else:
        count += 1

        if count >= 5:  # Increase this value to let more time between triggering Video
Recording
            count = 0
```

```
            resetInt = False
            accInt = False


'''Threashold update on screen'''
def screenUpdate():
    while True:
        thsScreen = screen.getThreshold()

        if thsScreen != None:
            if thsScreen != threshold:
                initialise(thsScreen)


try:
    screen.setMessage(" ")
    initialise(4)
    screen.setThreshold(4)
    is_Connected()
    screen.setTime()

except KeyboardInterrupt:
    signal.signal(signal.SIGINT, signal_handler)
    signal.pause()
```

Screen.py

```
import serial, time
from binascii import unhexlify
from datetime import datetime
from pair_and_connect_gopro8 import *
import os
import globals
import logging
from pathsLinux import *

'''Serial initialization'''

ser = serial.Serial()
ser.baudrate = 9600
ser.port = '/dev/ttyO5'
ser.open()

locked = False
threshold = 10
```

```
animating = False

changed_text = True
logging.basicConfig(filename=log_url, level=logging.INFO, filemode='a')

#Send command to screen for message
def sendCmd(cmd):
    checksum = 0
    strCMD = "".join(str(x) for x in cmd)

    for d in cmd:
        checksum ^= ord(d)

    strCMD = strCMD + chr(checksum)

    ser.write(bytes.fromhex(strCMD.encode("utf-8").hex()))


# Set Threshold on the screen
def setThreshold(value):
    c = ["\x01", "\x05", "\x00", "\x00"]

    c.append(chr(value - 2))
# Set threshold start from 3

    sendCmd(c)


#Function responsible for all the states of user interaction. In this function, user may power off the device, pair camera, and change threashold
def getThreshold():
    global threshold
    if ser.in_waiting:

        if (ser.read() == b'\x07'):
            serialState = ser.read(5)
            powerOffState = serialState

            thresholdState = str(serialState)[:14] + """
            #Check for threshold changed command
            if (thresholdState == str(b'\x05\x00\x00')):
                ans = str(serialState)[16:18]

                thresholdNew = int(ans, 16)

                if thresholdNew != threshold:
                    threshold = thresholdNew
                    setTime()
                    return threshold + 2
```

```
        #Check for poweroff command
         if (str(powerOffState) == str(b'\x06\x02\x00\x00\x03')):
            lockScreen()
            lines = os.popen("sudo systemctl poweroff").readlines()
        if (str(powerOffState) == str(b'\x0a\x00\x00\x00\x0d')):
            globals.is_connected_to = "Connected to: -------   "
        #Check for pairing command
        if (str(powerOffState) == str(b'\x06\x00\x00\x00\x01')):


            camera_connection_status = connect_camera_8()
            start_time = time.time()
            curr_time = time.time()
            counter = 1
            while (curr_time - start_time < 2):
                curr_time = time.time()
                if camera_connection_status == "SUCCESS":

                    if counter == 1:
                        logging.info("Camera Wifi succesfully updated ")
                    setMessage("Camera Wifi On")

                elif camera_connection_status == "ERROR":
                    setMessage("Camera Wifi Error")
                    if counter == 1:
                        logging.info("Camera Wifi Not Working")
                    # print("ERROR")
                elif camera_connection_status == "ERROR STATUS":
                    setMessage("Please Reboot Camera")
                    if counter == 1:
                        logging.info("Camera Wifi request did not reach device")
                else:
                    break
                counter = 0


# Camera Red Led [On when camera is connected]
def setCameraLed(flag):
    if flag == True:
        c = ["\x01", "\x0E", "\x00", "\x00", "\x01"]
        sendCmd(c)
    else:
        c = ["\x01", "\x0E", "\x00", "\x00", "\x00"]
        sendCmd(c)


# Wifi Red Led [On when wifi is connected]
def setWifiLed(flag):
    if flag == True:
```

```
        c = ["\x01", "\x0E", "\x01", "\x00", "\x01"]
        sendCmd(c)
    else:
        c = ["\x01", "\x0E", "\x01", "\x00", "\x00"]
        sendCmd(c)


# Error Red Led [On when there is an error]
def setErrorLed(flag):
    if flag == True:
        c = ["\x01", "\x0E", "\x02", "\x00", "\x01"]
        sendCmd(c)
    else:
        c = ["\x01", "\x0E", "\x02", "\x00", "\x00"]
        sendCmd(c)


# Vibration Red Led [On when vibration interrupt]
def setVibrationLed(flag, timer):
    if flag == True:
        c = ["\x01", "\x0E", "\x03", "\x00", "\x01"]
        sendCmd(c)
        time.sleep(timer)
        c = ["\x01", "\x0E", "\x03", "\x00", "\x00"]
        sendCmd(c)


# Set time showing on the screen
def setTime():
    # \x02 is the write Str command
    # \x01 is the id of the string object
    # \x11 is the count of characters
    # \x07 is the count of characters
    # \x20 is spaces to align in the middle
    now = datetime.now()
    h = now.strftime("%H")
    m = now.strftime("%M")
    c = ["\x02", "\x01", "\x05", h[0], h[1], ':', m[0], m[1]]

    sendCmd(c)


# Set message on the middle of the screen
def setMessage(msg):
    global changed_text

    # Editing For Master Degree
    if "device" in msg.lower():
        msg = msg.lower().replace("device", "Device")
        msg = msg.capitalize()
```

```
    if "sgx" in msg.lower():
        msg = msg.lower().replace("sgx", "wifi")
        msg = msg.capitalize()
    if "Connection" in msg:

        if globals.is_connected_to != msg.split(":")[1] or changed_text == True:

            c = ["\x02", "\x00"]

            c.append(chr(len(msg)))
            for ch in msg:
                c.append(ch)
            sendCmd(c)
            globals.is_connected_to = msg.split(":")[1]
            changed_text = False
        else:
            pass
    else:
        c = ["\x02", "\x00"]

        c.append(chr(len(msg)))
        for ch in msg:
            c.append(ch)
        sendCmd(c)
        changed_text = True


def setAnimatedMessage(msg):
    for i in range(9):
        setMessage(msg + "")
        time.sleep(0.3)
        setMessage(msg + ".")
        time.sleep(0.3)
        setMessage(msg + "..")
        time.sleep(0.3)
        setMessage(msg + "...")
        time.sleep(0.3)
    setMessage("  Done.")


def lockScreen():
    c = ['\x04', '\x00']
    sendCmd(c)
    locked = True


setThreshold(threshold)
time.sleep(1)
# getThreshold()
```

```
setWifiLed(True)
time.sleep(1)
setCameraLed(True)
time.sleep(1)
setErrorLed(True)
time.sleep(1)
setVibrationLed(True, 1)
```

connecting_to_network.py

```
import os
import subprocess
from registerNetworks import registerNetworks
import globals
import logging
import time
import datetime
from pathsLinux import *
from threading import Thread
import screen
from uploadVideos import upload_files
from connectingToCamera import Camera
import signal
import conImage_rtc_with_internet

"""Connecting to network class"""


class connecting_to_network:
    candidate_networks = []
    """Initial settings for connecting_to_network class"""
    def __init__(self, screen):

        self.counter_of_anomaly_on_camera = 0
        self.first_time_anomaly = 0
        self.curr_time_anomaly = 0
        self.rNM = registerNetworks()
        self.connectingCamera = Camera()
        self.timer = 30

        """Counter of Recording Video Mode"""
        self.threadForvideoShooting = Thread(target=self.start_timer_video_shooting,
args=(90,))
        self.threadForvideoShooting.setDaemon(True)

        logging.basicConfig(filename=log_url, level=logging.INFO, filemode='a')
```

```
    self.register_networks_to_connect = []

    if len(self.rNM.wifiNetworks) != 0:
        self.register_networks_to_connect = self.rNM.wifiNetworks
    else:
        return

    self.connect()

''' Find candidate networks in the preconImaged list'''
def find_candidate_networks(self):
    self.candidate_networks.clear()

    self.rNM.getAvailableNetworksUnix()

    self.available_networks_to_connect = self.rNM.availableNetworks

    available_network_ssids = self.rNM.availableNetworks.keys()

    for net in self.register_networks_to_connect.keys():
        for anet in available_network_ssids:
            if anet == net:
                self.candidate_networks.append(anet)

'''Error Handler for Camera malfunctioning'''
def start_timer_camera(self, timer):
    start_time = time.time()
    current_time = 0
    globals.camera_is_anomaly_detected = True
    screen.setErrorLed(globals.camera_is_anomaly_detected)
    while current_time - start_time < timer:
        current_time = time.time()
    globals.camera_is_anomaly_detected = False

    globals.again_vibration = False

'''Error Handler for Network malfunctioning'''
def start_timer_net(self, timer):
    start_time = time.time()
    current_time = 0
    globals.network_is_anomaly_detected = True
    screen.setErrorLed(globals.network_is_anomaly_detected)
    while current_time - start_time < timer:
        current_time = time.time()
    globals.network_is_anomaly_detected = False
    screen.setErrorLed(globals.network_is_anomaly_detected)

'''Timer responsible for the time in recording, for this example is 7 sec recording time'''
def start_timer_video_shooting(self, timer):
```

```
        while True:
            if globals.state_video_shooting == True:

                start_time = time.time()
                current_time = 0
                while current_time - start_time < timer:
                    current_time = time.time()

                    if globals.again_vibration == True:
                        start_time = time.time()
                        current_time = 0
                        globals.again_vibration = False

                    if globals.camera_is_anomaly_detected == True:
                        break

                globals.state_video_shooting = False
            else:
                pass

    '''Connecting to network function, responsible for every possible scenario that will happen in run time.
        Responsible for download, upload of videos, error handling and connection between networks.
    '''
    def connect(self):

        self.find_candidate_networks()

        currently_connected = self.rNM.alreadyConnectedNetwork()
        now = str(datetime.datetime.now())

        if globals.camera_is_anomaly_detected and self.counter_of_anomaly_on_camera == 0:
            self.first_time_anomaly = time.time()
            self.curr_time_anomaly = time.time()
            self.counter_of_anomaly_on_camera = 1
        elif globals.camera_is_anomaly_detected:
            self.curr_time_anomaly = time.time()
            if self.curr_time_anomaly - self.first_time_anomaly > self.timer + 5:
                globals.camera_is_anomaly_detected = False
                self.counter_of_anomaly_on_camera = 0
                self.first_time_anomaly = 0
                self.curr_time_anomaly = 0

        if currently_connected == False:

            screen.setMessage("Connection: -------    ")
            logging.info(now + " : " + "Connected to -------")
            screen.setCameraLed(False)
            screen.setWifiLed(False)
```

```
        else:
            screen.setMessage("Connection: " + str(currently_connected) + "       ")
            logging.info(now + " : " + "Connected to: " + str(currently_connected) + "     ")
            if (str(currently_connected) == globals.camera):
                globals.isConnectedToCamera = True
                globals.isConnectedToNet = False
            else:
                globals.isConnectedToCamera = False
                globals.isConnectedToNet = True

        screen.setTime()

        screen.setCameraLed(globals.isConnectedToCamera)
        screen.setWifiLed(globals.isConnectedToNet)
        screen.setErrorLed(globals.camera_is_anomaly_detected)

        string_of_candidate_networks = ""
        if len(self.candidate_networks) >= 1:
            string_of_candidate_networks = ' '.join(self.candidate_networks)
            for net in self.candidate_networks:
                if "Device" in net:
                    globals.camera = net

        if globals.wasVibrated and globals.camera in string_of_candidate_networks and
globals.camera_is_anomaly_detected == False:

            if currently_connected != globals.camera:
                command = "sudo nmcli -wait 10 con up " + globals.camera

                connection_result = subprocess.Popen(command, shell=True,
stdout=subprocess.PIPE,
                                                     stderr=subprocess.STDOUT)
                if_succesfull = str(connection_result.stdout.read().decode())

                if "successfully" not in if_succesfull or " The Wi-Fi network could not be found" in
if_succesfull:
                    now = str(datetime.datetime.now())
                    logging.info(now + " : " + "Error connecting to " + globals.camera + " for taking
video")

                    command = "sudo nmcli -wait 10 con up " + globals.camera
                    counter = 1
                    while counter < 3 and (
                            "successfully" not in if_succesfull or " The Wi-Fi network could not be
found" in if_succesfull):
                        connection_result = subprocess.Popen(command, shell=True,
stdout=subprocess.PIPE,
                                                             stderr=subprocess.STDOUT)
                        if_succesfull = str(connection_result.stdout.read().decode())
```

```
                logging.info(now + " : " + "Error connecting to " + globals.camera + " for
taking video")
                counter = counter + 1
            if "successfully" in if_succesfull:
                self.connectingCamera.connection_to_camera()
                globals.isConnectedToNet = False
                globals.wasVibrated = False
                globals.state_video_shooting = True
                if self.threadForvideoShooting.is_alive():
                    globals.again_vibration = True
                else:
                    self.threadForvideoShooting.start()


            else:
                globals.isConnectedToCamera = False
                globals.wasVibrated = False
                globals.state_video_shooting = False

            self.connect()
        else:
            self.connectingCamera.connection_to_camera()
            globals.state_video_shooting = True

            if self.threadForvideoShooting.is_alive():
                globals.again_vibration = True
            else:
                self.threadForvideoShooting.start()
            now = str(datetime.datetime.now())
            logging.info(now + " : " + "Connected to " + globals.camera + " for video
shooting")
            currently_connected = str(globals.camera)

            globals.isConnectedToNet = False

            globals.wasVibrated = False
    else:
        globals.state_video_shooting = True

        if self.threadForvideoShooting.is_alive():
            globals.again_vibration = True
        else:
            self.threadForvideoShooting.start()

        logging.info(now + " : " + "Connected to " + globals.camera + " for video
shooting")
        currently_connected = str(globals.camera)
        if globals.isConnectedToCamera == True and globals.isDownloading == True:
            globals.isConnectedToNet = False
            globals.isDownloading = False
```

```
            self.connectingCamera.connection_to_camera()
            globals.wasVibrated = False
        elif globals.isConnectedToCamera == True:
            self.connectingCamera.connect_to_camera_and_take_video()
            globals.isConnectedToNet = False

            globals.wasVibrated = False
        else:
            globals.isConnectedToNet = False
            self.connectingCamera.connection_to_camera()

            globals.wasVibrated = False

    if currently_connected == False and globals.state_video_shooting == False:
        globals.wasVibrated = False
        thisTime = time.time()
        while len(self.candidate_networks) == 0:

            if time.time() - thisTime > 5:
                print(time.time() - thisTime)
                break
            self.find_candidate_networks()

        for cnt, net in enumerate(self.candidate_networks):
            command = "sudo nmcli con up " + str(net)

            connection_result = os.popen(command).readlines()

            if len(connection_result) >= 1:
                now = str(datetime.datetime.now())
                if "successfully" not in connection_result[0]:
                    logging.info(now + " : " + "Error connecting to " + str(net))
                    if cnt == len(self.candidate_networks) - 1:
                        self.connect()
                else:
                    logging.info(now + " : " + "Connected to " + str(net))
                    currently_connected = str(net)
                    if "Device" in str(net):
                        globals.camera = str(net)
                    if currently_connected != globals.camera:
                        globals.isConnectedToCamera = False
                        globals.isConnectedToNet = True
                        screen.setMessage("Connection: " + str(currently_connected) + "      ")
                        screen.setCameraLed(globals.isConnectedToCamera)
                        screen.setWifiLed(globals.isConnectedToNet)

                    else:
                        globals.isConnectedToCamera = True
                        globals.isConnectedToNet = False
```

```
                    screen.setMessage("Connection: " + str(currently_connected) + "      ")
                    screen.setCameraLed(globals.isConnectedToCamera)
                    screen.setWifiLed(globals.isConnectedToNet)

                break

        if currently_connected != False and globals.state_video_shooting == False:
            globals.wasVibrated = False

            if currently_connected != globals.camera and globals.camera in
self.candidate_networks and globals.hasNewVideo and globals.camera_is_anomaly_detected
== False:
                command = "sudo nmcli -wait 15 con up " + globals.camera

                connection_result = subprocess.Popen(command, shell=True,
stdout=subprocess.PIPE,
                                                     stderr=subprocess.STDOUT)
                returncode = connection_result.wait()
                now = str(datetime.datetime.now())
                if_succesfull = str(connection_result.stdout.read().decode())

                if "successfully" not in if_succesfull or " The Wi-Fi network could not be found" in
if_succesfull:

                    logging.info(now + " : " + "Error connecting to " + globals.camera)

                    self.threadForAnomalyDetectionOnCamera =
Thread(target=self.start_timer_camera, args=(self.timer,))
                    self.threadForAnomalyDetectionOnCamera.setDaemon(True)
                    self.threadForAnomalyDetectionOnCamera.start()
                    self.connect()
                else:
                    logging.info(now + " : " + "Connected to " + globals.camera)

                    currently_connected = str(globals.camera)
                    globals.isConnectedToCamera = True
                    globals.isConnectedToNet = False
                    self.connectingCamera.connection_to_camera()
                    self.connectingCamera.connect_to_camera_and_download_video()

            else:
                now = str(datetime.datetime.now())
                if currently_connected == globals.camera and globals.hasNewVideo and
globals.camera_is_anomaly_detected == True:
                    globals.camera_is_anomaly_detected = False
                    command = "sudo nmcli con down " + globals.camera
                    connection_result = subprocess.Popen(command, shell=True,
stdout=subprocess.PIPE,
                                                         stderr=subprocess.STDOUT)
                    returncode = connection_result.wait()
```

```
                self.threadForAnomalyDetectionOnCamera =
Thread(target=self.start_timer_camera, args=(self.timer,))
                self.threadForAnomalyDetectionOnCamera.setDaemon(True)
                self.threadForAnomalyDetectionOnCamera.start()
                self.connect()

            elif currently_connected == globals.camera and globals.hasNewVideo:
                logging.info(
                    now + " : " + "Connected to " + currently_connected)
                globals.isConnectedToCamera = True
                globals.isConnectedToNet = False
                screen.setCameraLed(globals.isConnectedToCamera)
                screen.setWifiLed(globals.isConnectedToNet)

                self.connectingCamera.connection_to_camera()
                if globals.isConnectedToCamera == False:
                    pass
                else:
                    self.connectingCamera.connect_to_camera_and_download_video()

            elif currently_connected == globals.camera and globals.hasNewVideo == False:
                self.candidate_networks.remove(globals.camera)
                # print("trying to connect to net after camera")
                if len(self.candidate_networks) > 0:
                    for net in self.candidate_networks:
                        if net != globals.camera:
                            to_connect = net
                            self.candidate_networks.remove(net)
                        command = "sudo nmcli con up " + to_connect
                    connection_result = os.popen(command).readlines()
                    if "successfully" not in connection_result:
                        logging.info(now + " : " + "Error connecting to " + to_connect)
                    else:
                        logging.info(now + " : " + "Connected to " + to_connect)
                        currently_connected = str(net)
                        globals.isConnectedToCamera = False
                        globals.isConnectedToNet = True


                else:
                    pass



            else:
                now = str(datetime.datetime.now())
                if currently_connected != False and currently_connected != globals.camera and
globals.network_is_anomaly_detected == False and globals.isNewVideoOnDevice:
                    logging.info(
                        now + " : " + "Connected to " + currently_connected + ". You have access
```

```
to Network!!!")
                if globals.inTimeSetFromInternet == False:
                    if conImage_rtc_with_internet.configuring_rtc_with_internet() == True:
                        globals.inTimeSetFromInternet == True

                globals.isConnectedToCamera = False
                globals.isConnectedToNet = True
                screen.setMessage("Connection: " + str(currently_connected) + "      ")
                screen.setCameraLed(globals.isConnectedToCamera)
                screen.setWifiLed(globals.isConnectedToNet)
                try:
                    logging.info(now + " : " + "Uploading Videos...")
                    upload_files()
                except Exception as e:
                    command = "sudo nmcli con down " + currently_connected

                    connection_result = subprocess.Popen(command, shell=True,
stdout=subprocess.PIPE,
                                                         stderr=subprocess.STDOUT)
                    returncode = connection_result.wait()
                    self.threadForAnomalyDetectionOnNet =
Thread(target=self.start_timer_net, args=(10,))
                    self.threadForAnomalyDetectionOnNet.setDaemon(True)
                    self.threadForAnomalyDetectionOnNet.start()
                    screen.setWifiLed(globals.isConnectedToNet)

            else:

                pass
```

connectingToCamera.py

```
from goprocam import GoProCamera, constants
import os
import asyncio
import datetime
import globals
import logging
import ffmpeg
import sys
from pprint import pprint
import time
import screen
import subprocess
import shutil
```

```
from threading import Thread
from sys import platform

"""Import paths needed for identification"""
if platform == "linux" or platform == "linux2":
    from pathsLinux import *

elif platform == "win32":
    from paths import *

import signal

"'Looking for specific Camera"'
camera_mac_address = globals.camera_mac_address

"'Initializing Camera Class"'


class Camera:

    def __init__(self):
        self.duration = globals.time_duration

        logging.basicConfig(filename=log_url, level=logging.INFO, filemode='a')

    def connection_to_camera(self):

        try:
            """Initial step of creating child of Camera Class"""
            self.gpCam = GoProCamera.GoPro()

            if len(str(self.gpCam)) < 5:
                raise ConnectionError("No camera connection")
            """Configuring Settings of GO Pro"""

            self.folder = "100GOPRO"
            self.gpCam.syncTime()

            self.gpCam.gpControlSet(constants.Video.PROTUNE_VIDEO,
constants.Video.ProTune.ON)
            self.gpCam.video_settings("720p", "60")
            globals.isConnectedToCamera = True

            "'Handling Errors"'
        except ConnectionError:

            globals.isConnectedToCamera = False
            globals.camera_is_anomaly_detected = True
            screen.setMessage("Camera Failure")
            return
```

```
        except Exception as e:
            if "timed out" in str(e):
                globals.isConnectedToCamera = False
                globals.camera_is_anomaly_detected = True
                screen.setMessage("Camera Failure")
                return
            globals.camera_is_anomaly_detected = True
            globals.isConnectedToCamera = False
            screen.setMessage("Camera Failure")

            return

    """Finding all videos on Camera """

    def find_media_list(self) -> list:
        media = list(map(lambda x: [x[1], x[3]],
                         filter(lambda z: "MP4" in z[1], self.gpCam.listMedia(format=True,
media_array=True))))

        return media

    """Finding all photos on Camera """

    def find_media_list_jpg(self) -> list:
        media = list(map(lambda x: [x[1], x[3]],
                         filter(lambda z: "JPG" in z[1], self.gpCam.listMedia(format=True,
media_array=True))))

        return media

    """Delete all photos from Camera """

    def delete_jpeg(self, medialist):

        cnt = 0

        for media in medialist:

            if len(media) > 1:

                if ".JPG" in media[0]:
                    self.gpCam.deleteFile(self.folder, media[0])

                    cnt += 1

    """Download video from file and then delete it """

    def download_file(self, name, timestamp):

        duration = int(self.gpCam.getVideoInfo(option=constants.Info.Duration,
```

```
folder=self.folder, file=name))

        if duration >= 9:
            self.gpCam.deleteFile(self.folder, name)
            return
        self.gpCam.downloadMedia(self.folder, file=name)
        shutil.move(name, dir_of_videos + str(timestamp) + ".mp4")

        self.gpCam.deleteFile(self.folder, name)

    """Handler that sets a limit on download duration"""

    def timeout_handler(self, num, stack):
        print("Received SIGALRM")
        raise Exception("Video Downloading Error")

    def timeout_handler_recording(self, num, stack):
        raise Exception("Video Recording Error")

    """Downloading all the videos from camera"""

    def connect_to_camera_and_download_video(self):
        try:
            statements = []
            media = self.find_media_list()
            media_jpg = self.find_media_list_jpg()
            size_of_media = str(len(media))
            self.delete_jpeg(media_jpg)

            globals.isDownloading = True
            screen.setMessage("Downloading... 0/" + size_of_media)
            screen.setCameraLed(globals.isConnectedToCamera)
            screen.setWifiLed(globals.isConnectedToNet)

            signal.signal(signal.SIGALRM, self.timeout_handler)

            for i, video in enumerate(media):
                #               try:
                signal.alarm(40)
                if globals.wasVibrated == True:
                    screen.setMessage("Downloading stops.")
                    globals.isConnectedToCamera = True
                    globals.hasNewVideo = True
                    globals.state_video_shooting = True
                    return

                self.download_file(video[0], video[1])
                screen.setMessage("Downloading... " + str(i + 1) + "/" + size_of_media)
                now = str(datetime.datetime.now())
                logging.info(now + " : " + "Video " + str(video[0]) + " has been downloaded")
```

```
            globals.isNewVideoOnDevice = True

        screen.setMessage("End of Downloading")
        globals.isDownloading = False
        globals.hasNewVideo = False

        '''Handling Errors'''
    except Exception as e:
        print("current error: " + str(e))
        globals.camera_is_anomaly_detected = True
        globals.isConnectedToCamera = False
        globals.isDownloading = False
        globals.hasNewVideo = True
        screen.setMessage("Camera Failure")
        if "Video Downloading Error" in str(e):
            now = str(datetime.datetime.now())
            logging.info(now + " : " + "Error Downloading video from camera")
        return
    finally:
        signal.alarm(0)

"""Function responsible for recording video"""

def connect_to_camera_and_take_video(self):
    try:

        signal.signal(signal.SIGALRM, self.timeout_handler_recording)
        signal.alarm(self.duration + 12)
        globals.isRecording = True

        screen.setMessage("Recording...")
        self.gpCam.shoot_video(self.duration)
        now = str(datetime.datetime.now())
        logging.info(now + " : " + "Recording...")
        screen.setMessage("End of Recording")

        globals.hasNewVideo = True
        globals.isRecording = False
        '''Handling Errors'''

    except Exception as e:
        if "Video Recording Error" in str(e):
            if self.gpCam.IsRecording():
                self.gpCam.shutter(constants.stop)
            now = str(datetime.datetime.now())
            logging.info(now + " : " + "Error Recording video from camera")
            return
        globals.camera_is_anomaly_detected = True
        globals.isConnectedToCamera = False
```

```
            globals.isRecording = False
            globals.hasNewVideo = True
            screen.setMessage("Camera Failure")
            print(e)

            if "invalid literal" in str(e):
                command = "gatttool -t random -b " + camera_mac_address + " --char-write-req -a
0x2f -n 03010100"

                try:
                    processed = subprocess.check_output(command, shell=True,
universal_newlines=True)
                    if "successfully" in processed:
                        print("success")

                    return
                except Exception as e:
                    print(e)
                    print("timeout")
                    return


        finally:
            signal.alarm(0)
```

10.2 Λογισμικό τοπικού διακομιστή για Rest Api

Main.py

```
import os
from app import app
import urllib.request
from flask import Flask, flash, request, redirect, url_for, render_template, Response
from werkzeug.utils import secure_filename
import socket

gw = os.popen("ip -4 route show default").read().split()
s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
s.connect((gw[2], 0))
ipaddr = s.getsockname()[0]


@app.route('/')
def upload_form():
```

```
    return render_template('upload.html')


'''Post request for video upload'''


@app.route('/', methods=['POST'])
def upload_video():
    try:
        if 'file' not in request.files:
            flash('No file part')

            return redirect(request.url)
        file = request.files['file']
        if file.filename == '':
            flash('No image selected for uploading')
            print("am here2")
            return redirect(request.url)
        else:
            filename = secure_filename(file.filename)
            print('upload_video filename: ' + filename)
            file.save(os.path.join(app.config['UPLOAD_FOLDER'], filename))
            print('upload_video filename: ' + filename)
            return Response("{'Video':'Succesfull'}", status=200, mimetype='application/json')

    except Exception as e:
        print(e)


'''Get request for displaying video'''


@app.route('/display/<filename>')
def display_video(filename):
    return redirect(url_for('static', filename='video/' + filename), code=301)


'''Execute flask api'''
if __name__ == "__main__":
    app.run(host=ipaddr, port=5002)
```

app.py

```
from flask import Flask
from flask_cors import CORS,cross_origin
```

```
UPLOAD_FOLDER = '/home/pi/Desktop/device_upload/static/video'

app = Flask(__name__)
CORS(app)
app.secret_key = "secret key"
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER
app.config['MAX_CONTENT_LENGTH'] = 16 * 1024 * 1024* 10
```

10.3 Λογισμικό τοπικού διακομιστή για επεξεργασία βίντεο και προσθήκη λεζάντας

GoPro_timestamp_FFMPEG_3.0.py

```
import os
import subprocess
import math
import sys
import time

import cv2, time, threading
import datetime as dt
import subprocess as sp
import json
from random import randrange
import logging

from create_jsons_from_csv import generate_json

from sys import platform

if platform == "linux" or platform == "linux2":
   from pathsLinux import *

   print("linux")

elif platform == "win32":
   from paths import *

   print("win32")
list_of_videos = []
logging.basicConfig(filename="/home/pi/Desktop/device_video_editing/edit_logger.log",
level=logging.INFO, filemode='a')
```

```
# Creating Json file from CSV
def make_the_json_from_file(file, out_name, my_date):
    """Check for binary data"""
    now = str(dt.datetime.now())
    command = "ffprobe " + dir_of_videos + file

    connection_result = subprocess.Popen(command, stdout=subprocess.PIPE,
stderr=subprocess.PIPE, shell=True)

    connection_result = connection_result.communicate()

    if "bin_data" not in connection_result[1].decode():
        raise cv2.error
    else:
        """Create Bin file"""
        command = "ffmpeg -y -loglevel panic -i " + dir_of_videos + file + " -codec copy -map 0:3
-f rawvideo " + dir_of_binary + \
                out_name.split(".")[0] + ".bin"
        connection_result = os.popen(command)
        # print(connection_result)
        # print("pass2")
        time.sleep(1)
        """Create csv files"""
        if platform == "linux" or platform == "linux2":
            command = "go run " + go_location + "/gpmd2csv.go -i " + dir_of_binary +
out_name.split(".")[
                0] + ".bin" + " -o " + dir_of_csvs + out_name.split(".")[0] + ".csv"

            connection_result = subprocess.Popen(command, stdout=subprocess.PIPE,
stderr=subprocess.PIPE, shell=True)

            connection_result = connection_result.communicate()
            logging.info(now + " : " + "Bin file of " + str(file) + " video, extracted")

        elif platform == "win32":
            command = go_location + "gpmd2csv.exe -i " + dir_of_binary + out_name.split(".")[
                0] + ".bin" + " -o " + dir_of_csvs + out_name.split(".")[0] + ".csv"
            # connection_result = os.popen(command)
            connection_result = subprocess.Popen(command, stdout=subprocess.PIPE,
stderr=subprocess.PIPE, shell=True)
            connection_result = connection_result.communicate()
        return generate_json(out_name.split(".")[0], my_date, now)


# Generating the data needed for rendering of video (Latitude,Longitude,Timestamp)
```

```
def make_data_overlay_list(size, file_name):
    try:
        fj = open(file_name)

    except FileNotFoundError:
        print("File does not exist")

        return False
    except Exception:
        print("File does not exist")
        return False

    data = json.load(fj)
    mylist = data["vibration"]
    text_list_data = []

    if len(mylist) == 0:
        return False
    if int(size) > len(mylist):
        num_repeats = int(size / len(mylist))

        extended_data = [val for val in mylist for _ in range(num_repeats)]
        while len(extended_data) < size:
            index = randrange(len(extended_data) - 1)
            extended_data.insert(index, extended_data[index])
        for entry in extended_data:
            text_list_data.append(
                "latitude: " + str(entry['latitude']) + " longitude: " + str(entry['longitude']) + "
altitude: " + str(
                    entry['altitude']))
        return text_list_data
    elif int(size) < len(mylist):

        times_in_list = int(math.ceil(len(mylist) / int(size)))
        shrinked_table = []
        for i, item in enumerate(mylist):
            if i % times_in_list == 0:
                shrinked_table.append(item)

        while len(shrinked_table) < size:
            index = randrange(1, len(shrinked_Πίνακας) - 1)
            shrinked_table.insert(index, shrinked_Πίνακας[index - 1])
        for entry in shrinked_table:
            text_list_data.append(
                "latitude: " + str(entry['latitude']) + " longitude: " + str(entry['longitude']) + "
altitude: " + str(
                    entry['altitude']))
```

```
        return text_list_data
    else:
        for entry in mylist:
            text_list_data.append(
                "latitude: " + str(entry['latitude']) + " longitude: " + str(entry['longitude']) + "
altitude: " + str(
                    entry['altitude']))
        return text_list_data


def drain_stderr():
    while keep_drain_stderr:
        try:
            stderr_output = pipe.stderr.readline()
        except:
            pass


'''Every 6 minutes the below is executing. Is responsible for reconstructing the videos with
timestamp and
coordinates '''


def creation_time(filename):
    time1 = filename.split(".")[0].split("_")[2]
    return time1


# Opens the video import and sets parameters

while True:

    for file in os.listdir(dir_of_videos):
        now = str(dt.datetime.now())
        if file.endswith(".mp4"):
            if "out" in file:
                continue
            list_of_videos.append(file)
            logging.info(now + " : " + "Video " + str(file) + " ready for processing.")

            print("list of videos: " + str(list_of_videos))

    for filename in list_of_videos:
        videoWithDir = dir_of_videos + filename
        device_index = "_".join(filename.split("_")[0:2])
        video = None
        unix_time = None
```

```
        json_file = None
        now = str(dt.datetime.now())
        try:
            video = cv2.VideoCapture(videoWithDir)

            t = creation_time(filename)
            unix_time = int(t)
            initial = dt.datetime.utcfromtimestamp(unix_time).strftime("%Y-%m-%d
%H:%M:%S")
            initial = dt.datetime.strptime(initial, "%Y-%m-%d %H:%M:%S")
            Output_name = dir_of_processed_video + device_index + "_" + (str(initial).replace("-
", "_")).replace(" ",
                                                                        "_").replace(
                ":", "_") + '_out' + filename[-4:]

            json_file = make_the_json_from_file(filename,
Output_name.replace(dir_of_processed_video, ''), initial)

            if video is None or not video.isOpened():
                raise cv2.error

        except cv2.error as error:
            print("this is video error: " + videoWithDir)
            os.remove(videoWithDir)
            logging.error(now + " : " + "Error opening " + str(filename) + " video.")
            continue
        except Exception as e:
            print("this is video error1: " + str(e))
            logging.error(now + " : " + "Error opening " + str(filename) + " video.")
            os.remove(videoWithDir)
            continue


        # Checks to see if a the video was properly imported
        status = video.isOpened()


        if status == True:
            FPS = video.get(cv2.CAP_PROP_FPS)
            width = video.get(cv2.CAP_PROP_FRAME_WIDTH)
            height = video.get(cv2.CAP_PROP_FRAME_HEIGHT)
            size = (int(width), int(height))
            total_frames = video.get(cv2.CAP_PROP_FRAME_COUNT)
            frame_lapse = (1 / FPS) * 1000

            # Initializes time origin of the video
```

```
datalist = make_data_overlay_list(total_frames, json_file)

if datalist == False:
   video.release()
   os.remove(videoWithDir)
   continue

timestamp = initial

# Initializes the frame counter
current_frame = 0
start = time.time()

# Command to send via the command prompt which specifies the pipe parameters
command = ['ffmpeg',
        '-y',  # (optional) overwrite output file if it exists
        '-f', 'rawvideo',  # Input is raw video
        '-vcodec', 'rawvideo',
        '-pix_fmt', 'bgr24',  # Raw video format
        '-s', str(int(width)) + 'x' + str(int(height)),  # size of one frame
        '-r', str(FPS),  # frames per second
        '-i', '-',  # The input comes from a pipe
        '-i', videoWithDir,
        '-vcodec', 'mpeg4',
        '-b:v', '10M',  # Sets a maximum bit rate
        Output_name]

# Open the pipe
pipe = sp.Popen(command, stdin=sp.PIPE, stderr=sp.PIPE)

keep_drain_stderr = True
thread = threading.Thread(target=drain_stderr)
thread.start()



# Reads through each frame, calculates the timestamp, places it on the frame and
# exports the frame to the output video.
while current_frame < total_frames:
   success, image = video.read()
   elapsed_time = video.get(cv2.CAP_PROP_POS_MSEC)
   current_frame = video.get(cv2.CAP_PROP_POS_FRAMES)
   timestamp = initial + dt.timedelta(microseconds=elapsed_time * 1000)
   cv2.putText(image, 'Date: ' + str(timestamp)[0:10], (50, int(height - 100)),
           cv2.FONT_HERSHEY_COMPLEX_SMALL, 1, (255, 255, 255), 1)
   cv2.putText(image, 'Time: ' + str(timestamp)[11:-4], (50, int(height - 75)),
           cv2.FONT_HERSHEY_COMPLEX_SMALL, 1, (255, 255, 255), 1)
```

```
            cv2.putText(image, datalist[int(current_frame) - 1], (50, int(height - 50)),
                        cv2.FONT_HERSHEY_COMPLEX_SMALL, 1, (255, 255, 255), 1)
            try:
               pipe.stdin.write(image.tobytes())
            except Exception as e:
               print(e)

         keep_drain_stderr = False

         video.release()
         pipe.stdin.close()
         pipe.stderr.close()

         # Calculate how long the timestampping took
         duration = (time.time() - float(start)) / 60

         # print("Video " +filename+" has been timestamped")
         logging.info(now + " : " + "Video " + filename + "(" + str(initial) + ")" + " has been
timestamped")
         # print('This video took:' + str(duration) + ' minutes')
         try:
            pipe.wait(3)
         except (sp.TimeoutExpired):
            pipe.kill()

         thread.join()
         os.remove(videoWithDir)

      else:
         print('Error: Could not load video')
   time.sleep(600)
```

sendJsonfile.py

```
import requests
from pathsLinux import *
import subprocess
def postRequest(data,json_file):
   url="http://142.11.210.23:5000/v0/vibration"
   x = requests.post(url, json=data,auth=("DeviceAdmin","DeviceV0Pass0"))
   #print(x.text)
   if (x.text) != "Success":
      pass
```

```
    else:

        command = "mv "+json_file+"  " + dir_of_jsons_uploaded
        #connection_result = os.popen(command).read()

        connection_result = subprocess.Popen(command, stdout=subprocess.PIPE,
stderr=subprocess.PIPE, shell=True)
        #print("pass")
        connection_result=connection_result.communicate()
```

11 Παράρτημα Β

Σε αυτό το κεφάλαιο θα παρουσιαστεί το λογισμικό που δημιουργήθηκε για την ανάλυση των βίντεο έτσι ώστε να επισημανθούν οι φθορές στο οδόστρωμα. Πιο αναλυτικά, γίνεται φιλτράρισμα και αφαίρεση φθορών από κάθε Frame ξεχωριστά και εύρεση λακκούβων. Το αποτέλεσμα είναι ένα καινούργιο βίντεο που υποδεικνύει της λακκούβες με περιγράμματα και τις μετρά.

Frame_extraction_per_video.py

```
import cv2
import os


'''frame extraction per video and cropping 1/3 of the frame on top'''
for filename in os.listdir(os.getcwd() + "\\video"):
  # if "1667393562" in filename:

      directory_name = filename.split(".")[0]
      if not os.path.exists(os.getcwd()+"\\frames_per_video\\"+directory_name):
         os.mkdir(os.getcwd()+"\\frames_per_video\\"+directory_name)

      vidcap = cv2.VideoCapture(os.getcwd() + "\\video\\"+filename)
      frame_count = vidcap.get(5)
      print('Frame count : '+str(frame_count))
      success, image = vidcap.read()
      count = 0
      while success:
         if count<10:
            count="0"+str(count)
         try:

            height, width, alpha = image.shape
            height_cutoff = height // 3
            s1 = image[:height_cutoff,:]
            s2 = image[height_cutoff:,:]

cv2.imwrite(os.getcwd()+"\\frames_per_video\\"+directory_name+"\\frame%s.jpg" %
str(count), s2)  # save frame as JPEG file
            count=int(count)
         except Exception as e :
            print(e)
            print("ERROR")
         success, image = vidcap.read()
```

```
            count += 1
            if count == 53:
                success=True
        vidcap.release()
```

shadow_remover.py

```
import cv2
import numpy as np
import os

pathe_original="C:\\Users\\piacovou\\Desktop\\thesis\\thesis-
Msc\\opencv_code\\frames_per_video\\"
pathe_shadow="C:\\Users\\piacovou\\Desktop\\thesis\\thesis-
Msc\\opencv_code\\shadow_remover_frame\\"


all_directories_with_frames=[x[0] for x in os.walk(pathe_original)][1:]
'''Shadow remover Function '''
for directory in all_directories_with_frames:
    if "1666178698" in directory:
        pass
    else:
        if not os.path.exists(pathe_shadow+directory.split("\\")[-1]):
            os.mkdir(pathe_shadow+directory.split("\\")[-1])
        print(directory)
        for filename in os.listdir(directory):
            img = cv2.imread(directory+"\\"+filename,-1)
            #split colors of images
            rgb_planes = cv2.split(img)
            result_planes = []
            result_norm_planes = []
            for plane in rgb_planes:
                #filtering image per color
                dilated_img = cv2.dilate(plane, np.ones((7, 7), np.uint8))
                bg_img = cv2.medianBlur(dilated_img, 21)
                diff_img = 255 - cv2.absdiff(plane, bg_img)
                norm_img = cv2.normalize(diff_img, None, alpha=0, beta=255,
norm_type=cv2.NORM_MINMAX, dtype=cv2.CV_8UC1)
                result_planes.append(diff_img)
                result_norm_planes.append(norm_img)


            result = cv2.merge(result_planes)
            result_norm = cv2.merge(result_norm_planes)
```

```
        cv2.imwrite(pathe_shadow + directory.split("\\")[-1] + "\\" + filename, result_norm)
```

frame_edges_and_contours.py

```
import cv2
import numpy as np
import os

pathe_original="C:\\Users\\piacovou\\Desktop\\thesis\\thesis-
Msc\\opencv_code\\frames_per_video\\"
pathe_edges="C:\\Users\\piacovou\\Desktop\\thesis\\thesis-
Msc\\opencv_code\\edge_frames\\"
pathe_contour="C:\\Users\\piacovou\\Desktop\\thesis\\thesis-
Msc\\opencv_code\\contour_frames\\"
pathe_shadow="C:\\Users\\piacovou\\Desktop\\thesis\\thesis-
Msc\\opencv_code\\shadow_remover_frame\\"


font = cv2.FONT_HERSHEY_SIMPLEX
fontScale = 4
# Blue color in BGR
color = (0, 255, 255)

# Line thickness of 2 px
thickness = 2
# all_directories_with_frames=[x[0] for x in os.walk(pathe_original)][1:]
all_directories_with_frames=[x[0] for x in os.walk(pathe_shadow)][1:]
'''Calculation of edges per frame'''
for directory in all_directories_with_frames:

    if not os.path.exists(pathe_edges+directory.split("\\")[-1]):
        os.mkdir(pathe_edges+directory.split("\\")[-1])
    if not os.path.exists(pathe_contour+directory.split("\\")[-1]):
        os.mkdir(pathe_contour+directory.split("\\")[-1])
    for filename in os.listdir(directory):
        im = cv2.imread(directory+"\\"+filename)

        gray1 = cv2.cvtColor(im, cv2.COLOR_BGR2GRAY)

        # CONTOUR DETECTION CODE
        imgray = cv2.cvtColor(im, cv2.COLOR_BGR2GRAY)

        ret, thresh = cv2.threshold(imgray, 50, 255, 0)

        contours2, hi = cv2.findContours(thresh, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
```

```
hi = hi[0]
img_for_rectangles = im.copy()

img2 = im.copy()

# guassian blur
gblur = cv2.GaussianBlur(im, (5, 5), 0)
# median
median = cv2.medianBlur(im, 5)
# erosion
kernel = np.ones((5, 5), np.uint8)
erosion = cv2.erode(median, kernel, iterations=1)

dilation = cv2.dilate(erosion, kernel, iterations=2)

# canny edge detection
edges = cv2.Canny(dilation, 9, 220)
cv2.imwrite('diff.jpg', im)
cv2.imwrite('diff.jpg', edges)
x = edges.shape[0]
y = edges.shape[1]
y_1_3 = y / 3
x_1_4 = x / 4
y_9_10 = y * 5 / 10
x_1_8 = x / 15
x_original = y
y_original = x
for i, item in enumerate(edges):
    item = item * 0
    edges[i] = item
    if i > x_1_4:
        break


contours1, hi1 = cv2.findContours(edges, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
counter=0

for c in contours1:
    x, y, w, h = cv2.boundingRect(c)
    if x > x_original - (x_original / 5):
        continue
    if w > 5 and h > 10 and w<400 and h<300:
        cv2.rectangle(img2, (x, y), (x + w, y + h), (255, 155, 0), 2)
        counter +=1
if counter <25:
    color=(0, 255, 0)
elif counter >= 25 and counter<50:
```

```
            color = (0, 165, 255)
        elif counter >= 50 and counter<75:
            color = (0, 100, 255)
        elif counter >= 75:
            color = (0, 0, 255)

        cv2.putText(img2, str(counter), (50,150), font,
                    fontScale, color, thickness, cv2.LINE_AA)
        print(str(filename)+": "+str(counter))


        cv2.imwrite(pathe_edges + directory.split("\\")[-1] + "\\" +
filename.split(".")[0].split("me")[1] + "." +
                    filename.split(".")[1], img2)
```