ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΕΠΙΚΟΙΝΩΝΙΩΝ
ΤΜΗΜΑ ΨΗΦΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ
«ΨΗΦΙΑΚΕΣ ΕΠΙΚΟΙΝΩΝΙΕΣ & ΔΙΚΤΥΑ»

Ανάπτυξη Συνεργατικού Συστήματος Ασφάλειας Γειτονιάς

Διπλωματική Εργασία

Βερβίτας Γεώργιος

Επιβλέπουσα: Αλεξίου Αγγελική
Αναπληρώτρια Καθηγήτρια Τμήματος Ψηφιακών Συστημάτων Πανεπιστημίου Πειραιώς

Πειραιάς, 2018

.....
Βερβίτας Γεώργιος
Πτυχιούχος Μηχανικός Ηλεκτρονικών Υπολογιστικών Συστημάτων ΤΕΙ Πειραιά

Στη γυναίκα μου, Βασιλική

Ευχαριστίες

Η παρούσα πτυχιακή υλοποιήθηκε σε συνεργασία με την εταιρεία COSMOTE Κινητές Τηλεπικοινωνίες Α.Ε., υπό την επίβλεψη του κου Λυμπερόπουλου Γιώργου, υπεύθυνου του R&D του Ομίλου ΟΤΕ, που με την πρότασή του για το παρόν θέμα, την καθοδήγησή του και την αμέριστη υποστήριξη, τόσο του ιδίου, όσο και του συνεργάτη του κου Θωμά Πάζιου συνετέλεσαν ώστε να υλοποιηθεί το καινοτόμο αυτό εγχείρημα. Επιπλέον θα ήθελα να τονίσω ότι ο υλικο-τεχνολογικός εξοπλισμός της παρούσης εργασίας ήταν εξ' ολοκλήρου παροχή του εργαστηρίου R&D του Ομίλου ΟΤΕ, με επικεφαλής τον κ. Λυμπερόπουλο. Συνεχίζοντας, θα ήθελα να ευχαριστήσω την επιβλέπουσα Αναπληρώτρια Καθηγήτρια του Π.Μ.Σ. "Ψηφιακές Επικοινωνίες και Δίκτυα" Αλεξίου Αγγελική για την υποστήριξή της στο παρών εγχείρημα αλλά και τον κο Γκότση Αντώνη, Μεταδιδάκτορα του Τμήματος Ψηφιακών Συστημάτων και Διδάσκοντα του Π.Μ.Σ. που εκτός από την υποστήριξή του έδωσε και το αρχικό έναυσμα για την συνεργασία με την COSMOTE.

Περίληψη

Μέσα από την αλματώδη ανάπτυξη της τεχνολογίας, έχουμε οδηγηθεί στην παραγωγή τερματικών συσκευών προδιαγραφών IoT (χαμηλής κατανάλωσης ενέργειας, μικρού βάρους/όγκου, απλοποιημένης μορφής και μεγάλης αξιοπιστίας) με υψηλή υπολογιστική ισχύ -αντίστοιχης μεγαλύτερων μονάδων υπολογιστών - διευρύνοντας τα όρια χρήσης σε εφαρμογές που μέχρι σήμερα απαιτούσαν σύνθετες υπολογιστικές μονάδες ελέγχου.

Ένας τομέας που απαιτεί την εναρμονισμένη συνεργασία όλων των περιφερειακών συστημάτων ελέγχου και έχει μεγάλο ποσοστό διείσδυσης - από πλευράς υλοποιήσεων/εφαρμογών - στον πληθυσμό [1] [2] είναι η ασφάλεια/παρακολούθηση χώρων. Μέχρι σήμερα το μεγαλύτερο ποσοστό τέτοιων υλοποιήσεων [3] έβρισκε εφαρμογή σε κάθε χώρο ξεχωριστά και υπήρχε η δυνατότητα ενημέρωσης – σε περίπτωση παραβίασης – της κεντρικής υπηρεσίας ελέγχου. Οι πρώτες υλοποιήσεις είχαν μόνο αισθητήρες ελέγχου που τοποθετούνταν σε κρίσιμα σημεία του εκάστοτε χώρου και παρείχαν ενημέρωση για παραβίαση μόνο τοπικά (πχ σειρήνα). Μετέπειτα έγινε διασύνδεση των συστημάτων μέσω τηλεφωνικής γραμμής (POTS,GSM) με κέντρο ελέγχου για λήψη απλού σήματος ειδοποίησης σε περίπτωση alarm και ενημέρωσης ιδιοκτήτη/αρχών. Τέλος, προστέθηκαν κάμερες για παρακολούθηση - ανεξάρτητα του alarm – από τους ιδιοκτήτες και καταγραφή τους σε τοπικό καταγραφικό.

Μέσω της παρούσης διπλωματικής, υλοποιήθηκε ένα καινοτόμο σύνθετο σύστημα ασφάλειας [4] για μια μεγάλη περιοχή (π.χ. δήμος, γειτονιά) οριζόμενη από το κέντρο ελέγχου, όπου τα εσωτερικά συστήματα παρακολούθησης που είναι τοποθετημένα σε κάθε οίκημα (κάμερες, PIR) διασυνδέονται με εξωτερικά συστήματα παρακολούθησης (κινούμενες κάμερες τοποθετημένες σε κατάλληλα σημεία του δρόμου, όπως κολώνες φωτισμού), ενώ πραγματοποιείται κεντρικοποιημένη καταγραφή (server, cloud) με χρήση πρωτοκόλλων και συσκευών IoT που ελαχιστοποιούν την κατανάλωση πόρων (ενέργειας, bandwidth).

Λέξεις-κλειδιά:

Ασφάλεια, IoT, Raspberry Pi, GPIO, MQTT, OpenCV

Abstract

Based on the technology development nowadays, we are producing terminal devices according to IoT specifications (low power consumption, lightweight, small, simple and reliable) with high computing power - like complex computing systems - allowing us to use them in complex applications.

One of the sectors that demands cooperation of all system peripherals and have high impact - and many implementations - on people is area/home security. Till now, there was a separated security system in each area and in case of invasion only a central audit service was informed. Furthermore, the first implementations had only control sensors placed in crucial spots of the area and the information of invasion was only in local area (e.g. sirens). Moreover, all the systems interconnected through telephone lines (POTS, GSM) to the control center so that the authorities/owner get informed in case of invasion. More recently implementations have cameras, so that the owner can watch/record (locally) the area.

In the context of this study, an innovative complex security system has been created to cover a wide area according to the needs of the control center (e.g. municipality, neighborhood) which has all the internal alarm peripherals (cameras, PIR) interconnected with external ones (moving cameras at the street level) while having centralized control (server, cloud) by using IoT devices and protocols thus minimizing the use of resources (energy, bandwidth).

Keywords:

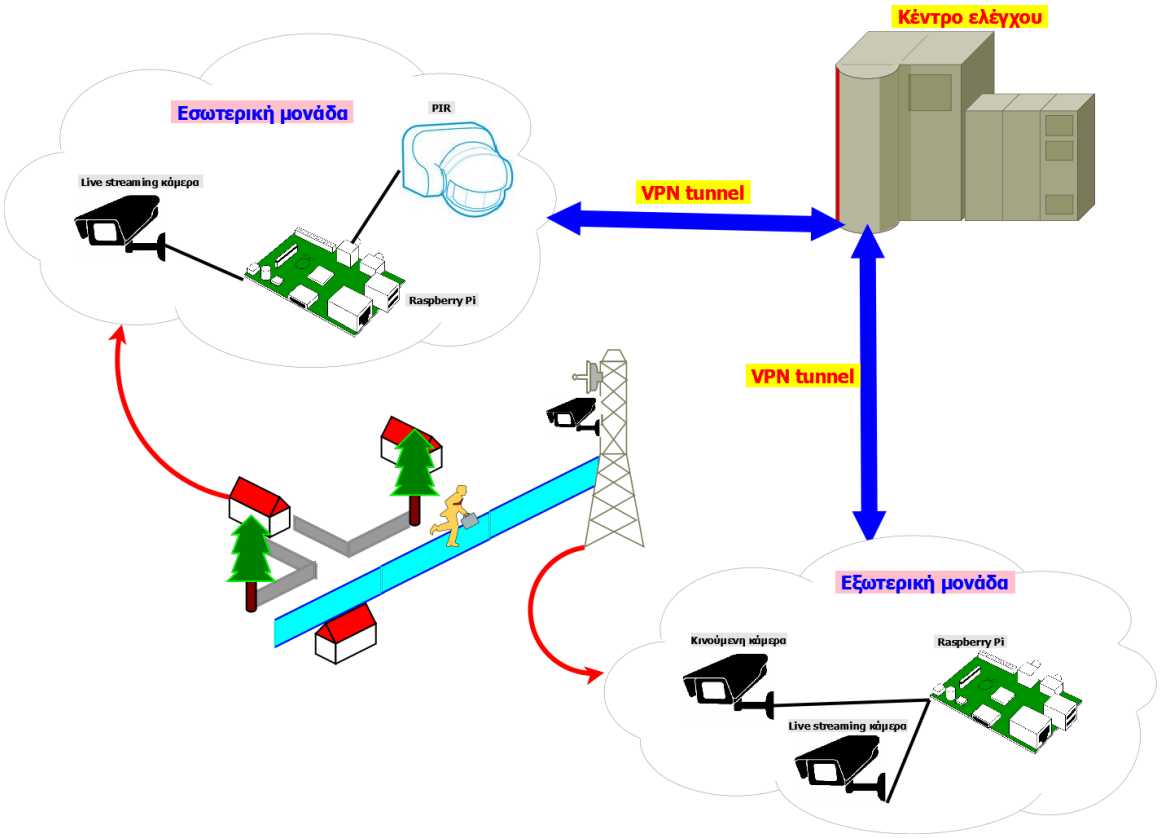
Security, IoT, Raspberry Pi, GPIO, MQTT, OpenCV

Πίνακας περιεχομένων

Κεφάλαιο 1 Εισαγωγή

1.1 Σκοπός της διπλωματικής εργασίας

Σκοπός της παρούσης μεταπτυχιακής διπλωματικής εργασίας είναι η ανάπτυξη συνεργατικού συστήματος ασφαλείας μιας ευρείας περιοχής (πχ γειτονιάς), βασιζόμενο στις προδιαγραφές της τεχνολογίας IoT (συσκευές και πρωτόκολλα) [5] ελεγχόμενη κεντρικά (πχ δήμος, γειτονιά). Το σύστημα ασφαλείας θα έχει διασυνδεδεμένα τα εσωτερικά συστήματα παρακολούθησης (κάμερες, PIR) που υπάρχουν σε κάθε οίκημα με εξωτερικά συστήματα παρακολούθησης (κινούμενες κάμερες) και θα διαθέτει κεντρικοποιημένη καταγραφή (server, cloud) κάνοντας χρήση πρωτοκόλλων και συσκευών IoT ελαχιστοποιώντας την κατανάλωση πόρων (ενέργειας, bandwidth). Το ολοκληρωμένο διάγραμμα της υλοποίησης έχει ως εξής:



Αναλυτικότερα το σύστημα αποτελείται:

Την εσωτερική μονάδα ελέγχου, η οποία θα εγκαθίσταται στους χώρους ελέγχου του τελικού χρήστη (πχ οίκημα) και θα είναι υπεύθυνη για τον έλεγχο παραβίασης αυτών και την αποστολή ενημερώσεων μέσω του δικτύου προς τον τελικό χρήστη αλλά και το κέντρο ελέγχου. Συγκεκριμένα θα αποτελείται από την κύρια υπολογιστική μονάδα -Raspberry Pi- που θα συνδέεται με το τοπικό οικιακό δίκτυο (μέσω ethernet στην παρούσα εργασία) για πρόσβαση στο Internet και πάνω του θα συνδέονται η σταθερή κάμερα καταγραφής του χώρου και ένας (ή περισσότεροι) ανιχνευτές κίνησης (PIR) που θα τοποθετούνται στα

σημεία ελέγχου. Το όλο σύστημα θα τροφοδοτείται από τροφοδοτικό που συνδέεται στο Raspberry και αυτό αναλαμβάνει να τροφοδοτήσει τα επιμέρους στοιχεία.

Η εσωτερική μονάδα συνδέεται με το κέντρο ελέγχου μέσω Internet κάνοντας χρήση VPN tunnel για ασφάλεια δεδομένων (κρυπτογράφηση μέσω OPENVPN πρωτοκόλλου).

Σε περίπτωση εντοπισμού κίνησης -alarm- από το PIR, η μονάδα στέλνει ενημέρωση στο κέντρο ελέγχου κάνοντας χρήση του πρωτοκόλλου MQTT ενημερώνοντας το topic που της έχει ανατεθεί (τα δεδομένα έχουν μορφή JSON stream) και παράλληλα αποστέλλει push notification στο κινητό του τελικού χρήστη για τον αύξοντα αριθμό της μονάδας (σε περίπτωση που έχει περισσότερες της μιας μονάδας ακόμα και σε ξεχωριστούς χώρους) και το link για το live streaming από την εν λόγω εσωτερική κάμερα. Επιπλέον των ενημερώσεων, εκκινεί υπηρεσία -MotionEye- για παροχή live streaming και ελέγχου της κάμερας (500 Kbps at 320*240), μέσω web interface, αναλαμβάνοντας να κάνει και τον έλεγχο εντοπισμού κίνησης από την κάμερα και σε περίπτωση εντοπισμού κίνησης (από την κάμερα) αποθηκεύει τόσο τοπικά όσο και στο cloud (Dropbox) εικόνες & video. Το alarm παραμένει ενεργό για αρκετή ώρα από την στιγμή που θα λάβει χώρα (πχ 1 ώρα) ώστε να είναι ενεργό και το streaming από την μονάδα.

Τέλος, στον τελικό χρήστη παρέχεται εφαρμογή για android που συνδέεται με την/τις μονάδες ώστε να μπορεί να κάνει arming/disarming και σε περίπτωση alarm να μπορεί να επιλέξει την επιθυμητή κάμερα (σε περίπτωση που έχει περισσότερες από μια) για δει το live streaming

Την εξωτερική μονάδα ελέγχου, η οποία θα εγκαθίσταται στους εξωτερικούς χώρους που θα ορίζεται από την αρμόδια υπηρεσία (πχ σε κολώνες φωτισμού) και θα πρέπει να καλύπτουν, στο σύνολό τους, την περιοχή που θέλουν να προστατεύσουν (πχ γειτονιά ή περιφέρεια) και έχουν ως σκοπό όταν συμβεί alarm από κάποια εσωτερική μονάδα να ξεκινήσουν την αναζήτηση προσώπων προς την κατεύθυνση που έλαβε χώρα το συμβάν αλλά και να καταγράψουν γενικά κίνηση.

Συγκεκριμένα θα αποτελείται από την κύρια υπολογιστική μονάδα -Raspberry Pi- που θα συνδέεται με το δημόσιο δίκτυο (ethernet στην παρούσα εργασία-αλλά σε μια πρακτική υλοποίηση αναμένεται να χρησιμοποιηθεί 3G/4G σύνδεση) για πρόσβαση στο Internet και πάνω του θα συνδέονται τόσο η σταθερή κάμερα καταγραφής της περιοχής αλλά και η κινούμενη κάμερα εντοπισμού και παρακολούθησης προσώπων. Το όλο σύστημα θα τροφοδοτείται από τροφοδοτικό που συνδέεται στο Raspberry και αυτό αναλαμβάνει να τροφοδοτήσει τα επιμέρους στοιχεία.

Σε περίπτωση εντοπισμού κίνησης -alarm- από κάποια εσωτερική μονάδα, ενημερώνεται από την αλλαγή στο topic (μέσω του MQTT broker) της συγκεκριμένης μονάδας και στρέφει (αν δύναται) την κινούμενη κάμερα στην κατεύθυνση που είναι η εσωτερική μονάδα που σήκωσε το alert και ελέγχει για εντοπισμό προσώπων. Όταν τα εντοπίσει, τα ακολουθεί -tracking- και τα καταγράφει σε εικόνες με μαρκάρισμα του προσώπου και καταγραφή ημερομηνίας/ώρας τις οποίες και αποθηκεύει τοπικά με παράλληλη μετατροπή τους σε live streaming (mjpeg) (~80 Kbps) το οποίο σηκώνει σε δικό του web server για live streaming από το κέντρο ελέγχου.

Η σταθερή κάμερα, όπως και στην εσωτερική μονάδα, όταν εντοπίσει γενικά κίνηση αντικειμένων τα καταγράφει τόσο τοπικά όσο και στο cloud (Dropbox) σε εικόνες & video για να είναι διαθέσιμα στο κέντρο ελέγχου.

Τέλος, παρόμοια με την εσωτερική μονάδα, συνδέεται με το κέντρο ελέγχου μέσω ίντερνετ κάνοντας χρήση VPN tunnel για ασφάλεια δεδομένων (κρυπτογράφηση μέσω OPENVPN πρωτοκόλλου).

Το κέντρο ελέγχου, το οποίο θα χειρίζεται η αρμόδια υπηρεσία (πχ δημοτική αστυνομία) και θα λαμβάνει τις ενημερώσεις για alert από τις εσωτερικές μονάδες. Διαθέτει κονσόλα ελέγχου που εμφανίζει λίστα με τις εσωτερικές μονάδες που έχουν ενεργό alarm όσο και το εργαλείο ελέγχου και επιτήρησης των εξωτερικών καμερών -MotionEye- όπου ελέγχει όλα τα live streaming (από τις σταθερές κάμερες). Τέλος, σε περίπτωση alarm, ανοίγει σε browser τους χάρτες -Google map- με το στίγμα του σημείου που έγινε το alarm (συντεταγμένες εσωτερικής μονάδας)

Το συνολικό σύστημα χαρακτηρίζεται από:

- Επεκτασιμότητα: μπορούν να προστεθούν/αφαιρεθούν όσες μονάδες εσωτερικές ή εξωτερικές επιθυμούμε αφού η κάθε εσωτερική μονάδα κάνει publish σε ξεχωριστό sub-topic του MQTT και οι εξωτερικές μονάδες εγγράφονται σε όλο το topic (συμπεριλαμβανομένων των sub-topic) χωρίς να γίνεται διακοπή λειτουργίας του όλου συστήματος
- Ασφάλεια: κρυπτογράφηση δεδομένων που μεταδίδονται μέσω VPN tunnel
- Ελάχιστο bandwidth: μέσω του MQTT μεταδίδονται ελάχιστες πληροφορίες ελέγχου (alarm flag, latitude, longitude) με το overhead να έχει μέγεθος 2 bytes [47].
- Χαμηλή κατανάλωση ενέργειας: τα Raspberry Pi τροφοδοτούνται από 5V/2A τροφοδοτικό με τυπική micro USB διασύνδεση.
- Τροποποίηση παρακολούθησης αντικειμένων: οι εξωτερικές μονάδες μπορούν να παρακολουθούν αντί για πρόσωπα, άλλα αντικείμενα, όπως αυτοκίνητα, πινακίδες οχημάτων με την αλλαγή μόνο του αρχείου xml με τα βιομετρικά δεδομένων αναγνώρισης από το κέντρο ελέγχου.
- Χρήση εργαλείων ανοιχτού κώδικα: τα συνεργατικά εργαλεία που χρησιμοποιούνται στην υλοποίηση, πέραν του κύριου κώδικα που αναπτύχθηκε σε Python, είναι ανοιχτού κώδικα.

1.2 Δομή της διπλωματικής εργασίας

Στο δεύτερο κεφάλαιο αναλύονται τα υλικά και οι συσκευές (hardware) που χρησιμοποιούνται κατά την υλοποίηση.

Ακολουθεί το τρίτο κεφάλαιο με το απαραίτητο λογισμικό-software με τα απαραίτητα βήματα εγκατάστασης στις επιμέρους συσκευές και τροποποίησή τους.

Στο τέταρτο κεφάλαιο παραθέτουμε τον πηγαίο κώδικα-source code- με την αναλυτική επεξήγησή του και τα απαραίτητα λογικά διαγράμματα της λογικής εκτέλεσης του κώδικα και στο τελευταίο κεφάλαιο κλείνουμε την παρούσα υλοποίηση με συμπεράσματα και μελλοντικές επεκτάσεις.

Κεφάλαιο 2 Υλικό του συστήματος-Hardware

2.1 Raspberry Pi

Οι υπολογιστικές μονάδες που χρησιμοποιήθηκαν στην παρούσα πτυχιακή ήταν τα single-board computers Raspberry Pi [6].

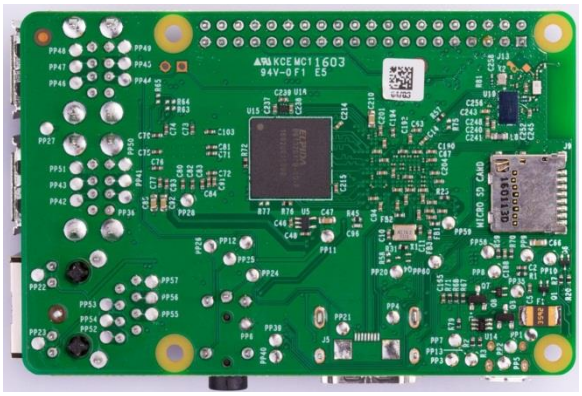
Τα Raspberry Pi [7] αναπτύχθηκαν στο Ηνωμένο Βασίλειο από το Raspberry Pi Foundation [8] και είχαν αρχικό σκοπό την βασική εκπαίδευση στην επιστήμη των υπολογιστών σε σχολεία και αναπτυσσόμενες χώρες. Το 2006 έλαβαν χώρα τα πρώτα concept και beta testings boards όπου 6 χρόνια μετά (Μάρτιος 2012) έλαβαν χώρα οι πρώτες πωλήσεις στο κοινό.

Τα Pi έγιναν αμέσως αποδεκτά από την κοινότητα των robotics και έκτοτε χρησιμοποιούνται ευρέως (μέχρι τον Μάρτιο 2017 είχαν πουληθεί 12.5 εκατομμύρια μονάδες κάνοντάς το τρίτο σε κατάταξη παγκοσμίως σε πωλήσεις υπολογιστή ευρέως σκοπού) σε υλοποιήσεις IoT (με άμεσο ανταγωνιστή του το Arduino) καθώς παρείχαν υψηλή υπολογιστική δύναμη με χαμηλή κατανάλωση ισχύος και ευελιξία σε σύνδεση περιφερειακών.

Το λειτουργικό σύστημα που έτρεχαν τα Raspberry Pi στις εργαστηριακές δοκιμές ήταν Linux και συγκεκριμένα το Raspbian (2016-05-27-raspbian-jessie). Αξίζει να αναφερθεί ότι τα λειτουργικά συστήματα που υποστηρίζουν είναι τόσο Linux-based (Raspbian, Ubuntu mate, Snappy Ubuntu core) όσο και non-Linux-based όπως Windows 10 IoT Core και RISC OS.

Ως γλώσσα προγραμματισμού για τα script χρησιμοποιήθηκε η Python που είναι εγκατεστημένη εξ αρχής στο λειτουργικό Raspbian.

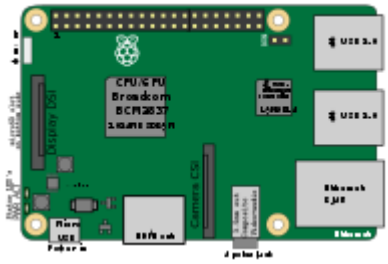
Raspberry Pi 3 model B up-side



Raspberry Pi 3 model B down-side

Οι 2 εκδόσεις που χρησιμοποιήθηκαν ήταν το Pi 3 model B για την εξωτερική μονάδα και το Pi model B για την εσωτερική μονάδα.

2.1.1 Τεχνικά χαρακτηριστικά της έκδοσης Pi 3 model B:



Quad Core 1.2GHz Broadcom BCM2837 64bit CPU

1GB RAM

40-pin extended GPIO

4 USB 2 ports

4 Pole stereo output and composite video port

Full size HDMI

BCM43438 wireless LAN and Bluetooth Low Energy (BLE) on board

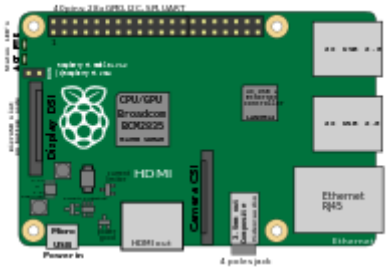
CSI camera port for connecting a Raspberry Pi camera

DSI display port for connecting a Raspberry Pi touchscreen display

Micro SD port for loading your operating system and storing data

Upgraded switched Micro USB power source up to 2.5A

2.1.2 Τεχνικά χαρακτηριστικά της έκδοσης Pi model B:



900MHz quad-core ARM Cortex-A7 CPU

1GB RAM

4 USB ports

40 GPIO pins

Full HDMI port

Ethernet port

Combined 3.5mm audio jack and composite video

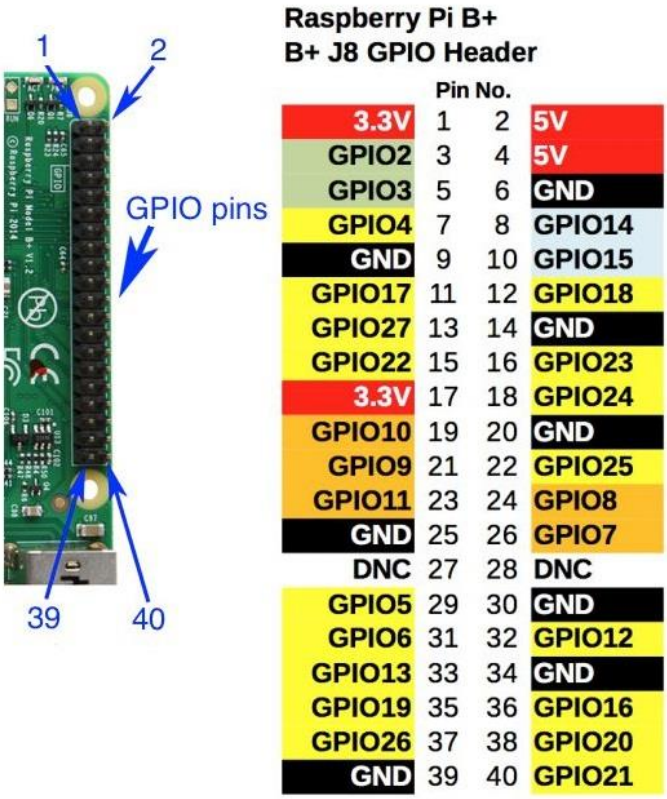
Camera interface (CSI)

Display interface (DSI)

Micro SD card slot

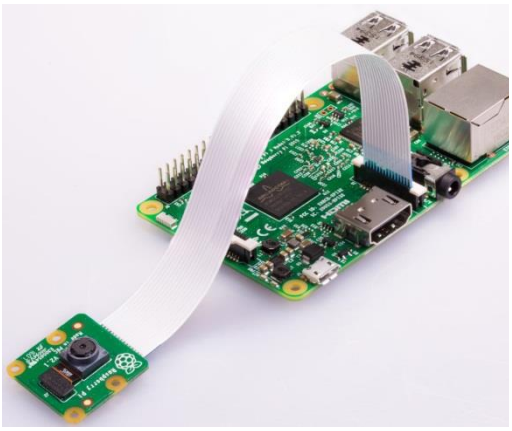
VideoCore IV 3D graphics core

Τα Raspberry διαθέτουν και ακροδέκτες GPIO (general-purpose input/output) στους οποίους μπορούμε να συνδέσουμε και διάφορα περιφερειακά για έλεγχο και λειτουργία (πχ servo, pir κτλ) και ο προγραμματισμός του κάθε pin (πχ input-output-clock-pulse) γίνεται μέσα από το εκάστοτε script (πχ python) που τρέχουμε εκείνη την στιγμή. Η διάταξή τους είναι:



GPIO schematic - explanation

Επίσης, εκτός από τις usb θύρες στις οποίες μπορούμε να χρησιμοποιήσουμε περιφερειακά (web-cameras, mouse, keyboard, wifi, Bluetooth κτλ) έχουν και δική τους ξεχωριστή θύρα (CSI) για σύνδεση on-board κάμερας (πλεονέκτημα η άμεση αναγνώριση/λειτουργία από το λειτουργικό και με ταχύτερη απόκριση). Η αρχική έκδοση των 5-megapixel ήταν διαθέσιμη το 2013 με την έκδοση των 8-megapixel (Camera Module v2) να ακολουθεί το 2016.



On board camera connection with ribbon cable

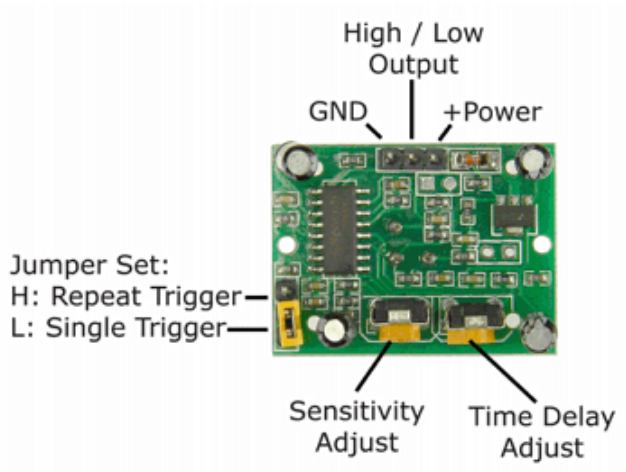
Hardware specification

	Camera Module v1	Camera Module v2
Net price	$25	$25
Size	Around 25 × 24 × 9 mm	
Weight	3g	3g
Still resolution	5 Megapixels	8 Megapixels
Video modes	1080p30, 720p60 and 640 × 480p60/90	1080p30, 720p60 and 64 × 480p60/90
Linux integration	V4L2 driver available	V4L2 driver available
C programming API	OpenMAX IL and others available	OpenMAX IL and others available
Sensor	OmniVision OV5647	Sony IMX219
Sensor resolution	2592 × 1944 pixels	3280 × 2464 pixels
Sensor image area	3.76 × 2.74 mm	3.68 x 2.76 mm (4.6 mm diagonal)
Pixel size	1.4 µm × 1.4 µm	1.12 µm x 1.12 µm
Optical size	1/4"	1/4"
Full-frame SLR lens equivalent	35 mm	
S/N ratio	36 dB	
Dynamic range	67 dB @ 8x gain	
Sensitivity	680 mV/lux-sec	
Dark current	16 mV/sec @ 60 C	
Well capacity	4.3 Ke-	
Fixed focus	1 m to infinity	
Focal length	3.60 mm +/- 0.01	3.04 mm
Horizontal field of view	53.50 +/- 0.13 degrees	62.2 degrees
Vertical field of view	41.41 +/- 0.11 degrees	48.8 degrees
Focal ratio (F-Stop)	2.9	2.0

2.2 PIR

Ως motion sensors [9] χρησιμοποιήθηκαν τα HC-SR501 [10]. Βασίζεται στον infrared sensor LHI778 (Passive) και για τον έλεγχο του εντοπισμού της κίνησης έχει το ολοκληρωμένο BISS0001. Φέρει trimmer ρύθμισης της ευαισθησίας εντοπισμού κίνησης για να πετύχει εύρος κάλυψης από 3 έως 7 μέτρα. Επίσης φέρει trimmer για time delay ελέγχου εντοπισμού κίνησης και triggering selection ανάλογα με την εκάστοτε εφαρμογή.

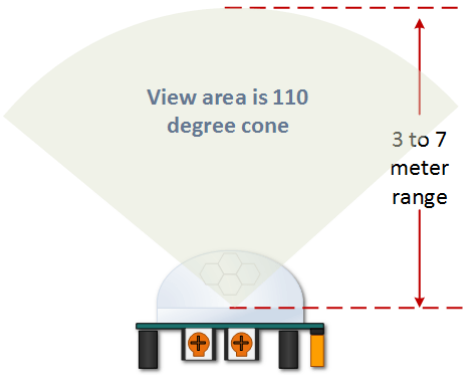
HC-SR501 up-side

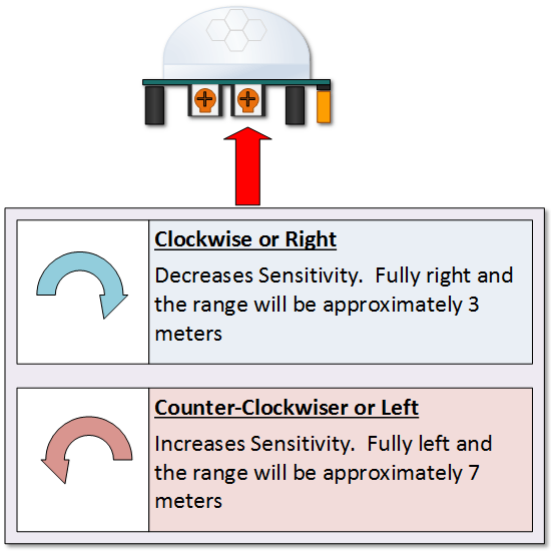


HC-SR501 down-side

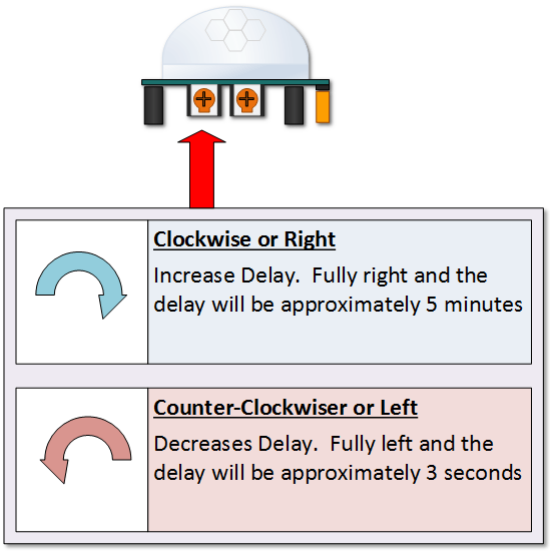
Το HC-SR501 [11] όταν εντοπίσει κίνηση θα εγείρει το λογικό «1» στην έξοδό του (3.3V) και όταν δεν εντοπίσει κίνηση εγείρει λογικό «0» στην έξοδό του (0V). Το time delay ορίζει πόσο χρονικό διάστημα (από τον εντοπισμό της κίνησης) θα διατηρείται το λογικό «1» στην έξοδό του με τιμές από 3 sec ως 5 minutes.

Η γωνία κάλυψης του infrared sensor είναι 110°:



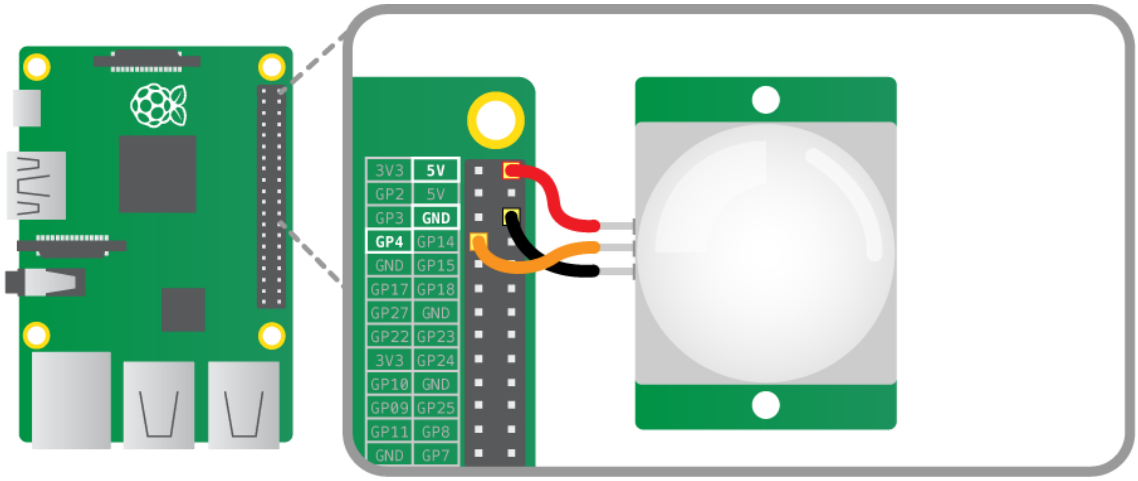


Ρύθμιση της απόστασης κάλυψης



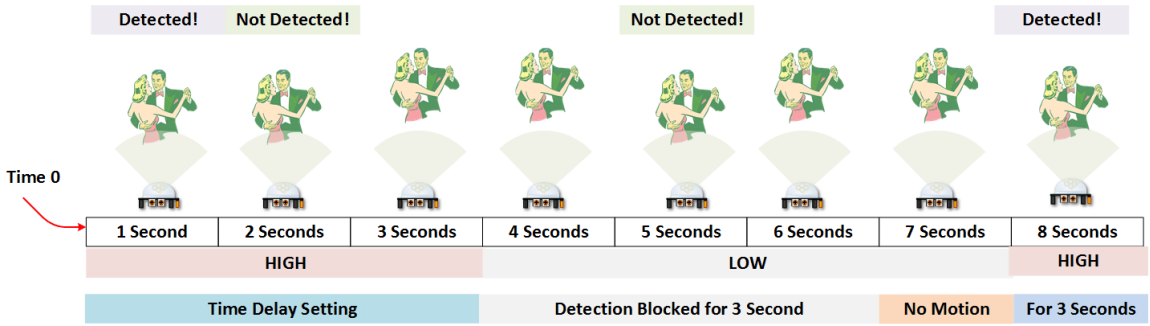
Ρύθμιση του time delay

Η σύνδεση του pir sensor με το GPIO του Raspberry Pi θα γίνει όπως απεικονίζεται παρακάτω με την τροφοδοσία να γίνεται μέσω των Pin 2 (5V) και 6 (Gnd) και η έξοδός του θα συνδεθεί στο pin 7 (το οποίο θα ορίσουμε ως είσοδο στον κώδικα της python):

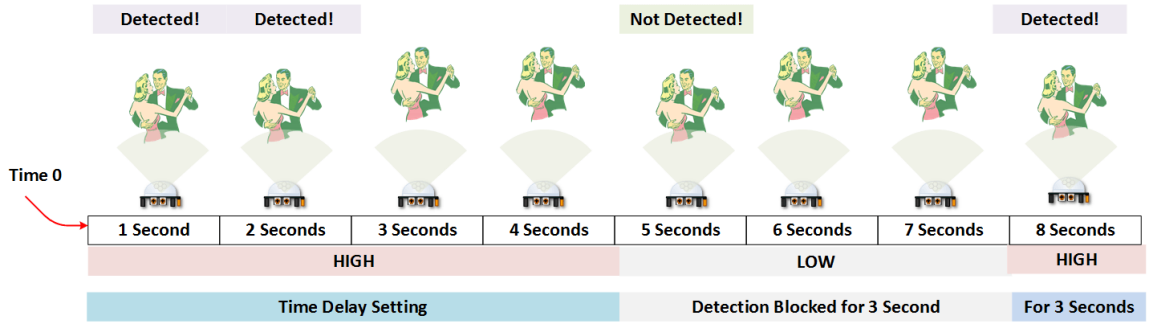


2.2.1 Παραδείγματα ρύθμισης και λειτουργίας:

Θέτουμε το time delay στην ελάχιστη θέση (3 sec) και το trigger mode σε single mode. Όπως φαίνεται και στο κάτωθι σχήμα υπάρχει μια περίοδος 6 sec (3+3 sec) που η κίνηση δεν μπορεί να εντοπιστεί:

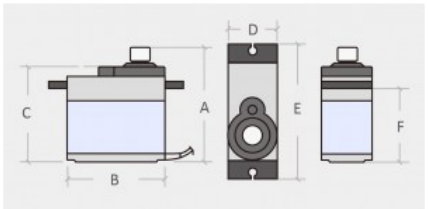


Σε επόμενη δοκιμή αλλάζουμε μόνο το trigger mode σε repeated mode και πάλι παρατηρούμε ότι υπάρχει η περίοδος των 3 sec που μετά τον μη-εντοπισμό κίνησης δεν παράγει τίποτα στην έξοδο του και ας υπάρξει κίνηση:



2.3 Servo

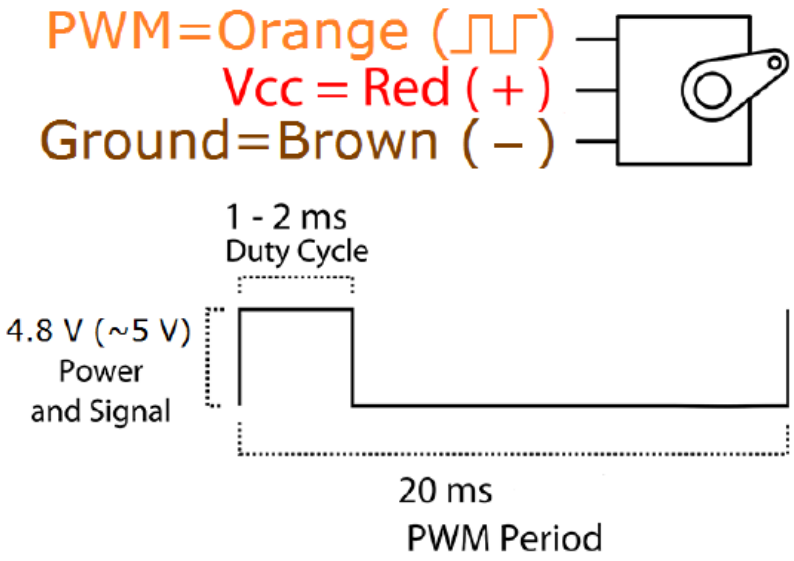
Ως μηχανισμό κίνησης της εξωτερικής κάμερας χρησιμοποιήσαμε το SG90 servo [12] της TowerPro [13] με δυνατότητα στρέψης 180 μοιρών (90 μοίρες σε κάθε κατεύθυνση) μαζί με βάση στήριξης:



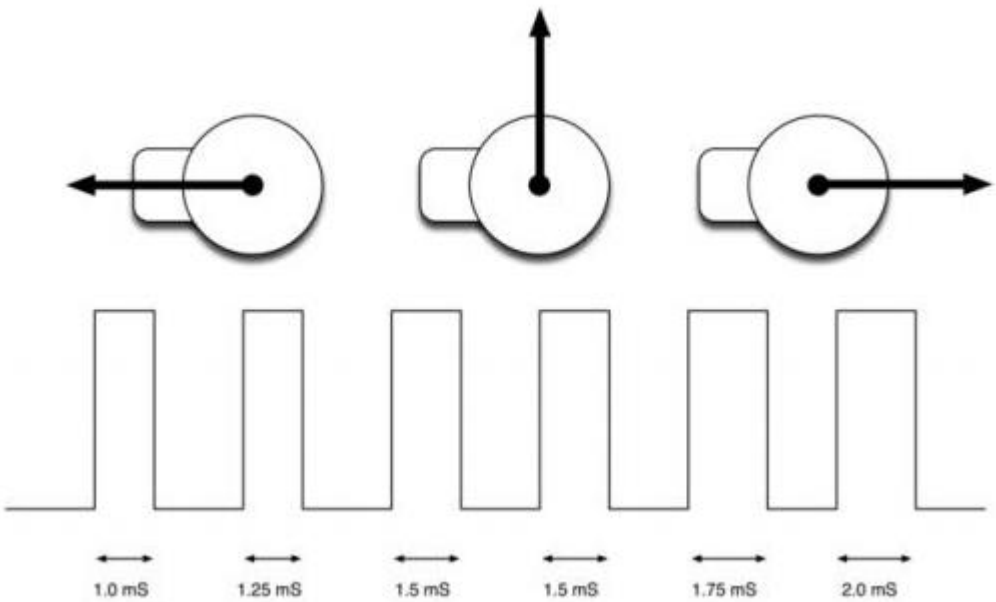
Weight(g)	9
Torque(kg)(4.8v)	1.8
Speed(sec/60deg)	0.1
A(mm)	34.5
B(mm)	22.8
C(mm)	26.7
D(mm)	12.6
E(mm)	32.5
F(mm)	16

Ο κόκκινος ακροδέκτης είναι η τάση λειτουργίας (+5V) και ο καφέ ακροδέκτης αποτελεί την γείωση [14]. Ο πορτοκαλί ακροδέκτης λαμβάνει τον παλμό λειτουργίας για την στρέψη

του σερβομηχανισμού. Η περίοδος παλμού είναι 20 msec που σημαίνει ότι το σερβό μπορεί να περιστραφεί θεωρητικά 50 φορές το δευτερόλεπτο (στην πράξη λόγω και των delay στρέψης δεν μπορεί να γίνει κάτι τέτοιο).



Το πλάτος του παλμού καθορίζει τις μοίρες στρέψης και παίρνει τιμές από 0.5ms έως 2.5ms αντιστοιχώντας σε μοίρες στρέψης από 0^o έως 180^o. Η ουδέτερη θέση του σερβό είναι στις 90^o και επιτυγχάνεται με παλμό 1.5ms. Αφού η συχνότητα είναι 100Hz το duty ratio του παλμού αντιστοιχεί σε 5%~25% (0.5ms~2.5ms):



2.3.1 Τεχνικά χαρακτηριστικά:

Weight: 9g

Dimension: 23×12.2x29mm

Stall torque: 1.8kg/cm(4.8v)

Gear type: POM gear set

Operating speed: 0.1sec/60degree(4.8v)

Operating voltage: 4.8v

Temperature range: 0°C~55 °C

Dead band width: 1us

Power Supply: Through External Adapter

servo wire length: 25 cm

Servo Plug: JR (Fits JR and Futaba)

Η βάση περιστροφής της κάμερας δέχεται 2 servo για οριζόντια και κάθετη κίνηση (pan-tilt):

2.4 Εξωτερική USB κάμερα

Ως κινούμενη κάμερα εξωτερικής χρήσης επιλέξαμε το μοντέλο Logitech QuickCam Communicate STX [15] με USB διασύνδεση με την μονάδα του Raspberry Pi.

2.4.1 Τεχνικά χαρακτηριστικά:

Color: Black

USB 2.0 interface

Sensor: High quality VGA

Video Capture: Up to 640 x 480 pixels

Still Image Capture: 1.3 megapixels (interpolated)

Video File Type: AVI

Still Image File Types: BMP, JPEG

Frame Rate: Up to 30 frames per second (with recommended system)

Field of View: 42 degrees horizontal

Optics: Fixed focus

Integrated microphone

Cable Length: 6-ft. USB cable

2.5 Κόστος υλικών

Παραθέτουμε το κόστος λιανικής της κάθε συσκευής που χρησιμοποιήσαμε:

Raspberry Pi 3 model B: **38€**

Raspberry Pi Camera Module: **19€**

Αισθητήρας Κίνησης HC-SR501: **0,8€**

Micro Servo Tower Pro SG90: **3,5€**

Βάση περιστροφής: **0,8€**

Εξωτερική USB κάμερα: **20~30€**

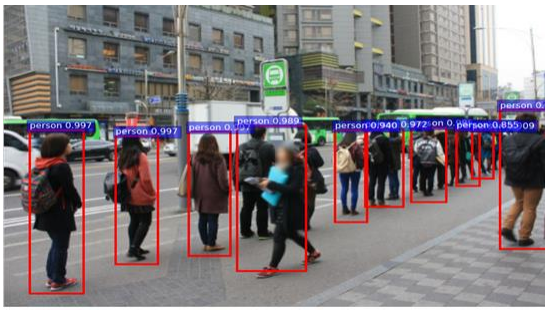
Κεφάλαιο 3 Λογισμικό του συστήματος-Software

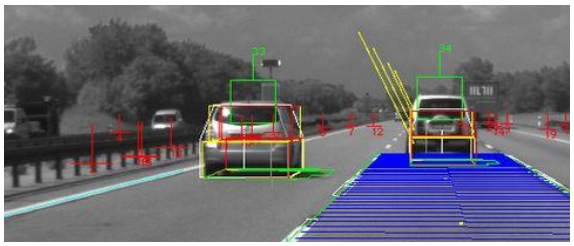
3.1 OpenCV

Το OpenCV (Open Source Computer Vision Library) [16] αποτελεί μια βιβλιοθήκη λογισμικού για εφαρμογές computer-vision και machine-learning και είναι υλοποιημένο σε C++ [17] από την Intel. Σχεδιάστηκε [18] για να παρέχει μια κοινή βάση στις εν λόγω εφαρμογές και για την επιτάχυνση της χρήσης της αντίληψης των μηχανών στις εμπορικές υλοποιήσεις. Ως Open Source (BSD-licensed product) καθίσταται εύκολη η υλοποίηση/τροποποίηση του κώδικα.

Η βιβλιοθήκη περιλαμβάνει περισσότερους από 2500 βελτιστοποιημένους αλγόριθμους, τόσο απλοποιημένους όσο και state-of-the-art, για machine-learning, οι οποίοι μπορούν να χρησιμοποιηθούν για:

- → ανίχνευση και αναγνώριση προσώπων-αντικειμένων
- → ταξινόμηση ανθρώπινων κινήσεων
- → παρακολούθηση (tracking) κινούμενων αντικειμένων-καμερών
- → εξαγωγή 3D μοντέλων των αντικειμένων
- → ενοποίηση εικόνων για παραγωγή εικόνας υψηλής ανάλυσης
- → εύρεση παρόμοιων εικόνων από βάση δεδομένων με εικόνες
- → αφαίρεση του φαινομένου των «κόκκινων ματιών» από φωτογραφίες
- → ακολουθία των κινήσεων των ματιών
- → αναγνώριση σκηνών (scenery) και χρήση markers για υλοποίηση σε περιβάλλον επαυξημένης πραγματικότητας (augmented reality)





Η κοινότητα του OpenCV αριθμεί περισσότερους από 47 χιλιάδες ανθρώπους και ο αριθμός των λήψεων των βιβλιοθηκών ξεπερνά τα 14 εκατομμύρια.

Οι βιβλιοθήκες χρησιμοποιούνται ευρέως από ερευνητικές ομάδες-κέντρα, κυβερνητικά σχήματα και εταιρείες (μεταξύ άλλων: Google, Yahoo, Microsoft, Intel, IBM, Sony, Honda, Toyota).

Υποστηρίζει Windows, Linux, Android και Mac OS και παρέχει βιβλιοθήκες σε C++, C, Python, Java και MATLAB. Απευθύνεται κυρίως σε εφαρμογές οπτικού περιεχομένου πραγματικού χρόνου και χρησιμοποιεί επιπλέον τις δυνατότητες MMX (Intel acronym for processor instruction set) και SSE (Streaming SIMD Extensions) όπου δύναται με μελλοντικές υλοποιήσεις να βασίζονται και στα CUDA και OpenCL.

Τα βήματα για την εγκατάσταση και χρήση του OpenCV στο Raspberry Pi:

→ Αρχικά κάνουμε αναβάθμιση στον πυρήνα του λειτουργικού συστήματος:

```
sudo apt-get update
sudo apt-get upgrade
sudo dist-upgrade
```

→ Εγκαθιστούμε όλα τα απαραίτητα πακέτα υποστήριξης για το Raspberry Pi και τέλος το OpenCV:

```
sudo apt-get install build-essential cmake pkg-config
sudo apt-get install libjpeg-dev libtiff5-dev libjasper-dev
libpng12-dev
sudo apt-get install libgtk2.0-dev libgstreamer0.10-0-dbg
libgstreamer0.10-0 libgstreamer0.10-dev libv4l-0 libv4l-dev
sudo apt-get install libavcodec-dev libavformat-dev
libswscale-dev libv4l-dev
sudo apt-get install libxvidcore-dev libx264-dev
sudo apt-get install libatlas-base-dev gfortran
sudo apt-get install python-numpy python-scipy python-
matplotlib
sudo apt-get install default-jdk ant
sudo apt-get install libgtkglext1-dev
sudo apt-get install v4l-utils
sudo modprobe bcm2835-v4l2
wget https://bootstrap.pypa.io/get-pip.py
sudo python get-pip.py
sudo apt-get install python2.7-dev
sudo pip install numpy
sudo apt-get install libopencv-dev python-opencv
```

3.2 FFmpeg 

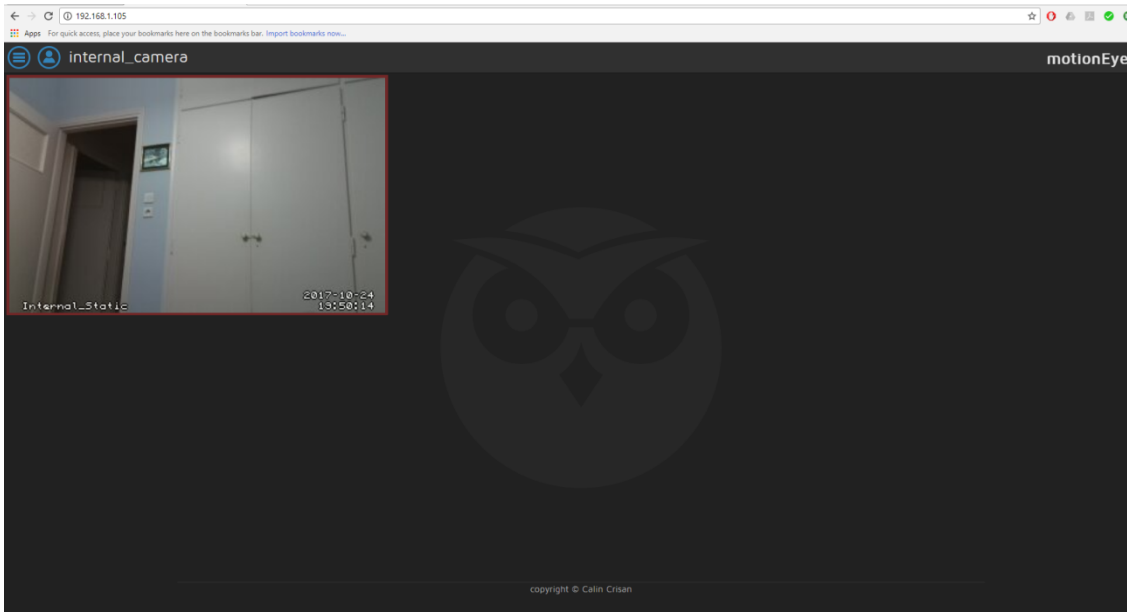
Απαραίτητο πρόγραμμα στα Raspberry για το streaming video (και την επεξεργασία του) ήταν το FFMpeg [19] [20]. Αποτελεί το multimedia framework που χρησιμοποιείται ευρέως σήμερα για media decoding, encoding, transcoding, multiplexing, de-multiplexing, streaming, filtering, playing. Υποστηρίζει όλα τα formats και υποστηρίζεται από όλες τις πλατφόρμες.

Τα βήματα για την εγκατάσταση του FFMpeg στο Raspberry Pi:

```
wget
https://github.com/ccrisan/motioneye/wiki/precompiled/ffmpeg_3
.1.1-1_armhf.deb
dpkg -i ffmpeg_3.1.1-1_armhf.deb
apt-get remove libavcodec-extra-56 libavformat56
libavresample2 libavutil54
```

3.3 MotionEye & Motion

Το MotionEye [21] αποτελεί το περιβάλλον διαχείρισης μέσω browser (web frontend) για το motion και είναι υλοποιημένο σε Python. Το motion [22] αποτελεί το πρόγραμμα που παρακολουθεί την πηγή που του ορίζουμε (πχ onboard camera) και ελέγχει αν υπάρχει αλλαγή στα περιεχόμενα της εικόνας, δλδ επιτελεί τη λειτουργία motion detection. Επίσης, μέσω του MotionEye, παρέχονται δυνατότητες live streaming, triggering settings, snapshot και video sampling (όταν γίνει ανίχνευση κίνησης) αλλά και αποθήκευση των media files τόσο τοπικά (Raspberry Pi) όσο και σε FTP ή cloud (Google Drive, DropBox) υπηρεσίες.



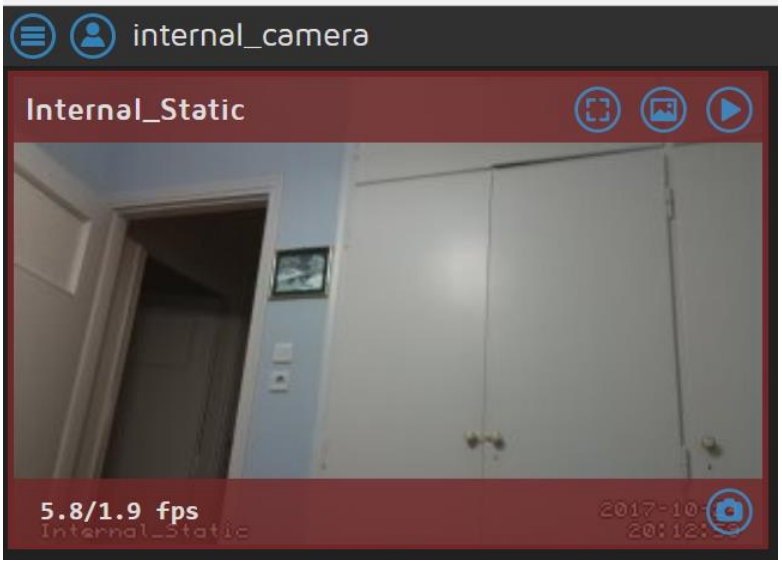
Αρχική οθόνη MotionEye

Στην αρχική οθόνη απεικονίζεται τόσο η live εικόνα από την κάμερα (με εμφάνιση ονόματος κάμερας και ημερομηνίας-ώρας στην κάτω πλευρά) όσο και οι 2 κύριες επιλογές με μπλε εικονίδια:

 Μενού ρυθμίσεων

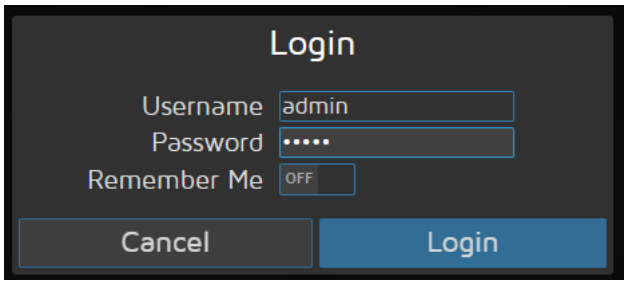
 Login

Αν πατήσουμε πάνω στην live εικόνα, εμφανίζονται επιλογές:

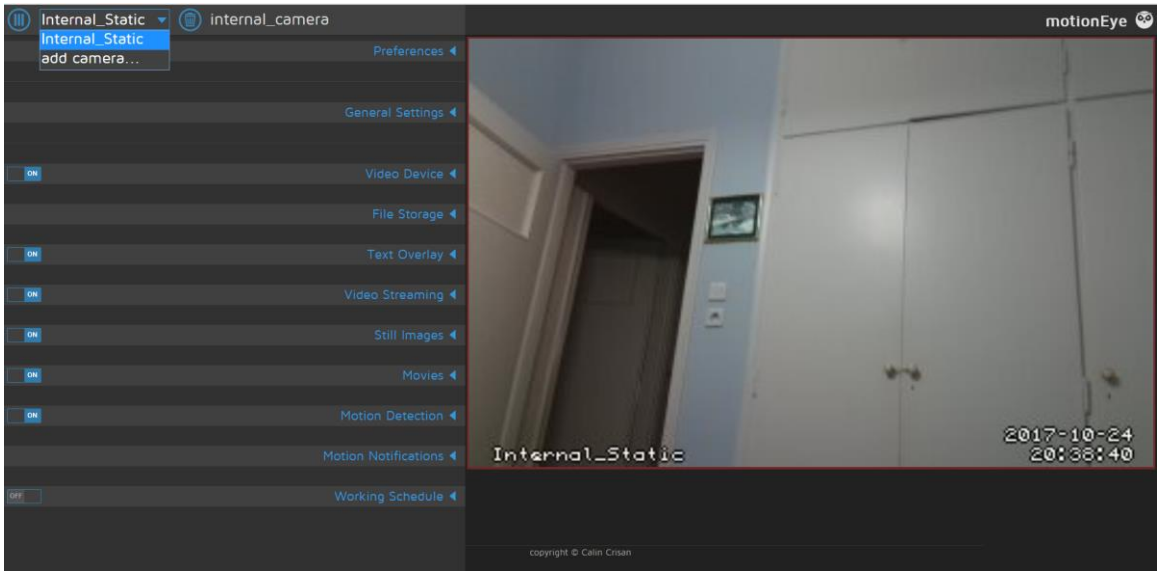


Από πάνω αριστερά προς τα δεξιά: Full screen, Αρχεία εικόνων που έχει αποθηκεύσει τοπικά μετά από motion detection, Αρχεία βίντεο που έχει αποθηκεύσει τοπικά μετά από motion detection. Κάτω έχει επιλογή χειροκίνητου screenshot της εικόνας.

Αφού κάνουμε Login:



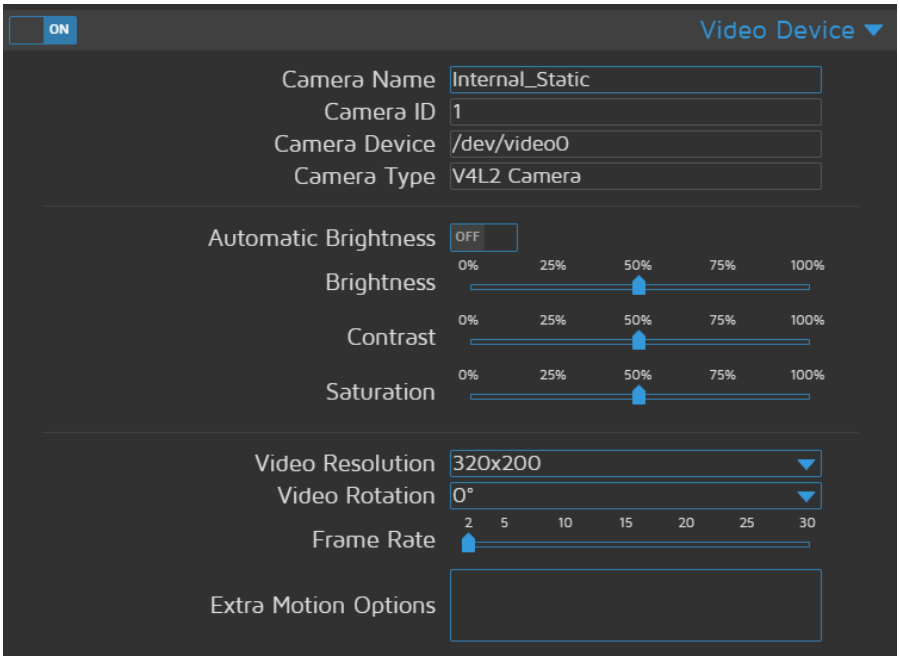
Πατάμε στο μενού των ρυθμίσεων και δεξιά στο panel που ανοίγει εμφανίζονται όλες οι διαθέσιμες επιλογές-ρυθμίσεις:



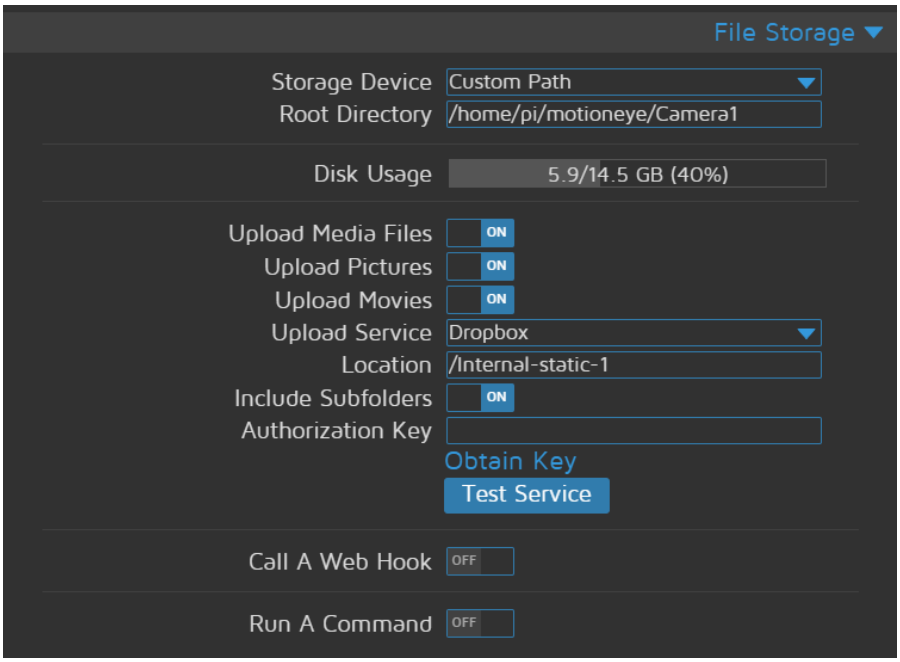
Στην πάνω πλευρά του panel μας δίνει την δυνατότητα να προσθέσουμε κάμερα (τοπική ή μέσω δικτύου), να διαλέξουμε την ενεργή κάμερα και να διαγράψουμε την εκάστοτε κάμερα από το MotionEye.

Ξεκινώντας με τις ρυθμίσεις, στην καρτέλα "Preferences" μας επιτρέπει να τροποποιήσουμε το δεξί panel με την live εικόνα ώστε να φιλοξενεί ταυτόχρονα περισσότερες κάμερες. Έπειτα στην καρτέλα "General Settings" μας επιτρέπει να τροποποιήσουμε το username/password, να επανεκινήσουμε/κλείσουμε το σύστημα και να κάνουμε backup/restore το configuration:

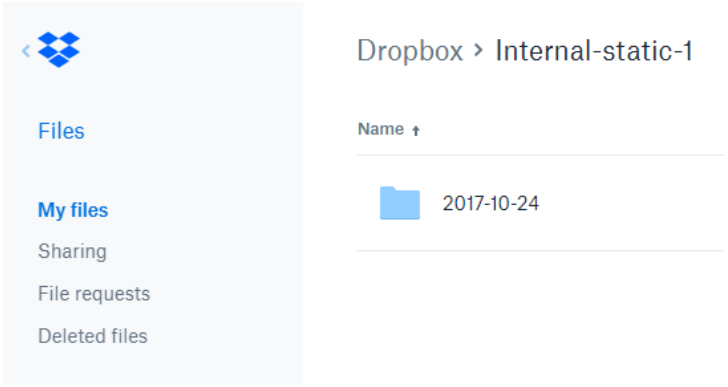
Έπειτα ακολουθούν ρυθμίσεις λειτουργίας της επιλεγμένης κάμερας:



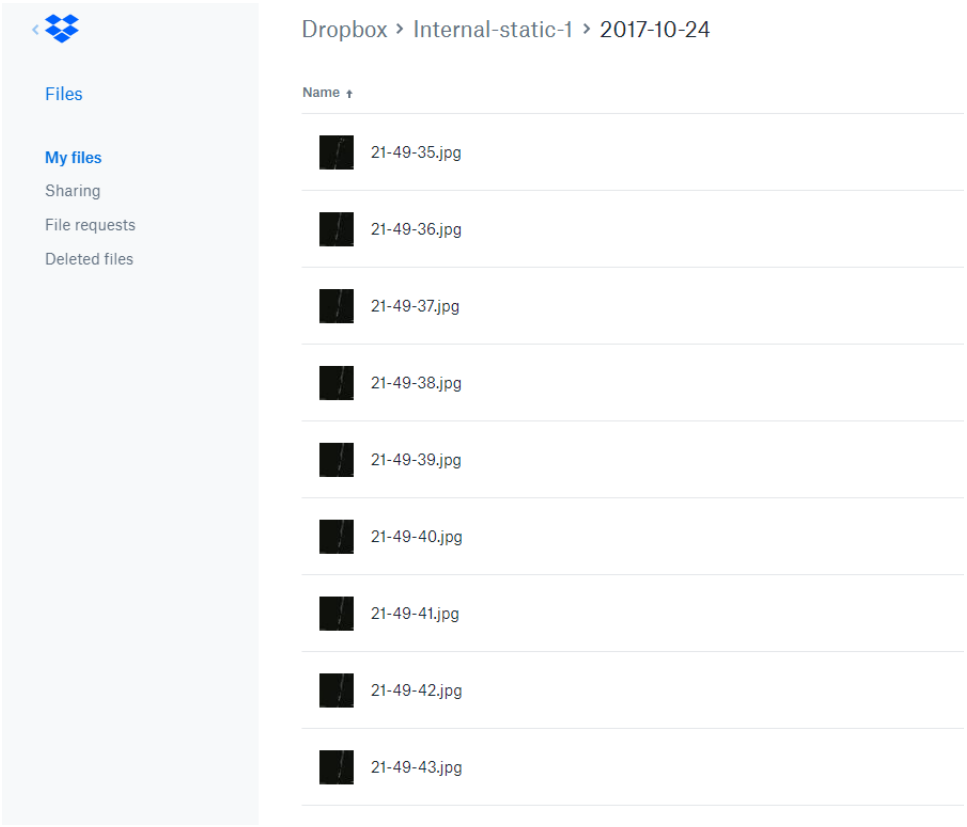
Στην καρτέλα "File Storage" μπορούμε να επιλέξουμε που θα σώζει τα media files τόσο τοπικά (επιλέγοντας το path) όσο και σε cloud (ρυθμίζοντας το account):



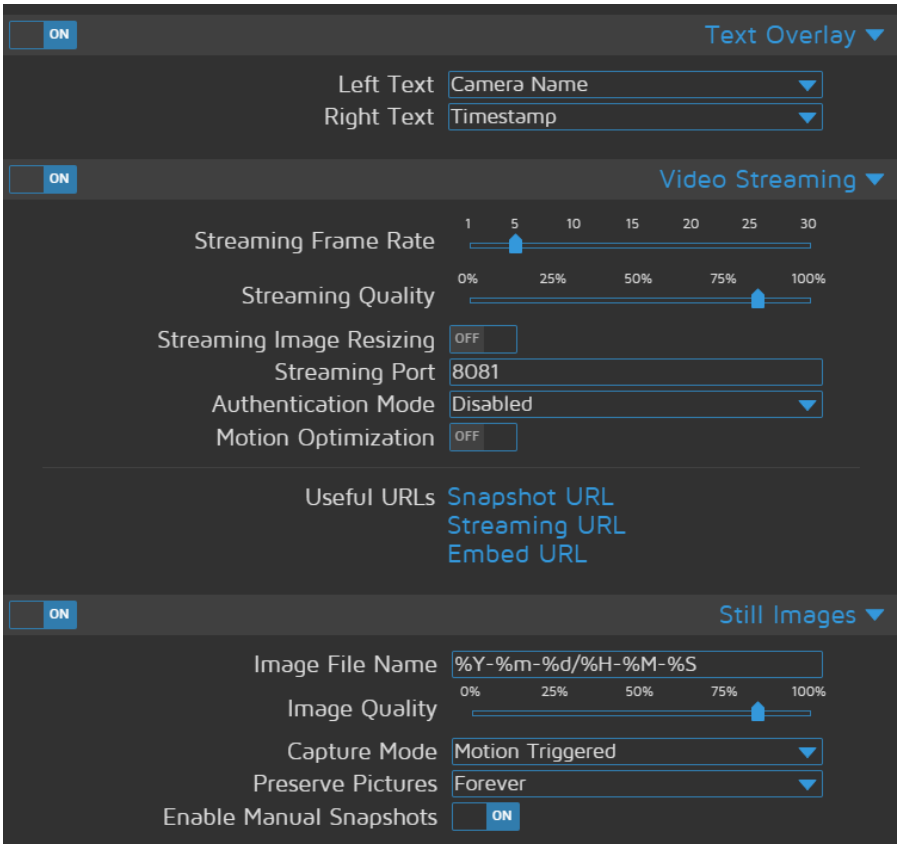
Έχοντας επιλέξει το DropBox [23] και έχοντας κάνει authorization μέσω της επιλογής"Obtain Key" του ορίζουμε έναν φάκελο που θα τα σώζει στο DropBox (πχ Internal-static-1):



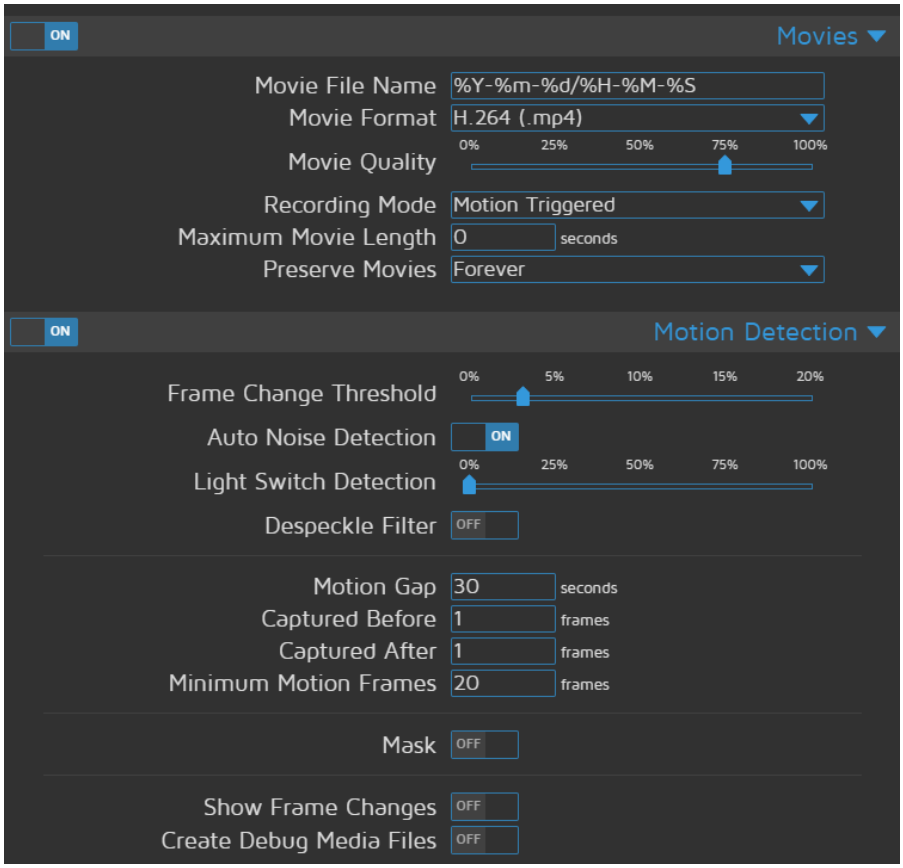
Το MotionEye φτιάχνει στο path που του έχουμε ορίσει φάκελο με την ημερομηνία και μέσα σε αυτόν αποθηκεύει τα media της εκάστοτε ημερομηνίας που έλαβαν χώρα:



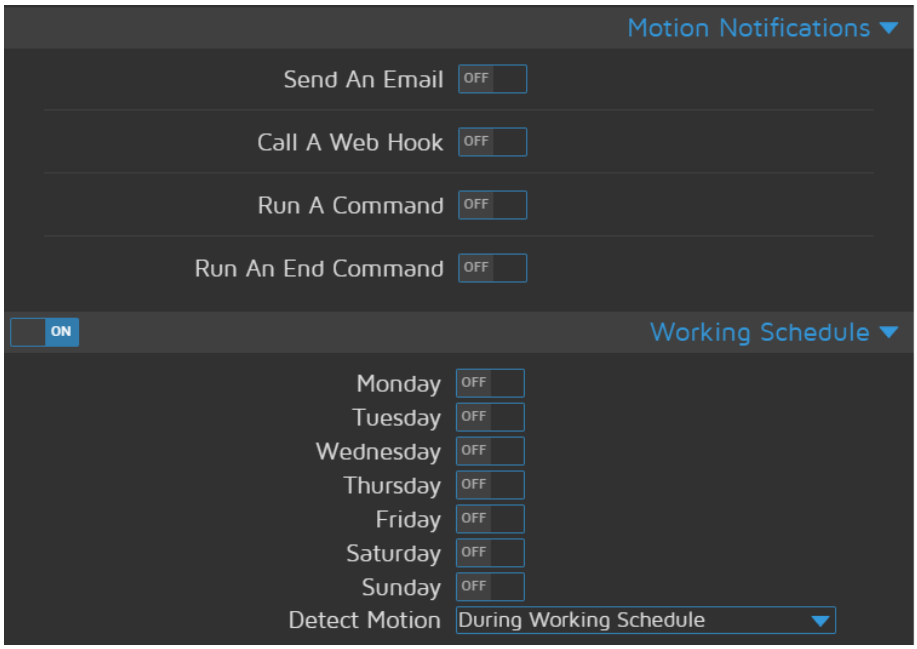
Στις επόμενες 3 ενότητες μας επιτρέπει να επιλέξουμε τι να εμφανίζεται σαν label στην live εικόνα, επιλογές του video streaming και επιλογές για τις εικόνες που θα αποθηκεύει (πχ αν θα αποθηκεύει συνέχεια εικόνες ή μόνο μετά από motion detection, το όνομα των αρχείων, την ποιότητα):



Αντίστοιχα στην καρτέλα "Movies" μας επιτρέπει να ρυθμίσουμε πότε θα σώζει βίντεο, το format, ποιότητα, ονόματα αρχείων και στην ενότητα "Motion Detection" μπορούμε να ρυθμίσουμε τις λεπτομέρειες για την ενεργοποίηση του motion triggering:



Τελευταίες 2 ενότητες είναι η "Motion Notifications" με επιπλέον actions σε περίπτωση motion detection (πχ αποστολή mail) και η "Working Schedule" για ορισμό λειτουργίας ορισμένων ημερών εβδομαδιαίως:

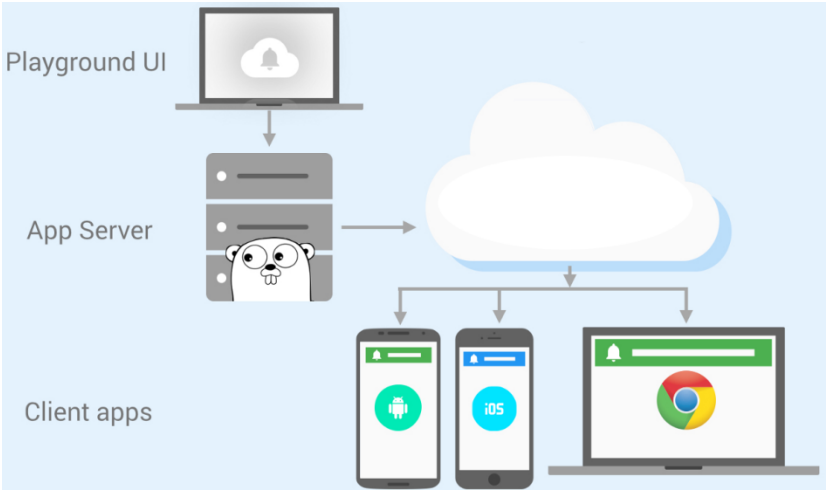


Τα βήματα για την εγκατάσταση του MotionEye στο Raspberry Pi:

```
apt-get install python-pip python-dev curl libssl-dev
libcurl4-openssl-dev libjpeg-dev libx264-142 libavcodec56
libavformat56 libmysqlclient18 libswscale3 libpq5

wget https://github.com/Motion-
Project/motion/releases/download/release-
4.0.1/pi_jessie_motion_4.0.1-1_armhf.deb
dpkg -i pi_jessie_motion_4.0.1-1_armhf.deb

pip install motioneye

mkdir -p /etc/motioneye

cp /usr/local/share/motioneye/extra/motioneye.conf.sample
/etc/motioneye/motioneye.conf

mkdir -p /var/lib/motioneye

cp /usr/local/share/motioneye/extra/motioneye.systemd-unit-
local /etc/systemd/system/motioneye.service

systemctl daemon-reload
```

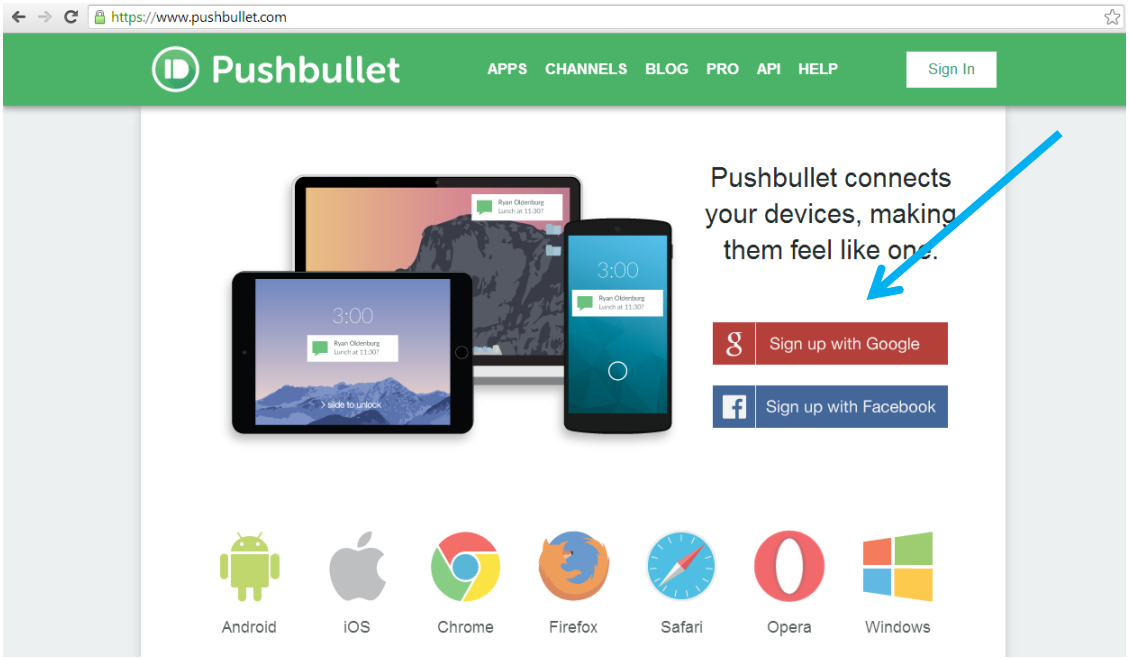
3.4 Pushbullet (Push Notification service)

Τα Push Notifications [24] αποτελούν μηνύματα που διοχετεύονται από τον back-end server (pushed) σε μια end-user συσκευή (πχ smartphone) και χρησιμοποιούνται κατά κόρον από τα smartphones-tablets σήμερα καθώς αποτελούν άμεσο και ευδιάκριτο τρόπο ενημέρωσης για κάποιο event (πχ νέο mail). Το push notification [25] μπορεί να είναι μήνυμα στην οθόνη είτε ήχος ειδοποίησης.

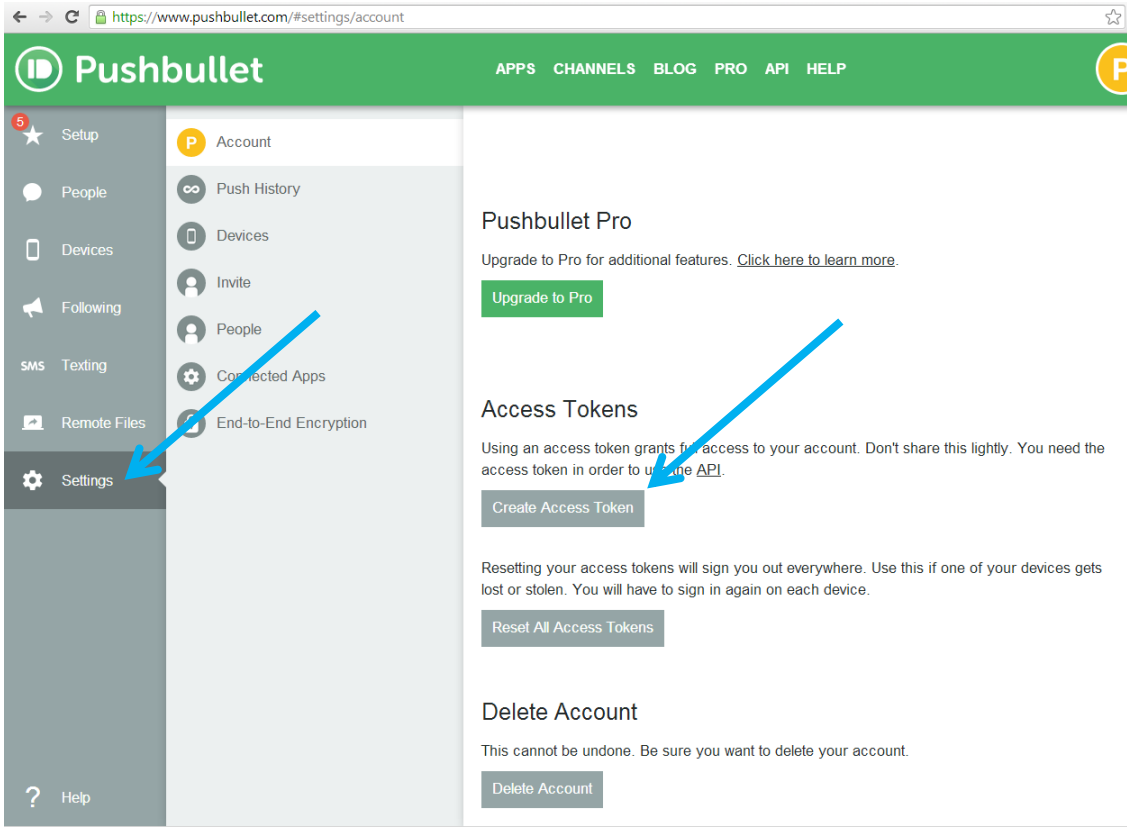


Για την προώθηση push notifications στις φορητές συσκευές χρησιμοποιούμε το εργαλείο PushBullet [26] το οποίο παρέχει υποστήριξη και εφαρμογές για όλες τις πλατφόρμες.

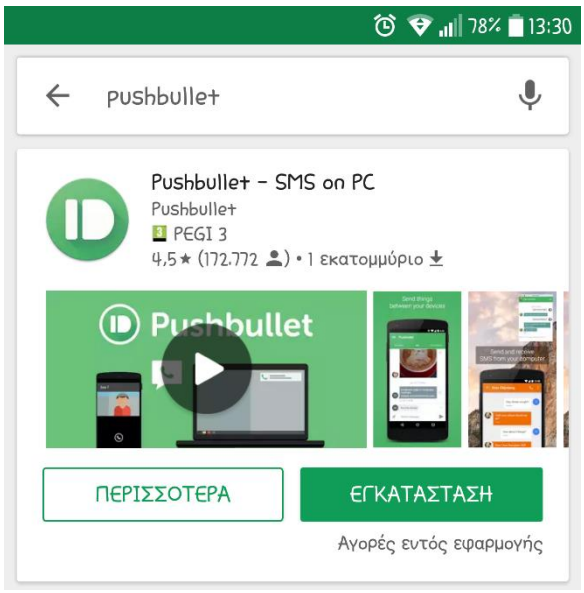
Αρχικά δημιουργούμε λογαριασμό στον server της υπηρεσίας:

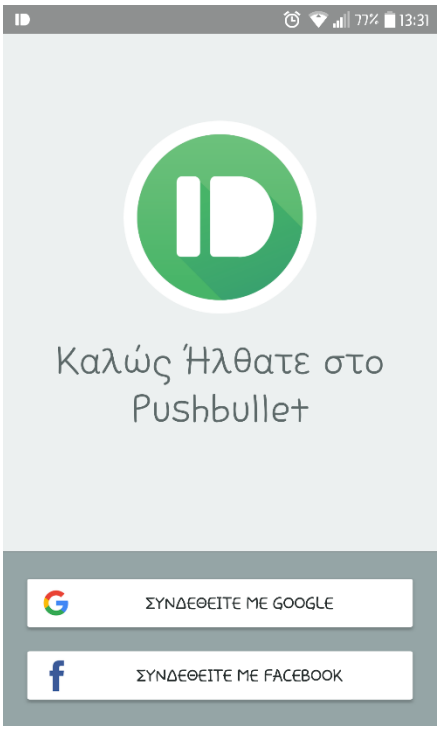


Έπειτα δημιουργούμε το κλειδί (access token) για να μπορέσουμε να στείλουμε push notifications από το script που θα τρέχει στο Raspberry Pi στο smartphone του χρήστη. Το κλειδί χρησιμοποιείται ως διαπιστευτήριο για να έχει πρόσβαση το python script που έχουμε φτιάξει στον λογαριασμό μας στο PushBullet και θα το εισάγουμε στο αρχείο με τις ρυθμίσεις (settings-internal.txt) που θα διαβάζει το script:



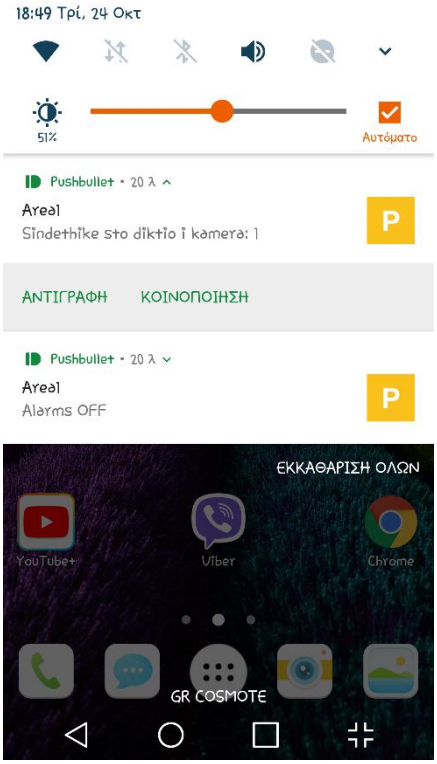
Αντίστοιχα εγκαθιστούμε την εφαρμογή στο τερματικό που θέλουμε, πχ smartphone:



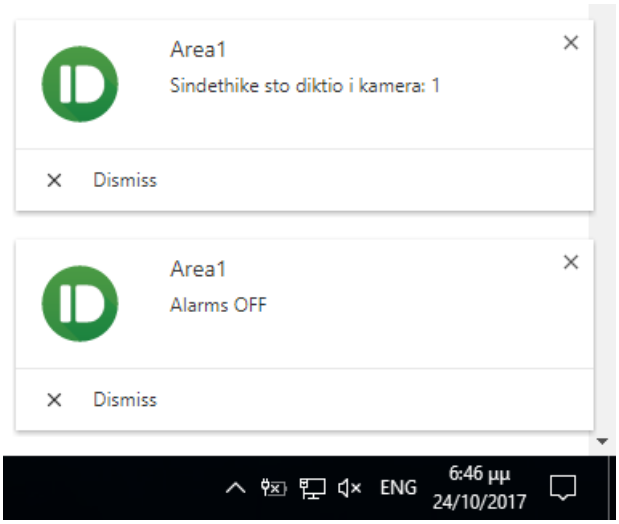


Και εν συνεχεία κάνουμε login με το ίδια στοιχεία όπως κάναμε και στο site. Αντίστοιχα, μπορούμε να εγκαταστήσουμε το PushBullet και σε άλλες συσκευές (πχ laptop) και να λειτουργεί ταυτόχρονα.

→ Στην δική μας περίπτωση όταν ενεργοποιηθεί η εσωτερική μονάδα στέλνει 2 push notifications, ένα που ενημερώνει για τον αύξοντα αριθμό της μονάδας-κάμερας που συνδέθηκε στο δίκτυο (σε περίπτωση που κάποιος έχει περισσότερες της μιας κάμερες) και αν το σύστημα είναι armed ή disarmed:

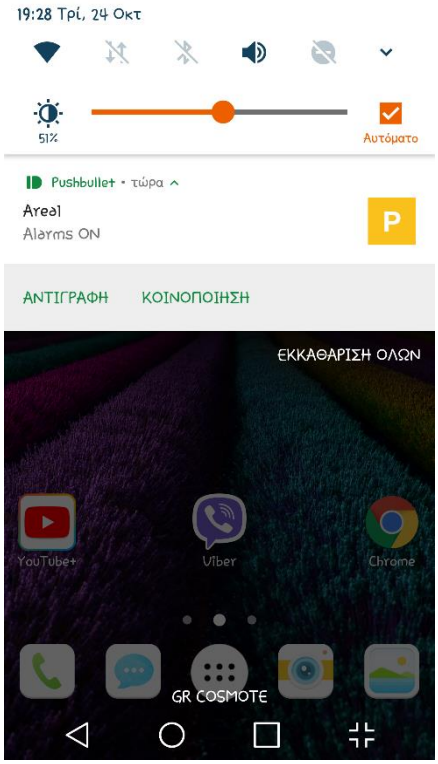


Notifications σε smartphone

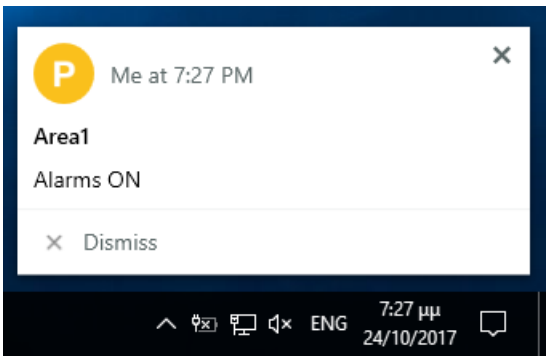


Notifications σε laptop

→ Όταν κάνουμε arming (ή disarming) του συστήματος συναγερμού αποστέλλει εκ νέου notification:

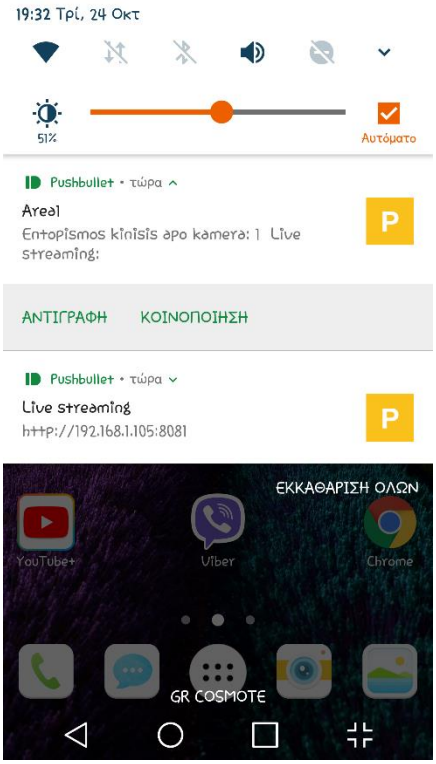


Notifications σε smartphone

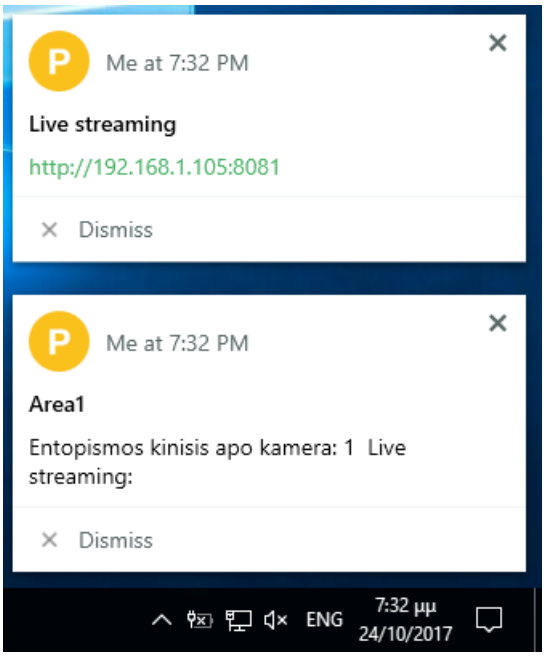


Notifications σε laptop

→ Έπειτα, όταν το PIR της εσωτερικής μονάδας εντοπίσει κίνηση, ενεργοποιεί την μετάδοση πάλι 2 push notifications, ένα για τον αύξοντα αριθμό της μονάδας-κάμερας που εντόπισε την κίνηση και άλλο ένα με το link της IP που κάνει streaming video μέσω του MotionEye (ώστε κάνοντας κλικ να ανοίξει απευθείας στον browser της εκάστοτε συσκευής το live video):



Notifications σε smartphone



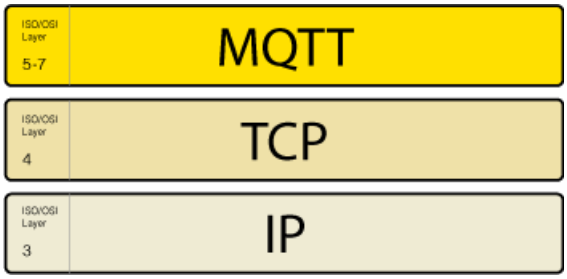
Notifications σε laptop

Τα βήματα για την εγκατάσταση του Pushbullet στο Raspberry Pi **[27]**:

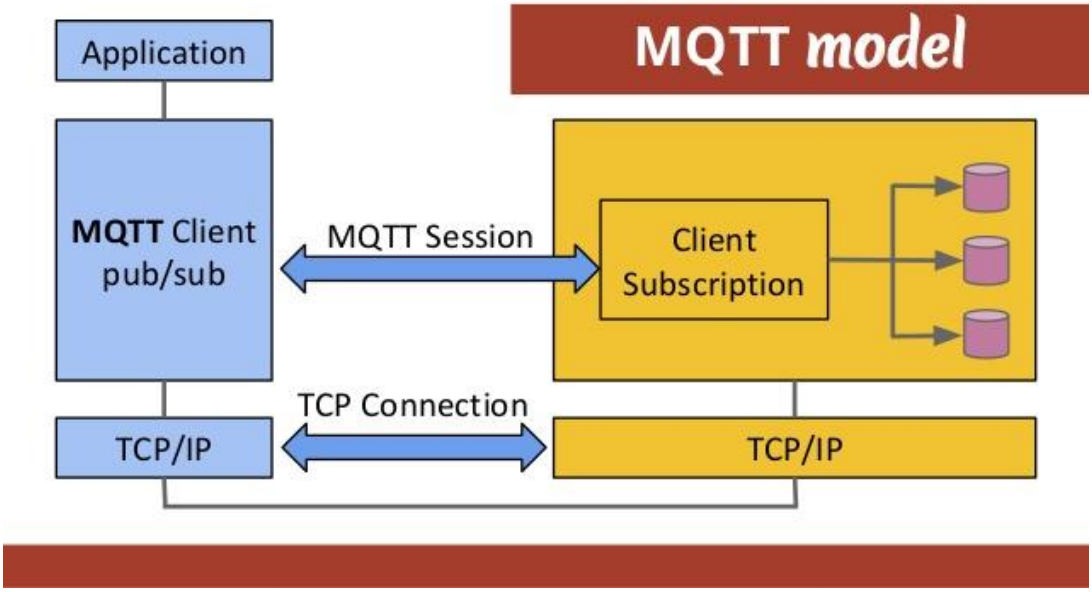
```
pip install pushbullet.py
```

3.5 MQTT protocol

Το MQTT (MQ Telemetry Transport - Message Queue Telemetry Transport) [28] [29] αποτελεί ένα ελαφρύ και απλοποιημένο πρωτόκολλο επικοινωνίας, βασιζόμενο στην λειτουργία publish-subscribe messaging. Αποτελεί ISO standard (ISO/IEC PRF 20922) [30] και από το 2013 αποτελεί και OASIS standard και τοποθετείται λειτουργικά πάνω από το TCP/IP protocol.

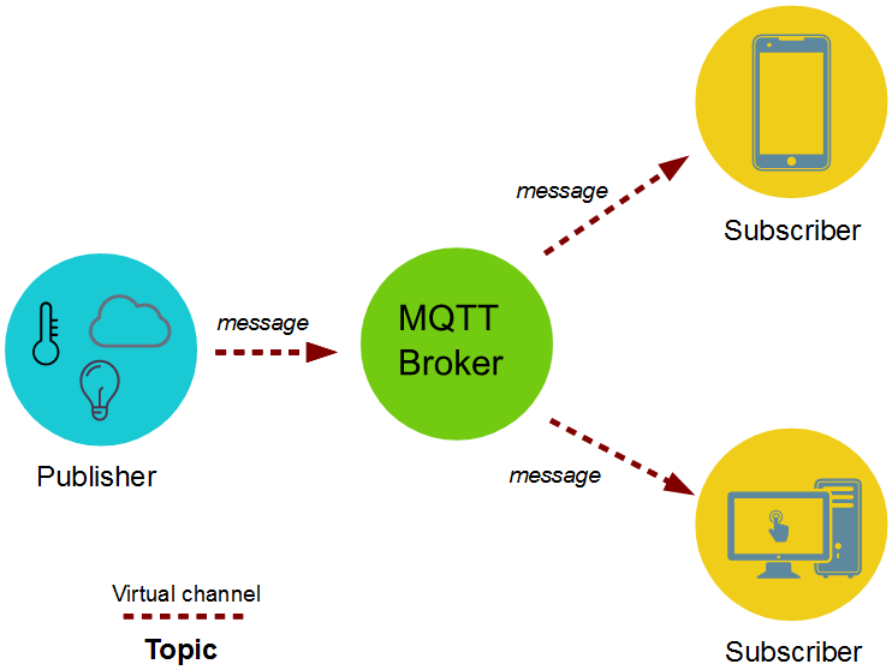


MQTT layer over TCP/IP



MQTT model

Είναι σχεδιασμένο να ελαχιστοποιεί το απαιτούμενο bandwidth επικοινωνίας και τις απαιτήσεις συστήματος αυξάνοντας την αξιοπιστία και την πιθανότητα επιτυχημένης μετάδοσης μηνύματος κάνοντάς το ιδανικό για χρήση σε "machine-to-machine" (M2M) ή "Internet of Things" δίκτυα όπου το bandwidth είναι περιορισμένο, οι καθυστερήσεις είναι μεγάλες, τα δίκτυα κάλυψης δεν είναι αξιόπιστα και η κατανάλωση ενέργειας πρέπει να είναι η μικρότερη δυνατή.



Η πρώτη μορφή του δημοσιοποιήθηκε το 1999 από τους Andy Stanford-Clark (IBM) και Arlen Nipper (Arcom- Eurotech) [31].

Το MQTT-SN αποτελεί μια μορφοποίηση του κυρίως πρωτοκόλλου απευθυνόμενο σε embedded non-TCP/IP συσκευές δικτύου όπως το ZigBee.

MQTT vs MQTT-S

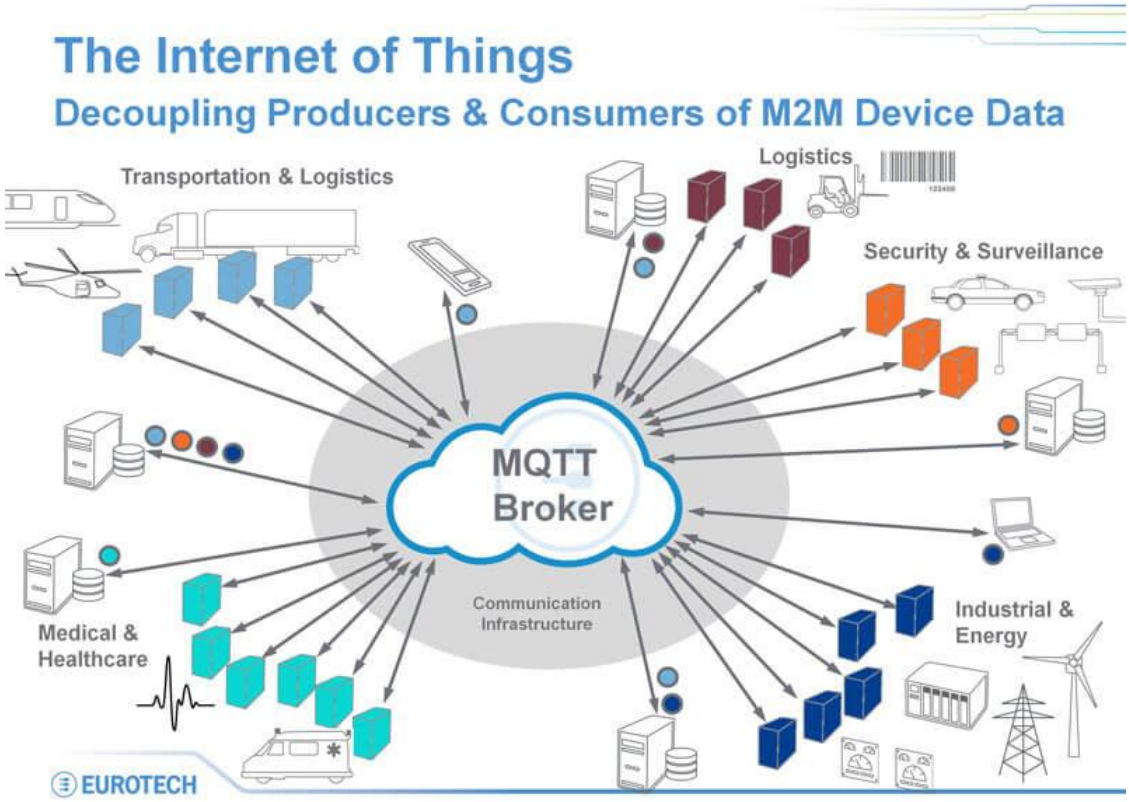
	MQTT	MQTT-S
Transport type	Reliable point to point streams	Unreliable datagrams
Communication	TCP/IP	Non-IP or UDP
Networking	Ethernet, WiFi, 3G	ZigBee, Bluetooth, RF
Min message size	2 bytes - PING	1 byte
Max message size	≤ 24MB	< 128 bytes (*)
Battery-operated		√
Sleeping clients		√
QoS: -1 "dumb client"		√
Gateway auto-discovery & fallbacks		√

Σύγκριση του MQTT & MQTT-S

Την λειτουργία παράδοσης των μηνυμάτων (που γίνονται publish από κάποιο τερματικό) αναλαμβάνει ο broker αποστέλλοντας το εκάστοτε μήνυμα στους ενδιαφερόμενους clients

που έχουν κάνει εγγραφή (subscribe) στο συγκεκριμένο topic (one-to-many message distribution) σε πραγματικό χρόνο. Υποστηρίζεται και η ταυτοποίηση με username/password (από την έκδοση 3.1 και μετά) για αυξημένη ασφάλεια και η κωδικοποίηση του tunnel επικοινωνίας μέσω SSL (αυξάνοντας το network overhead). Η απλή μορφή του MQTT χρησιμοποιεί την TCP/IP πόρτα 1883 ενώ η over-SSL μορφή την πόρτα 8883. Ορίζονται οι εξής μέθοδοι –status- λειτουργίας του MQTT Broker [32]:

→ ***Connect***: Αναμένει την ολοκλήρωση της σύνδεσης με τον MQTT server.
→ ***Disconnect***: Αναμένει την ολοκλήρωση των ενεργειών από τον MQTT client και την αποσύνδεση του TCP/IP session.
→ ***Subscribe***: Κάνει εγγραφή σε συγκεκριμένο topic.
→ ***UnSubscribe***: Αιτείται την αποδέσμευση της εγγραφής του client από ένα ή περισσότερα topics.
→ ***Publish***: Μετάδοση του μηνύματος στο topic από τον client και επαναμετάδοσή του από τον Broker στους listener του συγκεκριμένου topic.



Η επιτυχία της μετάδοσης μηνυμάτων βασίζεται σε 3 επίπεδα QoS:

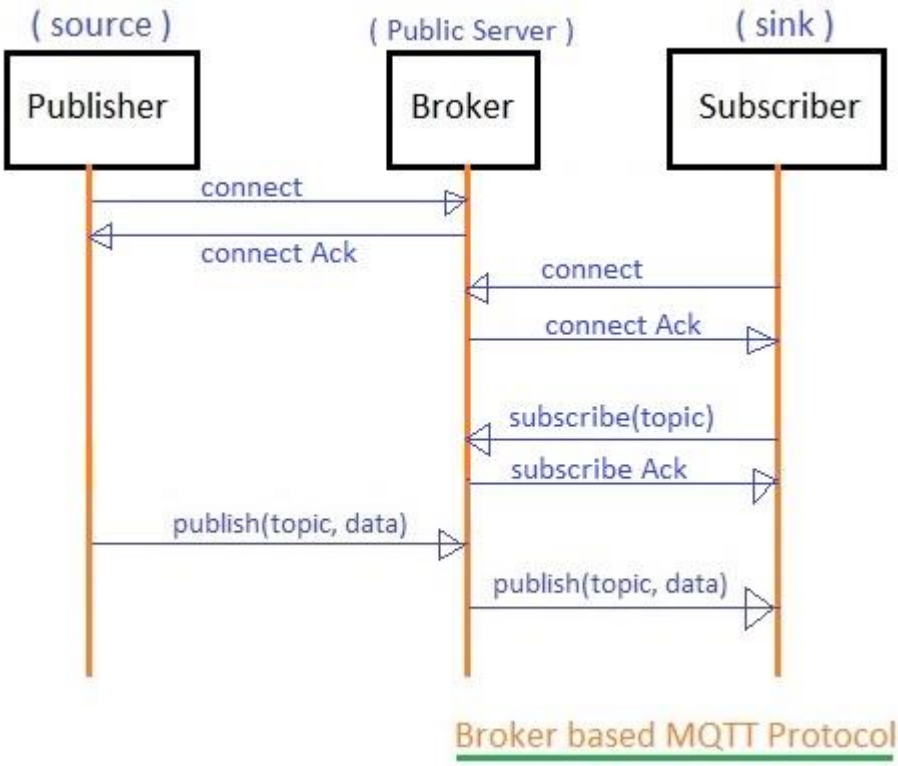
→ ***"At most once"***: Τα μηνύματα παραδίδονται βασιζόμενα στην καλύτερη δυνατή προσπάθεια του συστήματος –best efforts- και απώλειες μπορεί να συμβούν. Αυτό χρησιμοποιείται κυρίως σε μετάδοση δεδομένων από sensors όπου η απώλεια

ενδιάμεσης πληροφορίας δεν μας επηρεάζει αφού η επόμενη μετάδοση θα ξεκινήσει άμεσα.

→ ***"At least once"***: Επιβεβαίωση παραλαβής μηνύματος με ενδεχόμενο ύπαρξης duplicate μηνύματος από επαναμετάδοση σε ενδιάμεσο χρόνο.

→ ***"Exactly once"***: Επιβεβαίωση παραλαβής μόνο ενός αντίτυπου μηνύματος (χρήση κυρίως σε billing συστήματα όπου απώλειες ή διπλότυπα μηνυμάτων μπορεί να οδηγήσουν σε λανθασμένες χρεώσεις).

Σχηματική απεικόνιση λειτουργίας του MQTT πρωτοκόλλου βασιζόμενο σε broker:



Για την παρούσα εργασία, ως κύριο topic επιλέξαμε την ονομασία ***"Security_Area_MSc/"*** με sub-topic το *"/1"* που συμβολίζει την περιοχή 1 στην οποία θα εγκατασταθεί το πλέγμα από αισθητήρες και κάμερες. Έτσι η συνολική ονομασία του topic είναι: ***"Security_Area_MSc/1/"*** και η κάθε κάμερα/αισθητήρας θα κάνει publish σε ξεχωριστό, δικό του, sub-topic για να είναι εφικτός ο έλεγχος από ποια πηγή έρχεται το alert και να γίνεται και ορθά η ενημέρωση προς τον τελικό χρήστη. *Π.χ.* η κάμερα 2 σε περίπτωση ανίχνευσης κίνησης θα κάνει publish την τιμή «1» στο topic *"Security_Area_MSc/1/2"* και αντιστοίχως η κάμερα 1 στο topic *"Security_Area_MSc/1/1"* κλπ.

Σημείωση: Αν θέλουμε να εγγραφούμε (και να ακούμε) σε όλα τα sub-topic αυτόματα χωρίς να χρειάζεται να κάνουμε εγγραφή για καθένα χωριστά, αρκεί να κάνουμε εγγραφή στο ***"Security_Area_MSc/1/#"*** (η «#» καθορίζει όλα τα sub-topic)

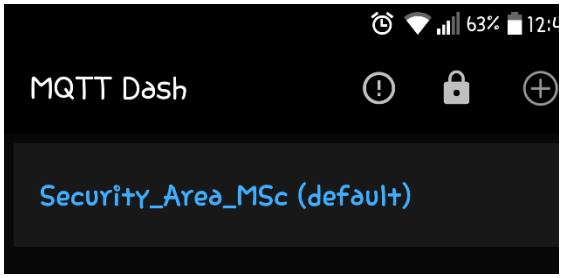
Τα βήματα για την εγκατάσταση του MQTT στο Raspberry Pi:

```
pip install paho-mqtt
```

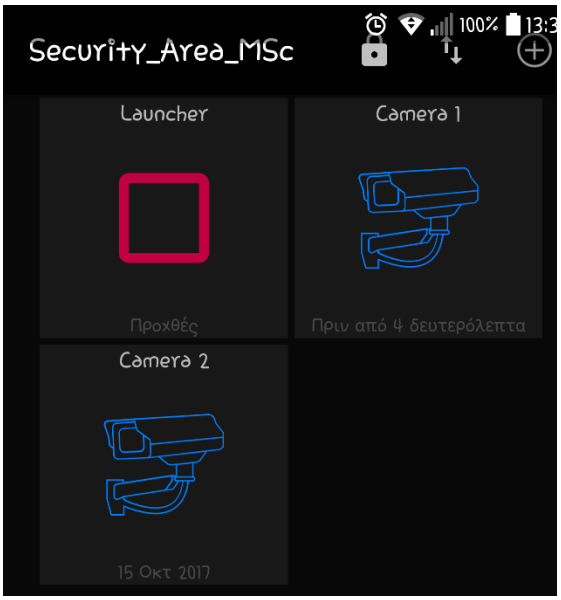
Τέλος, για την ενεργοποίηση/απενεργοποίηση του συστήματος, χρησιμοποιούμε το sub-topic **"Security_Area_MSc/1/0"** στο οποίο εγγράφουμε την τιμή «1» για εκκίνηση-arming και «0» για απενεργοποίηση- disarming του συστήματος.

3.6 MQTT Dash

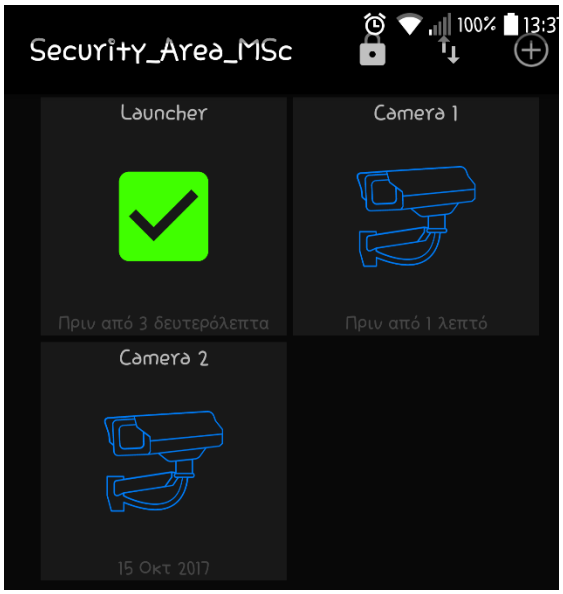
Ως dashboard ελέγχου του εσωτερικού συστήματος για τον τελικό χρήστη χρησιμοποιήσαμε την εφαρμογή για Android MQTT Dash. Η συγκεκριμένη εφαρμογή δίνει την δυνατότητα προσθήκης γραφικών στοιχείων ελέγχου (πχ buttons, labels) ώστε ο τελικός χρήστης να μπορεί να κάνει Arming-Disarming του εσωτερικού συστήματος, να δει (εκτός από το notification) ποια κάμερα εντόπισε κίνηση (και πατώντας πάνω της να ανοίξει στον browser το live streaming) χρησιμοποιώντας το MQTT πρωτόκολλο. Κάθε στοιχείο ελέγχου προγραμματίζεται να ακούει/διαβάζει/γράφει από/σε συγκεκριμένο topic του MQTT έτσι ώστε να παρέχει την ευκολία της εγκατάστασης/τροποποίησης ανάλογα με την εκάστοτε εφαρμογή του συστήματος ασφαλείας:



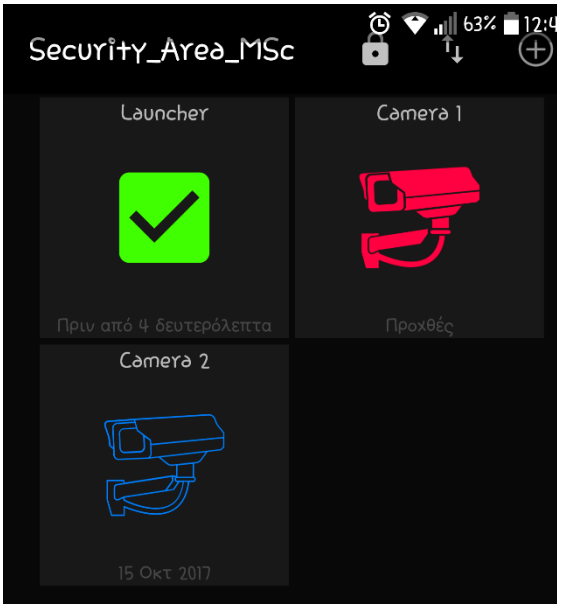
Topic εγγραφής του app



Τοποθέτηση κουμπιών ελέγχου

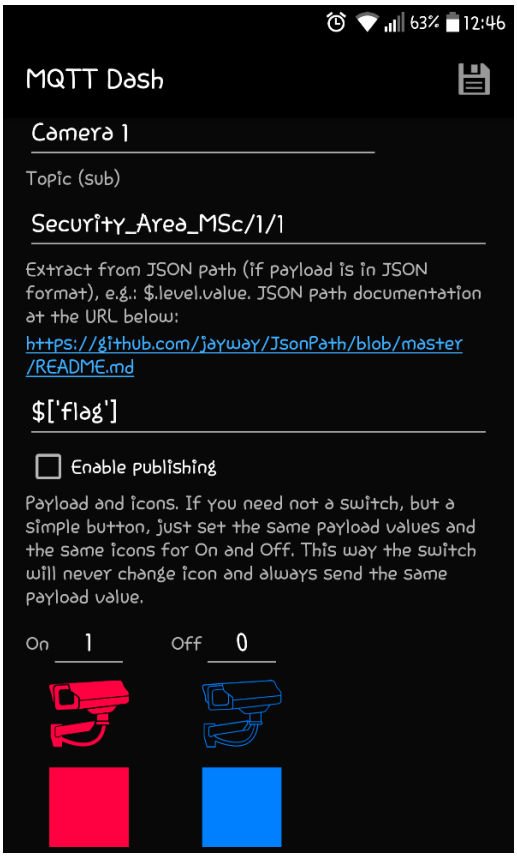


Arming/Disarming από το "Launcher"

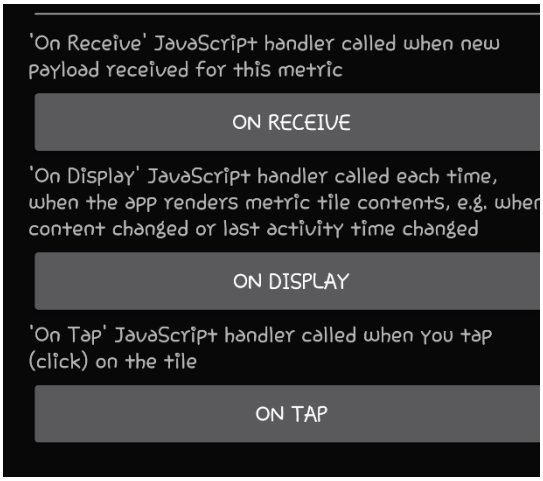


Armed με εντοπισμό κίνησης από κάμερα

Ρυθμίσεις του πλήκτρου Arming/Disarming



Ρυθμίσεις του εικονιδίου της εκάστοτε κάμερας



Συνέχεια ρυθμίσεων του εικονιδίου της κάμερας

Άνοιγμα του streaming URL της κάμερας κατά το πάτημα

3.7 JSON

Το JSON (JavaScript Object Notation) [33] [34] [35] είναι ένα πρότυπο ανταλλαγής δεδομένων που χαρακτηρίζεται από την ευκολία διαχείρισης από τους ανθρώπους. Είναι βασισμένο πάνω σε ένα υποσύνολο της γλώσσας προγραμματισμού JavaScript, Standard ECMA-262 (Έκδοση 3η - Δεκέμβριος 1999) [36] και είναι εύκολο για τις μηχανές να το αναλύσουν (parse) και να το παράγουν (generate). Το JSON είναι ένα πρότυπο κειμένου το οποίο είναι τελείως ανεξάρτητο από γλώσσες προγραμματισμού αλλά χρησιμοποιεί πρακτικές (conventions) οι οποίες είναι γνωστές στους προγραμματιστές της οικογένειας προγραμματισμού C, συμπεριλαμβανομένων των C, C++, C#, Java, JavaScript, Perl, Python [37], και πολλών άλλων. Αυτές οι ιδιότητες κάνουν το JSON μια ιδανική γλώσσα ανταλλαγής δεδομένων.

Το JSON είναι χτισμένο σε δύο δομές [38]:

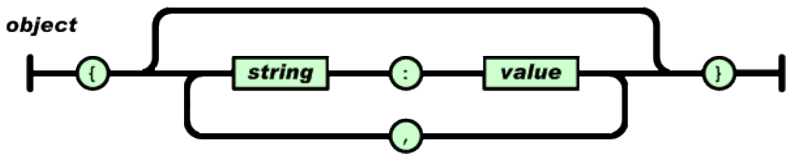
→ Μια συλλογή από ζευγάρια ονομάτων/τιμών. Σε διάφορες γλώσσες προγραμματισμού, αυτό αντιλαμβάνεται ως ένα object, καταχώριση, δομή, λεξικό, πίνακα hash (hash table), λίστα κλειδιών, ή associative πίνακα.

→ Μία ταξινομημένη λίστα τιμών. Στις περισσότερες γλώσσες προγραμματισμού, αυτό αντιλαμβάνεται ως ένας πίνακας (array), διάνυσμα, λίστα, ή ακολουθία.

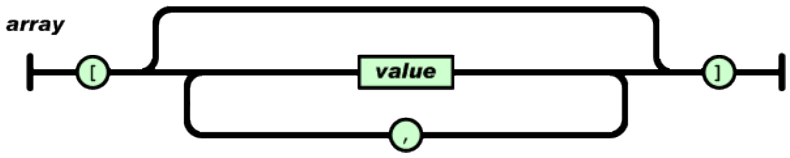
Αυτά είναι τα universal data structures. Ουσιαστικά όλες οι μοντέρνες γλώσσες προγραμματισμού τα υποστηρίζουν με τον έναν ή τον άλλον τρόπο.

Στο JSON, ακολουθούνται οι μορφές:

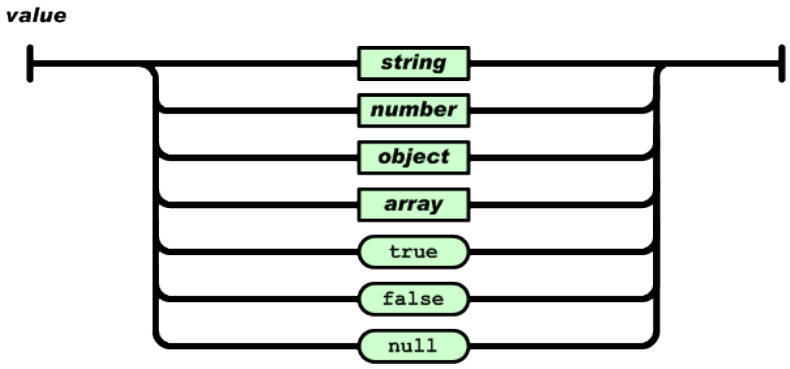
→ Ένα αντικείμενο (object) είναι ένα άτακτο σύνολο από ζευγάρια ονομάτων/τιμών. Ένα αντικείμενο (object) ξεκινάει με { (αριστερό άγκιστρο) και τελειώνει με } (δεξιό άγκιστρο). Κάθε όνομα ακολουθείται από : (άνω-κάτω τελεία) και τα ζευγάρια ονόματος/τιμής χωρίζονται από , (κόμμα):



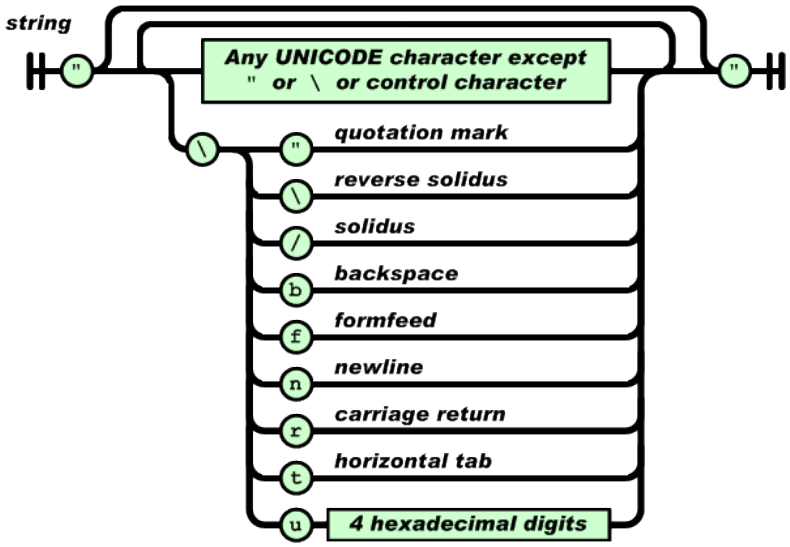
→ Ένας πίνακας (array) είναι μια συλλογή από τιμές σε σειρά. Ένας πίνακας (array) ξεκινάει με [(αριστερή αγκύλη) και τελειώνει με] (δεξιά αγκύλη). Οι τιμές χωρίζονται με , (κόμμα):



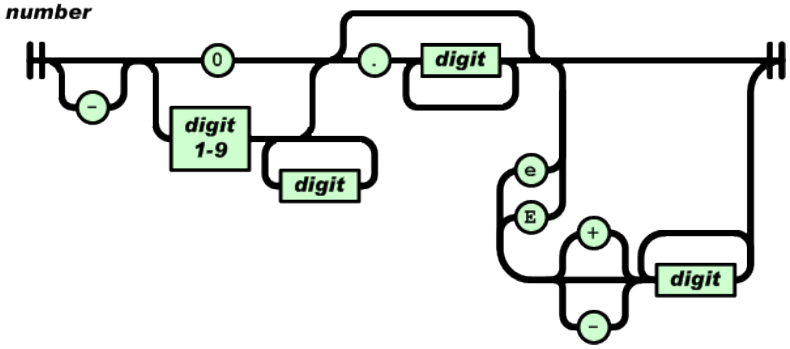
→ Μία τιμή μπορεί να είναι string μέσα σε διπλά quotes, ή αριθμός (number), ή true ή false ή null, ή αντικείμενο (object) ή πίνακας (array). Αυτές οι τιμές μπορεί να είναι και ανακατεμένες:



→ Ένα string είναι μια συλλογή από μηδέν ή περισσότερους Unicode χαρακτήρες, μέσα σε διπλά quotes, χρησιμοποιώντας αντίστροφους κάθετους "\" (backslash) για escapes. Ένας χαρακτήρας αντιπροσωπεύεται ως ένας μονός χαρακτήρας string. Ένα string μοιάζει πολύ σαν ένα C ή Java string.



→ Ένας αριθμός (number) μοιάζει πάρα πολύ με ένα C ή Java αριθμό (number), με την διαφορά πως τα οκταδικά και δεκαεξαδικά συστήματα δεν χρησιμοποιούνται.



Τα κενά (whitespace) μπορούν να εισαχθούν ανάμεσα σε οποιοδήποτε ζευγάρι tokens.

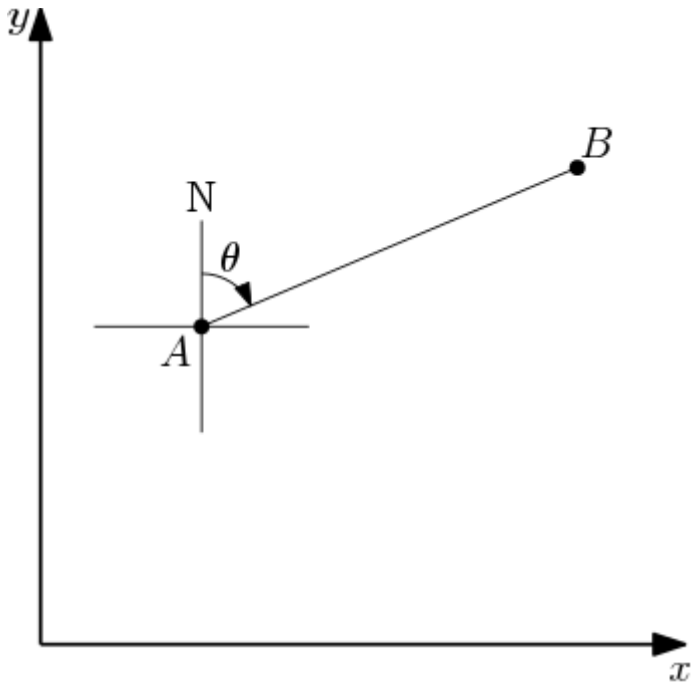
Στην παρούσα υλοποίηση χρησιμοποιήσαμε την δομή JSON για να μεταφέρουμε τις τιμές στον MQTT Broker και με την σειρά του σε όλους τους listeners του topic. Συγκεκριμένα η δομή του JSON που χρησιμοποιήσαμε είναι:

```
{"latitude":"38.048850","longitude":"23.788525","flag":"1"}
```

Συγκεκριμένα με τα string “latitude” & “longitude” δηλώνουμε τις συντεταγμένες της κάμερας του χρήστη που σηκώνει το alert έτσι ώστε να μπορέσει το εξωτερικό σύστημα να υπολογίσει την γωνία στρέψης “bearing”. Στο τελευταίο δεδομένο με το string “flag” δηλώνουμε αν η συγκεκριμένη κάμερα έχει ανιχνεύσει (“1”) ή όχι (“0”) κίνηση. Οπότε αν αρχικά ανιχνεύσει κίνηση θα ενημερώσει τον broker ότι το flag της είναι “1” ενώ όταν πάψει να ανιχνεύει κίνηση ενημερώνει τον broker ότι το flag της είναι “0”.

3.8 Angle (Bearing) μεταξύ 2 συντεταγμένων

Για την εύρεση της γωνίας θ [39] [40] μεταξύ 2 σημείων A(a1,a2) και B(b1,b2):



Ορίζουμε την γωνία θ που σχηματίζεται μεταξύ του κάθετου άξονα του σημείου A (Βορράς) και της ευθείας που ενώνει τα κέντρα των A και B πάντα με δεξιόστροφη κατεύθυνση ερχόμενης από τον κάθετο άξονα που ορίζει πάντα τον Βορρά:

$$(b_1, b_2) = (a_1 + r * \sin\theta\,, a_2 + r * \cos\theta)$$

όπου r είναι το μήκος της ευθείας που ενώνει τα κέντρα των A και B, οπότε:

$$\tan\theta = \frac{b_1 - a_1}{b_2 - a_2} \Rightarrow$$

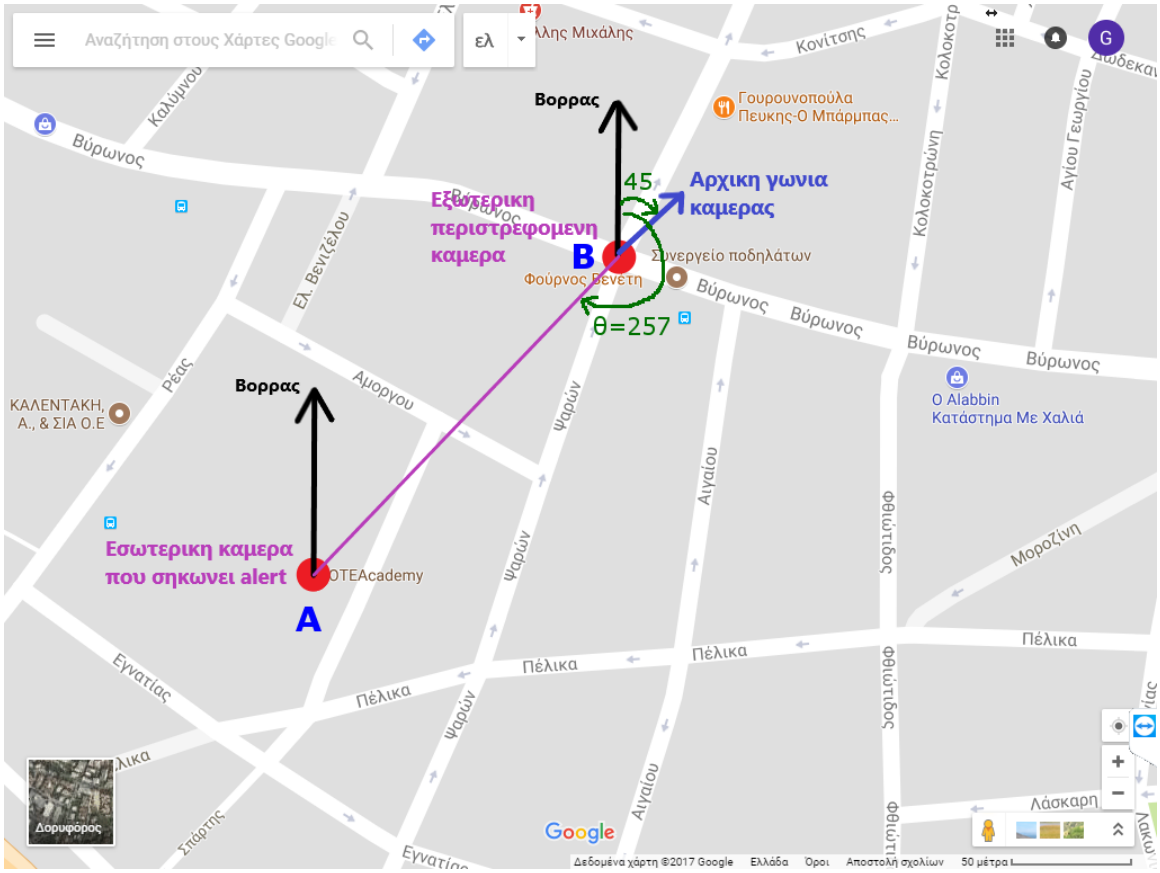
$$\hat{\theta} = arctan2(b_1 - a_1\,, b_2 - a_2) \in (-\pi, \pi]$$

Άρα η γωνία $\theta \in [0, 2\pi)$ είναι:

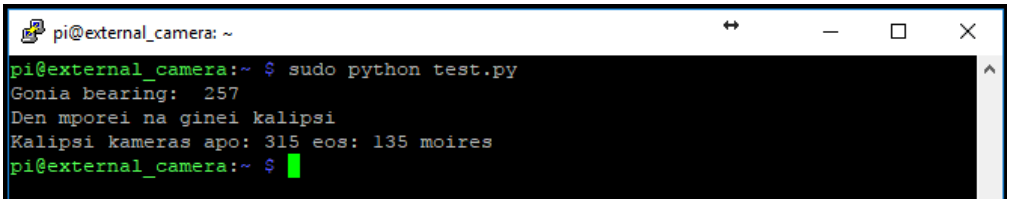
$$\theta = \begin{cases} \hat{\theta}, & \hat{\theta} \geq 0 \\ 2\pi + \hat{\theta}, & \hat{\theta} < 0 \end{cases}$$

3.8.1 Παράδειγμα υλοποίησης:

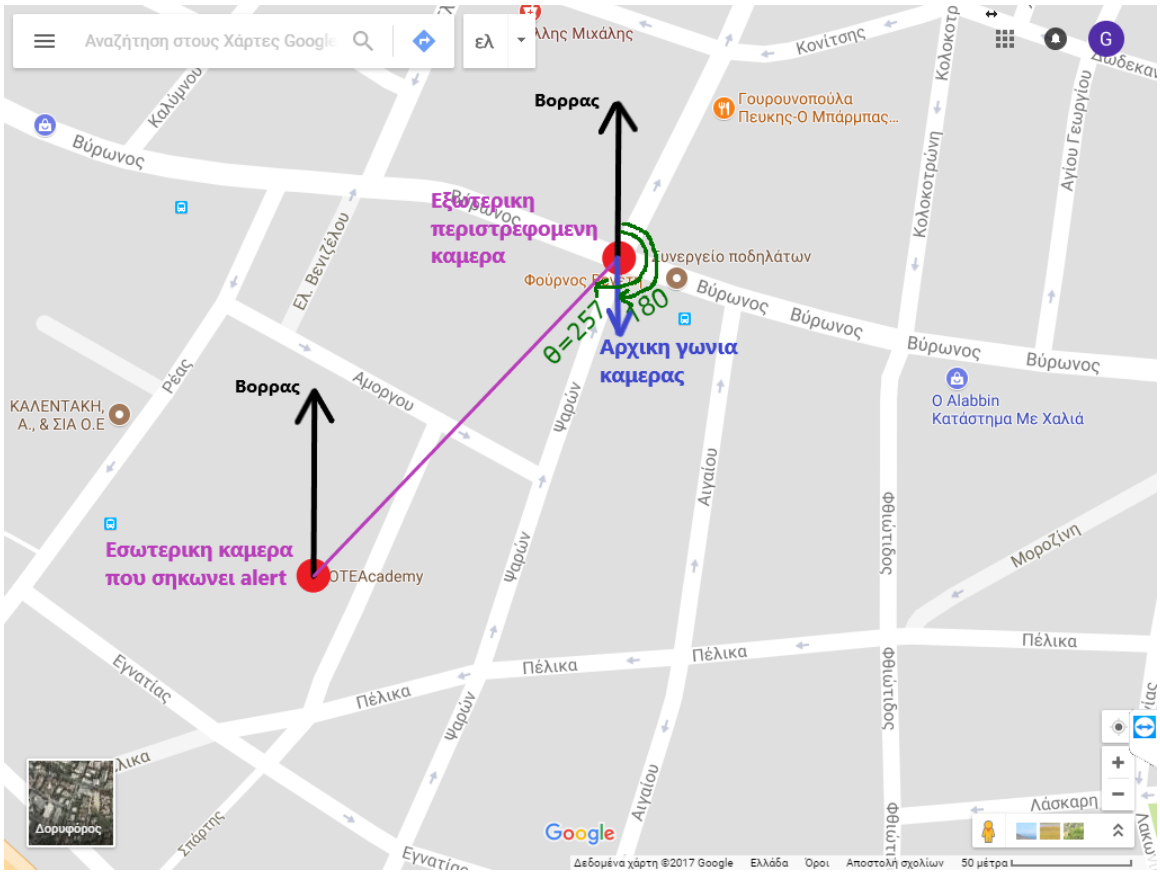
Έστω ότι το σύστημα συναγερμού (με το pir) που σηκώνει το alert βρίσκεται στο σημείο Α με συντεταγμένες Latitude-Longitude 38.049333-23.7872655 και το εξωτερικό σύστημα εντοπισμού κίνησης (κινούμενη κάμερα) βρίσκεται στο σημείο Β με συντεταγμένες 38.0497846-23.7897075 και η γωνία (ως προς τον Βορρά - bearing) που κοιτάζει η εξωτερική κάμερα είναι 45 μοίρες (μπλε βελάκι):

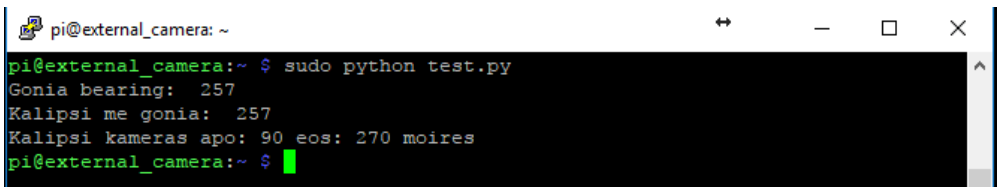


Αφού η εξωτερική μονάδα μπορεί να κινηθεί 90 μοίρες αριστερά και άλλες 90 δεξιά, θα έχει πεδίο κάλυψης από $45^o - 90^o = 315^o$ έως $45^o + 90^o = 135^o$. Για να περιστραφεί προς το σημείο Α χρειάζονται 257^o, οπότε δεν είναι δυνατή η κάλυψη:



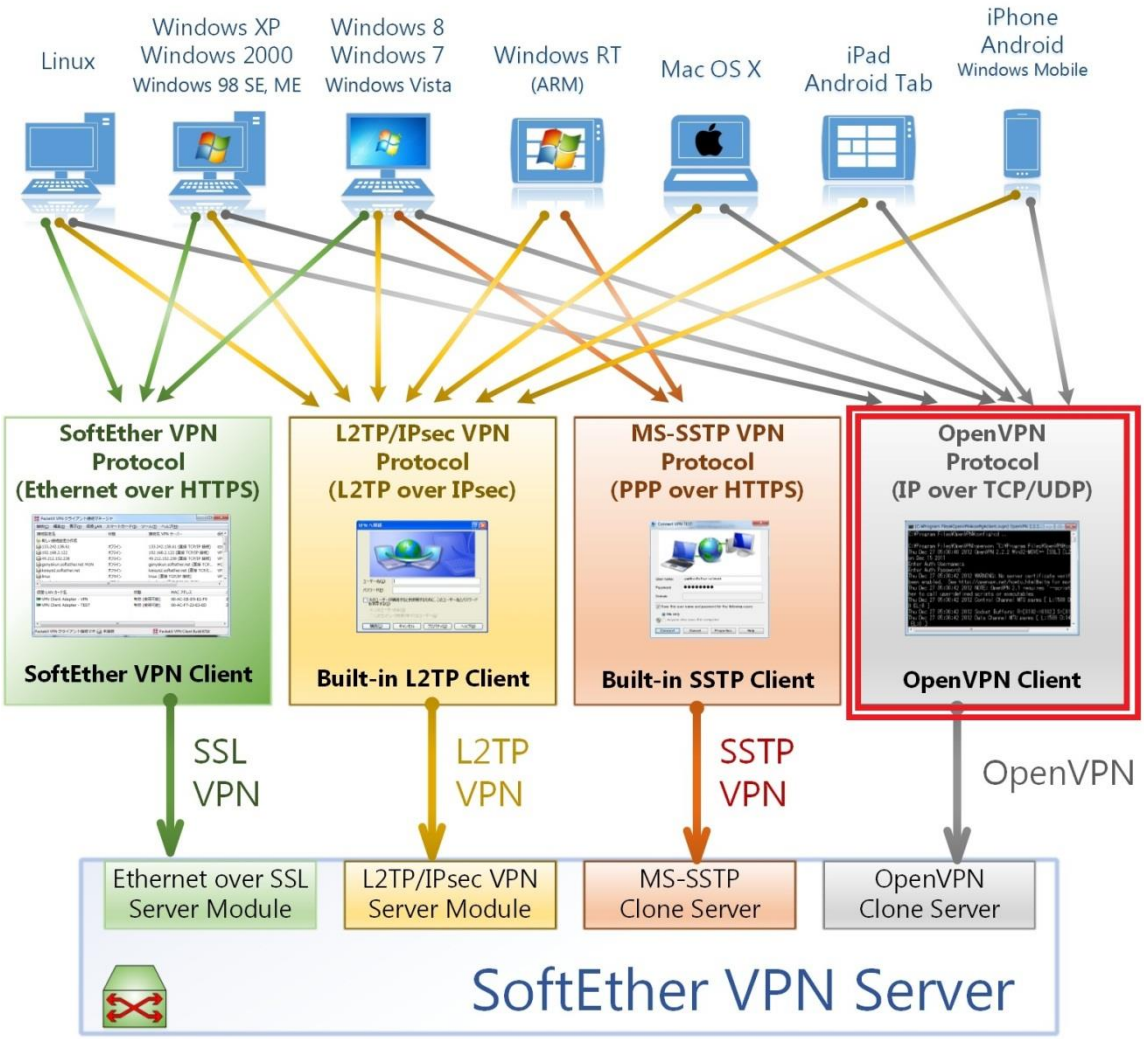
Αν η αρχική γωνία είναι στις 180 μοίρες (μπλε βελάκι), θα έχει πεδίο κάλυψης από $180^o - 90^o = 90^o$ έως $180^o + 90^o = 270^o$ μοίρες οπότε είναι δυνατή η κάλυψη:

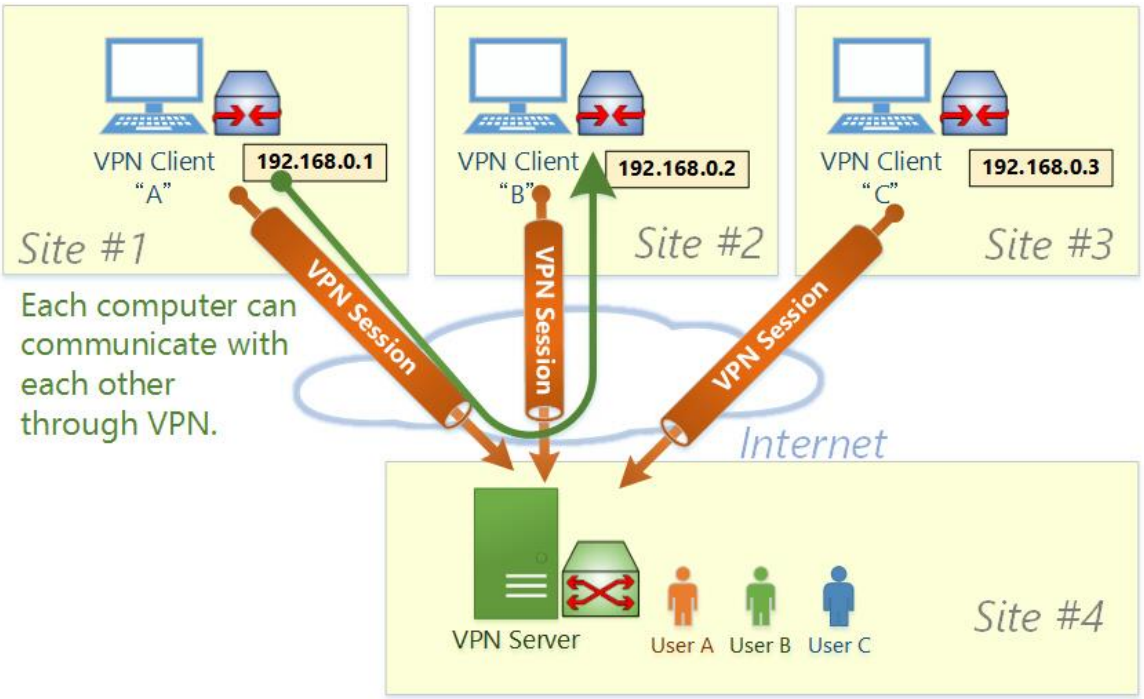




3.9 VPN – OpenVPN

Για την ασφάλεια της διασύνδεσης των τερματικών συσκευών με τον server του κέντρου ελέγχου και την κρυπτογράφηση των δεδομένων (push notification, MQTT, live streaming) χρησιμοποιείται VPN tunneling [41] [42] [43] μέσω του OpenVPN [44]. Η παροχή υπηρεσίας OpenVPN [45] υλοποιήθηκε με χρήση του (freeware) SoftEther VPN [46] server:





Τα βήματα για την εγκατάσταση του OpenVPN client στο Raspberry Pi:

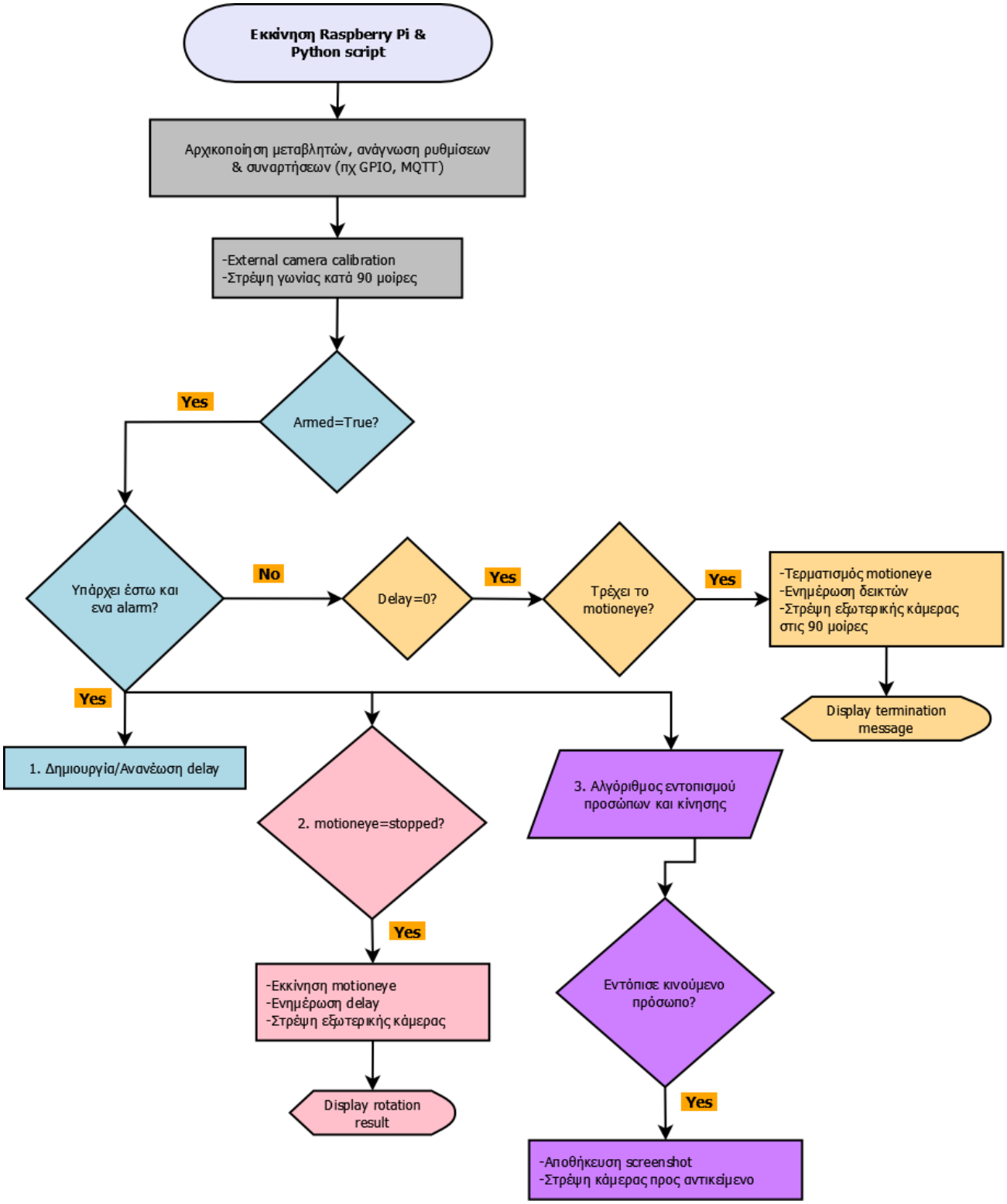
```
sudo apt-get install openvpn
```

Οπότε το κάθε τερματικό λαμβάνει 2 διευθύνσεις IP, μια από το τοπικό δίκτυο στο οποίο συνδέεται και άλλη μια από την ζεύξη VPN. Τόσο στα push notification της εσωτερικής μονάδας (streaming link) όσο και στην σύνδεση της εξωτερικής μονάδας με το κεντρικό πίνακα ελέγχου των streaming χρησιμοποιούνται οι IP από το VPN δίκτυο.

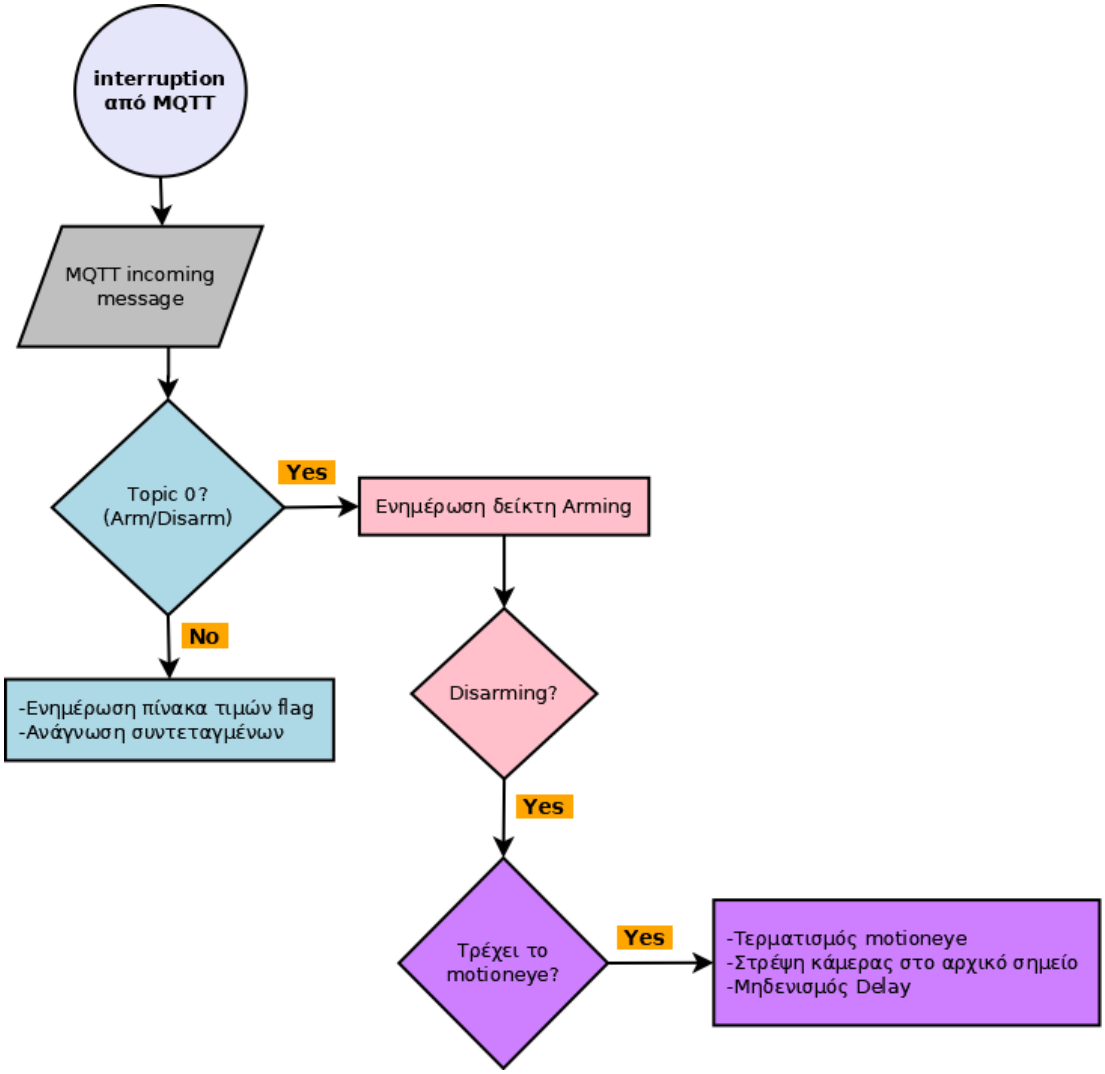
Κεφάλαιο 4 Υλοποίηση συστήματος – Πηγαίος Κώδικας

4.1 Εξωτερική μονάδα

Το διάγραμμα ροής του κώδικα είναι:



Σε περίπτωση που έρθει μήνυμα από τον MQTT Broker γίνεται interruption:


interruption από MQTT
MQTT incoming message
Topic 0? (Arm/Disarm)
Yes
Ενημέρωση δείκτη Arming
No
-Ενημέρωση πίνακα τιμών flag
-Ανάγνωση συντεταγμένων
Disarming?
Yes
Τρέχει το motioneye?
Yes
-Τερματισμός motioneye
-Στρέψη κάμερας στο αρχικό σημείο
-Μηδενισμός Delay

4.1.1 Ανάλυση κώδικα εξωτερικής μονάδας (external.py):

Αρχικά κάνουμε εισαγωγή των απαραίτητων βιβλιοθηκών που θα χρησιμοποιήσουμε:

```
import os
import sys
import time
import datetime
import argparse
import datetime
import imutils
import cv2
import dropbox
import paho.mqtt.client as mqtt
import numpy as np #Gia tous pinakes
import math #Gia prakseis pinakon
import signal #Gia keyboard interruption
import json
```

Έπειτα γίνεται ανάγνωση των ρυθμίσεων λειτουργίας της μονάδας από το αρχείο settings.txt που θα πρέπει να βάλουμε στον ίδιο φάκελο με το αρχείο του κώδικα. Μας δίνεται η δυνατότητα να ορίσουμε:

- → τον αύξοντα αριθμό της κάμερας.
- → τον συνολικό αριθμό καμερών στην περιοχή κάλυψης (δύναται να βάλουμε αρκετά μεγάλο αριθμό ώστε να υπάρχει συνεχόμενη λειτουργία ακόμα και σε μελλοντική αύξηση του αριθμού των καμερών/μονάδων χωρίς να χρειαστεί να του φορτωθεί εκ νέου το αρχείο ρυθμίσεων και να γίνει επανεκκίνηση της μονάδας).
- → το topic του MQTT που θα ακούνε και θα κάνουν publish όλες οι μονάδες.
- → Token πρόσβασης του script στο Dropbox της επιλογής μας για αποθήκευση media.
- → συντεταγμένες της μονάδας Latitude-Longitude. Συνήθως αποτελεί τις συντεταγμένες του χώρου τοποθέτησης (πχ στύλος) και μπορεί είτε να βρεθεί από το Google maps είτε να μετρηθεί με ακρίβεια (πχ GPS) κατά την τοποθέτηση.
- → Την επιθυμητή γωνία που θα είναι αρχικά στραμμένη -default- η εξωτερική κινούμενη κάμερα. Αυτό μπορεί να μετρηθεί με smartphone κατά την τοποθέτηση της κάμερας και γυρνώντας στην κατεύθυνση που κοιτάζει η κάμερα, μέσω application, μας βγάζει τις μοίρες ως προς Βορρά. Αυτό βοηθάει στον υπολογισμό της στρέψης της κάμερας αρχικά όταν υπάρξει alarm.

```
file=open("settings-external.txt","r")
with file:
      content = file.readlines() #Diavazei oles tis grammes kai tis
vazei ston pinaka content
content = [x.strip() for x in content] #Afairei ola ta special
character apo tin ka8e line tou pinaka content
file.close()
index=int(content[0]) #index kameras
total=int(content[1]) #sinolikos arithmos kameron
```

```
topic=content[2] #topic tou broker
dropboxhash=content[3] #token tou Dropbox
latitude=float(content[4]) #latitude kameras
longtitude=float(content[5]) #longtitude kameras
bearing=int(content[6]) #Gonia pou einai strammeni i kamera
```

Αρχικοποίηση του Dropbox object:

```
dropbox=dropbox.Dropbox(dropboxhash)
```

Αρχικοποίηση μεταβλητών:

```
matrix=np.zeros(shape=(1,total)) #Arxikos pinakas me toses stiles
oses o arithmos ton kameron
launcher=False #Arming/Disarming
diktis2=False #Diktis an trexei to motioneye
framenum=0 #metritis gia ta filename ton images
diktis=False #Diktis gia sosimo eikonon apo entopismo
goniax= bearing
xronos=0
moires=6
diktis4=2 #den 8elo na einai 0 h 1
```

Αρχικοποίηση του keyboard interrupt listener και συνάρτηση εξόδου σε περίπτωση που πατηθούν τα πλήκτρα Ctrl+C για τερματισμό λειτουργίας. Για τον ομαλό τερματισμό:

→ κλείνει την σύνδεση με τον MQTT broker
→ κλείνει την κάμερα
→ σταματάει το motioneye
→ επαναφορά της κάμερας στην θέση 0^o

```
def interrupt_handler(signal, frame):
      print "\n\nTerminating....\n"
      client.loop_stop()
      client.disconnect()
      camera.release()
      cv2.destroyAllWindows()
      os.system('sudo systemctl stop motioneye')
      servo_set(5)
      sys.exit(0)
#Arxikopoiisi interrupt lisener
signal.signal(signal.SIGINT, interrupt_handler)
```

Αρχικοποίηση του MQTT Broker με χρήση του freeware broker test.mosquitto.org

```
client = mqtt.Client()
client.connect("test.mosquitto.org", 1883, 60)
```

```
client.subscribe(topic+"#") #Kanei subscribe ste ola ta sub-topics
client.on_message = on_message
client.loop_start()
```

Συνάρτηση interruption σε περίπτωση που έρθει μήνυμα από τον Broker:

→ Αν είναι από το κεντρικό topic για arming/disarming, κάνει ενημέρωση της μεταβλητής ελέγχου και σε περίπτωση disarming ελέγχει επιπλέον αν τρέχει το motioneye και το τερματίζει (για εξοικονόμηση πόρων) και ταυτόχρονα στρέφει την κάμερα στο αρχικό σημείο και μηδενίζει και το delay.

→ Αν είναι από topic κάποιας μονάδας ενημερώνει τους ανάλογους δείκτες ελέγχου και κρατάει τις συντεταγμένες της εν λόγω μονάδας που έκανε publish.

```
def on_message(client, userdata, msg):
      global j
      global m
      global matrix
      global push
      global launcher
      global diktis2
      global latitude2
      global longtitude2
      global xronos
      global bearing
      m=msg.topic
      if (int(m[-1])==0): #an ginei arming/disarming
            j=int(msg.payload)
            if (int(j)==1):
                  launcher=True #enimerosi tou boolean
            if (int(j)==0):
                  launcher=False #enimerosi tou boolean
                  if (diktis2==True): #termatizei to motioneye an
trexei
                        print "\n\nTermatismos MotionEye"
                        os.system('sudo systemctl stop motioneye')
                        diktis2=False
                        servo_set(bearing)
                        xronos=0
      if (int(m[-1])>0): #elegxei oti den itan to arming/disarming
            j=json.loads(str(msg.payload))
            latitude2=float(j["latitude"])
            longtitude2=float(j["longtitude"])
            j=int(j["flag"])
            matrix[0,int(m[-1])-1]=j #mono tote kanei enimerosi
pinaka timon topic
```

Δημιουργία φακέλου στην SD card του Raspberry στο home directory για σώσιμο τοπικά των media:

```
directory="screenshots/"
if not os.path.exists(directory):
      os.makedirs(directory)
```

```
      os.system('sudo chmod 0777 '+directory)
```

Συνάρτηση ελέγχου αριστερόστροφης κίνησης της κάμερας. Ελέγχει αν η τελική γωνία στρέψης ξεπερνά το ανώτατο όριο (175°) και καλεί την συνάρτηση στρέψης της κάμερας:

```
def aristera(gonia):
      global goniax
      goniax=goniax+gonia
      if goniax>=175: #orio mikrotero ton 180 moiron
            goniax=175
      servo_set(goniax)
```

Αντίστοιχα για δεξιόστροφα:

```
def deksia(gonia2):
      global goniax
      goniax=goniax-gonia2
      if goniax<=5: #orio megalitero ton 0 moiron
            goniax=5
      servo_set(goniax)
```

Συνάρτηση στρέψης της κάμερας. Μετατρέπει τις μοίρες σε milliseconds και στρέφει την κάμερα:

```
def servo_set(angle):
      angle=int(((float(angle)/9.0)+5)*10) #metatropi apo moires se
palmo miliseconds
      os.system('sudo echo 0=%i > /dev/servoblaster' %angle)
```

Συνάρτηση αποθήκευσης σε επιθυμητό path του Dropbox των screenshot που έχει καταγράψει η κινούμενη κάμερα:

```
def backup():
      global directory
      global filename
      file2=directory+filename
      filedropbox='/External-moving-1/'+filename
      with open(file2, 'rb') as f:
            try:
                  dropbox.files_upload(f.read(), filedropbox) #upload
sto Dropbox
            except Exception:
                  print "Dropbox error"
```

Συνάρτηση υπολογισμού γωνίας στρέψης κάμερας -ως προς Βορρά (bearing)- κατά την περίπτωση alarm προς την εσωτερική μονάδα που έκανε publish στον broker. Αφού υπολογίσει την γωνία bearing κάνει μετατροπή από $0^o \sim 360^o$ αντί για $-180^o \sim 180^o$ που παράγει η συνάρτηση atan2. Έπειτα υπολογίζει τα 2 άκρα στρέψης της κάμερας από την αρχική θέση εγκατάστασης ώστε να ελέγξει αν το σημείο που έγινε το alarm βρίσκεται εντός των ορίων οπτικής κάλυψης της κάμερας και να κάνει την στρέψη ή όχι. Λόγω του ότι μετά τις 359^o ο κύκλος ξαναμηδενίζει ($0^o = 360^o$) εμφανίζεται πρόβλημα ελέγχου στα τεταρτημόρια 1 & 4 οπότε και κάνουμε ξεχωριστούς ελέγχους των ορίων κάλυψης. Αντίθετα, στα τεταρτημόρια 2 & 3 ελέγχουμε με απλή σύγκριση τα όρια κάλυψης και αρκεί να καλύπτονται και οι 2 προϋποθέσεις (λογικό AND) ώστε το τελικό σημείο να βρίσκεται όντως εντός του οπτικού πεδίου:

```
def calc_bearing(pointA, pointB):
      global bearing
      global goniax
      global metrisi
      #if (type(pointA) != tuple) or (type(pointB) != tuple):
      #     raise TypeError("Only tuples are supported as arguments")
      lat1 = math.radians(pointA[0])
      lat2 = math.radians(pointB[0])
      diffLong = math.radians(pointB[1] - pointA[1])
      x = math.sin(diffLong) * math.cos(lat2)
      y = math.cos(lat1) * math.sin(lat2) - (math.sin(lat1)*
math.cos(lat2) * math.cos(diffLong))
      initial_bearing = math.atan2(x, y)
      # Now we have the initial bearing but math.atan2 return values
      # from -180 to + 180 which is not what we want for a compass
bearing
      # The solution is to normalize the initial bearing as shown
below
      initial_bearing = math.degrees(initial_bearing)
      teliko_bearing = (initial_bearing + 360) % 360
      teliko_bearing = int(round(teliko_bearing))
      print "Gonia bearing: ",teliko_bearing
      orio1=((bearing-90)+360)%360 #normalize os pros 360 moires
      orio2=((bearing+90)+360)%360
      if (bearing>=90 and bearing<=269): #2 kai 3 tetartimorio
strepsis
            if (teliko_bearing>=orio1 and teliko_bearing<=orio2):
                  print "Kalipsi me teliko bearing: ",teliko_bearing
                  print "Kalipsi kameras apo: %i eos: %i moires"
%(orio1,orio2)
                  print "Strofi: ",abs((teliko_bearing-orio1)-180)
                  goniax=abs((teliko_bearing-orio1)-180)
                  servo_set(goniax)
            else:
                  print "Den mporei na ginei kalipsi. Gonia bearing:
",teliko_bearing
                  print "Kalipsi kameras apo: %i eos: %i moires"
%(orio1,orio2)
      if (bearing>=270 and bearing<=359): #4 tetartimorio strepsis
            if (teliko_bearing>=orio1 or teliko_bearing<=orio2):
                  print "Kalipsi me teliko bearing: ",teliko_bearing
                  print "Kalipsi kameras apo: %i eos: %i moires"
%(orio1,orio2)
                  if (teliko_bearing>=orio1):
                        print "Strofi: ",abs((teliko_bearing-orio1)-
```

```
180)
                        goniax=abs((teliko_bearing-orio1)-180)
                        servo_set(goniax)
                  if (teliko_bearing<=orio2):
                        print "Strofi: ",abs(orio2-teliko_bearing)
                        goniax=abs(orio2-teliko_bearing)
                        servo_set(goniax)
            else:
                  print "Den mporei na ginei kalipsi. Gonia bearing:
",teliko_bearing
                  print "Kalipsi kameras apo: %i eos: %i moires"
%(orio1,orio2)
      if (bearing>=0 and bearing<=89): #1 tetartimorio strepsis
            if (teliko_bearing>=orio1 or teliko_bearing<=orio2):
                  print "Kalipsi me teliko bearing: ",teliko_bearing
                  print "Kalipsi kameras apo: %i eos: %i moires"
%(orio1,orio2)
                  if (teliko_bearing>=orio1):
                        print "Strofi: ",abs((teliko_bearing-orio1)-
180)
                        goniax=abs((teliko_bearing-orio1)-180)
                        servo_set(goniax)
                  if (teliko_bearing<=orio2):
                        print "Strofi: ",abs(orio2-teliko_bearing)
                        goniax=abs(orio2-teliko_bearing)
                        servo_set(goniax)
            else:
                  print "Den mporei na ginei kalipsi. Gonia bearing:
",teliko_bearing
                  print "Kalipsi kameras apo: %i eos: %i moires"
%(orio1,orio2)
      metrisi=1
```

Κύρια ρουτίνα προγράμματος. Όλες οι εντολές εκτελούνται μόνο αν έχει γίνει armed το σύστημα.

- → Αρχικά κάνει import τα βιομετρικά δεδομένα ελέγχου και εντοπισμού προσώπων σε κάθε frame. Αλλάζοντας το xml αρχείο με τα βιομετρικά δεδομένα μπορούμε να εντοπίσουμε και άλλα αντικείμενα όπως σώμα, αυτοκίνητα, μάτια κτλ.
- → Calibration της κάμερας και τελική στρέψη στις 90°
- → Εκκίνηση του opencv για διάβασμα των frames από την usb κάμερα
- → Αν υπάρξει alarm δημιουργεί delay 1 ώρας ώστε αν σταματήσει το alarm να συνεχίσει να κάνει monitoring η κάμερα για επαυξημένο έλεγχο.
- → Αν υπάρχει alarm και δεν τρέχει το motioneye, το εκκινεί και ανανεώνει το delay και υπολογίζει την γωνία στρέψης (αφού τώρα εκκινεί το motioneye δεν είχε γίνει στρέψη σε συμβάν) ώστε να στρέψει την κάμερα.
- → Αν δεν υπάρχει κανένα alarm τότε κάνει τερματισμό του motioneye μόνο αν έχει τελειώσει το delay και τρέχει το process του motioneye και στρέψει την κάμερα στις 90°
- → Αν υπάρχει alarm τότε τρέχει ο αλγόριθμος ελέγχου των frames της εξωτερικής κάμερας για εντοπισμό προσώπων και στρέψη της ανάλογα για την παρακολούθηση αυτών.

Ο αλγόριθμος που αναπτύξαμε λειτουργεί ως εξής:

→ αρχικά μετατρέπει το frame σε gray scale διαστάσεων 320*240 pixels για πιο γρήγορη επεξεργασία και ψάχνει μέσα σε κάθε frame βάσει των βιομετρικών δεδομένων από το xml αρχείο.

→ Αν εντοπίσει πρόσωπα σχεδιάζει γύρω τους πράσινο τετράγωνο και υπολογίζει το κέντρο του εκάστοτε τετραγώνου (με συντεταγμένες x,y σε pixels ως προς το frame) και έπειτα υπολογίζει το μέσο όρο όλων των κέντρων ώστε να υπάρχει πιο ομαλή κίνηση για καταγραφή όλων των προσώπων (αν δύναται).

→ Αν γίνει εντοπισμός προσώπων αποθηκεύει το screenshot

- Στο πρώτο screenshot ελέγχει την θέση του μέσου κέντρου των προσώπων ως προς την οριζόντιο άξονα του frame (σε pixels) και δεν κινείται. Έχουμε θέσει την ενδιάμεση περιοχή από 140~180 pixels (από τα συνολικά 320 πλάτους του frame) κατά την οποία σημαίνει ότι το πρόσωπο είναι στο κέντρο οπτικού πεδίου της κάμερας και ορίζουμε ελάχιστες μοίρες στρέψης.
- Στο δεύτερο screenshot βρίσκει αρχικά την διαφορά του τωρινού κέντρου με του πρώτου frame ώστε να δούμε αν κινήθηκε αριστερά (αρνητική διαφορά) ή αν κινήθηκε δεξιά (θετική διαφορά). Επίσης πριν κινήσουμε την κάμερα ελέγχουμε, σύμφωνα με την φορά κίνησης του προσώπου, σε ποια περιοχή του frame βρίσκεται ώστε αν βρίσκεται στην αριστερή πλευρά και κινείται δεξιά να μην κινηθεί γιατί θα βρίσκεται σε όλο το οπτικό πεδίο κάλυψης της κάμερας. Αντίστοιχα αν βρίσκεται στο δεξιό τμήμα του frame και κινείται αριστερά πάλι δεν κινείται. Κίνηση κάμερας έχουμε όταν πχ είναι στην αριστερή πλευρά του frame και κινείται αριστερά ή αντίστοιχα όταν είναι στην δεξιά και κινείται δεξιά, δηλαδή όταν πάει να βγει εκτός frame οπτικής κάλυψης. Επίσης λαμβάνουμε υπόψιν και την προηγούμενη κίνηση της κάμερας ώστε να αυξήσουμε τις μοίρες στρέψης (πχ αν είχε κινηθεί δεξιά η κάμερα και σύμφωνα με τους τωρινούς υπολογισμούς πρέπει να ξανα-κινηθεί δεξιά σημαίνει ότι το πρόσωπο κινείται με μεγάλη ταχύτητα είτε ότι οι μοίρες στρέψης δεν ήταν αρκετές ώστε να τοποθετήσουμε το κέντρο του προσώπου είτε στο κέντρο του frame είτε στην αντίθετη πλευρά).
- Αντίστοιχα με το πρώτο, στο τρίτο screenshot δεν κινείται και βρίσκει εκ νέου το κέντρο.
- Αντίστοιχα με το δεύτερο, στο τέταρτο screenshot κινείται ανάλογα με την ανάλογη τροποποίηση των μοιρών.

```
people_cascade =
cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
for num in range (5,180,5):
      servo_set(num)
for num in range (175,85,-5):
      servo_set(num)
servo_set(bearing)
#we are reading from webcam
camera = cv2.VideoCapture(1)
time.sleep(3)
# initialize the first frame in the video stream
```

```
(grabbed, frame) = camera.read()
print "\nTermatismos me Ctrl+C \n"
while True:
      if (launcher==True):
            if (int(matrix.sum())!= 0): #An iparxei alarm to delay
ananaonetai
                  xronos=int(round(time.time()))+3600
            if (int(matrix.sum())!= 0 and diktis2==False): #An den
trexei to motion kanei ekkinisi kai exei ginei entopismos kinisis apo
kapoia kamera
                  print "\n\nEkkinisi MotionEye"
                  os.system('sudo systemctl start motioneye')
                  diktis2=True
                  xronos=int(round(time.time()))+3600 #Ftiaxnei to
delay gia epipleon katagrafi se periptosi pou stamaisei to alarm
                  pointA=(latitude,longtitude)
                  pointB=(latitude2,longtitude2)
                  calc_bearing(pointA, pointB)
            if (int(matrix.sum())==0 and diktis2==True and
int(round(time.time()))>xronos): #An trexei idi to motion kai den
exoume pleon entopismo kinisis
                  print "\n\nTermatismos MotionEye" #apo kapoia
kamera kai teleiosei to delay, termatizei to motioneye
                  os.system('sudo systemctl stop motioneye')
                  diktis2=False
                  goniax=90 #Epistrofi kameras sto kentriko simeio
                  servo_set(goniax)
            if (diktis2==True):
                  (grabbed, frame) = camera.read()
                  frame=imutils.resize(frame, width=320, height=240)
                  gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
                  people=people_cascade.detectMultiScale(gray, 1.3,
5)
                  text = 0
                  temp=0
                  orizontio_kentro=0
                  for (x,y,w,h) in people:

      cv2.rectangle(frame,(x,y),(x+w,y+h),(0,255,0),1)
                        if temp==0:
                              orizontio_kentro=x+(w/2)
                        else:

      orizontio_kentro=(orizontio_kentro+(x+(w/2)))/2
                        temp=temp+1
                        diktis=True
                  text=temp
                  cv2.putText(frame, "Antikeimena: {}".format(text),
(10, 20), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 255), 2)
                  cv2.putText(frame,
datetime.datetime.now().strftime("%A %d %B %Y %I:%M:%S%p"), (10,
frame.shape[0] - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.35, (0, 0, 255), 1)
                  filename=datetime.datetime.now().strftime('%Y-%m-
%d_%H-%M-%S')+'_'+str(framenum)+'.jpg'
                  if diktis==True:

      cv2.imwrite(os.path.join(directory,filename),frame)
                        framenum=framenum+1
                        diktis=False #Ksanamidenizei ton dikti

                  ##-------------------------------------------------
```

```
------------------------------------------------------------------------
---------------------------------------
                        diktis3=True
                        if (metrisi==1 and diktis3==True):
                              if (orizontio_kentro>140 or
orizontio_kentro<180):
                                    moires=6
                              simeioA=orizontio_kentro
                              metrisi=2
                              diktis3=False
                        if (metrisi==2 and diktis3==True):
                              diafora=orizontio_kentro-simeioA
                              if (orizontio_kentro<140 or
orizontio_kentro>180):
                                    if (orizontio_kentro<140 and
diafora<0):
                                          if (diktis4==0):
                                                moires=2*moires
                                          aristera(moires)
                                          diktis4=0
                                    elif (orizontio_kentro<140 and
diafora>0):
                                          moires=3
                                    elif (orizontio_kentro>180 and
diafora>0):
                                          if (diktis4==1):
                                                moires=2*moires
                                          deksia(moires)
                                          diktis4=1
                                    elif (orizontio_kentro>180 and
diafora<0):
                                          moires=3
                              metrisi=3
                              simeioA=orizontio_kentro
                              diktis3=False
                              diafora2=diafora
                        if (metrisi==3 and diktis3==True):
                              simeioA=orizontio_kentro
                              metrisi=4
                              diktis3=False
                              if (orizontio_kentro>140 or
orizontio_kentro<180):
                                    moires=6
                        if(metrisi==4 and diktis3==True):
                              diafora=orizontio_kentro-simeioA
                              if (orizontio_kentro<140 or
orizontio_kentro>180):
                                    if (orizontio_kentro<140 and
diafora<0):
                                          if (diktis4==0):
                                                if
(abs(diafora)>abs(diafora2)):
                                                      moires=3*moires
                                                else:
                                                      moires=2*moires
                                          aristera(moires)
                                          diktis4=0
                                    elif (orizontio_kentro<140 and
diafora>0):
                                          moires=3
                                    elif (orizontio_kentro>180 and
```

```
diafora>0):
                                                    if (diktis4==1):
                                                          if
(abs(diafora)>abs(diafora2)):
                                                                moires=3*moires
                                                          else:
                                                                moires=2*moires
                                                    deksia(moires)
                                                    diktis4=1
                                              elif (orizontio_kentro>180 and
diafora<0):
                                                    moires=3
                                        metrisi=1
```

Δείγματα screenshots από την κινούμενη κάμερα κατά τον εντοπισμό προσώπου:

Επίσης στην μονάδα τρέχει http server που κάνει live streaming μορφή (mjpeg) τις εικόνες που έχει αποθηκεύσει στον φάκελο "screenshots" με τα αναγνωρισμένα πρόσωπα και αποτελείται από 2 αρχεία, το pymjpeg.py που μετατρέπει τα αρχεία jpeg σε mjpeg:

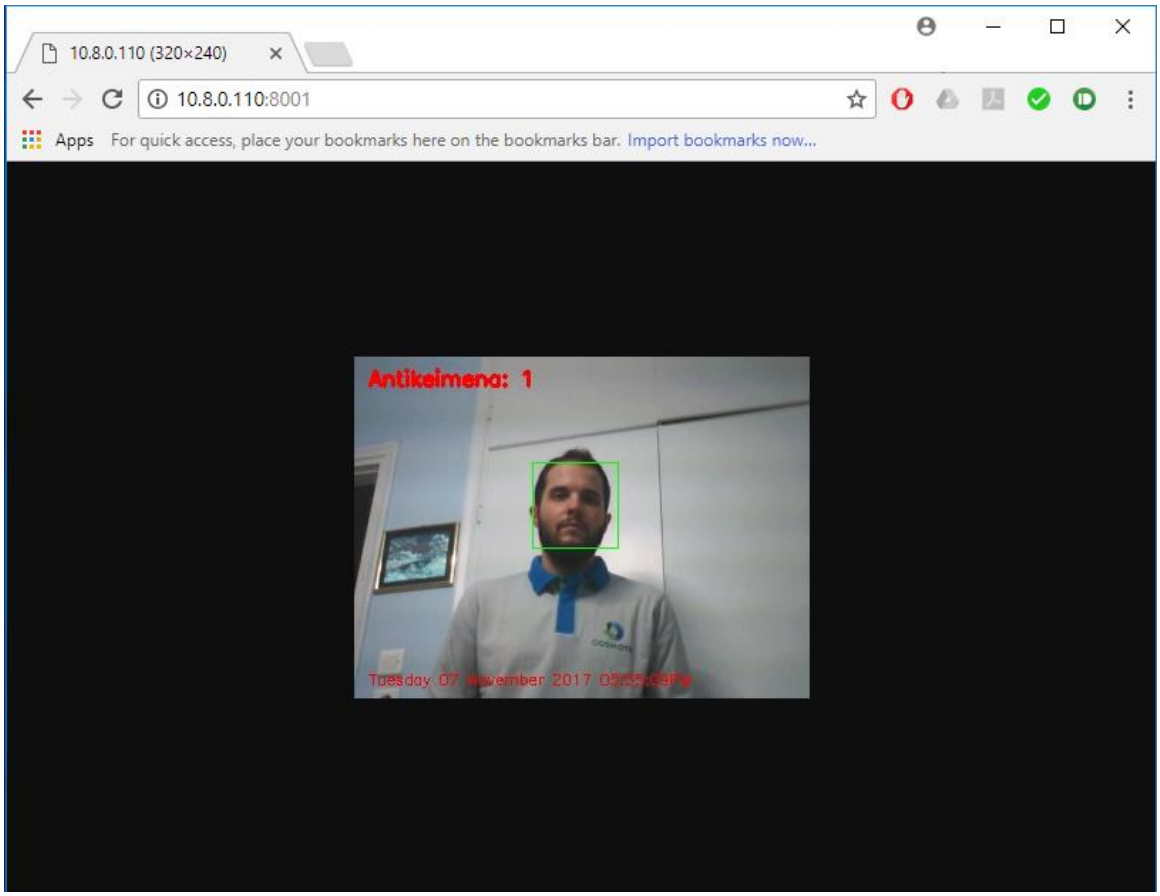
```
import os, time
boundary = '--boundarydonotcross'
def request_headers():
    return {
        'Cache-Control': 'no-store, no-cache, must-revalidate, pre-
check=0, post-check=0, max-age=0',
        'Connection': 'close',
        'Content-Type': 'multipart/x-mixed-replace;boundary=%s' %
boundary,
        'Expires': 'Mon, 3 Jan 2000 12:34:56 GMT',
        'Pragma': 'no-cache',
    }
def image_headers(filename):
    return {
        'X-Timestamp': time.time(),
        'Content-Length': os.path.getsize(filename),
        'Content-Type': 'image/jpeg',
    }
def image(filename):
    with open(filename, "rb") as f:
        byte = f.read(1)
        while byte:
            yield byte
            byte = f.read(1)
```

Και το stream.py που αρχικά καλεί το άνω αρχείο μετατροπής των εικόνων σε stream και έπειτα διαβάζει τα περιεχόμενα του φακέλου "screenshots" και εκκινεί τον server:

```
import pymjpeg
from glob import glob
import sys
from BaseHTTPServer import HTTPServer, BaseHTTPRequestHandler

class MyHandler(BaseHTTPRequestHandler):
```

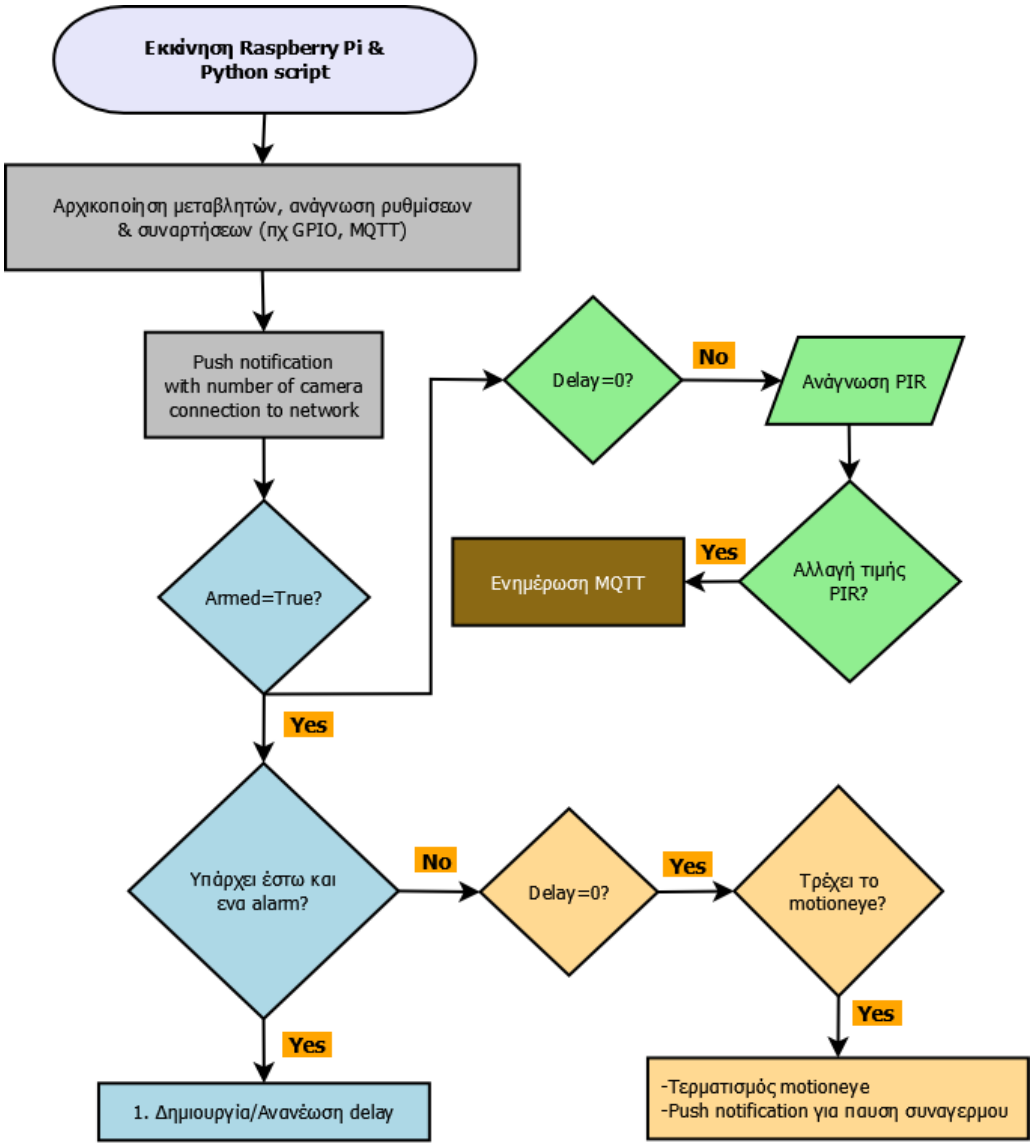
```
    def do_GET(self):
        self.send_response(200)
        # Response headers (multipart)
        for k, v in pymjpeg.request_headers().items():
            self.send_header(k, v)
        # Multipart content
        for filename in glob('screenshots/*'):
            # Part boundary string
            self.end_headers()
            self.wfile.write(pymjpeg.boundary)
            self.end_headers()
            # Part headers
            for k, v in pymjpeg.image_headers(filename).items():
                self.send_header(k, v)
            self.end_headers()
            # Part binary
            for chunk in pymjpeg.image(filename):
                self.wfile.write(chunk)
    def log_message(self, format, *args):
        return
httpd = HTTPServer(('', 8001), MyHandler)
httpd.serve_forever()
```

Έτσι ανοίγοντας τον browser και πατώντας την ip του vpn του συγκεκριμένου Raspberry στην πόρτα 8001 μας δείχνει το streaming των εικόνων:

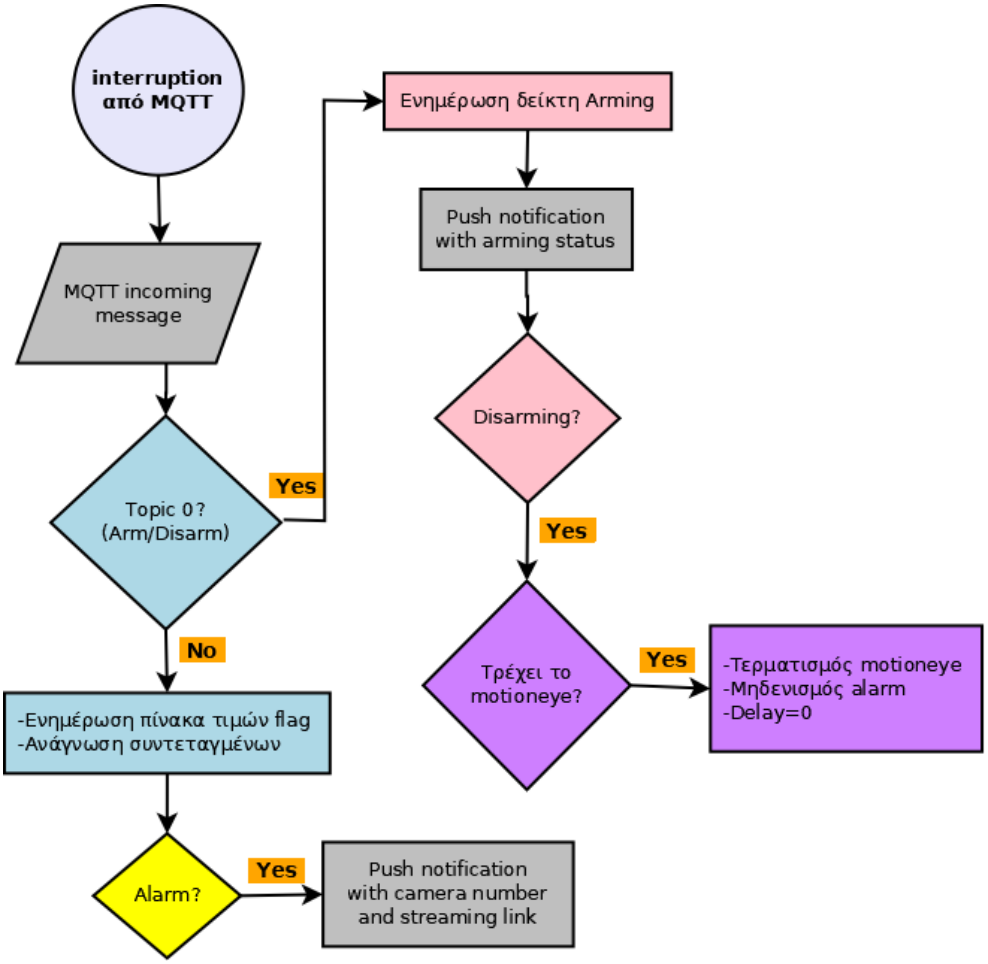


4.2 Εσωτερική μονάδα

Το διάγραμμα ροής του κώδικα της εσωτερικής μονάδας είναι:



Σε περίπτωση που έρθει μήνυμα από τον MQTT Broker γίνεται interruption:


interruption
από MQTT
MQTT incoming
message
Topic 0?
(Arm/Disarm)
Yes
No
Ενημέρωση δείκτη Arming
Push notification
with arming status
Disarming?
Yes
Τρέχει το
motioneye?
Yes
-Τερματισμός motioneye
-Μηδενισμός alarm
-Delay=0
-Ενημέρωση πίνακα τιμών flag
-Ανάγνωση συντεταγμένων
Alarm?
Yes
Push notification
with camera number
and streaming link

4.2.1 Ανάλυση κώδικα εσωτερικής μονάδας (internal.py):

Αρχικά κάνουμε εισαγωγή των απαραίτητων βιβλιοθηκών που θα χρησιμοποιήσουμε:

```
import os
import sys
import time
import datetime
import paho.mqtt.client as mqtt
import numpy as np #Gia tous pinakes
import math #Gia prakseis pinakon
import signal #Gia keyboard interruption
from pushbullet import Pushbullet #Gia push notifications
import RPi.GPIO as GPIO
import json
import socket
import fcntl
import struct
```

Έπειτα γίνεται ανάγνωση των ρυθμίσεων λειτουργίας της μονάδας από το αρχείο settings.txt που θα πρέπει να βάλουμε στον ίδιο φάκελο με το αρχείο του κώδικα. Μας δίνεται η δυνατότητα να ορίσουμε:

- → τον αύξοντα αριθμό της κάμερας.
- → τον συνολικό αριθμό καμερών στην περιοχή κάλυψης (δύναται να βάλουμε αρκετά μεγάλο αριθμό ώστε να υπάρχει συνεχόμενη λειτουργία ακόμα και σε μελλοντική αύξηση του αριθμού των καμερών/μονάδων χωρίς να χρειαστεί να του φορτωθεί εκ νέου το αρχείο ρυθμίσεων και να γίνει επανεκκίνηση της μονάδας).
- → το topic του MQTT που θα ακούνε και θα κάνουν publish όλες οι μονάδες.
- → συντεταγμένες της μονάδας Latitude-Longitude. Συνήθως αποτελεί τις συντεταγμένες του χώρου τοποθέτησης (πχ οικείας) και μπορεί είτε να βρεθεί από το Google maps είτε να μετρηθεί με ακρίβεια (πχ GPS) κατά την τοποθέτηση.
- → Το κλειδί του PushBullet του λογαριασμού του χρήστη που θα του αποστέλλονται τα notifications

```
file=open("settings-internal.txt","r")
with file:
      content = file.readlines() #Diavazei oles tis grammes kai tis
vazei ston pinaka content
content = [x.strip() for x in content] #Afairei ola ta special
character apo tin ka8e line tou pinaka content
file.close()
index=int(content[0]) #index kameras
total=int(content[1]) #sinolikos arithmos kameron
topic=content[2] #topic tou broker
latitude=float(content[3]) #latitude kameras
longtitude=float(content[4]) #longtitude kameras
pushkey=content[5] #kleidi Pushbullet
```

Αρχικοποίηση μεταβλητών:

```
matrix=np.zeros(shape=(1,total)) #Arxikos pinakas me toses stiles
oses o arithmos ton kameron
pir=0 #Metavliti pou diavazei tin timi tou pir
launcher=False #Arming/Disarming
diktis=False #Diktis gia enimerosi mono mia  fora gia mi-entopismo
kinisis
diktis2=False #Diktis gia ekkinisi face detection
xronos=0
```

Διαβάζουμε την IP address του VPN που έχει ανατεθεί στην μονάδα για να ενημερώσει κατάλληλα μέσω notification, σε περίπτωση alarm, για το streaming link:

```
def get_ip_address(ifname):
      s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
      return socket.inet_ntoa(fcntl.ioctl(s.fileno(),0x8915,
struct.pack('256s', ifname[:15]))[20:24])
time.sleep(10) #delay gia na kanei connection to OpenVPN stin
ekkinisi
try:
      ipaddr=get_ip_address('tun0') #IP addresses of streaming server
except Exception:
      ipaddr=get_ip_address('eth0')
```

Αρχικοποίηση του object για Push notification:

```
push = Pushbullet(pushkey)
```

Αρχικοποίηση του keyboard interrupt listener και συνάρτηση εξόδου σε περίπτωση που πατηθούν τα πλήκτρα Ctrl+C για τερματισμό λειτουργίας. Για τον ομαλό τερματισμό κλείνει την σύνδεση με τον MQTT broker και επαναφέρει σε default τα GPIO pins:

```
def interrupt_handler(signal, frame):
      print "\n\nTerminating....\n"
      client.loop_stop()
      client.disconnect()
      GPIO.cleanup()
      sys.exit(0)
#Arxikopoiisi interrupt lisener
signal.signal(signal.SIGINT, interrupt_handler)
```

Αρχικοποίηση του MQTT Broker με χρήση του freeware broker test.mosquitto.org

→ Κάνει subscribe σε όλα τα sub-topics του κυρίου topic

→ Ενημέρωση (publish) του topic της συγκεκριμένης μονάδας με τις συντεταγμένες και το flag=0 (δλδ δεν υπάρχει alarm).

→ Push notification στον κάτοχο για σύνδεση της εν λόγω μονάδας στο σύστημα

```
client = mqtt.Client()
client.connect("test.mosquitto.org", 1883, 60)
client.subscribe(topic+"#") #Kanei subscribe ste ola ta sub-topics
client.on_message = on_message
client.loop_start()
temp3='{"latitude":"%s","longtitude":"%s","flag":"0"}'
%(latitude,longtitude)
client.publish(topic+str(index),temp3)
push.push_note('Area1', 'Sindethike sto diktio i kamera:
'+str(index))
```

Συνάρτηση interruption σε περίπτωση που έρθει μήνυμα από τον Broker:

→ Αν είναι από το κεντρικό topic για arming/disarming, κάνει ενημέρωση της μεταβλητής ελέγχου και στέλνει αντίστοιχο push notification στον κάτοχο. Σε περίπτωση disarming ελέγχει επιπλέον αν τρέχει το motioneye και το τερματίζει (για εξοικονόμηση πόρων) και μηδενίζει τα delay, το alarm και κάνει publish την τιμή flag=0.

→ Αν είναι από topic κάποιας μονάδας, ελέγχει αν ήρθε διαφορετική τιμή από την προηγούμενη (για να μην ξαναεκτελεί ασκόπως κώδικα) και αν ήταν 1 (alarm) οπότε και ενημερώνει τους ανάλογους δείκτες ελέγχου και στέλνει push notifications στον κάτοχο τόσο με το ποια μονάδα εντόπισε κίνηση όσο και με το link του live streaming της εκάστοτε μονάδας.

```
def on_message(client, userdata, msg):
      global j
      global m
      global matrix
      global push
      global launcher
      global diktis
      global diktis2
      global ipaddr
      global latitude
      global longtitude
      global topic
      global index
      global xronos
      m=msg.topic
      if (int(m[-1])==0): #an ginei arming/disarming
            j=int(msg.payload)
            if (int(j)==1):
                  launcher=True #enimerosi tou boolean
                  diktis=True #gia na min ginei arxiki enimerosi gia
mi-entopismo
```

```
                print "\n\nAlarms ON"
                push.push_note('Area1', 'Alarms ON')
            if (int(j)==0):
                launcher=False #enimerosi tou boolean
                print "\n\nAlarms OFF"
                push.push_note('Area1', 'Alarms OFF')
                if (diktis2==True): #termatizei to motioneye an
trexei
                    print "\n\nTermatismos MotionEye"
                    os.system('sudo systemctl stop motioneye')
                    diktis2=False

      temp3='{"latitude":"%s","longtitude":"%s","flag":"0"}'
%(latitude,longtitude)
                    client.publish(topic+str(index),temp3)
                    xronos=0
                    matrix[0,int(index)-1]=0
      if (int(m[-1])>0 and launcher==True): #elegxei oti den itan to
arming/disarming
            j=json.loads(str(msg.payload))
            temp=int(matrix[0,int(m[-1])-1]) #kratame tin proigoumeni
timi tou pinaka prosorina
            j=int(j["flag"])
            matrix[0,int(m[-1])-1]=j #mono tote kanei enimerosi
pinaka timon topic
            if (j==1 and temp!=j): #elegxei oti ir8e asos
(entopismos) apo kapoia kamera kai oti prin itan miden gia na min
stelnei epanalamvanomena mails kai notifications
                print "\n\nEntopismos kinisis apo kamera: ",m[-1]
                diktis=False #otan ginei entopismos kinisis
ksanamidenizei o diktis enimerosis gia mi-entopismo
                push.push_note('Area1', 'Entopismos kinisis apo
kamera: '+m[-1]+'  Live streaming: ')
                push.push_link('Live streaming',
'http://'+ipaddr+':8081')
```

Ορισμός του Pin-7 του GPIO ως input:

```
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BOARD)
GPIO.setup(7, GPIO.IN)
```

Κύρια ρουτίνα προγράμματος. Όλες οι εντολές εκτελούνται μόνο αν έχει γίνει armed το σύστημα.

→ Αν υπάρξει alarm και δεν τρέχει το motioneye, το εκκινεί και δημιουργεί και ένα delay 1 ώρας ώστε αν σταματήσει το alarm να συνεχίσει να κάνει monitoring η κάμερα για επαυξημένο έλεγχο.

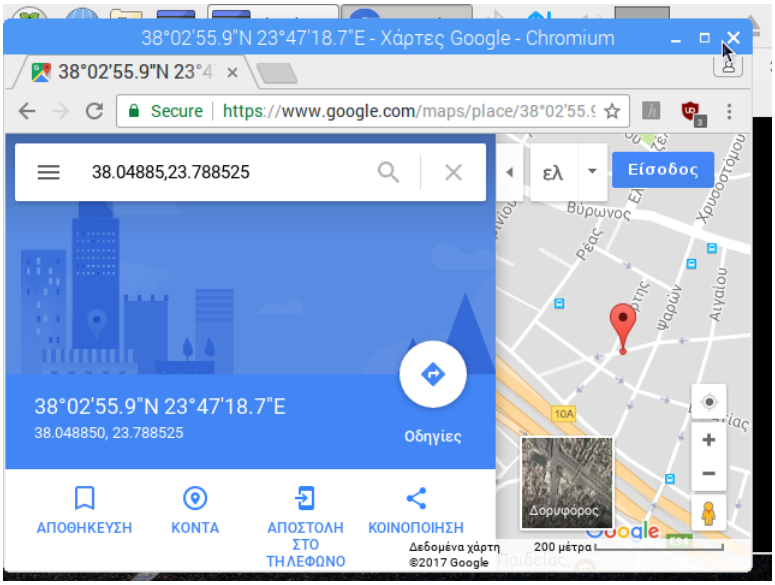
→ Αν δεν υπάρχει κανένα alarm τότε κάνει τερματισμό του motioneye μόνο αν έχει τελειώσει το delay και τρέχει το process του motioneye (και στέλνει και notification στον χρήστη για παύση συναγερμού)

→ Αν δεν έχει παρέλθει το delay, διαβάζει την τιμή του PIR και αν υπάρξει αλλαγή στην τιμή του ενημερώνει τον broker.

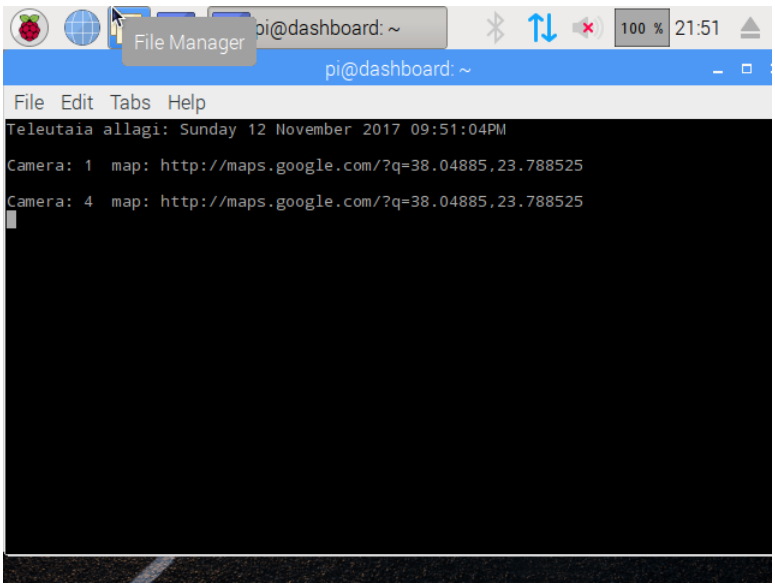
```
print "\nTermatismos me Ctrl+C \n"
while True:
      if (launcher==True):
            if (int(matrix.sum())!= 0 and diktis2==False): #An den
trexei to motion kanei ekkinisi kai exei ginei entopismos kinisis apo
kapoia kamera
                  print "\n\nEkkinisi MotionEye"
                  os.system('sudo systemctl start motioneye')
                  diktis2=True
                  xronos=int(round(time.time()))+3600 #Ftiaxnei to
delay gia epipleon katagrafi se periptosi pou stamaisei to alarm
            if (int(matrix.sum()==0) and diktis2==True and
int(round(time.time()))>xronos): #An trexei idi to motion kai den
exoume pleon entopismo kinisis
                  print "\n\nTermatismos MotionEye" #apo kapoia
kamera kai teleiosei to delay, termatizei to motioneye
                  os.system('sudo systemctl stop motioneye')
                  print "\n\nKanenas entopismos pleon"
                  push.push_note('Area1', 'Kanenas entopismos pleon')
                  diktis=True
                  diktis2=False

      #=================================================================
=====================================
            if (int(round(time.time()))>xronos):
                  pir=GPIO.input(7) #Diavazei to pin 7 tou GPIO kai
epistrofi timis
                  if (pir!=int(matrix[0,int(index)-1])): #Mono an
iparxei allagi stin timi tou pir

      jj='{"latitude":"%s","longtitude":"%s","flag":"%i"}'
%(latitude,longtitude,pir)
                        client.publish(topic+str(index),jj)
#Dilonoume tin allagi ston broker
      time.sleep(0.5)
```

4.3 Dashboard – Κεντρική μονάδα ελέγχου

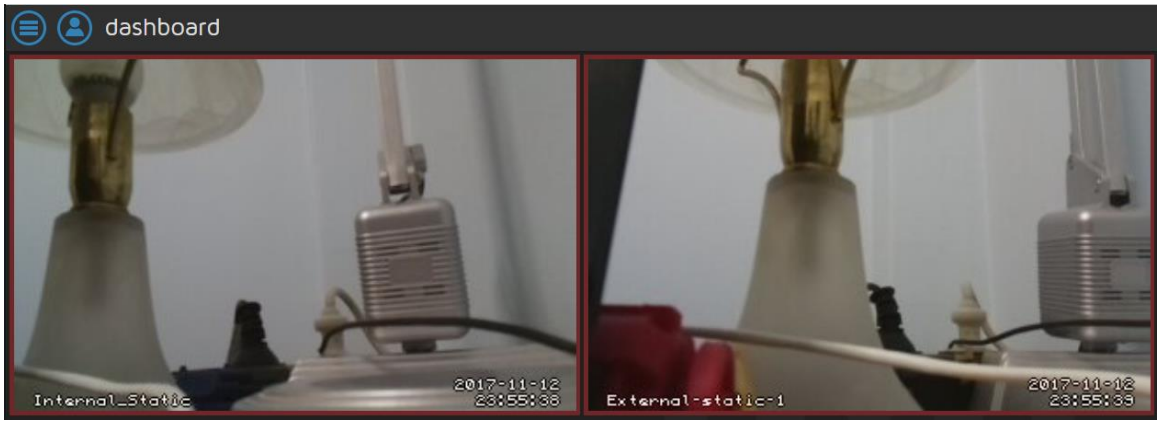
Η κεντρική μονάδα ελέγχου είναι συνδεμένη στο κύριο topic και κάθε φορά που θα γίνει publish μηνύματος από κάποια εσωτερική μονάδα με flag "1" (δλδ alert) ανοίγει στον browser το google map με το σημείο που έγινε το alert (συντεταγμένες της εσωτερικής μονάδας):



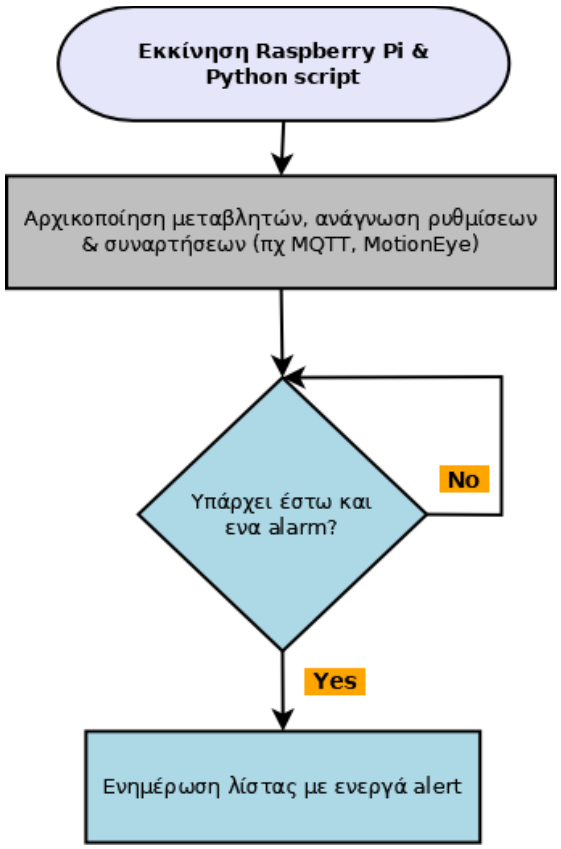
Επιπλέον στο shell εκτέλεσης του python script εμφανίζεται κάθε φορά ποια κάμερα έχει εκείνη την ώρα ενεργό alert και το link με το χάρτη (πατώντας ctrl και κάνοντας κλικ πάνω στο link ανοίγει παράθυρο στον browser με το σημείο της κάμερας):



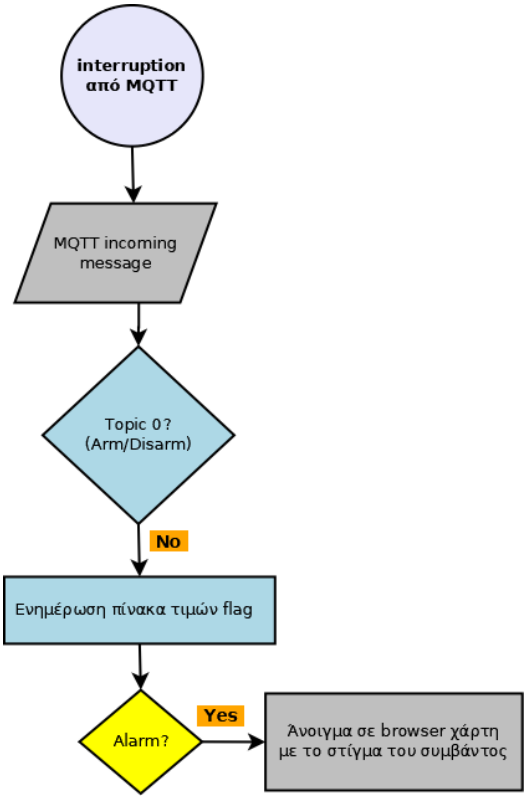
Τέλος, στοMotionEye της κεντρικής μονάδας έχουμε συνδέσει τις επιθυμητές κάμερες (εξωτερικές) ώστε να υπάρχει συγκεντρωτικός οπτικός έλεγχος:



Το διάγραμμα ροής του κώδικα της εσωτερικής μονάδας είναι:



Σε περίπτωση που έρθει μήνυμα από τον MQTT Broker γίνεται interruption:



4.3.1 Ανάλυση κώδικα κεντρικής μονάδας ελέγχου (dashboard.py):

Αρχικά κάνουμε εισαγωγή των απαραίτητων βιβλιοθηκών που θα χρησιμοποιήσουμε:

```
# import the necessary packages
import os
import sys
import time
import datetime
import argparse
import datetime
import paho.mqtt.client as mqtt
import numpy as np #Gia tous pinakes
import math #Gia prakseis pinakon
import signal #Gia keyboard interruption
import json
import webbrowser
```

Έπειτα γίνεται ανάγνωση των ρυθμίσεων λειτουργίας της μονάδας από το αρχείο settings.txt που θα πρέπει να βάλουμε στον ίδιο φάκελο με το αρχείο του κώδικα. Μας δίνεται η δυνατότητα να ορίσουμε:

→ τον συνολικό αριθμό καμερών στην περιοχή κάλυψης (δύναται να βάλουμε αρκετά μεγάλο αριθμό ώστε να υπάρχει συνεχόμενη λειτουργία ακόμα και σε μελλοντική

αύξηση του αριθμού των καμερών/μονάδων χωρίς να χρειαστεί να του φορτωθεί εκ νέου το αρχείο ρυθμίσεων και να γίνει επανεκκίνηση της μονάδας).

→ το topic του MQTT που θα ακούνε και θα κάνουν publish όλες οι μονάδες.

```
file=open("settings-dashboard.txt","r")
with file:
      content = file.readlines() #Diavazei oles tis grammes kai tis
vazei ston pinaka content
content = [x.strip() for x in content] #Afairei ola ta special
character apo tin ka8e line tou pinaka content
file.close()
total=int(content[0]) #sinolikos arithmos kameron
topic=content[1] #topic tou broker
```

Αρχικοποίηση μεταβλητών:

```
matrix=np.zeros(shape=(1,total)) #Arxikos pinakas me toses stiles
oses o arithmos ton kameron
matrix2=np.zeros(shape=(1,total)) #Pinakas gia to latitude tis ka8e
kameras
matrix3=np.zeros(shape=(1,total)) #Pinakas gia to longtitude tis ka8e
kameras
launcher=False #Arming/Disarming
diktis2=False
```

Αρχικοποίηση του keyboard interrupt listener και συνάρτηση εξόδου σε περίπτωση που πατηθούν τα πλήκτρα Ctrl+C για τερματισμό λειτουργίας. Για τον ομαλό τερματισμό κλείνει την σύνδεση με τον MQTT broker:

```
def interrupt_handler(signal, frame):
      print "\n\nTerminating....\n"
      client.loop_stop()
      client.disconnect()
      sys.exit(0)
#Arxikopoiisi interrupt lisener
signal.signal(signal.SIGINT, interrupt_handler)
```

Συνάρτηση interruption σε περίπτωση που έρθει μήνυμα από τον Broker:

→ Αν είναι από topic κάποιας μονάδας ενημερώνει τους πίνακες με τις αντίστοιχες για την εν λόγω κάμερα τιμές (flag, latitude, longitude) και αν ήταν 1 (alarm) ανοίγει στον browser παράθυρο με τους χάρτες Google maps που έχουν ως επίκεντρο το σημείο του συμβάντος.

```
def on_message(client, userdata, msg):
      global j
      global m
```

```
      global matrix
      global matrix2
      global matrix3
      global latitude2
      global longtitude2
      global webbrowser
      global diktis2
      m=msg.topic
      if (int(m[-1])>0): #elegxei oti den itan to arming/disarming
            diktis2=True
            j=json.loads(str(msg.payload))
            latitude2=float(j["latitude"])
            longtitude2=float(j["longtitude"])
            j=int(j["flag"])
            matrix[0,int(m[-1])-1]=j
            matrix2[0,int(m[-1])-1]=latitude2
            matrix3[0,int(m[-1])-1]=longtitude2
            if (j==1):

      url='http://maps.google.com/?q='+str(latitude2)+','+str(longtit
ude2)
                  webbrowser.open(url)
```

Αρχικοποίηση του MQTT Broker με χρήση του freeware broker test.mosquitto.org

```
client = mqtt.Client()
client.connect("test.mosquitto.org", 1883, 60)
client.subscribe(topic+"#") #Kanei subscribe ste ola ta sub-topics
client.on_message = on_message
client.loop_start()
```

Κύρια ρουτίνα προγράμματος.

→ Αν υπάρξει alarm τυπώνει την ημερομηνία & ώρα και ελέγχει τους πίνακες ώστε να ελέγξει ποιες μονάδες έχουν alarm (δλδ flag=1) και εκτυπώνει μια λίστα με τα αντίστοιχα ονόματα των καμερών και τα link με τις συντεταγμένες τους.

```
os.system('sudo clear')
print "\nTermatismos me Ctrl+C \n"
while True:
      if (diktis2==True):
            diktis2=False
            os.system('sudo clear')
            print 'Teleutaia allagi:
'+datetime.datetime.now().strftime("%A %d %B %Y %I:%M:%S%p")
            for diktis in range (1, total+1):
                  if (matrix[0,int(diktis)-1]==1):

      url='http://maps.google.com/?q='+str(matrix2[0,int(diktis)-
1])+','+str(matrix3[0,int(diktis)-1])
                        print '\nCamera: '+str(diktis)+'  map: '+url
            if (int(matrix.sum())== 0):
                  print 'Kanena alert'
```

Κεφάλαιο 5 Σύνοψη

5.1 Συμπεράσματα – Μελλοντικές επεκτάσεις

Μέσω της παρούσας διπλωματικής υλοποιήθηκε ένα πρωτοποριακό σύνθετο σύστημα ασφάλειας, το οποίο κάνει χρήση περιφερειακών και πρωτοκόλλων από τον καινοτόμο κόσμο του IoT, ελαχιστοποιώντας τις απαιτήσεις σε bandwidth και ενέργεια διατηρώντας παράλληλα τις δυνατότητες ελέγχου και παρακολούθησης (τόσο από τον τελικό χρήστη όσο και από το κέντρο ελέγχου), την αδιάλειπτη λειτουργία, την εύκολη επεκτασιμότητα και, τέλος, παρέχοντας ασφάλεια και κρυπτογράφηση των δεδομένων.

Πολλαπλές δοκιμές με διαφορετικά σενάρια επιβεβαίωσαν την ορθή λειτουργία του συστήματος, ενώ παρατηρήθηκαν και κάποιες αδυναμίες του συστήματος οι οποίες όμως δεν εμπίπτουν στους στόχους της παρούσας εργασίας. Συγκεκριμένα, η εξωτερική μονάδα που κάνει tracking μπορεί να χάσει τον στόχο της, δηλαδή το αντικείμενο προς παρακολούθηση, αν αυτό αλλάξει γρήγορα κατεύθυνση κίνησης (πχ ελιγμό) ή αν σταματήσει απότομα. Επίσης, σε περίπτωση προσωρινής διακοπής του internet διακόπτεται και η σύνδεση με τον Broker, οπότε αν την στιγμή εκείνη γίνεται μετάδοση (publish/read) μηνύματος, αυτό χάνεται και δεν είναι δυνατή η ανάκτησή του οπότε και δεν λειτουργεί ως αναμένεται η μονάδα.

Παρακάτω αναφέρονται δυνατές μελλοντικές βελτιώσεις λειτουργίας του συστήματος που βασίζονται στους ελέγχους και τις δοκιμές που πραγματοποιήθηκαν:

- Back-up σύνδεση μέσω κινητής: Σε περίπτωση connection failure του ethernet θα ήταν ζωτικής σημασίας η εναλλακτική σύνδεση μέσω usb dongle με κάρτα sim (3G/4G) ώστε κάθε μονάδα ξεχωριστά να επικοινωνεί απρόσκοπτα με το κέντρο ελέγχου.
- Εναλλακτικοί τρόποι ασύρματης δικτύωσης: Εκτός των ethernet/wifi/3G/4G μπορούν να εξεταστούν και να χρησιμοποιηθούν εναλλακτικοί τρόποι ασύρματης δικτύωσης που είναι σχεδιασμένοι κυρίως για IoT εφαρμογές όπως LTE-M, NB-IoT, LoRa/LoRaWAN.
- Τροφοδοσία από powerbank/solar panel: Λόγω της χαμηλής ενεργειακής απαίτησης από τα Raspberry θα ήταν χρήσιμη η δυνατότητα τροφοδοσίας των μονάδων μέσω μπαταρίας-powerbank ώστε να διατηρείται η απρόσκοπτη λειτουργία τους σε περίπτωση power failure. Ειδικότερα, οι εξωτερικές μονάδες θα μπορούσαν να φορτίζουν την μπαταρία τους μέσω solar panel.
- Χρήση servo-controller: Θα ήταν χρήσιμη η προσθήκη controller που να ελέγχει την κίνηση του servo σε αντίθεση με τον τωρινό έλεγχο που γίνεται μέσω του GPIO του Raspberry καθώς θα έδινε πιο ομαλή κίνηση του μηχανισμού στρέψης και θα ήταν καλύτερη η απόδοση της εξωτερικής μονάδας ελέγχου κατά το tracking των αντικειμένων
- Μέτρηση απόστασης αντικειμένου: Στις εξωτερικές μονάδες θα μπορούσε να γίνει χρήση εξαρτημάτων μέτρησης απόστασης, πχ laser, ώστε με κατάλληλη τροποποίηση του αλγορίθμου κίνησης της κάμερας (tracking), αυτή να στρέφεται

δυναμικά, ώστε να επιτυγχάνεται η παρακολούθηση των αντικειμένων ακόμα και όταν αυτά μεταβάλλουν την ταχύτητα ή τη διεύθυνση κίνησης

- Χρήση handshake: Κατά την επικοινωνία των μονάδων με τον Broker θα μπορούσε να υλοποιηθεί ένα σύστημα handshake με την κεντρική μονάδα ελέγχου ώστε σε κάθε μήνυμα που μεταδίδεται από μια εσωτερική μονάδα ελέγχου, οι εξωτερικές μονάδες να στέλνουν acknowledgement στην κεντρική μονάδα έτσι ώστε αν κάποια (εξωτερική) χάσει την σύνδεση να της αποσταλεί εκ νέου το μήνυμα (από την κεντρική μονάδα αυτή την φορά)

Βιβλιογραφία – Παραπομπές

1. Arshdeep Bahga and Vijay Madisetti, “Internet of Things: A Hands-On Approach”, VPT , 2014
2. Steven Goodwin, “Smart Home Automation with Linux and Raspberry Pi”, Apress, 2013
3. Gerard Honey, “The Complete Home Security Guide”, Crowood Press, 2013
4. Matthew Poole, “Building a Home Security System with Raspberry Pi”, Packt Publishing 2015
5. Olivier Hersent, David Boswarthick and Omar Elloumi, “The Internet of Things: Key Applications and Protocols”, Wiley, 2012
6. Simon Monk, “Programming the Raspberry Pi: Getting Started with Python”, McGraw-Hill Education TAB, 2012
7. Official site of Raspberry Pi
   https://www.raspberrypi.org/
8. Wikipedia: Raspberry Pi
   https://en.wikipedia.org/wiki/Raspberry_Pi
9. “PIR Motion Sensor”, Adafruit Industries White Paper, 2016
10. Official product site of HC-SR501
    https://www.mpja.com/download/31227sc.pdf
11. Arduino HC-SR501 Motion Sensor Tutorial
    http://henrysbench.capnfatz.com/henrys-bench/arduino-sensors-and-input/arduino-hc-sr501-motion-sensor-tutorial/
12. Matthew Scarpino , “Motors for Makers: A Guide to Steppers, Servos, and Other Electrical Machines”, Que Publishing , 2015
13. Official product site of SG90 servo
    http://www.towerpro.com.tw/product/sg90-analog/
14. Online store with specs of SG90 servo
    www.micropik.com/PDF/SG90Servo.pdf
15. USB Logitech web camera specification site
    https://www.evertek.com/viewpart.asp?auto=52245&cpc=NEWARR
16. Gary Bradski and Adrian Kaehler, “Learning OpenCV: Computer Vision with the OpenCV Library”, O'Reilly Media, 2008
17. Official site of OpenCV
    https://opencv.org/
18. Wikipedia: OpenCV
    https://en.wikipedia.org/wiki/OpenCV
19. Official site of FFmpeg
    https://www.ffmpeg.org/
20. Wikipedia: FFmpeg
    https://en.wikipedia.org/wiki/FFmpeg
21. Official site of MotionEye
    https://github.com/ccrisan/motioneye/wiki

22. Official site of Motion
https://motion-project.github.io/
23. Bill Gallagher, “Dropbox Essentials: The Complete Beginners Guide to Dropbox”, CreateSpace Independent Publishing Platform, 2015
24. Tal Ater, “Building Progressive Web Apps: Bringing the Power of Native to the Browser”, O'Reilly Media, 2017
25. Wikipedia: Push Notification
https://en.wikipedia.org/wiki/Push_Notification
26. PushBullet official site
https://www.pushbullet.com/
27. Deligence Technologies guide to PushBullet at Raspberry Pi
https://github.com/DeligenceTechnologies/Intruder-detector-with-Raspberry-Pi-and-Pushbullet/tree/master/pushbullet.py
28. Karl Joch, “Mosquitto - MQTT Broker for IoT (Internet of Things)”, CTS GMBH, 2017
29. Gaston C. Hillar, “MQTT Essentials - A Lightweight IoT Protocol”, Packt Publishing, 2017
30. ISO, “Information technology -- Message Queuing Telemetry Transport (MQTT)”, v3.1.1, 20922:2016, 2016
https://www.iso.org/standard/69466.html
31. MQTT official site
http://mqtt.org/
32. Wikipedia: MQTT
https://en.wikipedia.org/wiki/MQTT
33. Lindsay Bassett , “Introduction to JavaScript Object Notation: A To-the-Point Guide to JSON”, O'Reilly Media, 2015
34. Tom Marrs, “JSON at Work: Practical Data Integration for the Web”, O'Reilly Media, 2017
35. JSON official site
http://www.json.org
36. ECMA, “The JSON Data Interchange Format”, ECMA-404, October 2013
http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf
37. Python commands for json
https://docs.python.org/3/library/json.html
38. Wikipedia: JSON
https://en.wikipedia.org/wiki/JSON
39. Find the bearing angle between two points in a 2D space
https://math.stackexchange.com/questions/1596513/find-the-bearing-angle-between-two-points-in-a-2d-space
40. Calculate distance, bearing and more between Latitude/Longitude points
https://www.movable-type.co.uk/scripts/latlong.html
41. Ira Finch, “Build a Smart Raspberry Pi VPN Server: Auto Configuring, Plug-n-Play, Use from Anywhere”, Amazon Digital Services LLC, 2015
42. Jon C. Snader, “VPNs Illustrated: Tunnels, VPNs, and IPsec: Tunnels, VPNs, and IPsec”, Addison-Wesley Professional, 2005

43. How to set up a VPN (private internet access) in Raspberry Pi
https://dotslashnotes.wordpress.com/2013/08/05/how-to-set-up-a-vpn-private-internet-access-in-raspberry-pi/
44. OpenVPN official site
https://openvpn.net/
45. Wikipedia: OpenVPN
https://en.wikipedia.org/wiki/OpenVPN
46. SoftEther official site
https://www.softether.org/
47. IBM - 5 Things to Know About MQTT:
https://www.ibm.com/developerworks/community/blogs/5things/entry/5_things_to_know_about_mqtt_the_protocol_for_internet_of_things?lang=en