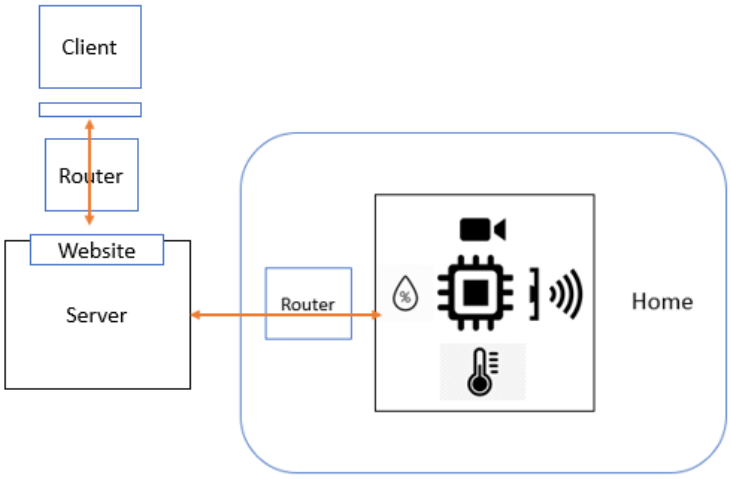
ΔΙΕΘΝΕΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΤΗΣ ΕΛΛΑΔΟΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

«Παρακολούθηση χώρου με κάμερα, αισθητήρες και raspberry μέσω διαδικτύου»



Του φοιτητή
Νικόλαου Βαΐδη
Αρ. Μητρώου: 531015

Επιβλέπων
Δρ. Κυριάκος Τσιακμάκης

Σεπτέμβριος 2020

Παρακολούθηση χώρου με κάμερα, αισθητήρες και raspberry μέσω διαδικτύου
Κωδικός: 20177
Φοιτητής: Νικόλαος Βαΐδης
Εισηγητής: Δρ Κυριάκος Τσιακμάκης
Ημερομηνία ανάληψης Π.Ε. 30-04-2020
Ημερομηνία περάτωσης Π.Ε. 01-09-2020

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

Περίληψη

Η εργασία αφορά την κατασκευή ενός συστήματος παρακολούθησης χώρου με κάμερα και αισθητήρες που υλοποιείται με Raspberry έχοντας αμφίδρομη επικοινωνία πραγματικού χρόνου με ένα απομακρυσμένο χρήστη μέσω διαδικτύου χρησιμοποιώντας tcp sockets.

Η επικοινωνία του χρήστη με το Raspberry πραγματοποιείται χωρίς να χρειάζεται port forwarding όταν βρίσκεται πίσω από δρομολογητές ή τείχη προστασίας.

Ο χρήστης να μπορεί να παρακολουθεί την κάμερα και τους αισθητήρες του Raspberry με εξουσιοδότηση μέσω του διαδικτύου από τον περιηγητή-browser του.

«Monitoring using camera, sensors and raspberry via internet»

«Nikolaos Vaidis»

Abstract

This thesis concerns the implementation of a monitoring system with camera and sensors using Raspberry and full-duplex real-time communication with a remote user via the internet using tcp sockets.

The user communicates with the Raspberry without the need for port forwarding when operates via routers or firewalls.

The authorized user can monitor the Raspberry's camera and sensors with internet from his browser.

Ευχαριστίες

Θέλω να ευχαριστήσω τους γονείς μου για τη συμπαράσταση τους και τον επιβλέπων κ. Τσιακμάκη Κυριάκο για τη συνεχή επιστημονική και παιδαγωγική του καθοδήγηση.

Περιεχόμενα

Κατάλογος Σχημάτων

Κεφάλαιο 1ο: Εισαγωγή

1.1 Εισαγωγή

Ο κύριος στόχος αυτής της εργασίας είναι η κατασκευή ενός συστήματος παρακολούθησης χώρου με κάμερα και αισθητήρες που υλοποιείται με Raspberry[1] έχοντας αμφίδρομη επικοινωνία πραγματικού χρόνου με ένα απομακρυσμένο χρήστη μέσω διαδικτύου χρησιμοποιώντας tcp sockets[2].

Συγκεκριμένα, οι επιμέρους στόχοι της εργασίας είναι:

Επικοινωνία του χρήστη με το Raspberry χωρίς να χρειάζεται port forwarding όταν βρίσκεται πίσω από router (ή nat boxes[3]).

Ο χρήστης να μπορεί να παρακολουθεί την κάμερα και τους αισθητήρες του Raspberry μέσω του διαδικτύου από τον περιηγητή-browser του.

Ο χρήστης που χρησιμοποιεί την ιστοσελίδα πρόσβασης στο σύστημα είναι εξουσιοδοτημένος.

1.1.1 Βιβλιογραφική Ανασκόπηση

Προτού αναλυθεί το σύστημα που υλοποιήθηκε στην εργασία θα αναφερθούν παρόμοια συστήματα που προτείνονται στη βιβλιογραφία και έχουν υλοποιηθεί.

Αρχικά θα αναφερθούν έργα που έχουν υλοποιήσει μικρά κομμάτια που χρησιμοποιούνται στην τελική εργασία και εργασίες που χρησιμοποιούν διαφορετικές τεχνικές.

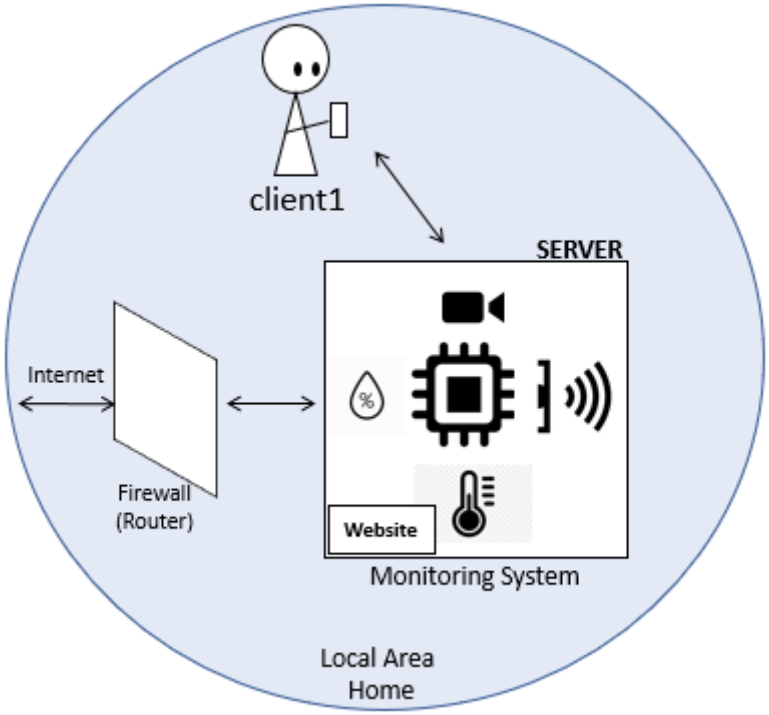
Υπάρχουν κάμερες που συνδέονται απευθείας με το Raspberry με ανάλυση που προσεγγίζει 4056 x 3040 pixels [4] και έχουν υλοποιηθεί και δημοσιευτεί αξιόπιστα έργα ανοιχτού κώδικα [5] .

Εκτός από την κάμερα module που συνδέεται απευθείας στην ειδική υποδοχή που έχει το raspberry υπάρχει η δυνατότητα χρήση κάμερας μέσω usb επικοινωνίας, χρησιμοποιώντας απλά μια web camera.

Υπάρχουν έτοιμα έργα με χρήση web camera με έτοιμες βιβλιοθήκες που δεν παρέχουν λεπτομέρειες και δυνατότητα επεξεργασίας με fswebcam [6] και με motion βιβλιοθήκη [7]

ενώ υπάρχουν έργα τα οποία έχουν υλοποιηθεί με OpenCV και δίνουν την δυνατότητα επεξεργασίας για κάθε frame-εικόνα [8-11]

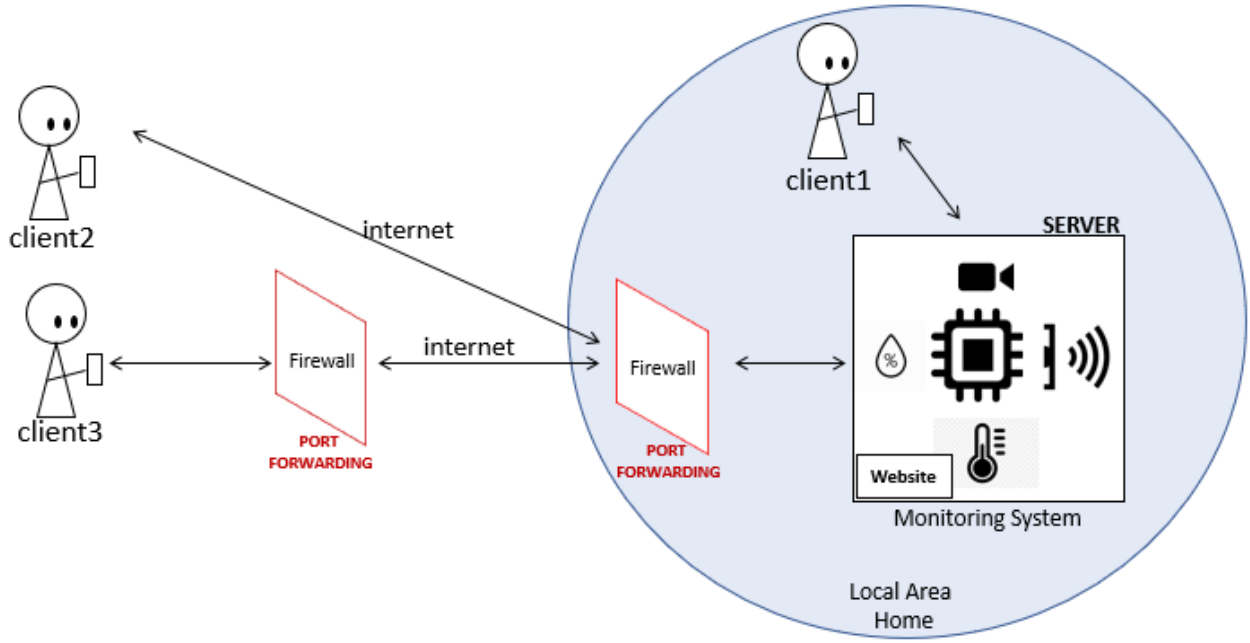
Το μεγαλύτερο ενδιαφέρον εστιάζεται στην υλοποίηση Web server στο Raspberry και απεικόνιση βίντεο μόνο στο τοπικό δίκτυο από ιστοσελίδα [12-14] και υλοποιημένο με imagezmq [15], όπως φαίνεται στο Σχήμα 1.1.



Σχήμα 1.1: Σύστημα παρακολούθησης εγκατεστημένο μόνο για τοπική χρήση

Ο χρήστης δεν μπορεί να έχει πρόσβαση στο σύστημα παρακολούθησης όταν βρίσκεται πίσω από έναν router-δρομολογητή, μια κατάσταση που ισχύει σχεδόν σε όλα τα σπίτια και χώρους εργασίας.

Μια προσωρινή λύση βρίσκεται στην υλοποίηση Web server στο Raspberry και απεικόνιση βίντεο μέσω internet από ιστοσελίδα μέσω port forwarding [16], όπως φαίνεται στο Σχήμα 1.2.

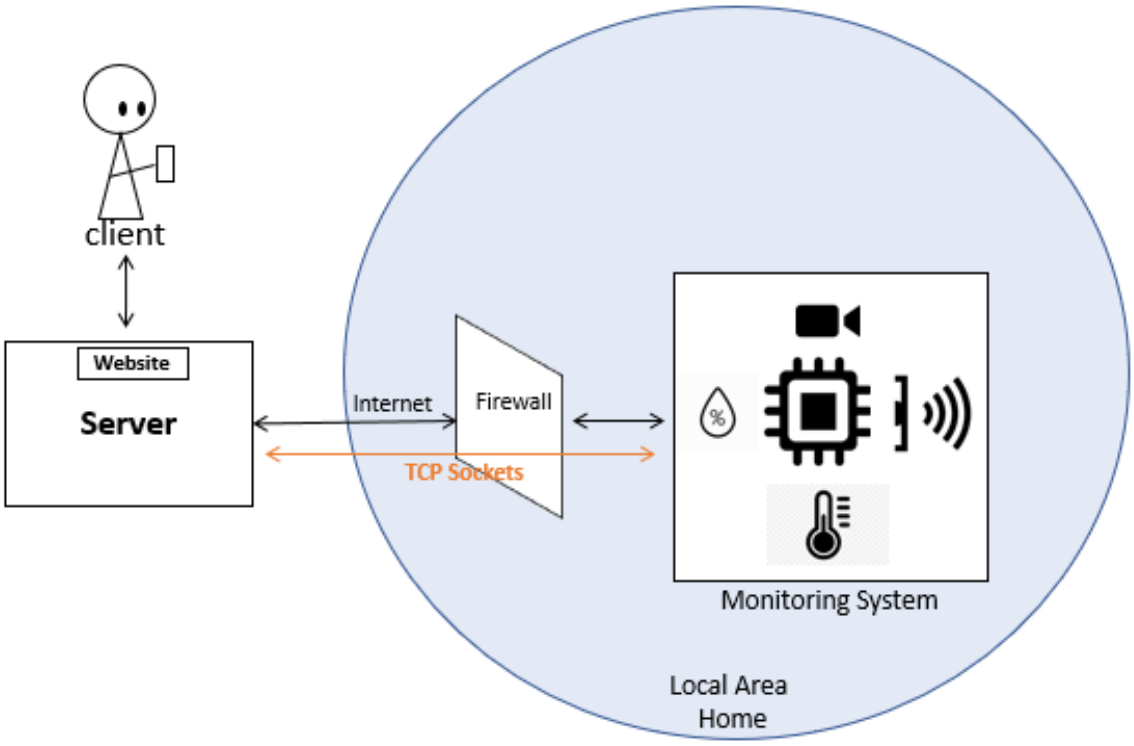


Σχήμα 1.2: Σύστημα παρακολούθησης εγκατεστημένο για απομακρυσμένη χρήση με Port Forwarding

Αυτό βέβαια προϋποθέτει ο κάθε χρήστης που κάνει εγκατάσταση το σύστημα στον χώρο του (μέσω ενός router) θα πρέπει να έχει πρόσβαση στο router και να ρυθμίζει τα ports ώστε να δίνεται πρόσβαση στο πρόγραμμα στο raspberry ώστε να επικοινωνεί με τον τελικό χρήση απευθείας. Κάτι τέτοιο είναι πολύ δύσκολο να γίνει από απλούς χρήστες που τοποθετούν το σύστημα στον χώρο τους και επίσης το σύστημα χάνει την αυτόματη ρύθμιση και λειτουργία του χωρίς επιπλέον επέμβαση του χρήστη.

Για να γίνει αποστολή των δεδομένων, αισθητήρων και βίντεο, σε απομακρυσμένη τοποθεσία θα πρέπει να χρησιμοποιηθούν τα πρωτόκολλα tcp ή udp όπου η αποστολή δεδομένων πραγματοποιείται μέσω tcp sockets σε server [17-18]. Η επικοινωνία του συστήματος παρακολούθησης που βρίσκεται πίσω από ένα firewall με έναν χρήστη μπορεί να γίνει μέσω των tcp sockets όπως φαίνεται στο Σχήμα 1.3.

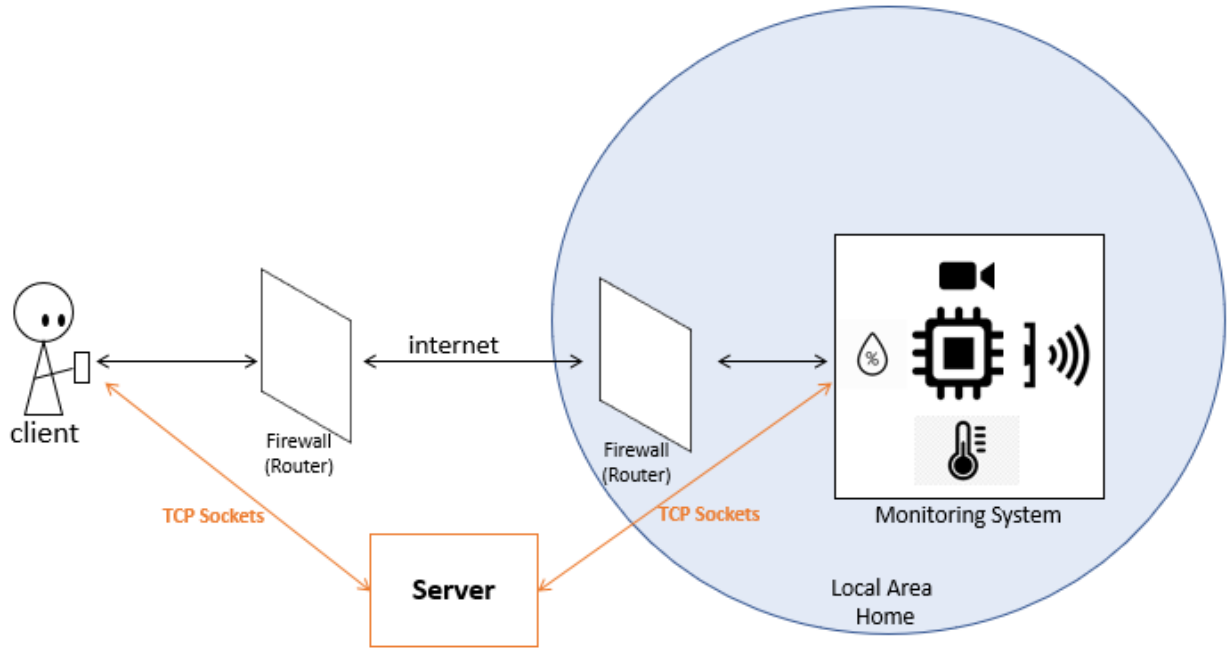
Υπάρχουν εργασίες που αναφέρονται σε αποστολή βίντεο μέσω tcp sockets σε server και λόγω μικρού tcp buffer και ταχύτητας upload η αποστολή παραμένει αργή. [19-20].



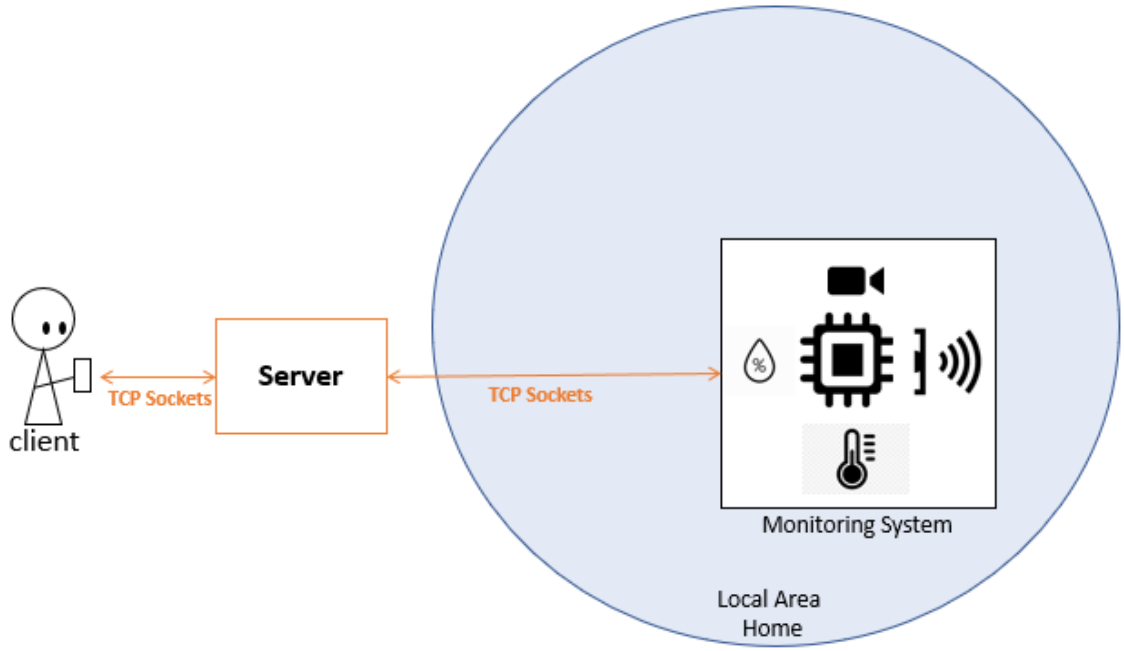
Σχήμα 1.3: Επικοινωνία του συστήματος παρακολούθησης που βρίσκεται πίσω από ένα firewall με έναν χρήστη μέσω των tcp sockets

Η εργασία που υλοποιήθηκε χρησιμοποιεί τον server για την αδιάλειπτη και αμφίδρομη επικοινωνία, χωρίς να επηρεάζεται από κάποιο firewall, του συστήματος παρακολούθησης με τον client. Παρέχεται στον χρήστη πρόσβαση στα δεδομένα μέσω ιστοσελίδας που είναι στον server.

Επίσης, μπορεί χρησιμοποιηθεί η τοπολογία που παρουσιάζεται στο Σχήμα 1.4



Σχήμα 1.4: Επικοινωνία του συστήματος παρακολούθησης με τον χρήστη όταν βρίσκονται πίσω από firewalls μέσω των tcp sockets και ενδιάμεσου server



Σχήμα 1.5: Επικοινωνία του συστήματος παρακολούθησης με τον χρήστη - hole punching

Όπου ο server έχει το ρόλο του μεσολαβητή για την εξυπηρέτηση της αποστολής και λήψης των δεδομένων. Η διαδικασία αυτή λέγεται hole punching ή NAT punch-through [26-27] και συμβαίνει όταν δύο συσκευές θέλουν να επικοινωνήσουν όταν βρίσκονται πίσω από Network address translation

(NAT)[3], δηλαδή firewall-router, με TCP επικοινωνία. Έτσι, η επικοινωνία πραγματοποιείται όπως φαίνεται στο Σχήμα 1.5.

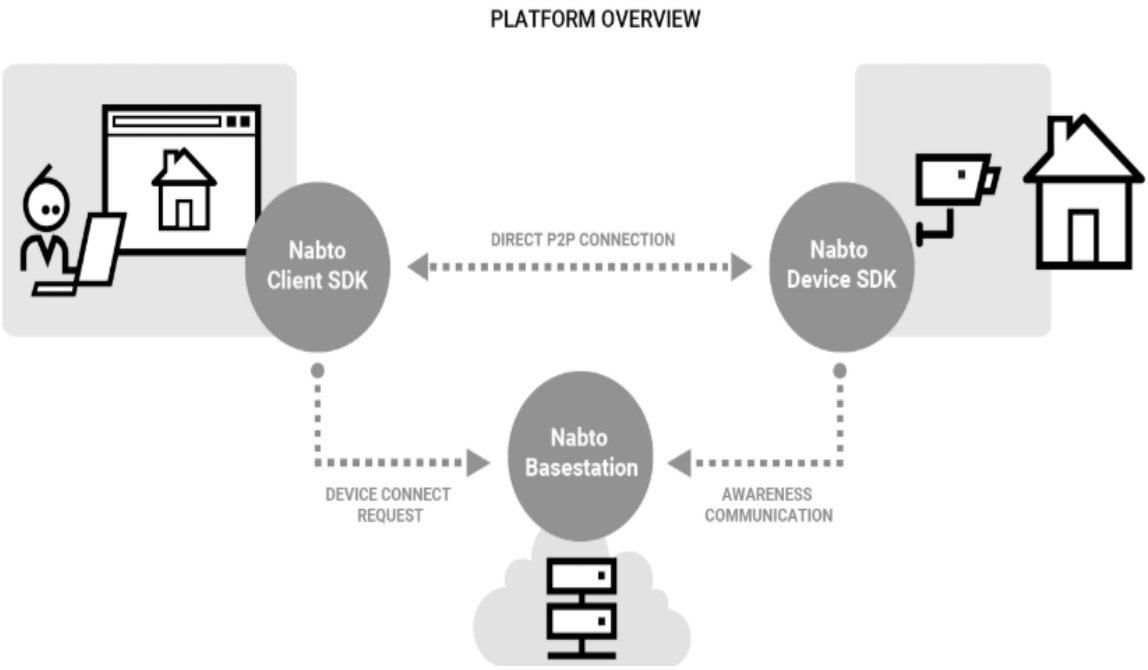
Το Hole punching (ή μερικές φορές punch-through)) είναι μια τεχνική στη δικτύωση υπολογιστών για τη δημιουργία άμεσης σύνδεσης μεταξύ δύο χρηστών όπου το ένα ή και τα δύο βρίσκονται πίσω από τείχη προστασίας ή πίσω από δρομολογητές που χρησιμοποιούν μετάφραση διευθύνσεων δικτύου (NAT). Για να γίνει μια 'τρύπα', κάθε πελάτης συνδέεται με έναν άλλο διακομιστή που αποθηκεύει προσωρινά εξωτερική και εσωτερική διεύθυνση και πληροφορίες θύρας για κάθε πελάτη. Ο διακομιστής στη συνέχεια μεταδίδει τις πληροφορίες κάθε πελάτη στον άλλο, και χρησιμοποιώντας αυτές τις πληροφορίες κάθε πελάτης προσπαθεί να δημιουργήσει άμεση σύνδεση.

Έτσι, οι δικτυακές συσκευές με δημόσιες προσβάσιμες διευθύνσεις IP μπορούν να δημιουργούν συνδέσεις μεταξύ τους εύκολα. Οι πελάτες με ιδιωτικές διευθύνσεις μπορούν επίσης να συνδεθούν εύκολα σε δημόσιους διακομιστές, αρκεί ο πελάτης πίσω από ένα δρομολογητή ή ένα τείχος προστασίας να ξεκινήσει τη σύνδεση. Ωστόσο, απαιτείται διάτρηση ή άνοιγμα 'τρύπας' (ή κάποια άλλη μορφή NAT traversal) για να δημιουργηθεί μια άμεση σύνδεση μεταξύ δύο πελατών που και οι δύο βρίσκονται πίσω από διαφορετικά τείχη προστασίας ή δρομολογητές που χρησιμοποιούν μετάφραση διευθύνσεων δικτύου (NAT).

Και οι δύο πελάτες ξεκινούν μια σύνδεση σε έναν απεριόριστο διακομιστή, ο οποίος σημειώνει πληροφορίες τελικού σημείου και περιόδου σύνδεσης, συμπεριλαμβανομένων δημόσιων IP και θύρας, μαζί με ιδιωτική IP και θύρα. Τα τείχη προστασίας σημειώνουν επίσης τα τελικά σημεία για να επιτρέψουν στις απαντήσεις από τον διακομιστή να περάσουν πίσω. Ο διακομιστής στέλνει έπειτα τις πληροφορίες του τελικού σημείου και της περιόδου σύνδεσης κάθε πελάτη στον άλλο πελάτη. Κάθε πελάτης προσπαθεί να συνδεθεί στον ομότιμό του μέσω της καθορισμένης διεύθυνσης IP και της θύρας που έχει ανοίξει το τείχος προστασίας του διακομιστή για τον διακομιστή. Η νέα προσπάθεια σύνδεσης ανοίγει μια 'τρύπα' στο τείχος προστασίας του πελάτη, καθώς το τελικό σημείο ανοίγει τώρα για να λάβει μια απάντηση από τον ομότιμό του. Ανάλογα με τις συνθήκες δικτύου, ένας ή και οι δύο πελάτες ενδέχεται να λάβουν ένα αίτημα σύνδεσης. Η επιτυχής ανταλλαγή ενός ελέγχου ταυτότητας μεταξύ των δύο πελατών υποδεικνύει την ολοκλήρωση μιας διαδικασίας χρήσης της λεγόμενης 'τρύπας'.

Τα VoIP [31] προϊόντα, οι διαδικτυακές εφαρμογές παιχνιδιών χρησιμοποιούν hole punching.

Μια πολύ καλή προσέγγιση αποστολής δεδομένων μέσω tcp sockets και εξυπηρέτηση από server ξεκίνησε από την Nabto [21-23] και παρουσιάζεται στο Σχήμα 1.6.



Σχήμα 1.6: Τρόπος επικοινωνίας μεταξύ συσκευών που βρίσκονται πίσω από τείχη προστασίας με την μέθοδο που εφαρμόζει η Nabto [21-22]

Η εταιρία παρέχει έτοιμο λογισμικό για να το χρησιμοποιήσει ο προγραμματιστής ώστε να δημιουργήσει εφαρμογές επικοινωνίας μεταξύ του client και του συστήματος παρακολούθησης. Χρησιμοποιεί έναν εξωτερικό server-basestation για να αποθηκεύει στοιχεία πρόσβασης και συνδεδεμένες λίστες που περιέχουν πληροφορίες συσκευής και χρήστη για ασφάλεια στην πρόσβαση αλλά και εγκατάσταση στην επικοινωνία χρήστη-συσκευής. Η εταιρία υποστηρίζει ότι η επικοινωνία μέσω της τεχνικής nat-travesal [24] γίνεται p2p (peer two peer) με απευθείας σύνδεση χωρίς να μεσολαβεί στη συνέχεια ο server-basestation. Βασίζεται σε plugin στον browser και επιχειρεί να εδραιώνει την επικοινωνία μέσω της port 80, η οποία χρησιμοποιείται από το http πρωτόκολλο.

Πολλές εφαρμογές P2P θέλουν να συνδεθούν μεταξύ τους μέσω TCP, αλλά εμποδίζονται τα NAT boxes, δηλαδή από firewalls ή δρομολογητές κτλ. [24-27]

Ορισμένες δημοφιλείς εφαρμογές P2P δεν αντιμετωπίζουν αυτό το εμπόδιο ή το κάνουν άσχημα. Μερικές νεότερες δρομολογούν επικοινωνίες μεταξύ NATed συσκευών μέσω διακομιστών αναμετάδοσης ή μέσω ομότιμων μη NATed, ή ζητούν από τους χρήστες να αναδιαμορφώσουν τους δρομολογητές τους. Ορισμένες αναδυόμενες λύσεις προτείνουν τη χρήση του πρωτόκολλου συνεδρίας για τη ρύθμιση σήραγγας μέσω UDP, τη χρήση UPnP ή ακόμη και την ανάπτυξη IPv6.

Στη συνέχεια θα περιγραφούν διάφορα πρωτόκολλα επικοινωνίας για video streaming.

Rtsp

Το Real Time Streaming Protocol (RTSP) είναι ένα πρωτόκολλο ελέγχου δικτύου που έχει σχεδιαστεί για χρήση σε συστήματα ψυχαγωγίας και επικοινωνιών για τον έλεγχο διακομιστών πολυμέσων. Το πρωτόκολλο χρησιμοποιείται για τον καθορισμό και τον έλεγχο των συνεδριών πολυμέσων μεταξύ των τελικών σημείων [28].

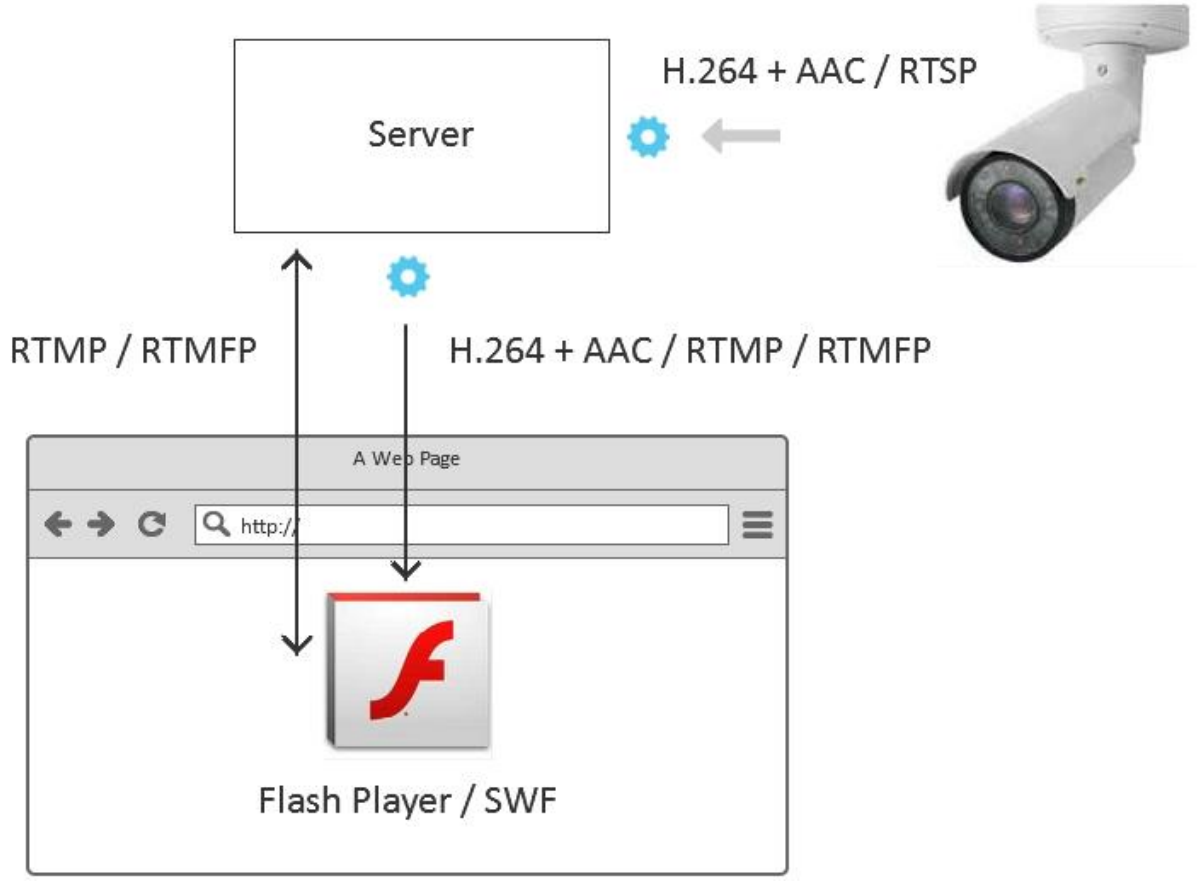
Πολλές IP κάμερες έχουν ενσωματωμένη την πρόσβαση στο πρωτόκολλο αυτό ώστε με ένα απλό Link τις ip διεύθυνσης της κάμερας και εισαγωγή των στοιχείων πρόσβασης ο εξουσιοδοτημένος χρήστης να έχει πρόσβαση στην κάμερα.

Σχήμα 1.7: RTSP πρωτόκολλο

Ένα απλό παράδειγμα είναι η χρήση του VLC player και γράφοντας το URL rtsp://IP Camera's IP Address/channel1 θα μπορεί κάποιος να δει το βίντεο που προέρχεται από την κάμερα. Για πχ rtsp://192.168.1.100/channel1.

Rtmp

Το Real-Time Messaging Protocol (RTMP) ήταν αρχικά ένα ιδιόκτητο πρωτόκολλο που αναπτύχθηκε από τη Macromedia για ροή ήχου, βίντεο και δεδομένων μέσω του Διαδικτύου, μεταξύ ενός προγράμματος αναπαραγωγής Flash και ενός διακομιστή. Η Macromedia ανήκει πλέον στην Adobe, η οποία κυκλοφόρησε μια ελλιπή έκδοση των προδιαγραφών του πρωτοκόλλου για δημόσια χρήση. Στο RTMP Tunneled (RTMPT), τα δεδομένα RTMP ενθυλακώνονται και ανταλλάσσονται μέσω HTTP και τα μηνύματα από τον πελάτη (το πρόγραμμα αναπαραγωγής πολυμέσων, σε αυτήν την περίπτωση) απευθύνονται στη θύρα 80 (η προεπιλογή για HTTP) στον διακομιστή. [29-30]

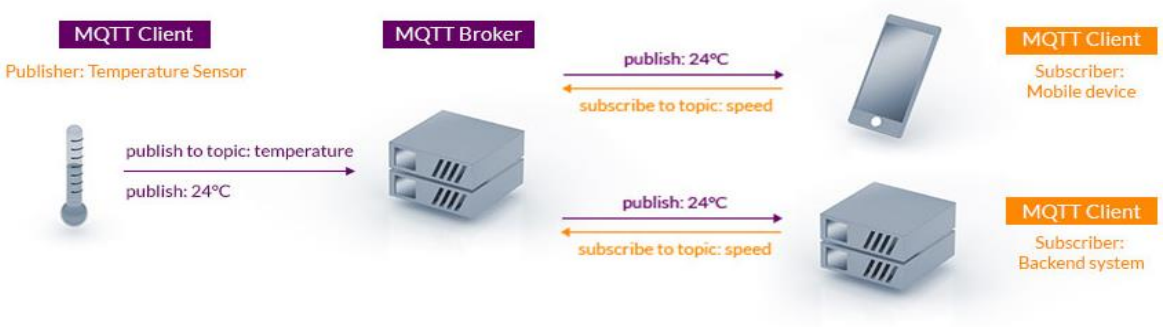


Σχήμα 1.8: RTMP πρωτόκολλο: https://flashphoner.com/7-ways-to-stream-rtsp-on-the-page/

MQTT

Το MQTT (Transport Queue Telemetry Transport) είναι ένα πρότυπο πρωτοκόλλου ανταλλαγής μηνυμάτων "μικρού μήκους" βάσει προτύπου ISO (ISO / IEC PRF 20922) για χρήση πάνω από το πρωτόκολλο TCP / IP. Έχει σχεδιαστεί για συνδέσεις με απομακρυσμένες τοποθεσίες όπου απαιτείται μικρή μεταφορά δεδομένων ή το εύρος ζώνης δικτύου είναι περιορισμένο.

Είναι πολύ εύκολο να χρησιμοποιηθεί το πρωτόκολλο MQTT για την ανταλλαγή μικρών μηνυμάτων μεταξύ πολλών συσκευών αρκεί να υπάρχει ένας μεσολαβητής-server που να έχει το ρόλο του συντονιστή και να εξυπηρετεί όποιον συνδέεται σε αυτόν, όπως φαίνεται στο Σχήμα 1.9. [49-50]



Σχήμα 1.9: MQTT πρωτόκολλο: https://mqtt.org/

1.2 Συνεισφορά της εργασίας

Στην εργασία παρουσιάζεται ένα σύστημα παρακολούθησης χώρου που χρησιμοποιεί tcp sockets για την επικοινωνία του χρήστη με το σύστημα όταν και οι δύο μπορεί να βρίσκονται πίσω από δρομολογητές ή firewalls. Η εργασία μέσω του κώδικα και της ανάλυσης που παρέχει συνεισφέρει στα επιμέρους μέρη:

- Κώδικας για λήψη και επεξεργασία βίντεο στο Raspberry με μια απλή web camera
- Κώδικας για λήψη και επεξεργασία δεδομένων από αισθητήρες που είναι συνδεδεμένοι στο Raspberry
- Αμφίδρομη και συνεχή επικοινωνία του συστήματος παρακολούθησης με τους χρήστες με tcp sockets
- Αποστολή βίντεο με tcp sockets σε εξυπηρετητή και προβολή σε ιστοσελίδα
- Με τη μέθοδο hole punching υλοποιήθηκε επικοινωνία μεταξύ συστήματος και χρήστη όταν βρίσκονται πίσω από ένα δρομολογητή ή firewall.
- Χαμηλής κατανάλωσης και μικρού κόστους σύστημα παρακολούθησης

1.3 Δομή της εργασίας

Στο πρώτο κεφάλαιο παρουσιάζεται μια μικρή εισαγωγή της εργασίας και οι στόχοι της και μια εκτενής βιβλιογραφική ανασκόπηση παρόμοιων συστημάτων. Επιπρόσθετα, αναφέρεται η συνεισφορά της εργασίας και σε ποια σημεία επικεντρώνεται η ανάλυση της.

Στο δεύτερο κεφάλαιο περιγράφεται η ηλεκτρονική διάταξη του συστήματος και περιγράφονται το Raspberry και οι αισθητήρες που χρησιμοποιήθηκαν. Στη συνέχεια αναφέρονται η γλώσσα προγραμματισμού Python, το πακέτο βιβλιοθηκών OpenCV που απαιτείται για την λήψη βίντεο και αναλύεται η TCP επικοινωνία και τα TCP sockets που είναι απαραίτητα για την επικοινωνία του συστήματος με τον απομακρυσμένο χρήστη.

Στο τρίτο κεφάλαιο αναλύονται τα επιμέρους κομμάτια τη εργασία, όπως η επικοινωνία αισθητήρων με το Raspberry, η λήψη βίντεο και φωτογραφιών από την κάμερα με το Raspberry, η επικοινωνία client και server με αμφίδρομη επικοινωνία μέσω sockets, η αποστολή βίντεο-frames στον server, η κατασκευή server-side στον server και η περιγραφή Ιστοσελίδας για την Διεπαφή Χρήστη-Συστήματος μέσω Server. Στο τέλος, παρουσιάζεται η ανάλυση απόδοσης της αποστολής δεδομένων και ανάλυση ασφάλειας στο δίκτυο και στα δεδομένα.

Στο τέταρτο κεφάλαιο παρουσιάζονται τα συμπεράσματα της εργασίας και θέματα για μελλοντική έρευνα ενώ στο τέλος της εργασίας παρατίθεται το παράρτημα με τους κώδικες που χρησιμοποιήθηκαν στην εργασία.

Κεφάλαιο 2ο: Το σύστημα παρακολούθησης

2.1 Εισαγωγή

Το σύστημα παρακολούθησης που υλοποιήθηκε αποτελείται κυρίως από μια web κάμερα που παρακολουθεί τον χώρο και από ενδεικτικούς αισθητήρες όπως φαίνεται στο παρακάτω Σχήμα 2.1.

Σχήμα 2.1: Σύστημα Παρακολούθησης με τους Αισθητήρες

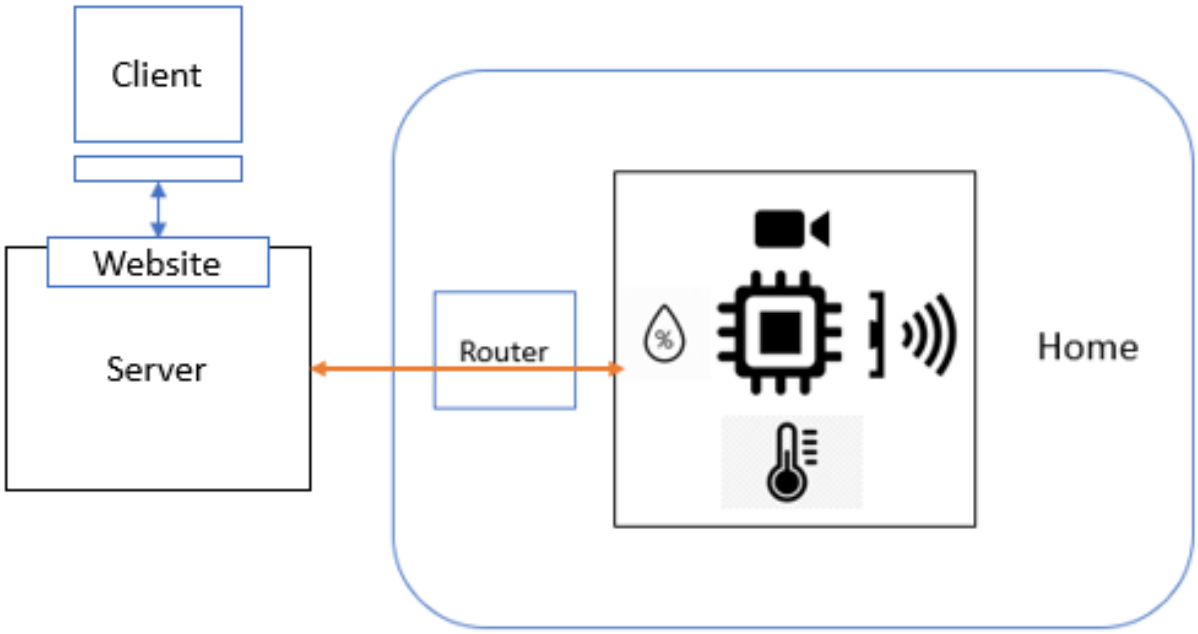
Η κάμερα που χρησιμοποιήθηκε είναι η Logitech HD Webcam C270

Οι αισθητήρες που χρησιμοποιήθηκαν είναι:

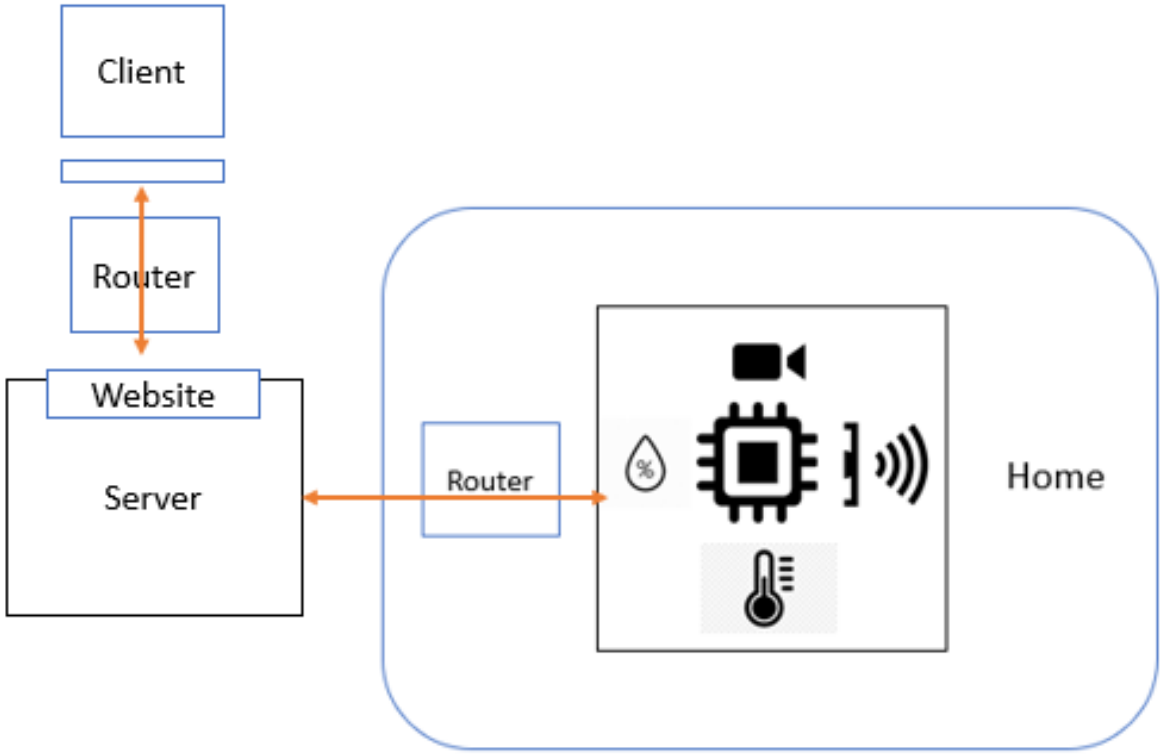
Θερμοκρασίας-Υγρασίας: DHT11 [32]

Ανίχνευσης κίνησης: HC-SR501 [33]

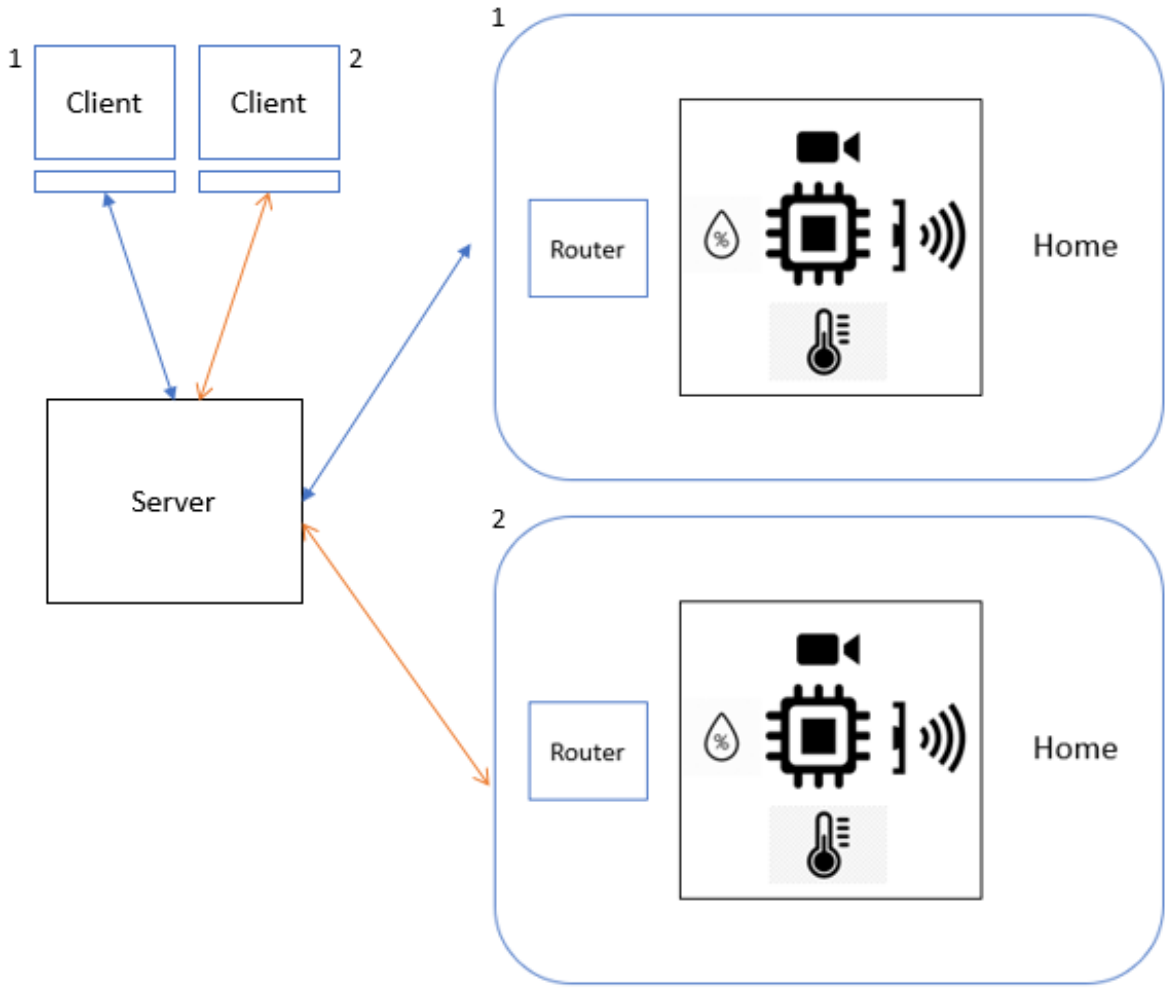
Η επικοινωνία του τελικού χρήστη με το σύστημα υλοποιήθηκε με tcp sockets μέσω ενός server ώστε να μπορούν να επικοινωνούν ακόμα και αν βρίσκονται πίσω από router ή firewall όπως φαίνεται στα Σχήματα 2.2 και 2.3.



Σχήμα 2.2: Επικοινωνία χρήστη και συστήματος παρακολούθησης (όταν το σύστημα βρίσκεται πίσω από δρομολογητή ή τείχος προστασίας)



Σχήμα 2.3: Επικοινωνία χρήστη και συστήματος παρακολούθησης (όταν και οι δύο βρίσκονται πίσω από δρομολογητή ή τείχος προστασίας)



Σχήμα 2.4: Επικοινωνία πολλών χρηστών και συστημάτων παρακολούθησης

Μέσω του server μπορούν να συνδεθούν παραπάνω χρήστες στο ίδιο σύστημα ή σε διαφορετικό ανάλογα την αντιστοιχία Χρήστη-Συσκευής που αποθηκεύεται στη βάση του server, όπως φαίνεται στο Σχήμα 2.4.

2.2 Ηλεκτρονική Διάταξη

Στο Σχήμα 2.5 παρουσιάζεται η ηλεκτρονική διάταξη που χρησιμοποιήθηκε για την υλοποίηση του συστήματος παρακολούθησης. Η προσθήκη και χρήση αισθητήρων είναι ενδεικτική διότι το σύστημα εστιάζει περισσότερο στην επικοινωνία του συστήματος με τον χρήστη μέσω διαδικτύου και στην αποστολή βίντεο-frames.

Σχήμα 2.5: Ηλεκτρονική Διάταξη

2.2.1 Raspberry

Το Raspberry Pi 3 [34] είναι ένας πλήρης υπολογιστής σε μέγεθος πιστωτικής κάρτας.

Παρά τον μικρό όγκο του, το Raspberry Pi στη τελευταία του έκδοση διαθέτει έναν τετραπύρηνο επεξεργαστή 1200MHz, μια διπύρηνη κάρτα γραφικών με GPU, 1GB RAM, τέσσερις θύρες USB, έχει έξοδο HDMI, τροφοδοτείται μέσω Micro USB και έχει 40 pins γενικής χρήσης για σύνδεση με άλλες ηλεκτρονικές συσκευές και περιφερειακά. Συνδέεται σε μια οθόνη και με ένα πληκτρολόγιο/ποντίκι γίνεται ένας υπολογιστής με λειτουργικό Linux.

2.2.1.1 Βασικά Χαρακτηριστικά

Τα βασικά χαρακτηριστικά του Raspberry Pi 3 είναι:

SoC: Broadcom BCM2837

CPU: 4× ARM Cortex-A53, 1.2GHz

GPU: Broadcom VideoCore IV

RAM: 1GB LPDDR2 (900 MHz)

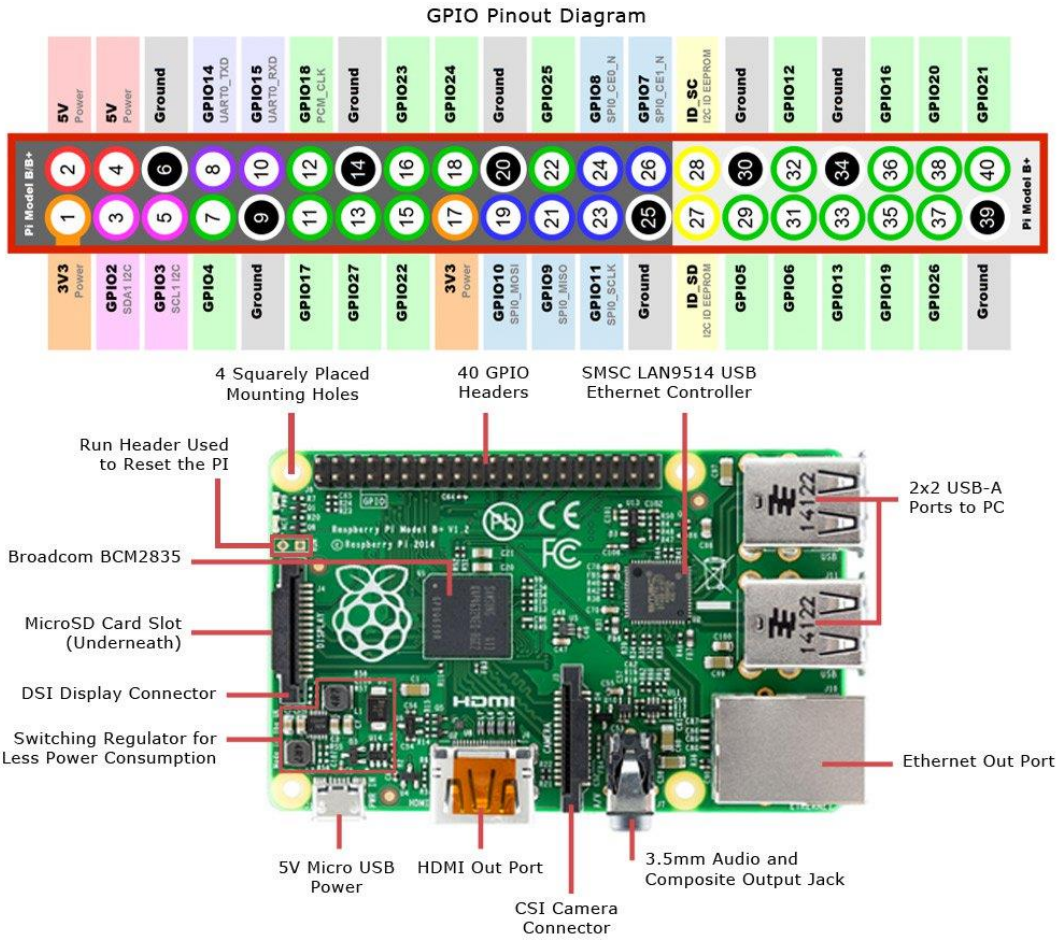
Networking: 10/100 Ethernet, 2.4GHz 802.11n wireless

Bluetooth: Bluetooth 4.1 Classic, Bluetooth Low Energy

Storage: microSD

GPIO: 40-pin header, populated

Ports: HDMI, 3.5mm analogue audio-video jack, 4× USB 2.0, Ethernet, Camera Serial Interface (CSI), Display Serial Interface (DSI)



Σχήμα 2.6: Πλακέτα Raspberry Pi 3 με τους ακροδέκτες

Κατά την περίοδο υλοποίησης της εργασίας το Raspberry Pi 3 κόστιζε πάνω από 40 ευρώ. Για την εργασία χρησιμοποιήθηκε και το μικρότερο μοντέλο Raspberry Pi Zero W [35] με κόστος 10 ευρώ.

Σχήμα 2.7: Πλακέτα Raspberry Pi Zero W

του οποίου τα χαρακτηριστικά παρουσιάζονται παρακάτω:

- 802.11 b/g/n wireless LAN
- Bluetooth 4.1
- Bluetooth Low Energy (BLE)
- 1GHz, single-core CPU
- 512MB RAM
- Mini HDMI and USB On-The-Go ports
- Micro USB power
- HAT-compatible 40-pin header
- Composite video and reset headers
- CSI camera connector

2.2.1.2 Ιστορική Αναδρομή

Το 2008 κυκλοφόρησε το πρώτο Raspberry Pi, με τα Model A, Model A+ και Model B.

Τα μοντέλα αυτά διέθεταν επεξεργαστή ARMv6k στα 700 MHz, 256MB RAM, κάρτα γραφικών Broadcom VideoCore IV, και κατανάλωση από 1 έως 3.5 watt, ενώ η αποθήκευση των δεδομένων γινόταν σε κάρτες SD, SDHC και MicroSD. Σε δύο χρόνια το Raspberry Pi Model B πούλησε πάνω από δύο εκατομμύρια κομμάτια. Στη συνέχεια κυκλοφόρησαν οι εκδόσεις Model B rev 2 και Model B+ με 512MB RAM. Τον Οκτώβριο του 2014, οι συνολικές πωλήσεις άγγιζαν τα 4 εκατομμύρια.

Το Φεβρουάριο του 2015 κυκλοφόρησε το Raspberry Pi Generation 2 Model B, με RAM 1GB και 6 φορές μεγαλύτερη ταχύτητα επεξεργαστή. Χρησιμοποιούσε τον τετραπύρηνο Cortex-A7 (ARMv7) και διπύρηνη κάρτα γραφικών Broadcom VideoCoreIV.

Το 2015 κυκλοφόρησε το Raspberry Pi Zero, το μικρό του Raspberry Pi, 512MB RAM, και επεξεργαστή ARM1176JZF-S στα 1GHz.

Το raspberry που χρησιμοποιήθηκε στην εργασία είναι το Raspberry Pi Generation 3 Model B κυκλοφόρησε το Φεβρουάριο του 2016. Ακόμα ταχύτερος επεξεργαστής ARM Cortex-A53 στα 1.2GHz, 1GB RAM και κάρτα γραφικών Broadcom VideoCore IV χρονισμένη στα 250MHz, συχνότητα υψηλότερη από κάθε προηγούμενη έκδοση.

Αξίζει να σημειωθεί ότι το Raspberry Pi έχει κατανάλωση μόλις 4W, ενώ το Raspberry pi zero w κάτω από 1W.

2.2.1.3 Διανομές Λειτουργικού Συστήματος

Όπως αναφέρθηκε συνδέοντάς το σε μια οθόνη και προσθέτοντας πληκτρολόγιο και ποντίκι είναι ένας πλήρης υπολογιστής, ο οποίος υποστηρίζει συγκεκριμένες διανομές Linux.

Αυτή τη στιγμή, οι διανομές που υποστηρίζονται στην τελευταία έκδοση του Raspberry Pi είναι οι εξής:[36]

- Raspbian
- Kali Linux
- Pidora
- Windows 10 IoT Core
- Ubuntu Core
- RISC OS
- Arch Linux ARM
- FreeBSD
- RetroPie

2.2.1.4 Τρόποι επικοινωνίας του Raspberry με το περιβάλλον - αισθητήρες

Το Raspberry έχει δύο συγκεκριμένες λειτουργίες μέσω των ακροδεκτών του, να μπορεί να γίνει έξοδος ή είσοδος. Χρησιμοποιώντας αυτούς τους ακροδέκτες μπορούμε να χρησιμοποιήσουμε κάθε ψηφιακό αισθητήρα αλλά όταν θέλουμε να χρησιμοποιήσουμε αναλογικό αισθητήρα δεν μπορούμε να το συνδέσουμε απευθείας στο Raspberry αλλά μόνο με την χρήση ειδικών μετατροπέων αναλογικού σε ψηφιακό σήμα.

Ενώ μπορούν να διαβάσουν 1 ή 0 και να γράψουν 1 ή 0 στην έξοδο μπορούν να εκτελέσουν πολλές λειτουργίες γιατί το πρόγραμμα τους γράφεται σε γλώσσα προγραμματισμού όπως η Python προσομοιώνοντας πολλές λειτουργίες μικροελεγκτών, όπως SPI, I2C, serial, usb κτλ.

2.2.2 Αισθητήρες

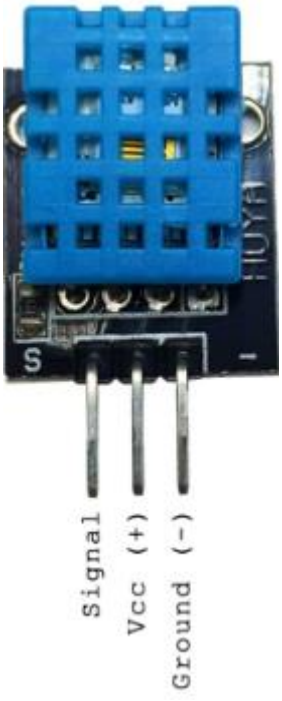
Το σύστημα παρακολούθησης έχει σαν βασικό περιφερειακό την κάμερα με την οποία εποπτεύεται ο χώρος. Σε αυτό μπορούν να τοποθετηθούν διάφοροι αισθητήρες και διάφορες διατάξεις εξόδου όπως είναι leds, σειρήνες, ηχείο, ηλεκτρονόμος κτλ. Οι αισθητήρες διακρίνονται σε δύο κατηγορίες, όσον αφορά τη σύνδεση τους με το raspberry, τους ψηφιακούς και τους αναλογικούς,

Οι ψηφιακοί αισθητήρες επικοινωνούν πιο εύκολα με το raspberry ενώ οι αναλογικοί χρειάζονται A/D μετατροπέα. Στην εργασία για λόγους απλότητας χρησιμοποιήθηκαν δύο βασικοί ψηφιακής εξόδου αισθητήρες, ένας αισθητήρας θερμοκρασίας-υγρασίας και ένας αισθητήρας κίνησης.

2.2.2.1 Αισθητήρας Θερμοκρασίας - Υγρασίας

Οι αισθητήρες DHT11 και DHT22 μπορούν να μετρήσουν την υγρασία και τη θερμοκρασία. Χρησιμοποιείται μόνο ένα GPIO.

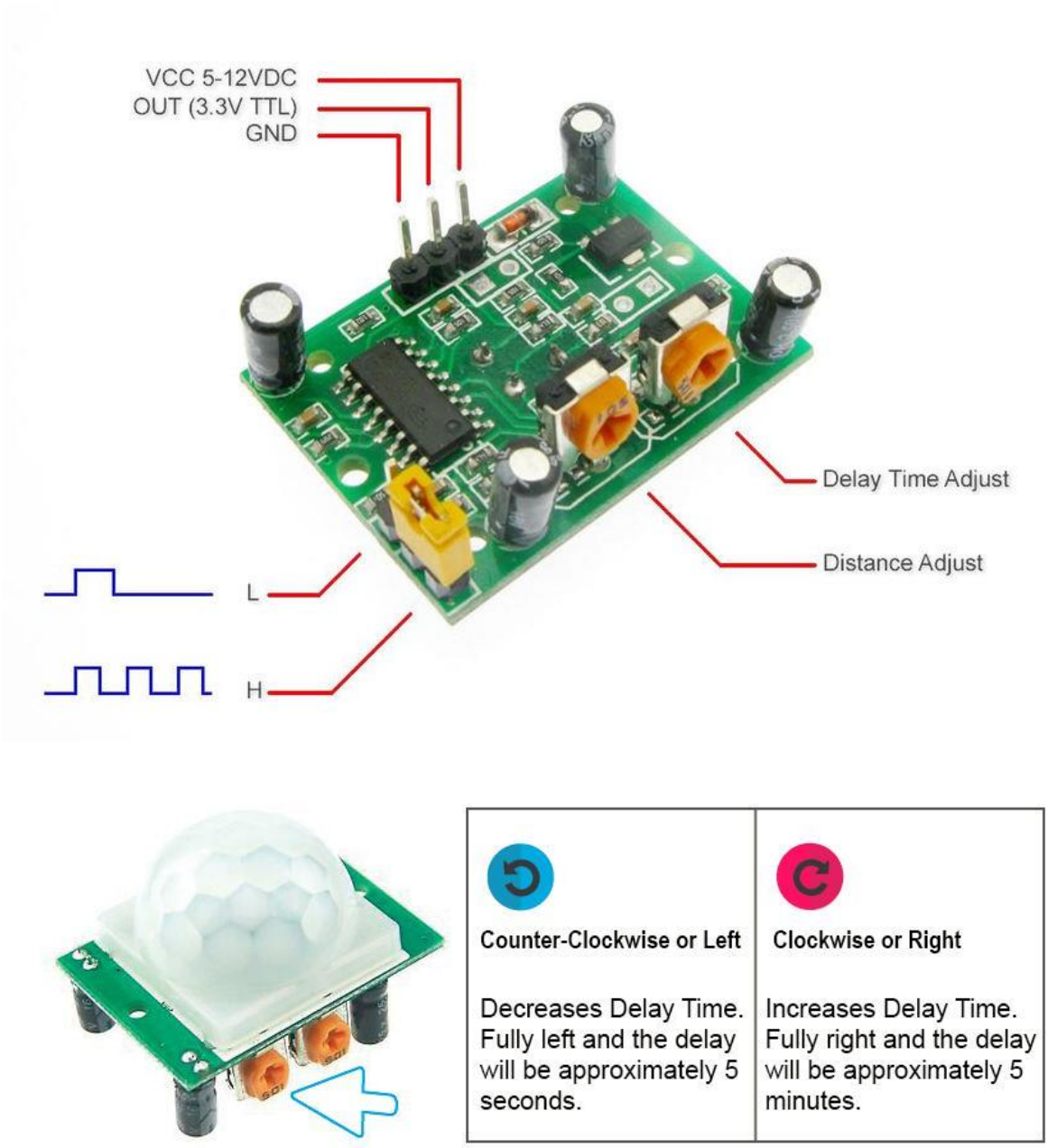
Η διαφορά μεταξύ των δύο είναι κυρίως η περιοχή μέτρησης και η ακρίβεια.



Σχήμα 2.8: Αισθητήρας DTH11

2.2.2.2 Αισθητήρας Κίνησης

Ο αισθητήρας κίνησης PIR έχει κάποια πλεονεκτήματα έναντι άλλων παρόμοιων προϊόντων: Εκτός από τη χαμηλή τιμή, αποστέλλεται σήμα μόνο αν ανιχνεύσει κίνηση στον χώρο.



Σχήμα 2.9: Αισθητήρας κίνησης PIR [37-38]

Όταν το Jumber είναι σε κατάσταση Low τότε όταν ο αισθητήρας ανιχνεύσει κίνηση η έξοδος γίνεται λογικό 1 και παραμένει σε αυτήν μέχρι να περάσει ο χρόνος που ορίστηκε με το Delay Time Adjust (5s-5min). Σε High κατάσταση η έξοδος εναλλάσσεται ανάμεσα σε λογικό 0 και 1 όσο ανιχνεύει κίνηση.

2.3 Python

Η Python [39] είναι μια γλώσσα γενικής χρήσης - που σημαίνει ότι, σε αντίθεση με HTML, CSS και JavaScript, μπορεί να χρησιμοποιηθεί για σε άλλους τύπους προγραμματισμού και ανάπτυξης λογισμικού εκτός από την ανάπτυξη web εφαρμογών.

Αυτό περιλαμβάνει, μεταξύ άλλων, ανάπτυξη back-end εφαρμογών, ανάπτυξη λογισμικού, επιστήμη δεδομένων κτλ.

2.3.1 Χρήσεις Python

1. Εφαρμογές Διαδικτύου

Επειδή η Python διαθέτει προεγκατεστημένες βιβλιοθήκες και frameworks για Εφαρμογές Διαδικτύου, συμπεριλαμβανομένων των Pyramid, Django και Flask, είναι ιδιαίτερα εξαιρετική για ανάπτυξη front-end και back-end έργων, μειώνοντας τον χρόνο σε έργα, επιτρέποντάς την επαναχρησιμοποίηση κώδικα.

2. Επιστημονικός Υπολογισμός + Επιστήμη Δεδομένων

Το Python χρησιμοποιείται επίσης για την επιστημονική έρευνα και την πληροφορική και διαθέτει ακόμη και βιβλιοθήκες φιλικές προς την επιστήμη ή ειδικές για την επιστήμη, όπως:

Astropy για την αστρονομία

Biopython για βιολογία και βιοπληροφορική

Γράφημα-εργαλείο για στατιστική ανάλυση γραφημάτων

3. Μηχανική Μάθηση

Η μηχανική μάθηση εμπίπτει στην επιστήμη των δεδομένων. Η μηχανική εκμάθηση περιλαμβάνει πράγματα όπως αναγνώριση ομιλίας, χρηματοοικονομικές υπηρεσίες κτλ. Η python περιέχει πληθώρα από βιβλιοθήκες που μπορούν με λίγες γραμμές εντολές κώδικά να εξάγουν πολύ ικανοποιητικά αποτελέσματα. [40]

2.3.2 Γιατί χρησιμοποιήθηκε η Python

Η python χρησιμοποιήθηκε στην εργασία καταρχήν λόγω πληθώρας βιβλιοθηκών για την χρήση των ακροδεκτών και των λειτουργιών του Raspberry. Επιπλέον, λόγω ευκολίας στην χρήση της βιβλιοθήκης OpenCV για την λήψη και επεξεργασία βίντεο από την κάμερα και για την εύκολη διαχείριση των TCP sockets μέσω των ενσωματωμένων πακέτων της.

2.4 OpenCV

2.4.1 Εισαγωγή

Το OpenCV [41] (Open Source Computer Vision Library) είναι μια βιβλιοθήκη προγραμματισμού που στοχεύει σε πραγματικού χρόνου υπολογιστικής όρασης[42]. Αρχικά αναπτύχθηκε από την Intel, στη συνέχεια υποστηρίχθηκε από τον Willow Garage - Itseez [43](το οποίο αργότερα εξαγοράστηκε από

την Intel). Η βιβλιοθήκη είναι πολλαπλής πλατφόρμας και δωρεάν για χρήση με την άδεια BSD ανοιχτού κώδικα.

Οι περιοχές εφαρμογής του OpenCV περιλαμβάνουν:

2D and 3D feature toolkits

Egomotion estimation

Facial recognition system

Gesture recognition

Human–computer interaction (HCI)

Mobile robotics

Motion understanding

Object identification

Segmentation and recognition

Stereopsis stereo vision: depth perception from 2 cameras

Structure from motion (SFM)

Motion tracking

Augmented reality

Για να υποστηρίξει ορισμένες από τις παραπάνω περιοχές, το OpenCV περιλαμβάνει μια βιβλιοθήκη στατιστικής μηχανικής μάθησης που περιέχει:

Boosting

Decision tree learning

Gradient boosting trees

Expectation-maximization algorithm

k-nearest neighbor algorithm

Naive Bayes classifier

Artificial neural networks

Το OpenCV είναι γραμμένο σε C++

Υπάρχουν πακέτα για Python, Java και MATLAB/OCTAVE.

Τελευταία αναπτύχθηκαν πακέτα υποστήριξης και για την Javascript.

2.4.2 Λήψη Βίντεο με το OpenCV

Συχνά, χρειάζεται η καταγραφή ζωντανή ροής - βίντεο με κάμερα. Το OpenCV παρέχει μια πολύ απλή διεπαφή για να το κάνει αυτό [45].

Για την καταγραφή ενός βίντεο, πρέπει να δημιουργηθεί ένα αντικείμενο VideoCapture. Πρέπει να δηλωθεί ποια κάμερα θα ενεργοποιήσει. Η πρώτη κάμερα βρίσκεται στο 0, η δεύτερη στο 1 και ούτω καθεξής. Μετά από αυτό καταγράφει καρέ-καρέ. Στο τέλος, απελευθερώνεται η λήψη – το αντικείμενο.

Παρακάτω παρουσιάζεται ο κώδικας για την λήψη βίντεο από την κάμερα, μετατροπή σε γκρι απόχρωση και προβολή του βίντεο στην οθόνη

```
import numpy as np
import cv2 as cv
cap = cv.VideoCapture(0)
if not cap.isOpened():
    print("Cannot open camera")
    exit()
while True:
    # Capture frame-by-frame
    ret, frame = cap.read()
    # if frame is read correctly ret is True
    if not ret:
        print("Can't receive frame (stream end?). Exiting ...")
        break
    # Our operations on the frame come here
    gray = cv.cvtColor(frame, cv.COLOR_BGR2GRAY)
    # Display the resulting frame
    cv.imshow('frame', gray)
    if cv.waitKey(1) == ord('q'):
        break
# When everything done, release the capture
cap.release()
cv.destroyAllWindows()
```

Από τις ποιο σημαντικές εντολές είναι η cap.read() η οποία επιστρέφει True ή False. Επιστρέφει True όταν το frame λήφθηκε σωστά. Επίσης τότε λαμβάνει το frame και το αποθηκεύει προσωρινά στη μεταβλητή frame η οποία μπορεί να επεξεργαστεί.

Μερικές φορές, το cap μπορεί να μην έχει αρχικοποιήσει τη λήψη. Σε αυτήν την περίπτωση, αυτός ο κωδικός εμφανίζει σφάλμα. Ο έλεγχος αν έχει αρχικοποιηθεί ή όχι γίνεται με τη μέθοδο cap.isOpened (). Εάν είναι True τότε είναι σωστή. Διαφορετικά, πρέπει να χρησιμοποιηθεί το cap.open ().

Η αναπαραγωγή βίντεο από αρχείο είναι η ίδια με τη λήψη από την κάμερα, αλλάζοντας το όνομα της κάμερας σε όνομα αρχείου βίντεο. Επίσης, για την διευθέτηση του ρυθμού μετάδοσης, ρυθμίζεται ο κατάλληλος χρόνος για cv.waitKey (). Εάν είναι πολύ λιγότερο, το βίντεο θα είναι πολύ γρήγορο και αν είναι πολύ υψηλό, το βίντεο θα είναι αργό.

2.5 TCP Επικοινωνία

TCP Πρωτόκολλο

Το TCP (Transmission Control Protocol) είναι ένα πρότυπο που καθορίζει τον τρόπο δημιουργίας και συντήρησης μιας επικοινωνίας δικτύου μέσω του οποίου τα προγράμματα μπορούν να ανταλλάξουν δεδομένα. Το TCP λειτουργεί με το Internet Protocol (IP)[46], το οποίο καθορίζει τον τρόπο με τον οποίο οι υπολογιστές στέλνουν πακέτα δεδομένων μεταξύ τους. Μαζί, το TCP και το IP είναι οι βασικοί κανόνες που καθορίζουν το Διαδίκτυο.

Το TCP είναι προσανατολισμένο στη σύνδεση και αυτή δημιουργείται μεταξύ πελάτη και διακομιστή πριν από την αποστολή δεδομένων. Ο διακομιστής πρέπει να ακούει (παθητικό ανοιχτό) για αιτήματα σύνδεσης από πελάτες πριν από τη σύνδεση. Η three-way χειραψία (ενεργή ανοιχτή), η αναμετάδοση και η ανίχνευση σφαλμάτων αυξάνουν την αξιοπιστία, αλλά επιμηκύνουν την καθυστέρηση. Οι εφαρμογές που δεν απαιτούν αξιόπιστη υπηρεσία ροής δεδομένων μπορούν να χρησιμοποιούν το User Datagram Protocol (UDP)[47].

UDP Πρωτόκολλο

Το UDP χρησιμοποιεί ένα απλό μοντέλο επικοινωνίας χωρίς σύνδεση με ελάχιστους μηχανισμούς πρωτοκόλλου. Το UDP παρέχει αθροίσματα ελέγχου για ακεραιότητα δεδομένων και αριθμούς θύρας για την αντιμετώπιση διαφορετικών λειτουργιών στην πηγή και τον προορισμό του datagram. Δεν έχει διαλόγους χειραψίας, και έτσι εκθέτει το πρόγραμμα του χρήστη σε οποιαδήποτε αναξιοπιστία του υποκείμενου δικτύου. Δεν υπάρχει εγγύηση παράδοσης, παραγγελίας ή διπλής προστασίας. Εάν απαιτούνται ενέργειες διόρθωσης σφαλμάτων σε επίπεδο διασύνδεσης δικτύου, μια εφαρμογή μπορεί να χρησιμοποιήσει το Transmission Control Protocol (TCP) που έχει σχεδιαστεί για το σκοπό αυτό.

Εύρος Αριθμών για τις Θύρες-Ports

Ένας αριθμός θύρας χρησιμοποιεί 16 bit και συνεπώς μπορεί να έχει τιμή από 0 έως 65535.

Οι αριθμοί θύρας χωρίζονται σε περιοχές ως εξής:

Αριθμοί θύρας 0-1023: Γνωστές Θύρες. Αυτά κατανέμονται σε υπηρεσίες για το διακομιστή από το Internet Assigned Numbers Authority (IANA) [48] π.χ. οι διακομιστές Web χρησιμοποιούν συνήθως τη θύρα 80 και οι διακομιστές SMTP χρησιμοποιούν τη θύρα 25.

Θύρες 1024-49151: Καταχωρημένες Θύρες - Αυτές μπορούν να εγγραφούν για υπηρεσίες στο IANA και θα πρέπει να αντιμετωπίζονται ως ημι-δεσμευμένα. Οι χρήστες δεν πρέπει να χρησιμοποιούν αυτές τις θύρες.

Θύρες 49152-65535: Χρησιμοποιούνται από προγράμματα-εφαρμογές και είναι δωρεάν. Όταν ένα πρόγραμμα περιήγησης Web συνδέεται με έναν διακομιστή ιστού, το πρόγραμμα περιήγησης θα διαθέσει μια θύρα σε αυτό το εύρος. Είναι γνωστές ως εφήμερες θύρες.

Σε ένα δίκτυο TCP / IP κάθε συσκευή πρέπει να έχει διεύθυνση IP.

Η διεύθυνση IP προσδιορίζει τη συσκευή π.χ. υπολογιστή.

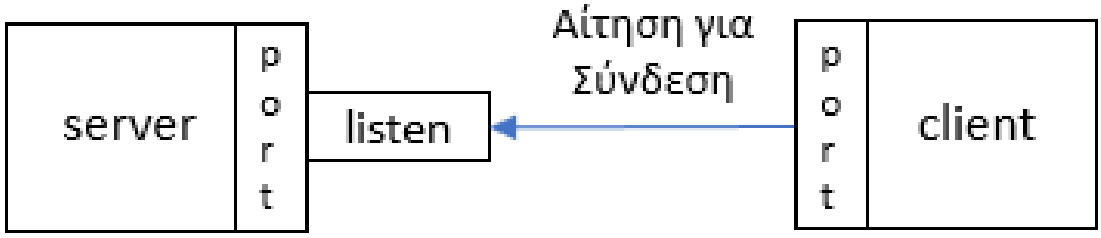
Ωστόσο, μόνο μια διεύθυνση IP δεν επαρκεί για την εκτέλεση εφαρμογών δικτύου, καθώς ένας υπολογιστής μπορεί να εκτελεί πολλαπλές εφαρμογές ή / και υπηρεσίες.

TCP Sockets

Το socket είναι ένα τελικό σημείο μιας αμφίδρομης σύνδεσης επικοινωνίας μεταξύ δύο προγραμμάτων που τρέχουν στο δίκτυο. Ένα socket συνδέεται με έναν αριθμό θύρας έτσι ώστε το επίπεδο TCP να μπορεί να αναγνωρίσει την εφαρμογή στην οποία προορίζονται να σταλούν τα δεδομένα.

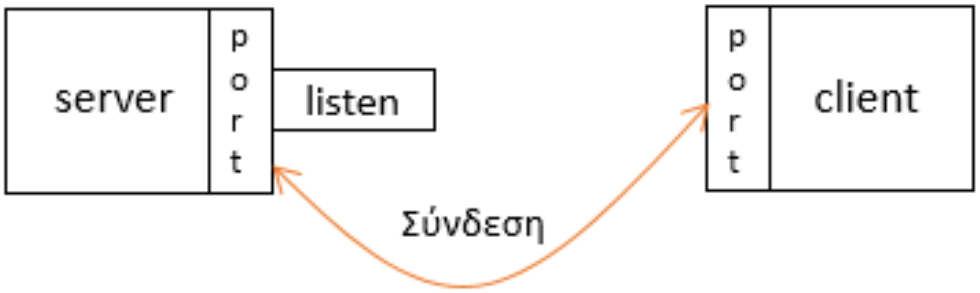
Κανονικά, ένας διακομιστής εκτελείται σε έναν συγκεκριμένο υπολογιστή και έχει ένα socket που συνδέεται με έναν συγκεκριμένο αριθμό θύρας. Ο διακομιστής περιμένει, ακούγοντας το socket για έναν πελάτη, να κάνει ένα αίτημα σύνδεσης.

Από την πλευρά του πελάτη: Ο πελάτης γνωρίζει το όνομα κεντρικού υπολογιστή του μηχανήματος στον οποίο εκτελείται ο διακομιστής και τον αριθμό θύρας στην οποία ακούει ο διακομιστής. Για να υποβάλει ένα αίτημα σύνδεσης, ο πελάτης προσπαθεί να συναντηθεί με τον διακομιστή στο μηχάνημα και τη θύρα του διακομιστή. Ο πελάτης πρέπει επίσης να ταυτιστεί με τον διακομιστή, ώστε να συνδέεται με έναν τοπικό αριθμό θύρας που θα χρησιμοποιήσει κατά τη διάρκεια αυτής της σύνδεσης. Αυτό ανατίθεται συνήθως από το σύστημα.



Σχήμα 2.10: Αίτηση για σύνδεσης με tcp socket από τον client

Εάν όλα πάνε καλά, ο διακομιστής αποδέχεται τη σύνδεση. Μετά την αποδοχή, ο διακομιστής παίρνει ενα νέα socket συνδεδεμένο στην ίδια τοπική θύρα και έχει επίσης το απομακρυσμένο τελικό σημείο του στη διεύθυνση και τη θύρα του πελάτη. Χρειάζεται ένα νέο socket έτσι ώστε να μπορεί να συνεχίσει να ακούει στο αρχικό socket για αιτήματα σύνδεσης ενώ φροντίζει για τις ανάγκες του συνδεδεμένου πελάτη.



Σχήμα 2.11: Σύνδεση με tcp socket client-server

Από την πλευρά του πελάτη, εάν γίνει αποδεκτή η σύνδεση, δημιουργείται επιτυχώς ένα socket και ο πελάτης μπορεί να χρησιμοποιήσει το socket για να επικοινωνήσει με τον διακομιστή.

Ο πελάτης και ο διακομιστής μπορούν πλέον να επικοινωνούν γράφοντας ή διαβάζοντας στο buffer του socket.

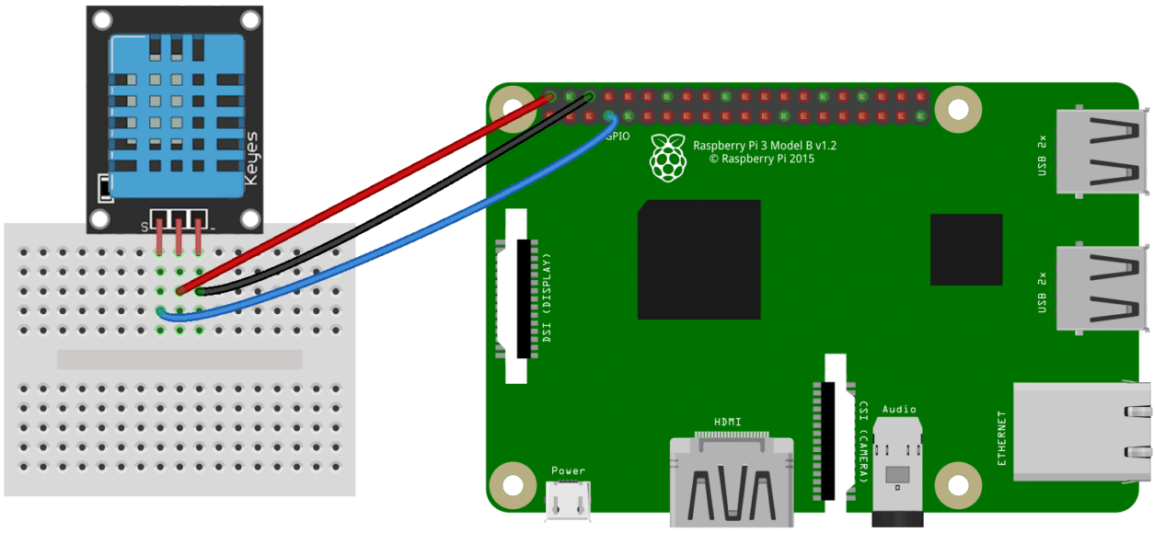
Ένα τελικό σημείο είναι ένας συνδυασμός μιας διεύθυνσης IP και ενός αριθμού θύρας. Κάθε σύνδεση TCP μπορεί να αναγνωριστεί μοναδικά από τα δύο τελικά σημεία της. Με αυτόν τον τρόπο είναι εφικτές πολλές συνδέσεις μεταξύ του κεντρικού υπολογιστή και του διακομιστή.

Κεφάλαιο 3ο: Ανάλυση και αποτελέσματα

3.1 Επικοινωνία αισθητήρων με το Raspberry

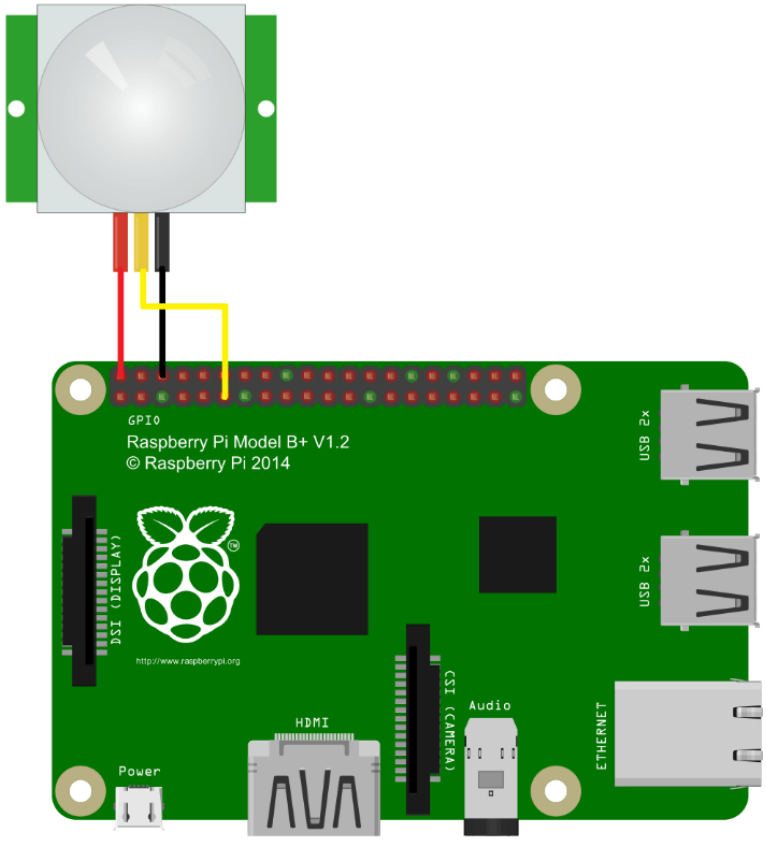
Οι αισθητήρες που χρησιμοποιήθηκαν είναι ο αισθητήρας θερμοκρασίας-υγρασίας dht11 και ο αισθητήρας ανίχνευσης κίνησης HC-SR501.

Στο Σχήμα 3.1 φαίνεται η σύνδεση του αισθητήρα dht11 με το Raspberry. Χρησιμοποιήθηκε η GPIO4, όπως φαίνεται στο Σχήμα 3.3 για την λήψη των ψηφιακών τιμών μέσω της έτοιμης βιβλιοθήκης Adafruit.

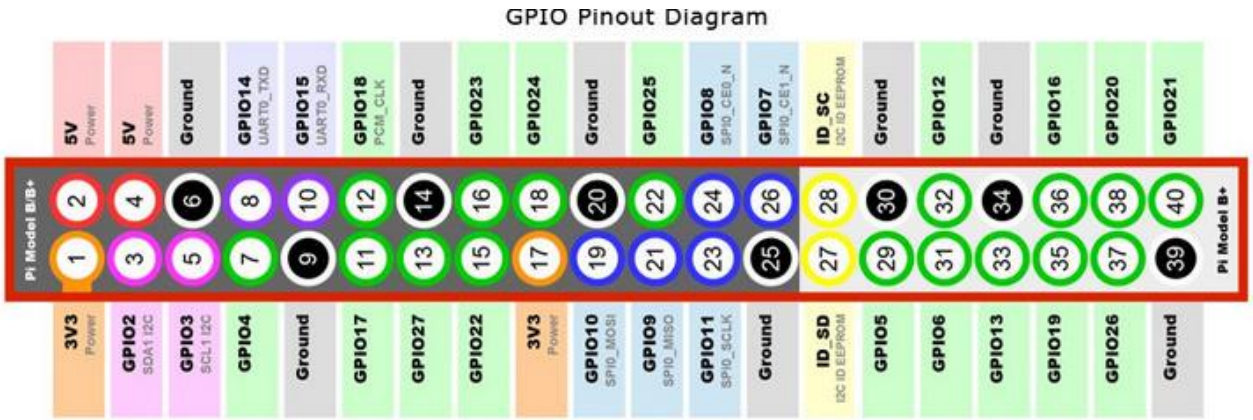


Σχήμα 3.1: Διάταξη Raspberry με τον αισθητήρα dht11

Στο Σχήμα 3.2 φαίνεται η σύνδεση του αισθητήρα HC-SR501 με το Raspberry. Χρησιμοποιήθηκε η GPIO17, όπως φαίνεται στο Σχήμα 3.3, για την λήψη της ψηφιακής εξόδου του αισθητήρα.

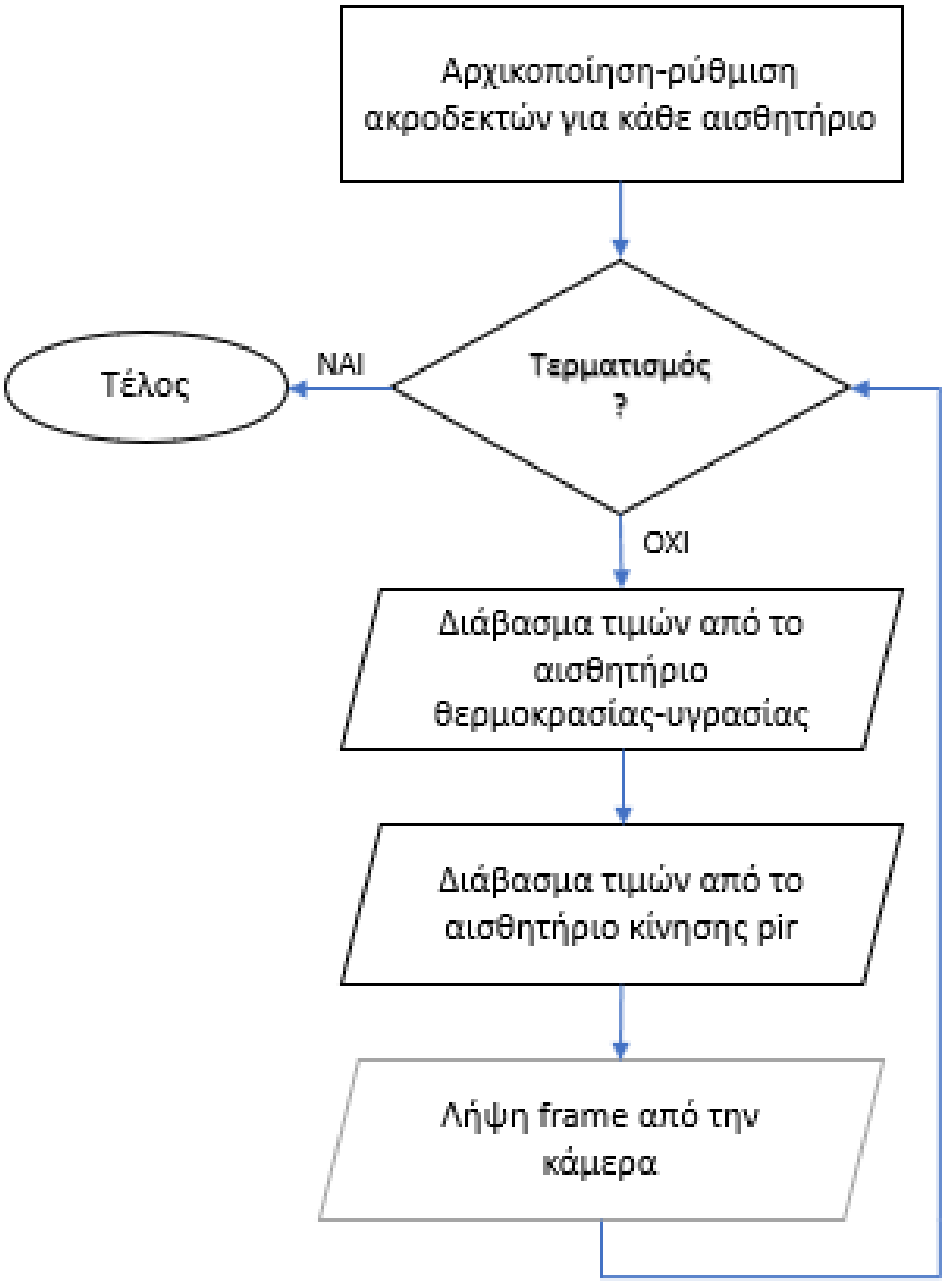


Σχήμα 3.2: Διάταξη Raspberry με τον αισθητήρα ανίχνευσης κίνησης pir



Σχήμα 3.3: Ακροδέκτες για το Raspberry Pi 3

Στη συνέχεια θα αναλυθεί ο κώδικας που χρησιμοποιήθηκε για την υλοποίηση του έργου με αναφορές μόνο στα σημαντικά σημεία του ώστε να βοηθήσουν τον αναγνώστη να κατανοήσει τις επιμέρους λειτουργίες του και για να μπορεί να χρησιμοποιήσει κομμάτια του όπου αυτός χρειάζεται.



Σχήμα 3.4: Διάγραμμα ανάγνωσης τιμών από τους αισθητήρες

Για την χρησιμοποίηση των ακροδεκτών του Raspberry αρχικά θα πρέπει να εισάγουμε δύο σημαντικές βιβλιοθήκες:

```
import RPi.GPIO as GPIO
import Adafruit_DHT
```

Ορίζουμε τους ακροδέκτες για το κάθε αισθητήριο

```
DHT_SENSOR = Adafruit_DHT.DHT11
```

```
DHT_PIN = 4 #DHT11
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BOARD)
GPIO.setup(11, GPIO.IN)        # PIR motion sensor
```

Διαβάζουμε την έξοδο του αισθητήρα dht11 μέσω της συνάρτησης Adafruit

```
humidity, temperature = Adafruit_DHT.read(DHT_SENSOR, DHT_PIN)
```

Σε περίπτωση που ο αισθητήρας παρέχει στην έξοδο του ψηφιακές τιμές μπορούμε να τις χρησιμοποιήσουμε για απεικόνιση ή επεξεργασία ή αποστολή των τιμών στον server.

```
  if humidity is not None and temperature is not None:
    print("Temp={0:0.1f}C Humidity={1:0.1f}%".format(temperature, humidity))
  else:
    print("Sensor failure. Check wiring.");
```

Διαβάζουμε την ψηφιακή έξοδο του PIR αισθητήρα

```
  i=GPIO.input(11)
  print("k= ",k)
  if i==0:              #When output from motion sensor is LOW
    print ("No intruders",i)
  elif i==1:             #When output from motion sensor is HIGH
    print ("Intruder detected",i)
```

3.2 Λήψη βίντεο και φωτογραφιών από την κάμερα με το Raspberry

Για την λήψη βίντεο με κάμερα με το Raspberry αρχικά θα πρέπει να εισάγουμε δύο σημαντικές βιβλιοθήκες:

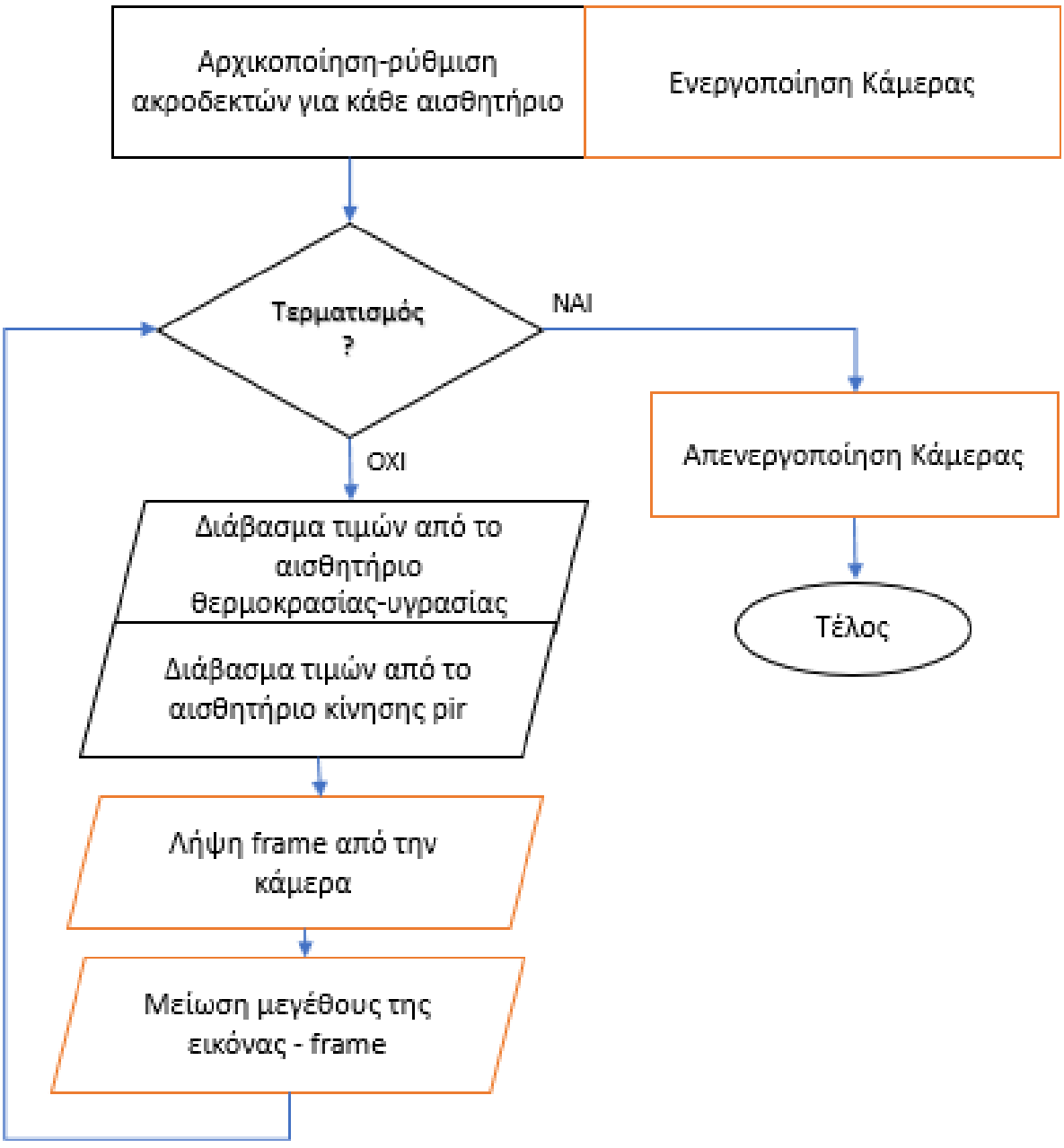
```
import cv2
import numpy as np
```

Χρησιμοποιείται η εντολή VideoCapture για την ενεργοποίηση της κάμερας

```
cap = cv2.VideoCapture(0)
```

Χρησιμοποιείται ειδική συνάρτηση για να μειώσει το μέγεθος της εικόνας

```
def rescale_frame(frame, percent=40):
    width = int(frame.shape[1] * percent/ 100)
    height = int(frame.shape[0] * percent/ 100)
    dim = (width, height)
    return cv2.resize(frame, dim, interpolation =cv2.INTER_AREA)
```



Σχήμα 3.5: Διάγραμμα για τη διαδικασία λήψης τιμών από τους αισθητήρες και το κάθε frame από την κάμερα

Η κάμερα ανοίγει με την εντολή cap.isOpened():

```
while cap.isOpened():
```

Με την παρακάτω εντολή λαμβάνεται frame by frame

```
_, frame = cap.read()
```

Και αμέσως μετά εφαρμόζεται στο frame-εικόνα μείωση ανάλυσης-μεγέθους

```
    frame = rescale_frame(frame, percent=40)
```

Αν χρειαστεί με την παρακάτω εντολή απεικονίζεται το frame στην οθόνη

```
cv2.imshow('frame',frame)
```

Είναι πολύ σημαντικό στο τέλος της λήψης όλων των frames να ελευθερώνεται η δέσμευση της κάμερας.

```
cap.release()
```

3.3 Επικοινωνία client και server με αμφίδρομη επικοινωνία μέσω sockets

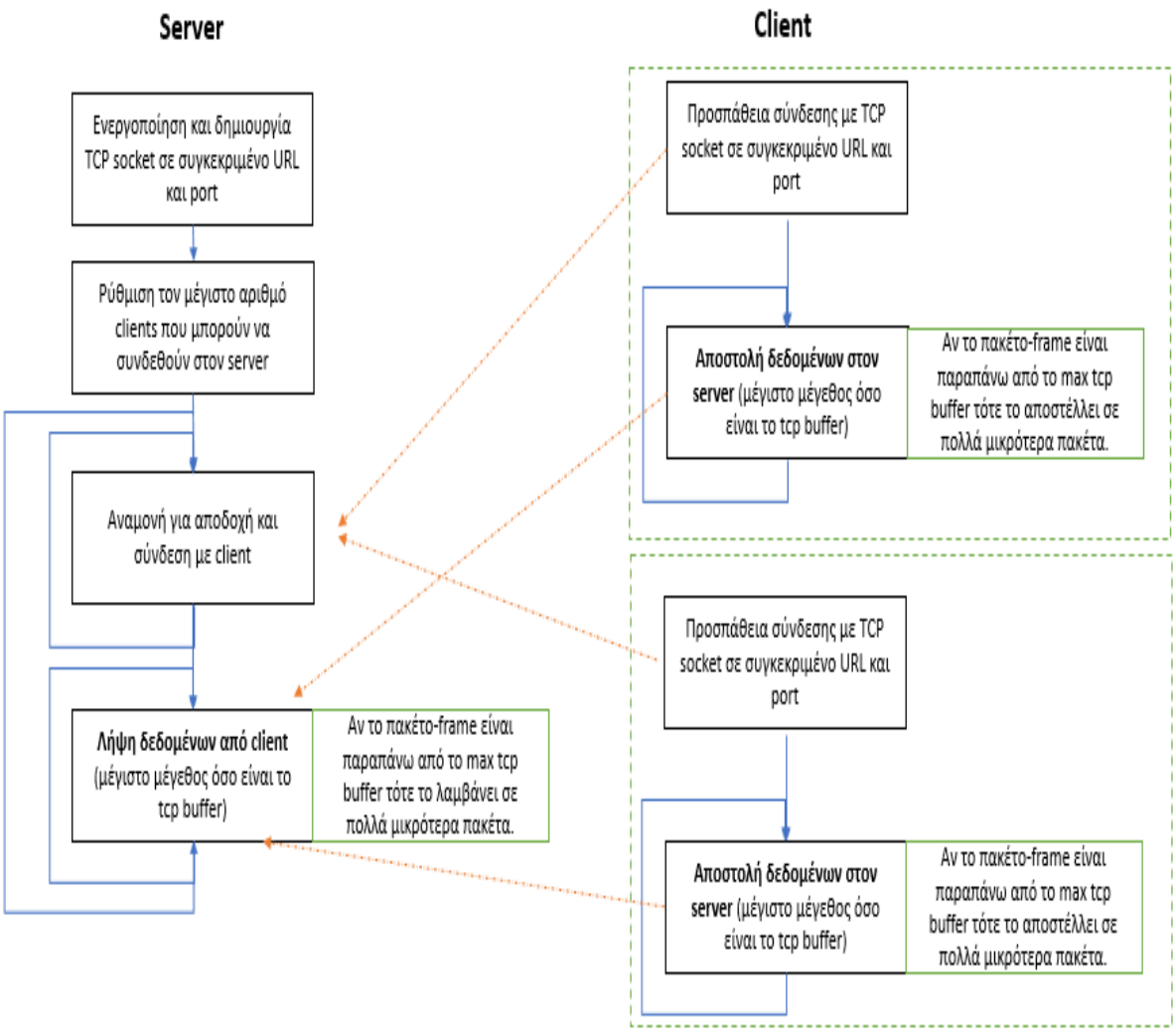
Client

Με τις παρακάτω εντολές συνδέεται ένας χρήστης-client με τον server σε συγκεκριμένη διεύθυνση και συγκεκριμένο port, με ένα socket που παραμένει ανοιχτό όσο διαρκεί επικοινωνία.

```
---------------------------------------client
client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
client_socket.connect(('localhost', 6001))
```

και αποστέλλονται τα δεδομένα των αισθητήρων

```
data = b"temperature:" + str(temperature).encode('UTF-8')
data = b"humidity:" + str(humidity).encode('UTF-8')
data = b"intruder:" + str(intruder).encode('UTF-8')
client_socket.sendall(struct.pack("I", len(data)) + data)
```



Σχήμα 3.6: Διάγραμμα της διαδικασίας της επικοινωνίας client και server μέσω sockets

Με τις παρακάτω εντολές ο server ενεργοποιεί τα sockets server σε συγκεκριμένη διεύθυνση και συγκεκριμένο port και αναμένει νέες συνδέσεις από τους χρήστες (μέχρι 10).

Server

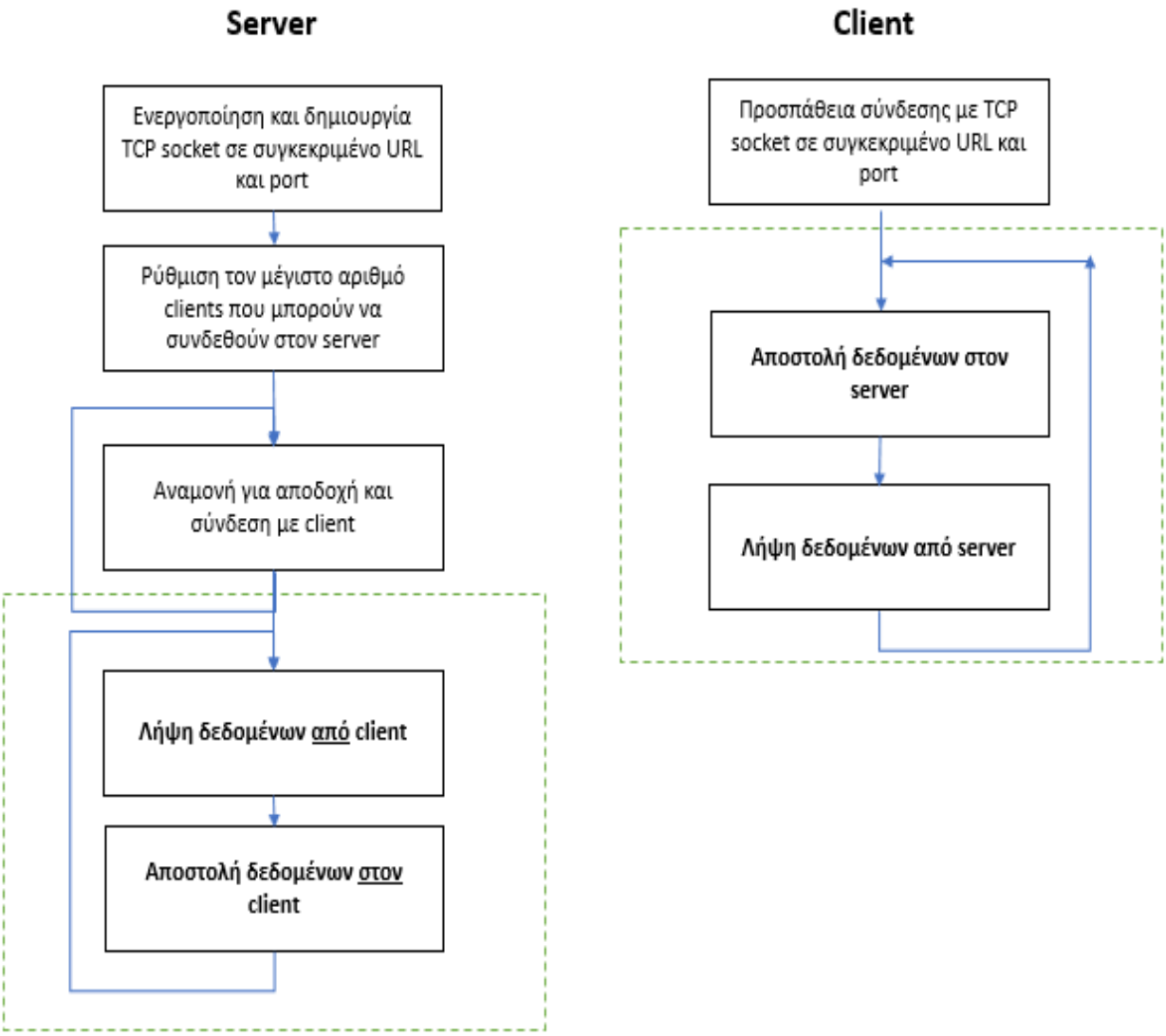
```
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.bind((self.hostname, self.port))
payload_size = struct.calcsize("I")
s.listen(10)
conn, client_address = s.accept()
```

Σε μια ατέρμων επανάληψη ο server συνεχώς λαμβάνει τα πακέτα δεδομένων που του στέλνει ένας client. Το μέγεθος του κάθε πακέτου προσδιορίζεται από τον μέγεθος του buffer του λειτουργικού ή του συστήματος που είναι εγκατεστημένος ο server.

```
while True:
        data, addr = conn.recvfrom(payload_size)
        msg_size = struct.unpack("I", data)[0]
        if msg_size > 0 and msg_size < 50:
            print("Einai data msg_size= ",msg_size)
            mydata, address = conn.recvfrom(msg_size)


            mydata = mydata.decode('utf-8')


            if (mydata.find('temperature:') != -1):
                self.temperature = mydata[12:];
                print("Einai self.temperature= ",self.temperature)
            if (mydata.find('humidity:') != -1):
                self.humidity = mydata[8:];
            if (mydata.find('intruder:') != -1):
                self.intruder = mydata[8:];
            continue;
```



Σχήμα 3.7 Διάγραμμα της διαδικασίας της επικοινωνίας client και server με αμφίδρομη επικοινωνία μέσω sockets

3.4 Αποστολή βίντεο-frames στον server

Χρειάζονται οι παρακάτω βιβλιοθήκες για να γίνει λήψη του βίντεο από μια web κάμερα στο Raspberry.

```
import cv2
import numpy as np
```

Ανοίγει η κάμερα

```
cap = cv2.VideoCapture(0)
```

και με τις παρακάτω εντολές αλλάζει το μέγεθος του κάθε frame-εικόνας για την μείωση των δεδομένων που στέλνει ο client για να αυξηθεί η ταχύτητα προβολής βίντεο, όπως αναλύεται στο τρίτο κεφάλαιο.

```
def rescale_frame(frame, percent=40):
    width = int(frame.shape[1] * percent/ 100)
    height = int(frame.shape[0] * percent/ 100)
    dim = (width, height)
    return cv2.resize(frame, dim, interpolation =cv2.INTER_AREA)
```

Αφού το σύστημα διαβάσει το frame (και κάθε frame μέσω επανάληψης) από την κάμερα το αποστέλλει μέσω της εντολής sendall στον server.

```
while cap.isOpened():
    _, frame = cap.read()
    frame = rescale_frame(frame, percent=40)
    memfile = BytesIO()
    np.save(memfile, frame)
    memfile.seek(0)
    data = memfile.read()
    client_socket.sendall(struct.pack("I", len(data)) + data)
```

Στο τέλος πρέπει να ελευθερωθεί η δέσμευση της κάμερας

```
cap.release()
```

3.5 Κατασκευή server-side στον server

Πρέπει πρώτα να πραγματοποιηθεί η λήψη του μεγέθους του πακέτου

```
        data, addr = conn.recvfrom(payload_size)
```

και στην συνέχεια να γίνει λήψη του πακέτου σε μικρά πακετάκια με μέγεθος όσο επιτρέπει το μέγεθος του buffer

```
        if data:
                msg_size = struct.unpack("I", data)[0]
                data = b"
```

```
while len(data) < msg_size:


    missing_data, address = conn.recvfrom(msg_size - len(data))
    if missing_data:
        data += missing_data
    else:
        break
```

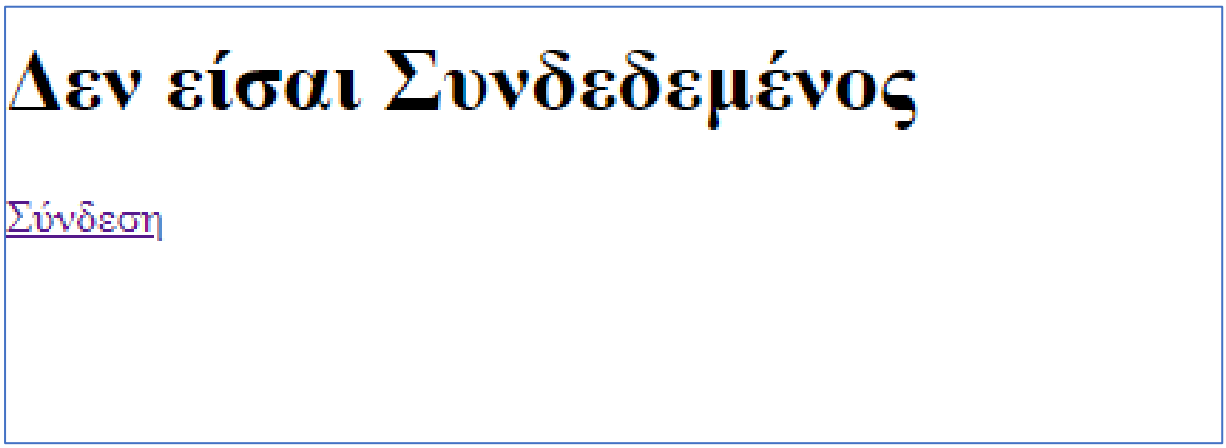
Το frame μετατρέπεται σε jpeg εικόνα για να μπορεί να προβληθεί το βίντεο εικόνα-εικόνα

```
memfile = BytesIO()
memfile.write(data)
memfile.seek(0)
frame = numpy.load(memfile)


ret, jpeg = cv2.imencode('.jpg', frame)
self.jpeg = jpeg
```

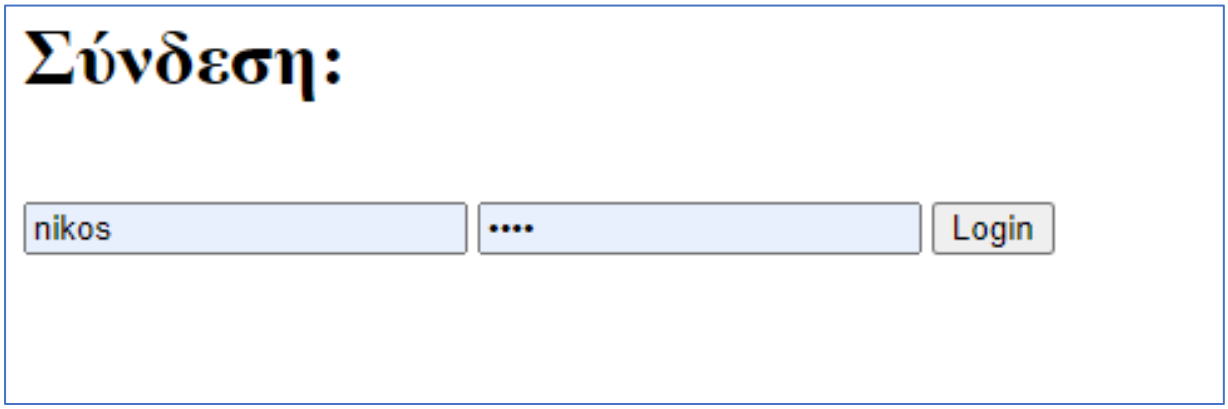
3.6 Περιγραφή Ιστοσελίδας για την Διεπαφή Χρήστη-Συστήματος μέσω Server

Η σελίδα του Σχήματος 3.8 εμφανίζεται όταν κάποιος χρήστης προσπαθεί να εισαχθεί στον ιστοχώρο πρόσβασης για τα συστήματα παρακολούθησης.



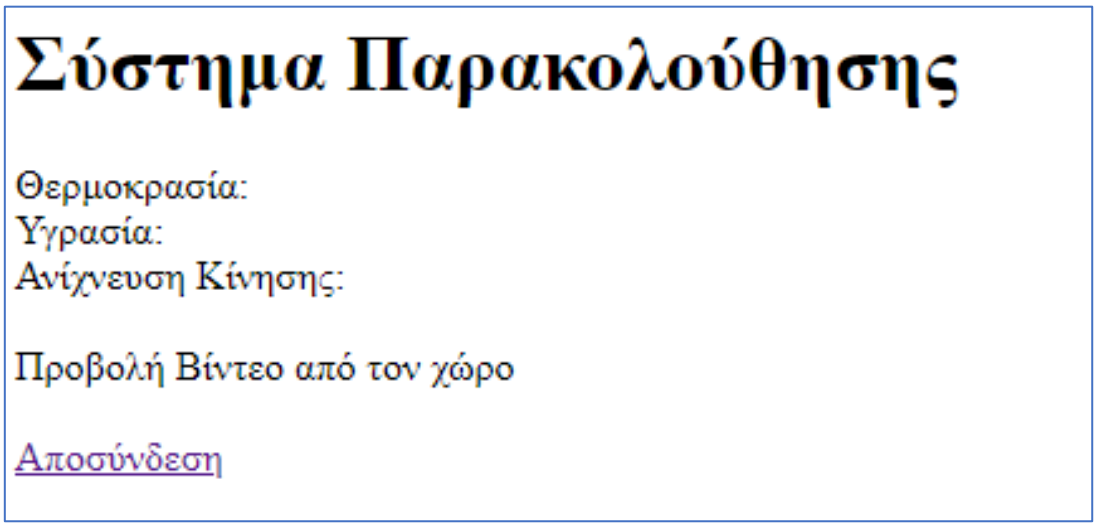
Σχήμα 3.8: Μη εξουσιοδοτημένη πρόσβαση

Η σελίδα του Σχήματος 3.9 εμφανίζεται όταν ο χρήστης θέλει να συνδεθεί και να εισάγει τα στοιχεία πρόσβασης.

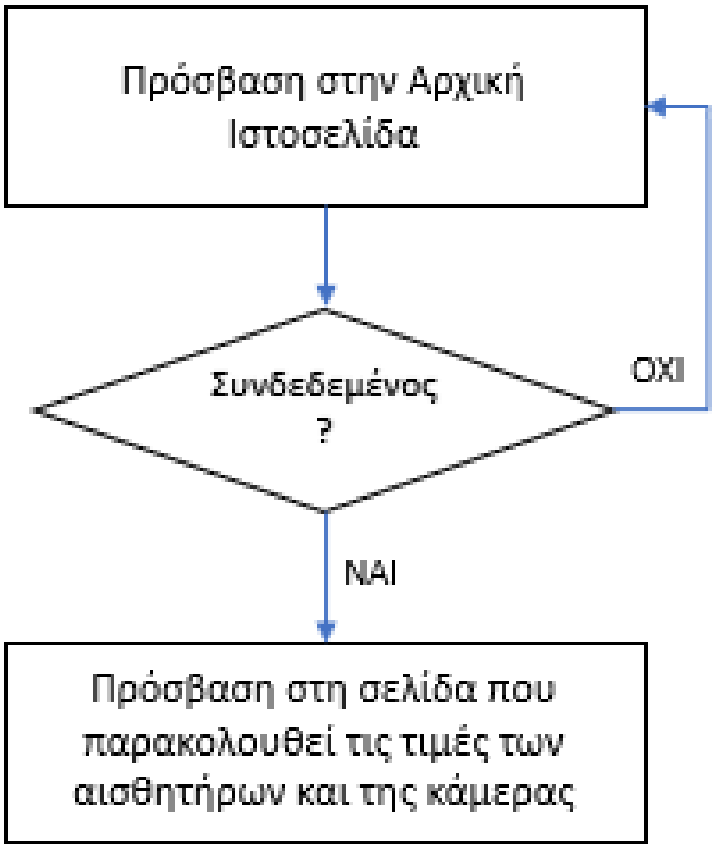


Σχήμα 3.9: Εισαγωγή στοιχείων πρόσβασης

Η σελίδα του Σχήματος 3.10 εμφανίζεται όταν ο χρήστης έχει συνδεθεί αλλά το σύστημα παρακολούθησης δεν είναι συνδεδεμένο με το server.

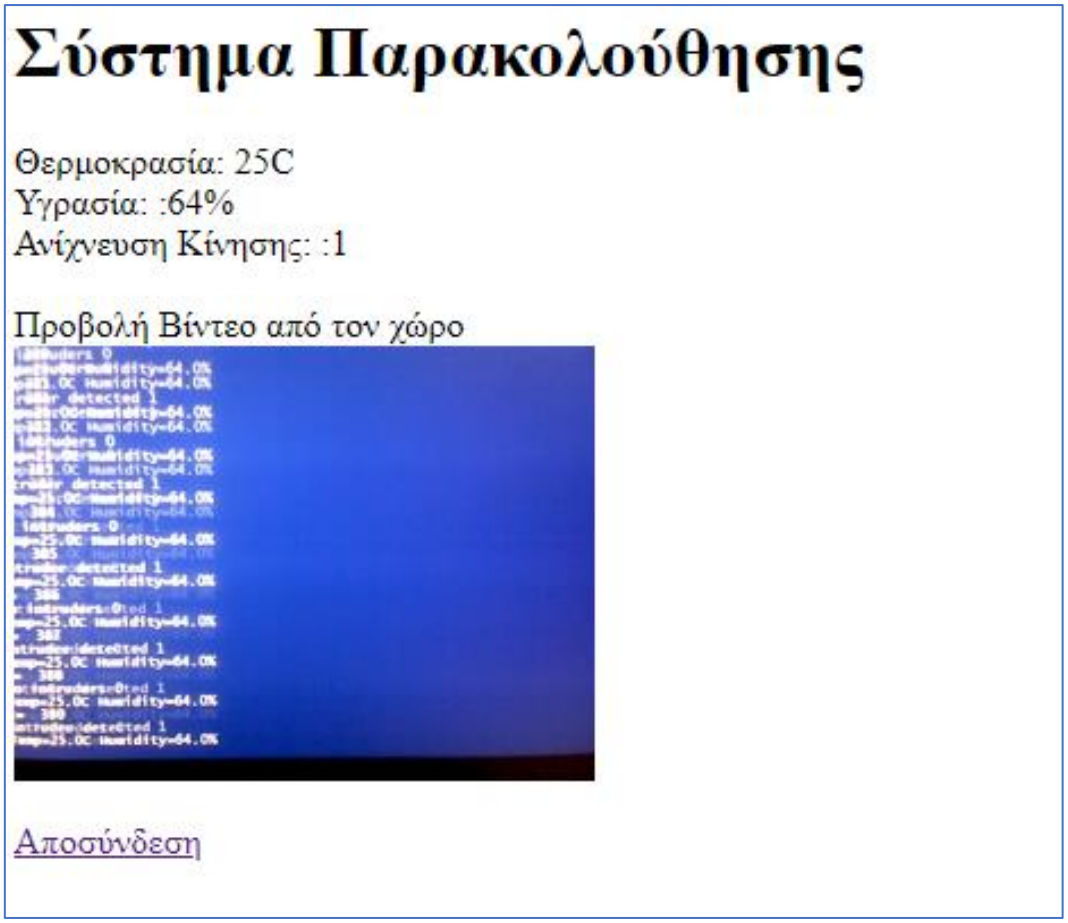


Σχήμα 3.10: Το σύστημα παρακολούθησης δεν είναι συνδεδεμένο με το server



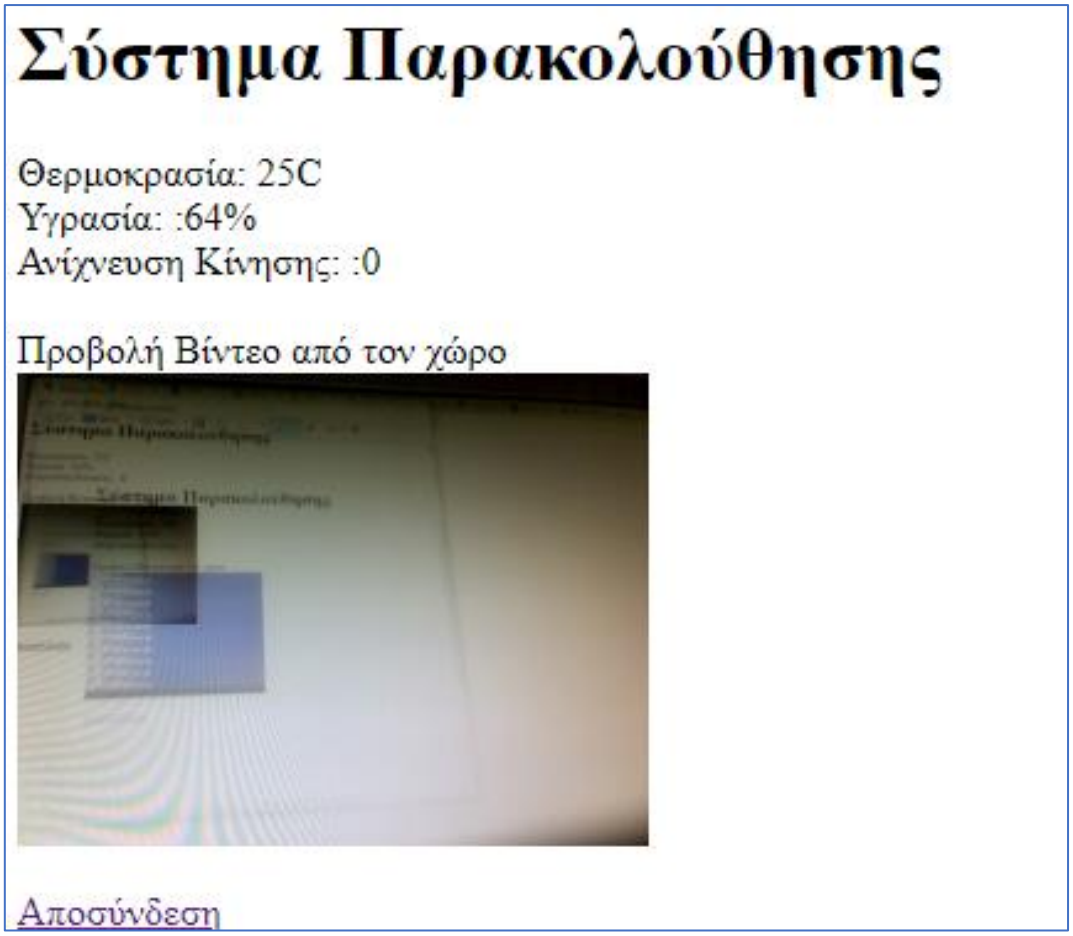
Σχήμα 3.11: Διάγραμμα για την περιγραφή πλοήγησης του χρήστη στον ιστοχώρο του συστήματος

Οι σελίδες των Σχημάτων 3.12 και 3.13 εμφανίζονται όταν τελικά το σύστημα παρακολούθησης που ανήκει στον χρήστη επικοινωνεί με το server.



Σχήμα 3.12: Προβολή τιμών των αισθητήρων και βίντεο από τον χώρο - 1

Απεικονίζονται οι τιμές της θερμοκρασίας, υγρασίας και ανίχνευσης κίνησης. Επίσης, προβάλλεται το βίντεο από τον χώρο σε ειδικό παράθυρο.



Σχήμα 3.13: Προβολή τιμών των αισθητήρων και βίντεο από τον χώρο - 2

3.7 Ανάλυση απόδοσης της αποστολής δεδομένων

Κάθε πακέτο δεδομένων, από ένα απλό γράμμα – byte μέχρι ένα frame ή εικόνα ενός βίντεο θα πρέπει να σταλθεί μέσω της εντολής sendall. Θα πρέπει να κατασκευαστεί ένα «κουστούμι» με την εντολή struct.pack για κάθε πακέτο, θα πρέπει να περιέχεται σε κάθε αρχή του πακέτου το μέγεθος των δεδομένων του πακέτου.

client_socket.sendall(struct.pack("I", len(data)) + data)

Για την λήψη χρησιμοποιείται η recvfrom

όπου αρχικά γίνεται ανάγνωση του μεγέθους του πακέτου

data, addr = conn.recvfrom(payload_size)

με τη βοήθεια της εντολή unpack

msg_size = struct.unpack("I", data)[0]

Στη συνέχεια πραγματοποιείται διάβασμα με recvfrom με την εντολή μέσα σε ένα Βρόγχο.

```
while len(data) < msg_size:
missing_data, address = conn.recvfrom(msg_size - len(data))
if missing_data:
        data += missing_data
else:
break;
```

Όλα τα παραπάνω περιγράφηκαν για να εξηγηθεί το εξής:

Σε κάθε συσκευή-υπολογιστής υπάρχει ένα μέγιστο όριο για τον TCP Buffer, δηλαδή για το πόσο μεγάλο μέγεθος γίνεται να αποστείλει ή να λάβει μέσω TCP επικοινωνίας.

Συνήθως αυτό το μέγεθος είναι μικρό και ορίζεται πολλές φορές σε 8KB.

Αυτό σημαίνει ότι για μια εικόνα-frame 20KB μπορεί να σταλθούν τρία διαδοχικά πακέτα με το τελευταίο να είναι 4KB.

Έστω μια εικόνα ενός βίντεο με ανάλυση εικόνας 720p, δηλαδή 1280x720 pixels δηλαδή

για μονόχρωμη εικόνα 921.600 bytes, περίπου 900KB. Αυτό σημαίνει ότι η εικόνα μπορεί να σταλθεί σε 112 πακέτα των 8KB περίπου. Άρα για κάθε frame έχουμε 112 καλέσματα της recvfrom. Δεν είναι τόσο σημαντική καθυστέρηση αν αναλογιστεί κανείς ότι κάθε κάλεσμα είναι της τάξης των 10μs.

Αν απαιτούμε το frame rate στη λήψη να είναι 30fps τότε θα πρέπει να λαμβάνουμε το κάθε frame κάτω από 1/30 = 33ms.

Σε τοπικό επίπεδο, με τον server να βρίσκεται σε localhost ικανοποιούνται οι προδιαγραφές και το βίντεο από την κάμερα εμφανίζεται κανονικά.

Όταν η αποστολή γίνεται σε server τότε θα πρέπει να ληφθεί υπόψη ο πιο σημαντικό παράγοντας, η ταχύτητα download/upload της γραμμής.

Ας υποθέσουμε ότι έχουμε ταχύτητα upload 8Mbps=1MB/s.

Μέσα σε 1 δευτερόλεπτο μπορεί να αποσταλθεί μια εικόνα ή ένα πακέτο δεδομένων 900KB. Αλλά, ο τελικός στόχος είναι να σταλθεί ένα πακέτο 900KB κάθε <33ms.

Για αυτό τον λόγο θα πρέπει να μειωθεί το μέγεθος του frame-εικόνας και το frame rate της απεικόνισης του βίντεο στην τελική συσκευή – ιστοσελίδα.

Αν σταλθεί ένα frame 150KB (320x480), με 1MB/s upload, δηλαδή 6 frames/s, θα διαρκέσει 166ms περίπου. Άρα το fps=6.

3.8 Ανάλυση ασφάλειας στο δίκτυο και στα δεδομένα

Ξεχωρίζουμε τριών ειδών ασφάλεια στην επικοινωνία.

Ασφάλεια με κωδικούς πρόσβασης στην ιστοσελίδα.

Ο χρήστης μπορεί να είναι εγγεγραμμένος στην ιστοσελίδα αλλά να μην έχει πρόσβαση ή να μην του ανήκει καμία συσκευή-σύστημα παρακολούθησης.

Ασφάλεια με κωδικούς πρόσβασης στο σύστημα παρακολούθησης

Κάθε εγγεγραμμένος χρήστης θα πρέπει να έχει δηλώσει στη βάση ποια συσκευή του ανήκει ώστε να συνδεθεί με αυτή. Στη βάση πρέπει να αποθηκεύονται επιπλέον πληροφορίες όπως οι ip των χρηστών-συστημάτων.

Ασφάλεια αποστολής δεδομένων

Σε κάθε πακέτο TCP που αποστέλλεται (και αντίθετα) στο server πρέπει να περιέχει και άλλες πρόσθετες πληροφορίες για να γίνει ασφαλής και μοναδική η αποστολή. Μερικές από αυτές είναι το username του χρήστη, την ip και ίσως την MAC διεύθυνση της συσκευής.

Κεφάλαιο 4ο: Συμπεράσματα και προτάσεις βελτίωσης

Παρουσιάστηκε και αναλύθηκε ένα σύστημα παρακολούθησης χώρου που υλοποιήθηκε με raspberry και συνδεδεμένα αισθητήρια και επικοινωνία του συστήματος με τον χρήστη μέσω εξωτερικού server και χρήση των tcp sockets. Με τα tcp sockets έχει επιτευχθεί ο κύριος στόχος της εργασίας, να υπάρχει συνεχή και αμφίδρομη επικοινωνία όταν οι χρήστες είναι πίσω από δρομολογητές ή τείχη προστασίας και δεν έχουν public ip ή δεν είναι εφικτό το port forwarding.

Παρουσιάσθηκε κώδικας για λήψη και επεξεργασία βίντεο στο Raspberry με μια απλή web camera και κώδικας για λήψη και επεξεργασία δεδομένων από αισθητήρες που είναι συνδεδεμένοι στο Raspberry. Περιγράφηκε η αμφίδρομη και συνεχή επικοινωνία του συστήματος παρακολούθησης με τους χρήστες με tcp sockets και η αποστολή βίντεο με tcp sockets σε εξυπηρετητή και προβολή σε ιστοσελίδα. Με τη μέθοδο hole punching υλοποιήθηκε επικοινωνία μεταξύ συστήματος και χρήστη όταν βρίσκονται πίσω από ένα δρομολογητή ή firewall.

Το σύστημα που υλοποιήθηκε είναι χαμηλής κατανάλωσης (<4W) και μικρού κόστους (<15 ευρώ).

Η ταχύτητα προβολής του βίντεο του χώρου είναι αργή και αυτό οφείλεται στην μικρή ταχύτητα αποστολής-upload στην γραμμή Internet, όπως αναφέρθηκε στο προηγούμενο κεφάλαιο. Για να αντιμετωπιστεί αυτό πρέπει να μειωθεί η ποιότητα του βίντεο, δηλαδή το μέγεθος της εικόνας που αποστέλλεται. Κατά συνέπεια θα υπάρχει και χαμηλό frame per second.

Η πιο σημαντική πρόταση βελτίωσης αφορά την ασφάλεια του συστήματος. Αρχικά, πρέπει να κατασκευαστεί βάσης δεδομένων ώστε κάθε εγγεγραμμένος χρήστης να πρέπει να έχει δηλώσει στη βάση ποια συσκευή του ανήκει ώστε να συνδεθεί με αυτή. Στη βάση πρέπει να αποθηκεύονται επιπλέον πληροφορίες όπως οι ip των χρηστών-συστημάτων. Επίσης, σε κάθε πακέτο TCP επικοινωνίας με το server πρέπει να περιέχονται και άλλες πρόσθετες πληροφορίες για να γίνει ασφαλής και μοναδική. Μερικές από αυτές είναι το username του χρήστη, την ip και ίσως την MAC διεύθυνση της συσκευής.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] https://www.raspberrypi.org/

[2] https://docs.oracle.com/javase/tutorial/networking/sockets/definition.html

[3] https://en.wikipedia.org/wiki/Network_address_translation

[4] https://www.raspberrypi.org/documentation/hardware/camera/

[5] https://projects.raspberrypi.org/en/projects/getting-started-with-picamera

[6] https://www.raspberrypi.org/documentation/usage/webcams/

[7] https://pimylifeup.com/raspberry-pi-webcam-server/

[8] https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_gui/py_video_display/py_video_display.html

[9] https://www.geeks3d.com/hacklab/20200308/python-3-and-opencv-part-3-how-to-read-the-webcam-with-opencv-cv2/

[10] https://www.e-consystems.com/blog/camera/how-to-access-cameras-using-opencv-with-python/

[11] https://medium.com/datadriveninvestor/video-streaming-using-flask-and-opencv-c464bf8473d6

[12] https://blog.miguelgrinberg.com/post/video-streaming-with-flask

[13] https://www.pyimagesearch.com/2019/09/02/opencv-stream-video-to-web-browser-html-page/

[14] https://towardsdatascience.com/python-webserver-with-flask-and-raspberry-pi-398423cc6f5d

[15] https://www.pyimagesearch.com/2019/04/15/live-video-streaming-over-network-with-opencv-and-imagezmq/

[16] https://hackernoon.com/how-to-access-your-raspberry-pi-camera-from-anywhere-544ab9e5bacc

[17] https://docs.python.org/3/howto/sockets.html

[18] https://pymotw.com/2/socket/tcp.html

[19] https://gist.github.com/tehmas/4a1b600e7bc95ee3d48b6fae85e5fad9

[20] https://www.pubnub.com/blog/socket-programming-in-python-client-server-p2p/

[21] https://www.nabto.com/

[22] https://www.nabto.com/developer/

[23] https://venturebeat.com/2010/12/08/nabto-internet-of-things-funding/

[24] https://en.wikipedia.org/wiki/NAT_traversal

[25] http://reports-archive.adm.cs.cmu.edu/anon/isri2005/CMU-ISRI-05-104.pdf

[26] https://en.wikipedia.org/wiki/TCP_hole_punching

[27] https://en.wikipedia.org/wiki/Hole_punching_(networking)

[28] https://en.wikipedia.org/wiki/Real_Time_Streaming_Protocol#:~:text=The%20Real%20Time%20Streaming%20Protocol,media%20sessions%20between%20end%20points.&text=The%20transmission%20of%20streaming%20data%20itself%20is%20not%20a%20task%20of%20RTSP.

[29] https://en.wikipedia.org/wiki/Real-Time_Messaging_Protocol#:~:text=Real%2DTime%20Messaging%20Protocol%20(RTMP,the%20protocol%20for%20public%20use.

[30] https://www.scaleway.com/en/docs/setup-rtmp-streaming-server/

[31] https://en.wikipedia.org/wiki/Voice_over_IP

[32] https://learn.adafruit.com/dht

[33] https://learn.adafruit.com/pir-passive-infrared-proximity-motion-sensor/how-pirs-work

[34] https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/

[35] https://www.raspberrypi.org/products/raspberry-pi-zero-w/

[36] https://www.raspberrypi.org/downloads/

[37] https://osoyoo.com/2017/07/27/arduino-lesson-pir-motion-sensor/

[38] http://famillemoreau.hopto.org/Arduino/HC-SR501%20Motion%20sensor%20Arduino%20IR%20Body%20Infrared%20Sensor%20Module.htm

[39] https://www.python.org/

[40] https://towardsdatascience.com/best-python-libraries-for-machine-learning-and-deep-learning-b0bd40c7e8c

[41] https://opencv.org/

[42] https://en.wikipedia.org/wiki/Machine_vision

[43] https://opencv.org/intel-acquires-itseez/

[45] https://docs.opencv.org/4.2.0/dd/d43/tutorial_py_video_display.html

[46] https://www.cloudflare.com/learning/network-layer/internet-protocol/#:~:text=The%20Internet%20Protocol%20(IP)%20is%20a%20protocol%2C%20or%20set,into%20smaller%20pieces%2C%20called%20packets.

[47] https://www.cloudflare.com/learning/ddos/glossary/user-datagram-protocol-udp/

[48] https://www.iana.org/

[49] https://mqtt.org/

[50] https://www.ev3dev.org/docs/tutorials/sending-and-receiving-messages-with-mqtt/

ΠΑΡΑΡΤΗΜΑ Α : Κώδικας

A1. server.py

```
# -*- coding: utf-8 -*-
from flask import Flask, session, escape, render_template, Response, redirect, url_for, request
#from flask.ext.session import Session
from streamer import Streamer
import sys
import time
from flask_socketio import SocketIO
from flask_socketio import emit
app = Flask(__name__)
SESSION_TYPE = 'filesystem'
app.config['SECRET_KEY'] = 'randomSecretKey@123'
app.secret_key = 'any random string'
#socket = SocketIO(app, cors_allowed_origins="*")
socket = SocketIO(app)
streamer = Streamer('localhost', 6001)
def gen():
  streamer.start()
  while True:
    if streamer.streaming:
      yield(b'--frame\r\n'b'Content-Type: image/jpeg\r\n\r\n' + streamer.get_jpeg() + b'\r\n\r\n');
  @app.route('/')
def index():
  if 'logged' in session:
    print("logged_in= "+str(session['logged']))
  else:
    session['logged'] = 0;
  print("logged index= "+str(session['logged']))
  return render_template('index.html',logged_in=session['logged'])
@app.route('/video_feed')
```

```
def video_feed():
  return Response(gen(), mimetype='multipart/x-mixed-replace; boundary=frame')
# Route for handling the login page logic
@app.route('/login', methods=['GET', 'POST'])
def login():
    print("login")
    error = None
    if request.method == 'POST':
        if request.form['username'] != 'nikos' or request.form['password'] != '1234':
            error = 'Λάθος στοιχεία πρόσβασης'
        else:
            session['logged'] = 1;
            return redirect(url_for('index'))
    return render_template('login.html', error=error)
@app.route('/logout')
def logout():
   session['logged'] = 0;
   session.pop('logged', None)
   return redirect(url_for('index'))
@socket.on('start_all')
def start_all():
    while True:
      if 'logged' in session:
        if session['logged'] == 1:
          time.sleep(1)
          print("logged start_all= "+str(session['logged']))
          print("Read from client and send data")
          emit('data_temp', streamer.get_temperature())
          emit('data_hum', streamer.get_humidity())
          emit('data_intr', streamer.get_intruder())
          time.sleep(1)
if __name__ == '__main__':
    app.run(host='localhost',port=5000, threaded=True, debug = True)
```

```
    socket.run(app)
```

A2. Streamer.py

```
import cv2
import numpy
import socket
import struct
import threading
from io import BytesIO
import time
from PIL import Image
class Streamer(threading.Thread):
    def __init__(self, hostname, port):
        threading.Thread.__init__(self)
        self.hostname = hostname
        self.port = port
        self.running = False
        self.streaming = False
        self.jpeg = None
        self.temperature = None
        self.humidity = None
        self.intruder = None

    def run(self):
        s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        print('Socket created')
        s.bind((self.hostname, self.port))
        print('Socket bind complete')
        payload_size = struct.calcsize("I")
        s.listen(10)
        print('Socket now listening')
```

```
self.running = True
while self.running:
    print('Start listening for connections...')
    conn, client_address = s.accept()
    print("New connection accepted.")
    k=0;
    measure = False;
    while True:
        data, addr = conn.recvfrom(payload_size)
        print("payload_size= ",payload_size)
        msg_size = struct.unpack("I", data)[0]
        if msg_size > 0 and msg_size < 50:
            print("Einai data msg_size= ",msg_size)
            mydata, address = conn.recvfrom(msg_size)
            mydata = mydata.decode('utf-8')
            if (mydata.find('temperature:') != -1):
                self.temperature = mydata[12:];
                print("Einai self.temperature= ",self.temperature)
            if (mydata.find('humidity:') != -1):
                self.humidity = mydata[8:];
            if (mydata.find('intruder:') != -1):
                self.intruder = mydata[8:];
            continue;
        if data:
            # Read frame size
            print("----------Read frame size")
            #msg_size = struct.unpack("I", data)[0]
            if k%10==0:
                start = time.time()
                measure = True
            print("msg_size= ",msg_size)
            # Read the payload (the actual frame)
            data = b"
```

```
    while len(data) < msg_size:

        #print("len(data)= ",len(data))
        missing_data, address = conn.recvfrom(msg_size - len(data))
        #print("len(missing_data)= ",len(missing_data))
        if missing_data:
            data += missing_data
        else:
            # Connection interrupted
            self.streaming = False
            break
    # Skip building frame since streaming ended
    if self.jpeg is not None and not self.streaming:
        continue
    #print("len(data)= ",len(data))
    # Convert the byte array to a 'jpeg' format
    memfile = BytesIO()
    memfile.write(data)
    memfile.seek(0)
    frame = numpy.load(memfile)
    ret, jpeg = cv2.imencode('.jpg', frame)
    self.jpeg = jpeg
    self.streaming = True

else:
    conn.close()
    print('Closing connection...')
    self.streaming = False
    self.jpeg = None
    break
print('ending.....')
if measure == True:
    end = time.time() #0.161527109146s
```

```
                print((end - start)/10)
                measure = False
        print('Exit thread.')
    def stop(self):
        self.running = False
    def get_jpeg(self):
        #return self.jpeg.tobytes()
        return bytearray(self.jpeg)
    def get_temperature(self):
        return self.temperature;
    def get_humidity(self):
        return self.humidity;
    def get_intruder(self):
        return self.intruder;
```

A3. client_sensors.py

```
import cv2
import numpy as np
import socket
import struct
from io import BytesIO
import time
import RPi.GPIO as GPIO
import Adafruit_DHT

DHT_SENSOR = Adafruit_DHT.DHT11
DHT_PIN = 4
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BOARD)
GPIO.setup(11, GPIO.IN)         #Read output from PIR motion sensor
# Capture frame
cap = cv2.VideoCapture(0)
cap.set(3, 640)
```

```
cap.set(4, 360)
w = cap.get(3)
h = cap.get(4)
print ("WIDTH:%d" %w)
print ("HEIGHT:%d" %h)
def rescale_frame(frame, percent=40):
    width = int(frame.shape[1] * percent/ 100)
    height = int(frame.shape[0] * percent/ 100)
    dim = (width, height)
    return cv2.resize(frame, dim, interpolation =cv2.INTER_AREA)
client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
bufsize = client_socket.getsockopt(socket.SOL_SOCKET, socket.SO_SNDBUF)
print ("Buffer size SO_SNDBUF [Before]:%d" %bufsize)
bufsize = client_socket.getsockopt(socket.SOL_SOCKET, socket.SO_RCVBUF)
client_socket.connect(('localhost', 6001))
k=0
measure = False
while cap.isOpened():
    if k%100==0:
        start = time.time()
        measure = True
    _, frame = cap.read()
    frame = rescale_frame(frame, percent=40)
    memfile = BytesIO()
    np.save(memfile, frame)
    memfile.seek(0)
    data = memfile.read()
    if measure == True:
        end = time.time()
        #print((end - start)/100)
        measure = False
    client_socket.sendall(struct.pack("I", len(data)) + data)
    temperature = 25 #32+k
```

```
    humidity = 64 #30+k
    #humidity, temperature = Adafruit_DHT.read(DHT_SENSOR, DHT_PIN)
    if humidity is not None and temperature is not None:
        print("Temp={0:0.1f}C Humidity={1:0.1f}%".format(temperature, humidity))
        data = b"temperature:" + str(temperature).encode('UTF-8')
        client_socket.sendall(struct.pack("I", len(data)) + data)
        data = b"humidity:" + str(humidity).encode('UTF-8')
        client_socket.sendall(struct.pack("I", len(data)) + data)
    else:
        print("Sensor failure. Check wiring.");
    #i=GPIO.input(11)
    i = k%2
    print("k= ",k)
    intruder = 0;
    if i==0:                #When output from motion sensor is LOW
        print ("No intruders",i)
    elif i==1:              #When output from motion sensor is HIGH
        print ("Intruder detected",i)
        intruder = 1;
    data = b"intruder:" + str(intruder).encode('UTF-8')
    client_socket.sendall(struct.pack("I", len(data)) + data)
    k=k+1
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
cap.release()
```