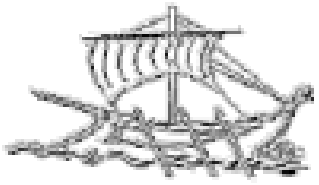


ΑΕΙ ΠΕΙΡΑΙΑ Τ.Τ.
ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΗΛΕΚΤΡΟΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ Τ.Ε.

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Συσκευή αυτόματης παροχής τροφής για κατοικίδια ζώα.

Αθανάσιος – Παναγιώτης Μαλαμάκης
Ιωάννης Μπαλαφούτας

Εισηγητής: Δρ Σταμάτης Αλατσαθιανός, Καθηγητής

ΑΘΗΝΑ
ΔΕΚΕΜΒΡΙΟΣ 2015

Αθανάσιος – Παναγιώτης Μαλαμάκης & Ιωάννης Μπαλαφούτας

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Συσκευή αυτόματης παροχής τροφής για κατοικίδια ζώα.

Αθανάσιος – Παναγιώτης Μαλαμάκης
Α.Μ. 42046
Ιωάννης Μπαλαφούτας
Α.Μ. 41760

Εισηγητής:

Δρ Σταμάτης Αλατσαθιανός, Καθηγητής

Εξεταστική Επιτροπή:

Ημερομηνία εξέτασης

ΕΥΧΑΡΙΣΤΙΕΣ

Η παρούσα πτυχιακή εργασία ολοκληρώθηκε έπειτα από αρκετές δοκιμές και συζητήσεις για την καλύτερη δυνατή επίτευξή της. Σε αυτό το σημείο, μιας και όλα κύλησαν ομαλά θα θέλαμε να ευχαριστήσουμε αρχικά τον καθηγητή μας, κ. Σταμάτη Αλατσαθιανό, για την πολύτιμη βοήθεια και στήριξη που μας παρείχε όποτε του ζητήθηκε.

Τέλος ευχαριστούμε πολύ τις οικογένειές μας για την στήριξη που μας έδωσαν στο διάστημα που αναπτύσσαμε την εργασία αυτή.

ΠΕΡΙΛΗΨΗ

Σε αυτή την εργασία αναπτύξαμε μια κατασκευή η οποία χρησιμοποιεί ένα raspberry pi 2 και διαθέτει δύο σερβοκινητήρες οι οποίοι βοηθάνε στην απελευθέρωση της τροφής. Συνδέεται επίσης με μια κάμερα (ενσύρματα) με σκοπό την λήψη και αποστολή της φωτογραφίας στο e-mail που έχουμε δώσει. Ασύρματα, μέσω tcp socket, επικοινωνεί με το κινητό μας τηλέφωνο μέσω του οποίου έχουμε την δυνατότητα να δώσουμε εντολή ταΐσματος στην ταΐστρα και να απελευθερώσει την τροφή. Η ποσότητα τροφής που απελευθερώνεται είναι ρυθμισμένη από πριν και δεν μπορεί ο χρήστης να την αυξομειώσει. Στην ταΐστρα υπάρχει ένα raspberry pi 2 που είναι κατάλληλα προγραμματισμένο για να δέχεται αιτήματα (requests) από το κινητό που υπάρχει η εφαρμογή και να κάνει τις κατάλληλες ενέργειες.

ABSTRACT

For this project we developed a construction which uses a Raspberry Pi 2 to control 2 servo motors that help the system release the food. It also has a wired camera module in order to take a snapshot and email it to the email address which we have in the configurations. Wirelessly, via a TCP Socket the system communicates with our mobile phone from which we have the power to send a command to the system in order to free some food. The amount of food released is preconfigured and the user cannot change it. The feeder uses a server socket based program which is configured on the raspberry pi 2 in order to recieve the various request/commands that the user sends in order to make the appropriate actions.

ΕΠΙΣΤΗΜΟΝΙΚΗ ΠΕΡΙΟΧΗ: Μικροϋπολογιστές και raspberry pi
ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: raspberry pi, ταΐστρα, σερβοκινητήρας, TCP socket, αισθητήρας κίνησης.

ΠΕΡΙΕΧΟΜΕΝΑ

ΚΕΦΑΛΑΙΟ 1

ΕΙΣΑΓΩΓΗ

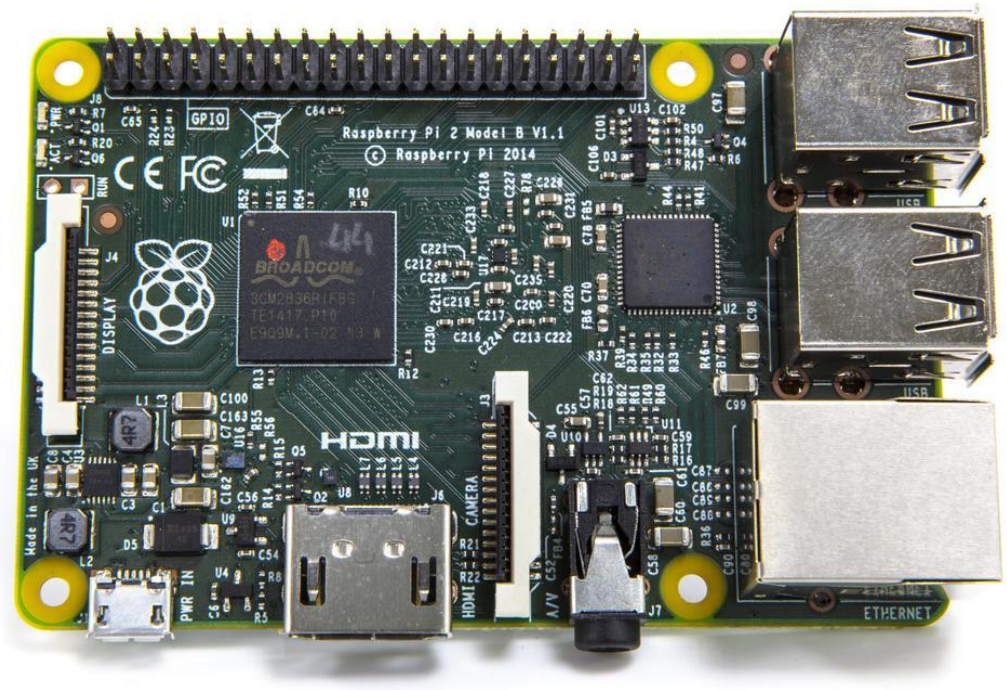
Η παρούσα εργασία εστιάζει στην ανάπτυξη μιας κατασκευής η οποία θα διευκολύνει τους ιδιοκτήτες σκυλιών/γατιών δίνοντας την δυνατότητα να παρέχουν τροφή στα κατοικίδια τους με χρήση της εφαρμογής (μέσω κινητού) σε συνεργασία με την συσκευή-ταΐστρα όποτε το επιθυμούν. Με αυτόν τον τρόπο καθίσταται εφικτή η καταπολέμηση της μεγαλύτερης, ίσως, δυσκολίας που αντιμετωπίζει ένας άνθρωπος όταν θελήσει να αποκτήσει ένα κατοικίδιο, που δεν είναι άλλη από την ερώτηση «όταν δεν θα μπορώ να είμαι στο σπίτι για μερικές μέρες και επίσης δεν θα μπορώ να πάρω μαζί μου το κατοικίδιο, τι θα κάνω;». Αυτή την ερώτηση αποφασίσαμε να την απαντήσουμε με στόχο από εδώ και στο εξής όλοι να έχουν την δυνατότητα να αποκτήσουν το αγαπημένο τους κατοικίδιο.

ΚΕΦΑΛΑΙΟ 2

RASPBERRY PI

2.1 Ανάλυση του raspberry

Το Raspberry Pi είναι ένας μικρός ως προς το μέγεθος υπολογιστής τον οποίο μπορεί ο καθένας να τον προγραμματίσει χρησιμοποιώντας κυρίως την γλώσσα python. Όπως είναι λογικό διαθέτει έναν επεξεργαστή, μια κάρτα γραφικών και μια ram. Έχει διαθέσιμες θύρες usb και Secure Digital (SD) ή MicroSD, ανάλογα το μοντέλο. Αυτή την στιγμή στο εμπόριο υπάρχουν τα Raspberry Pi 1 model A, A+, B, B+ και Raspberry Pi 2 model B και οι διαφορές που φέρουν είναι ως προς την μνήμη και την ταχύτητα. Από τα παραπάνω μοντέλα χρησιμοποιήσαμε το Raspberry Pi 2 model B το οποίο φαίνεται και παρακάτω.



Εικόνα 2.1: Raspberry pi 2 model B.

Τα χαρακτηριστικά του είναι τα εξής :

- A 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

Πέρα από τις προαναφερθείσες σημαντικές διαφορές που παρουσιάζουν τα μοντέλα raspberry μεταξύ τους, διαφέρουν επίσης και στον αριθμό των GPIO pins. Τα μοντέλα raspberry pi A και B έχουν 26, ενώ τα υπόλοιπα διαθέτουν 40. Τα General Purpose Iinput Output pins, όπως καταλαβαίνουμε και από το όνομα, είναι τα pins που μέσω του προγράμματός μας μπορούμε να τα καθορίσουμε ως είσοδο ή έξοδο κι από εκεί να χειριζόμαστε τα άλλα εξαρτήματα. Όπως για παράδειγμα κινητήρες, αισθητήρες, κάμερες κλπ.

Στην ακόλουθη φωτογραφία φαίνονται τα 40 gpio pins.

GPIO#	2nd func	pin#		pin#	2nd func	GPIO#
N/A	+3V3	1		2	+5V	N/A
GPIO2	SDA1 (I2C)	3		4	+5V	N/A
GPIO3	SCL1 (I2C)	5		6	GND	N/A
GPIO4	GCLK	7		8	TXD0 (UART)	GPIO14
N/A	GND	9		10	RXD0 (UART)	GPIO15
GPIO17	GEN0	11		12	GEN1	GPIO18
GPIO27	GEN2	13		14	GND	N/A
GPIO22	GEN3	15		16	GEN4	GPIO23
N/A	+3V3	17		18	GEN5	GPIO24
GPIO10	MOSI (SPI)	19		20	GND	N/A
GPIO9	MISO (SPI)	21		22	GEN6	GPIO25
GPIO11	SCLK (SPI)	23		24	CE0_N (SPI)	GPIO8
N/A	GND	25		26	CE1_N (SPI)	GPIO7
(Models A and B stop here)						
EEPROM	ID_SD	27		28	ID_SC	EEPROM
GPIO5	N/A	29		30	GND	N/A
GPIO6	N/A	31		32	-	GPIO12
GPIO13	N/A	33		34	GND	N/A
GPIO19	N/A	35		36	N/A	GPIO16
GPIO26	N/A	37		38	Digital IN	GPIO20
N/A	GND	39		40	Digital OUT	GPIO21

Εικόνα 2.2: GPIO των raspberry.

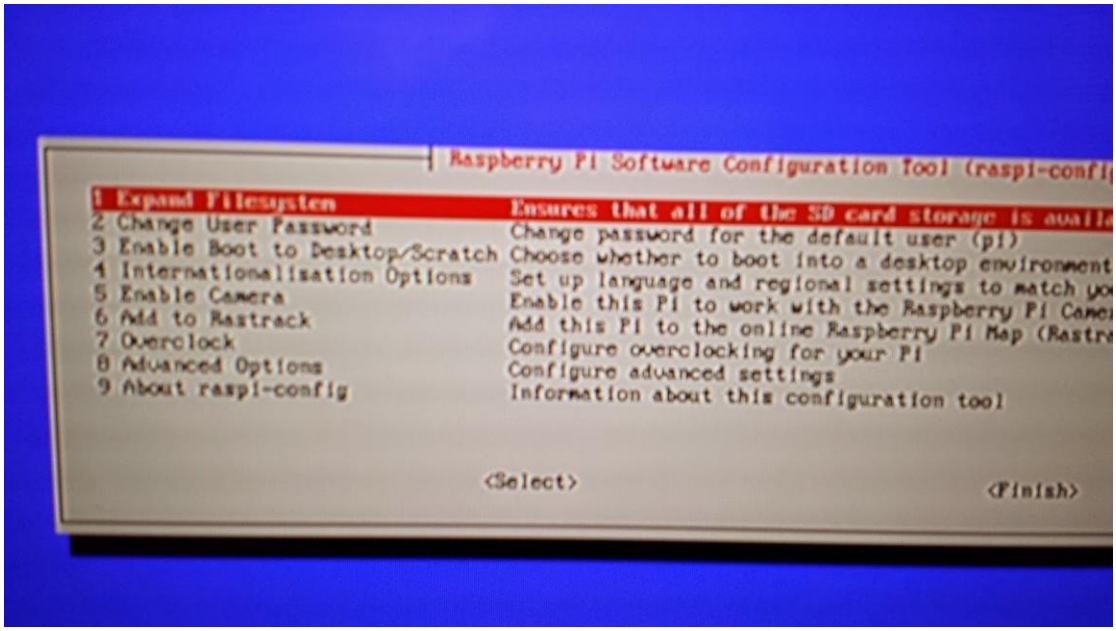
Το λειτουργικό σύστημα του raspberry είναι βασισμένο στα linux, όμως στο raspberry pi model B υπάρχει η δυνατότητα εγκατάστασης των windows 10. Πιο αναλυτικά υποστηρίζει τις εξής κύριες διανομές linux :

- Raspbian
- OpenELEC
- RISC OS
- Ubuntu

Το λειτουργικό σύστημα που θα διαλέξουμε να εγκαταστήσουμε στο raspberry μας το βρίσκουμε από το επίσημο site https://www.raspberrypi.org/downloads/. Η επιλογή γίνεται ανάλογα με το σε ποιο περιβάλλον είμαστε περισσότερο εξοικειωμένοι.

2.2 Setup Raspberry

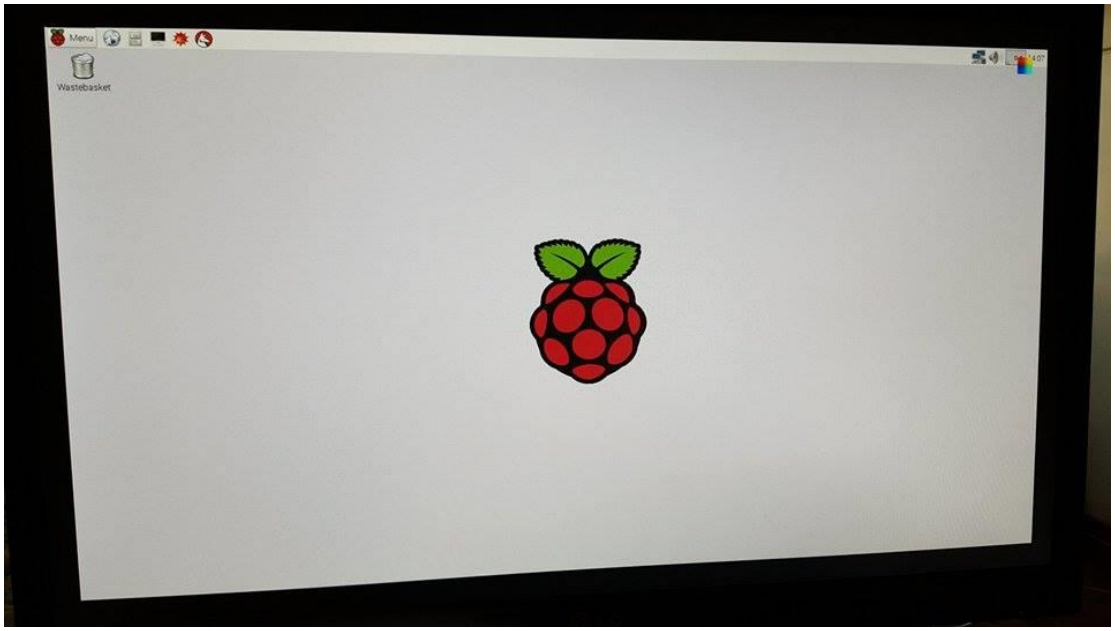
Αφού διαλέξουμε το λειτουργικό μας σύστημα και το περάσουμε στο raspberry όταν το ανοίξουμε (βάζοντάς το στην πρίζα) και ξεκινήσουμε την εγκατάστασή του, όταν ολοκληρωθεί θα δούμε στην οθόνη μας να γίνονται αυτόματα κάποια configuration. Έπειτα θα έχουμε μπροστά μας ένα παράθυρο Setup Options όπως φαίνεται στην εικόνα από κάτω.



Εικόνα 2.3: Raspberry Setup.

Σε αυτό το παράθυρο μας δίδεται η δυνατότητα να κάνουμε κάποια πρόσθετα configuration, όπως το να δηλώσουμε την χρήση της κάμερας του raspberry μας.

Αφού τελειώσουμε με το παραπάνω βήμα, είμαστε έτοιμοι να βρεθούμε στην επιφάνεια εργασίας του raspberry η οποία φαίνεται και στην ακόλουθη φωτογραφία.

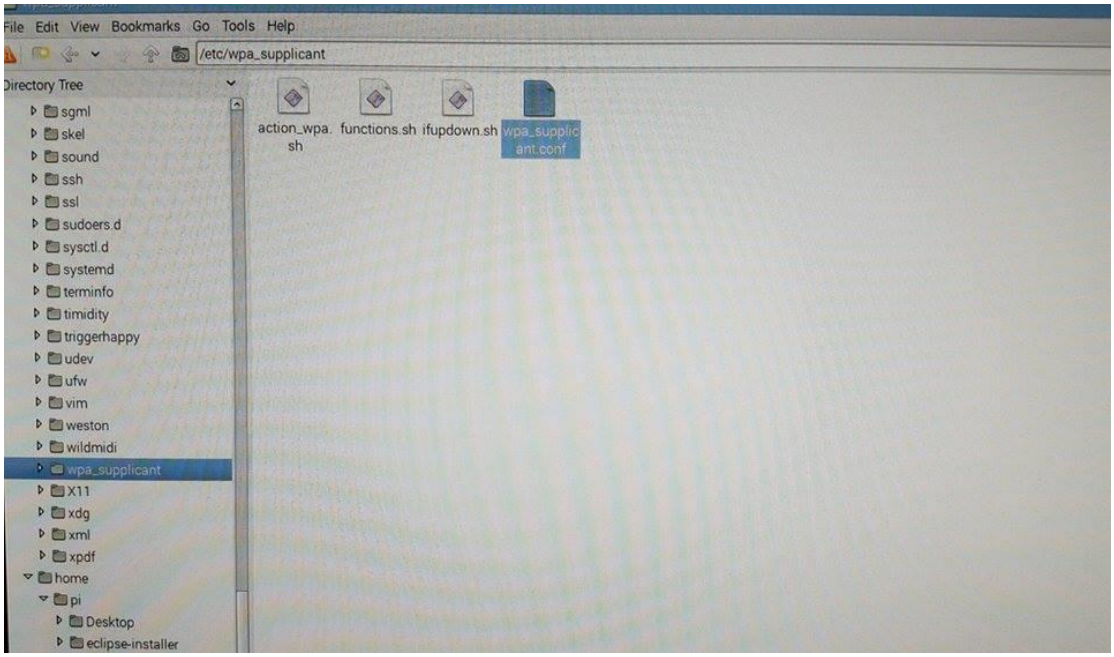


Εικόνα 2.4: Επιφάνεια εργασίας του raspberry.

Ανακεφαλαιώνοντας, στην παρούσα στιγμή έχουμε συνδέσει στο raspberry μας ένα ποντίκι και ένα πληκτρολόγιο μέσω usb, μία οθόνη μέσω HDMI, έχουμε συνδεθεί στο internet μέσω Ethernet, έχουμε συνδέσει την κάμερα module του raspberry μέσω ενός flex cable, και έχουμε εγκαταστήσει το λειτουργικό σύστημα της επιλογής μας. Τέλος έχουμε κάνει όλα τα κατάλληλα configuration και το μόνο που μας λείπει είναι το configuration του wifi.

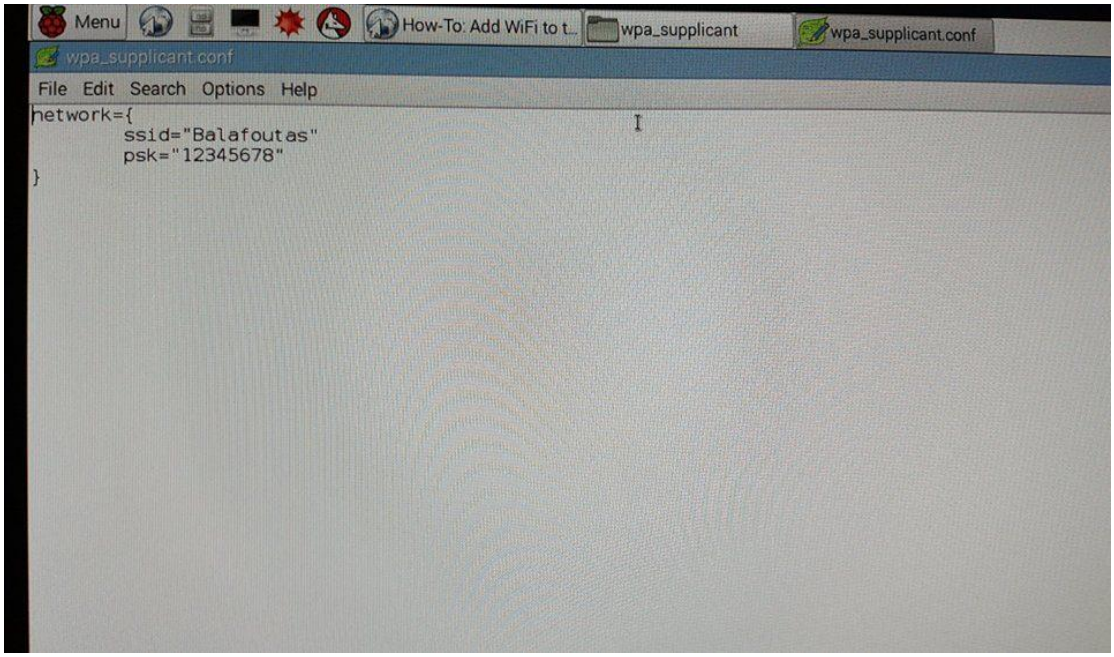
Το wifi configuration μπορούμε να το κάνουμε όπως ακριβώς φαίνεται στις παρακάτω φωτογραφίες.

Βρίσκουμε το αρχείο που είναι στο μονοπάτι που φαίνεται στην εικόνα 2.5.



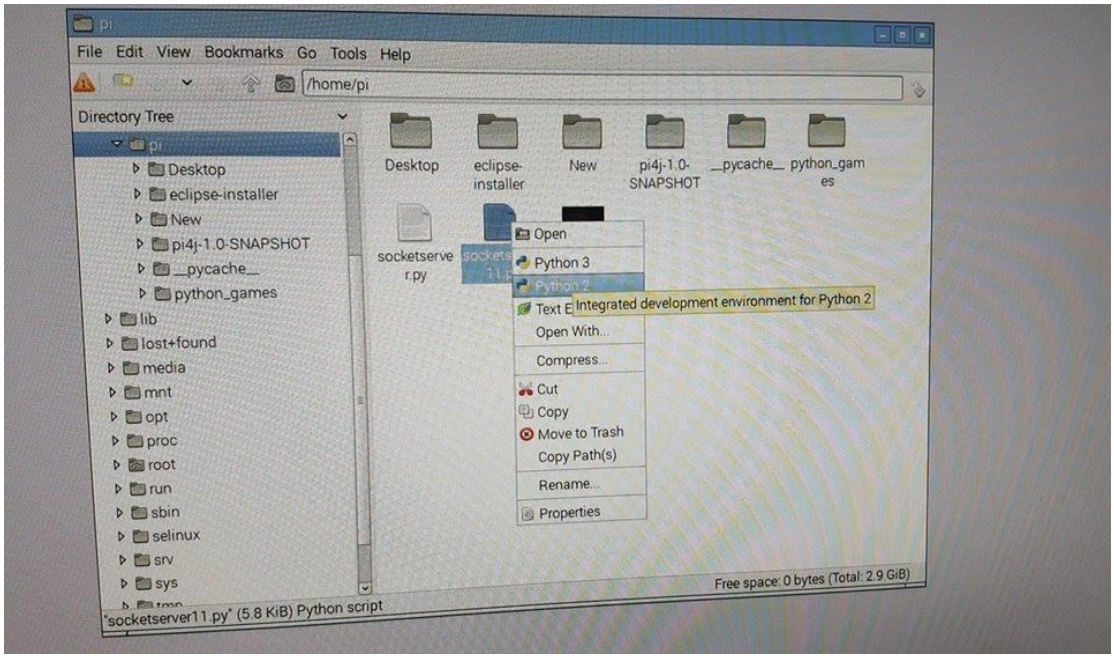
Εικόνα 2.5: Wifi configuration (1).

Ανοίγουμε το αρχείο αυτό, και γράφουμε το όνομα του wifi και τον κωδικό του με τον τρόπο που φαίνεται παρακάτω στην εικόνα 2.6.

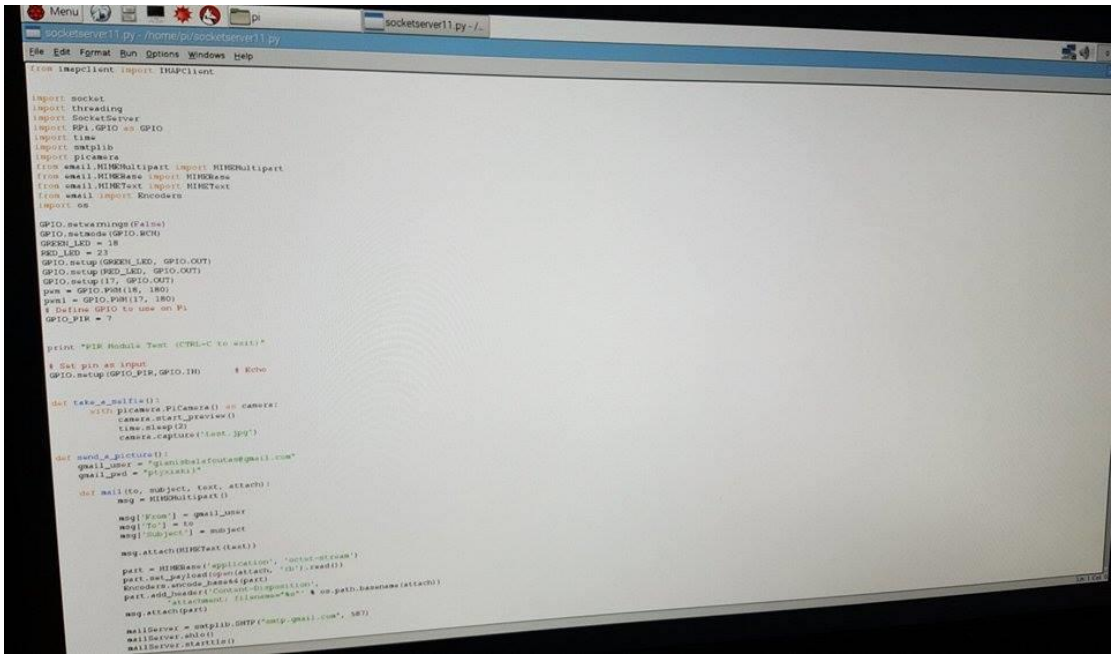


Εικόνα 2.6: Wifi configuration (2).

Τώρα είμαστε έτοιμοι να ανοίξουμε το πρόγραμμά μας. Με δεξί κλικ στο αρχείο που είναι ο κώδικας και επιλέγουμε το python 2, όπως φαίνεται παρακάτω.

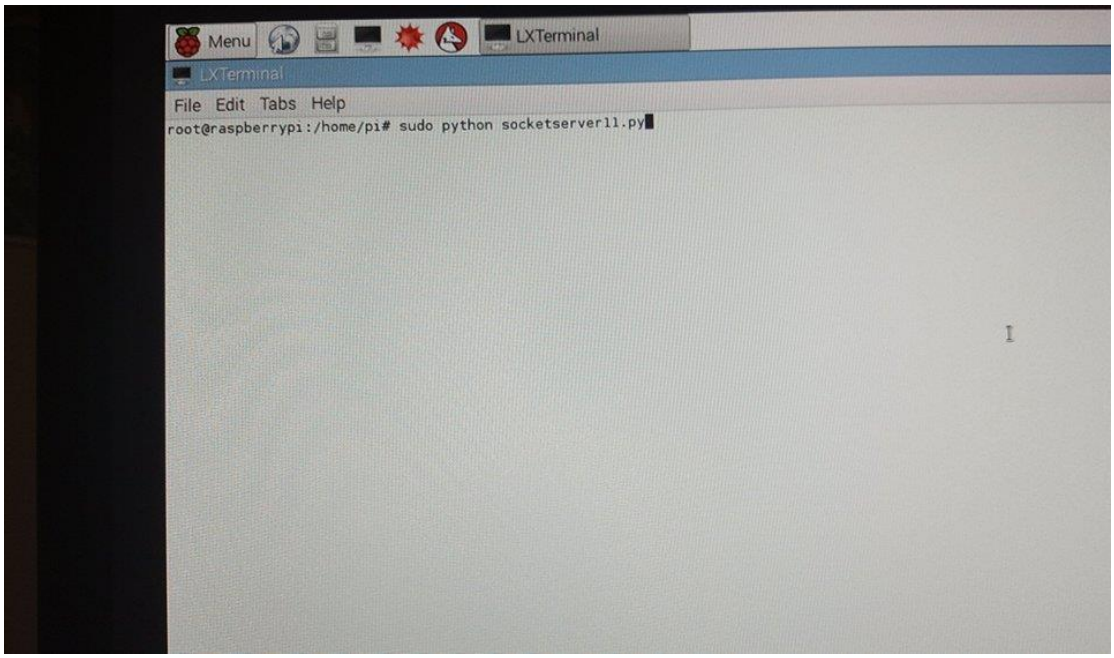


Εικόνα 2.7: Άνοιγμα του προγράμματός μας με σκοπό την επεξεργασία του.

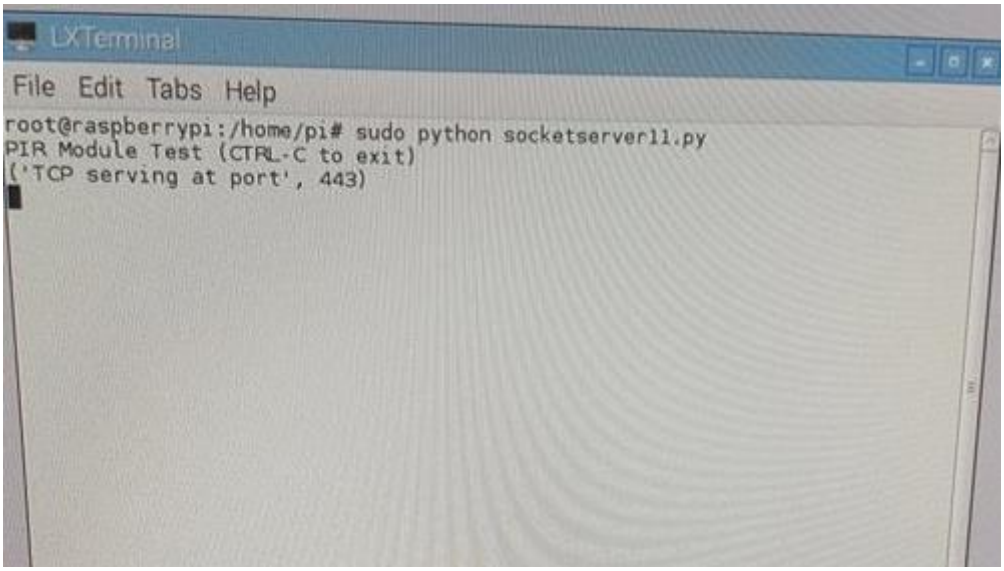


Εικόνα 2.8: Το πρόγραμμά μας διαθέσιμο προς επεξεργασία.

Ο κώδικας μας είναι γραμμένος σε γλώσσα python, οπότε το πρόγραμμά μας έχει κατάληξη «.py». Για να τρέξουμε το πρόγραμμα που έχουμε γράψει το κάνουμε μέσω του τερματικού (terminal) όπως φαίνεται στην συνέχεια.

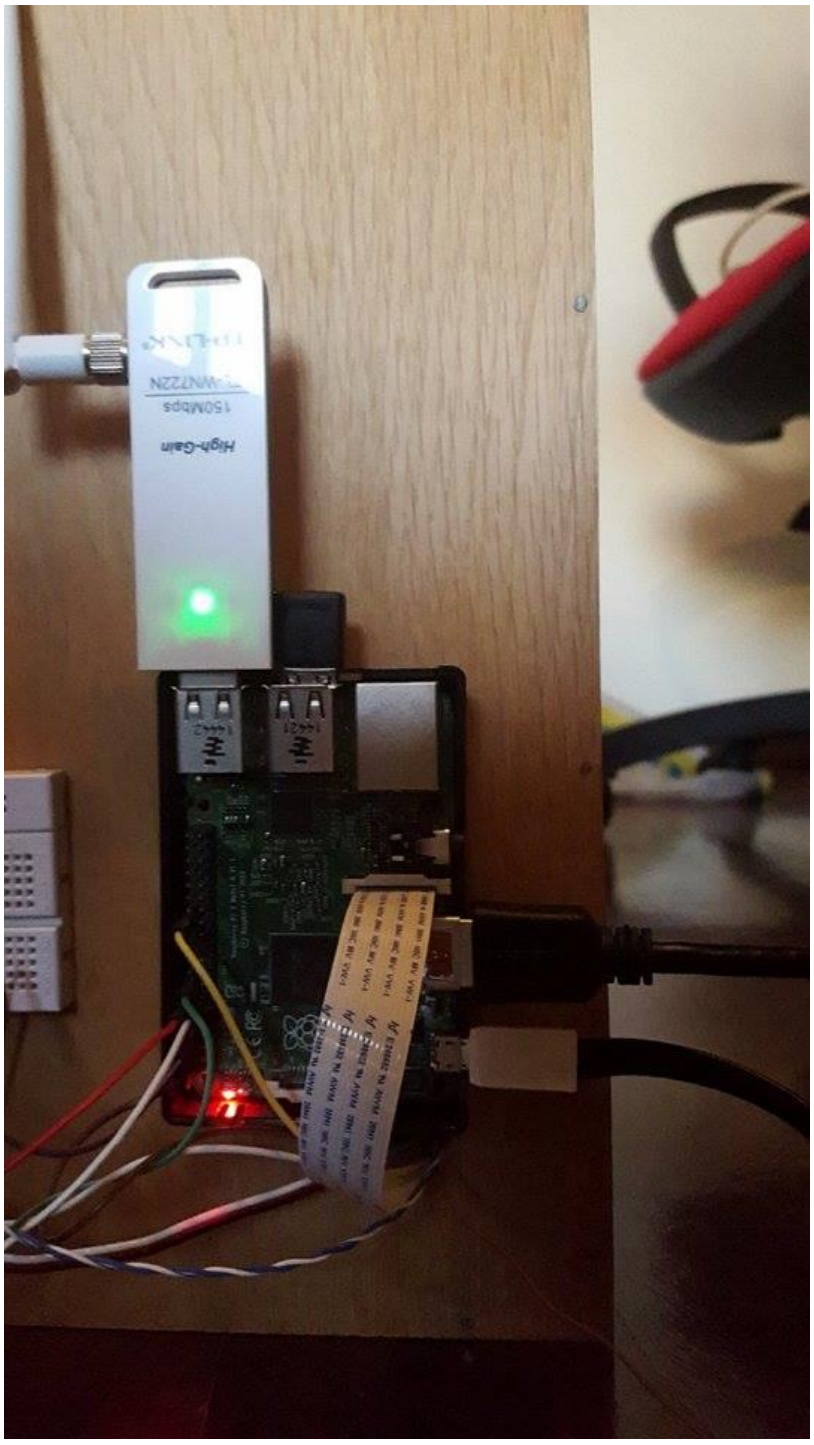


Εικόνα 2.9: Εισαγωγή εντολής εκτέλεσης του προγράμματος μέσω του τερματικού.



Εικόνα 2.10: Αποτέλεσμα της εντολής εκτέλεσης.

Για την υλοποίηση της εργασία μας χρησιμοποιήσαμε κάποια gpio pins έτσι ώστε να συνδέσουμε τα εξαρτήματα, 2 σερβοκινητήρες και 1 αισθητήρα κίνησης.



Εικόνα 2.11: Συνδεσμολογία του raspberry.

ΚΕΦΑΛΑΙΟ 3

ΣΕΡΒΟΚΙΝΗΤΗΡΕΣ

Ο σερβοκινητήρας αποτελείται από τρία μέρη.

1. Ηλεκτροκινητήρας συνεχούς ρεύματος.
2. Ηλεκτρονικό κύκλωμα για τον έλεγχο θέσης του τελικού άξονα κίνησης.
3. Κιβώτιο υποβιβασμού της σχέσης μετάδοσης του κινητήρα.

Ο τελικός άξονας κίνησης δεν εκτελεί ολόκληρες περιστροφές, αλλά περιστρέφεται μεταξύ δύο ακραίων θέσεων.

Για να λειτουργήσουμε έναν σερβοκινητήρα χρειάζεται να του δώσουμε την κατάλληλη τάση και ένα σήμα (παλμό) το οποίο θα δηλώνει την θέση περιστροφής του τελικού άξονα.

Πλεονεκτήματα :

- Χαμηλό κόστος.
- Μικρές διαστάσεις.
- Υψηλή παραγόμενη ροπή.
- Προσδιορισμός θέσης άξονα χωρίς κύκλωμα ανάδρασης.

Μειονεκτήματα :

- Δεν μπορεί να εκτελέσει ολόκληρη περιστροφή.

Εικόνα 3.1: Σερβοκινητήρας.

Στην κατασκευή μας χρησιμοποιούμε, όπως ήδη έχουμε αναφέρει, δύο σερβοκινητήρες. Ο ένας χρησιμοποιείται για το άνοιγμα και κλείσιμο της παροχής τροφής, και ο άλλος για να κουνάει την τροφή που υπάρχει μέσα στην κατασκευή.

Και οι δύο παίρνουν σήμα από το raspberry όταν πρέπει να ξεκινήσουν την περιστροφή τους.

Εικόνα 3.2: Το μπροστινό μέρος του πρώτου σερβοκινητήρα.

Εικόνα 3.3: Το μπροστινό μέρος του δεύτερου σερβοκινητήρα.

ΚΕΦΑΛΑΙΟ 4

ΑΙΣΘΗΤΗΡΑΣ ΚΙΝΗΣΗΣ

Ο αισθητήρας κίνησης ενεργοποιείται όταν αντιληφθεί κίνηση στον χώρο δράσης του. Τότε θα στείλει ένα σήμα στο raspberry και μέσω του κώδικα που έχουμε γράψει θα χειριστούμε την κατάσταση.

Εικόνα 4.1: Ο αισθητήρας κίνησης που χρησιμοποιούμε.

Ο αισθητήρας τίθεται σε ενεργή κατάσταση όταν του στείλουμε το κατάλληλο σήμα από το raspberry και τότε περιμένουμε μέχρι να ανιχνεύσει κάποια κίνηση και να μας αποστείλει το σήμα. Όταν το λάβουμε συνεχίζεται η εκτέλεση του κώδικά μας.

ΚΕΦΑΛΑΙΟ 5

ΑΝΑΛΥΣΗ ΕΡΓΑΣΙΑΣ

5.1 Περιγραφή εργασίας

Εικόνα 5.1: Η κατασκευή μας - πρόσοψη.

Εικόνα 5.2: Η κατασκευή μας – κάτοψη.

Στις δύο παραπάνω εικόνες φαίνονται οι 2 σερβοκινητήρες, η 1 κάμερα και ο 1 αισθητήρας κίνησης.

Ο κώδικας που έχει γραφτεί είναι βασισμένος σε 3 διαφορετικές λειτουργίες.

1. Να ανοίξουμε την παροχή τροφής και να ταΐσουμε το κατοικίδιο μας.
2. Να θέσουμε σε εφαρμογή τον αισθητήρα κίνησης και όταν ανιχνεύσει κάποια κίνηση να τραβήξει φωτογραφία η κάμερα που έχουμε εγκατεστημένη και να μας την στείλει στο email μας.
3. Να βγάλει φωτογραφία όταν ζητηθεί.

Η πρώτη, είναι και η βασική λειτουργία που γι αυτήν αναπτύξαμε την παρούσα εργασία. Όταν το raspberry πάρει εντολή, ενεργοποιεί πρώτα τον σερβοκινητήρα που χειρίζεται την παροχή τροφής, και έπειτα ενεργοποιεί τον δεύτερο σερβοκινητήρα έτσι ώστε να παραχθεί, μέσα στον χώρο αποθήκευσης τροφής, κινητική ενέργεια με σκοπό να μπορέσει η τροφή να κινηθεί και να πέσει από την ανοιχτή τρύπα. Όταν ολοκληρωθεί η κίνηση του εσωτερικού κινητήρα και έχει πέσει η επιθυμητή ποσότητα τροφής τότε ο εξωτερικός κινητήρας ξεκινάει ξανά με σκοπό να κλείσει την παροχή τροφής.

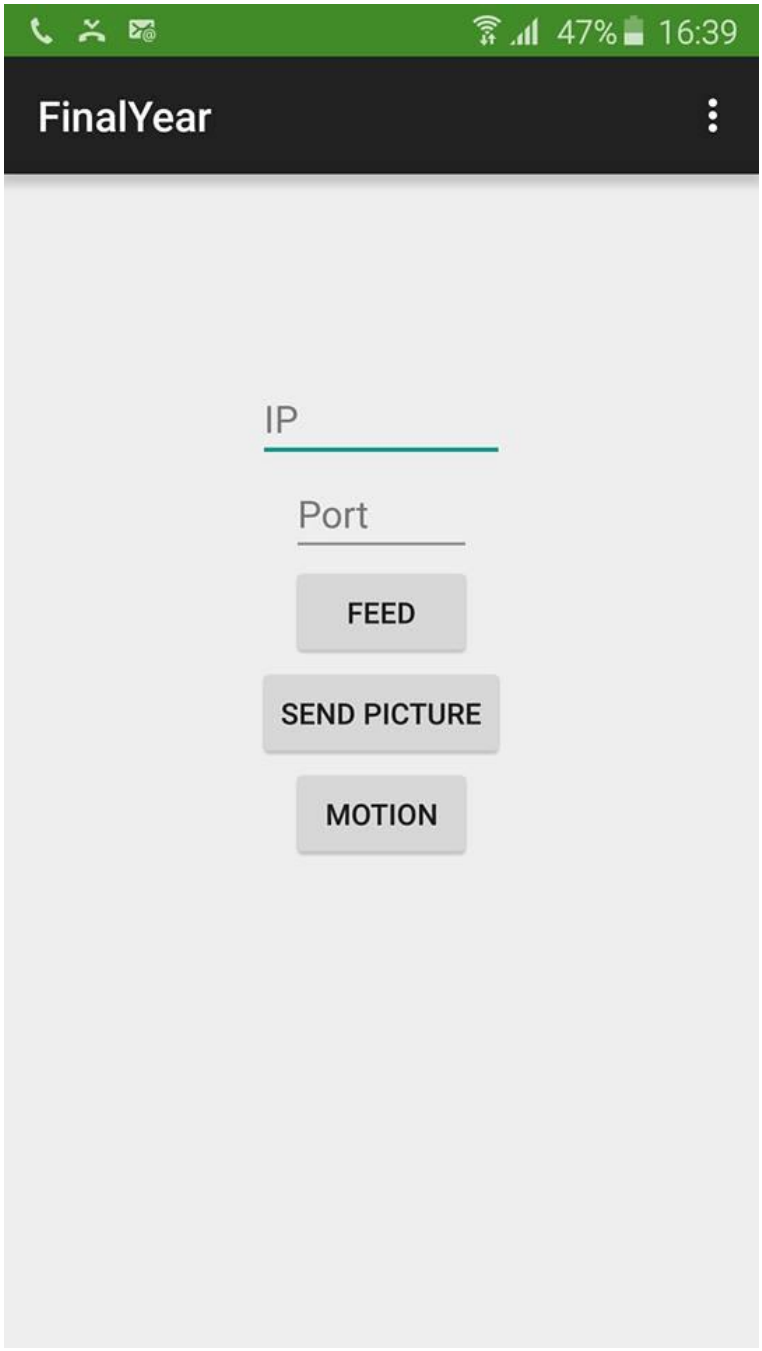
Η δεύτερη λειτουργία έχει ως στόχο να φωτογραφίσουμε το κατοικίδιό μας με σκοπό να δούμε πως είναι καλά. Για να το πετύχουμε αυτό πρέπει να πιάσει ο αισθητήρας κάποια κίνηση και τότε να τραβήξει την φωτογραφία. Αυτή την φωτογραφία την αποθηκεύει τοπικά στο raspberry και αφού ανοίξει μία σύνδεση με τον mail server την στέλνει στο email μας.

Η τρίτη λειτουργία αναπτύχθηκε λόγω του ότι μπορεί να θέλουμε να φωτογραφίσουμε την εναπομένουσα τροφή για να ξέρουμε οποιαδήποτε στιγμή πόση τροφή έχει μείνει και αναλόγως να στείλουμε σήμα ενεργοποίησης της λειτουργίας 1. Η διαδικασία λήψης της φωτογραφίας γίνεται χωρίς να περιμένουμε κάποια κίνηση, όπως γινόταν στην προηγούμενη λειτουργία. Η αποθήκευση και η αποστολή της φωτογραφίας όμως γίνεται με τον ίδιο ακριβώς τρόπο.

Οι τρεις παραπάνω λειτουργίες δεν μπορούν να γίνουν παράλληλα, αλλά σε σειρά. Δηλαδή αν θέλουμε να κάνουμε τις λειτουργίες 3, 1 και 2, θα πρέπει να ολοκληρωθεί η 3, για να αρχίσει η 1 κ.ο.κ για να έχουμε το αποτέλεσμα που θέλουμε.

Αυτές τις τρεις λειτουργίες τις θέτουμε σε εφαρμογή μέσω του κινητού μας και της εφαρμογής που έχουμε αναπτύξει.

5.2 Η εφαρμογή



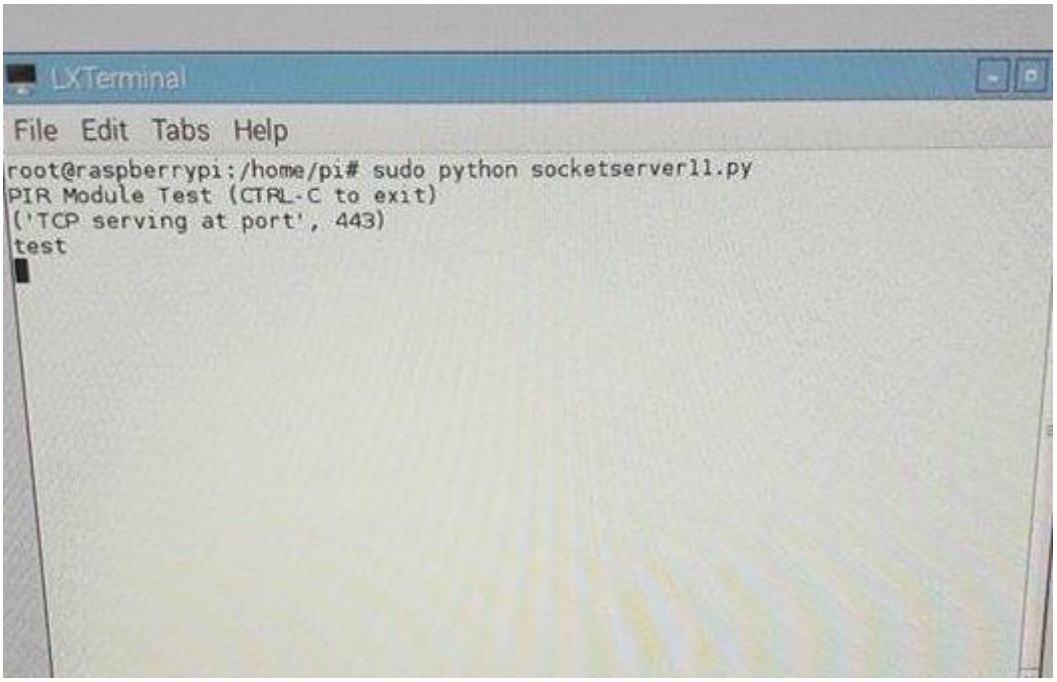
Εικόνα 5.3: Η εφαρμογή που έχουμε αναπτύξει για την επικοινωνία με το raspberry.

Όπως βλέπουμε στην εικόνα 18 η εφαρμογή έχει δύο πεδία προς εισαγωγή στοιχείων και 3 επιλογές (λειτουργίες).

Στο πεδίο IP εισάγουμε την IP του raspberry και στο πεδίο της πόρτας, την πόρτα που έχουμε καθορίσει στον κώδικά μας στην οποία «ακούει» το raspberry.

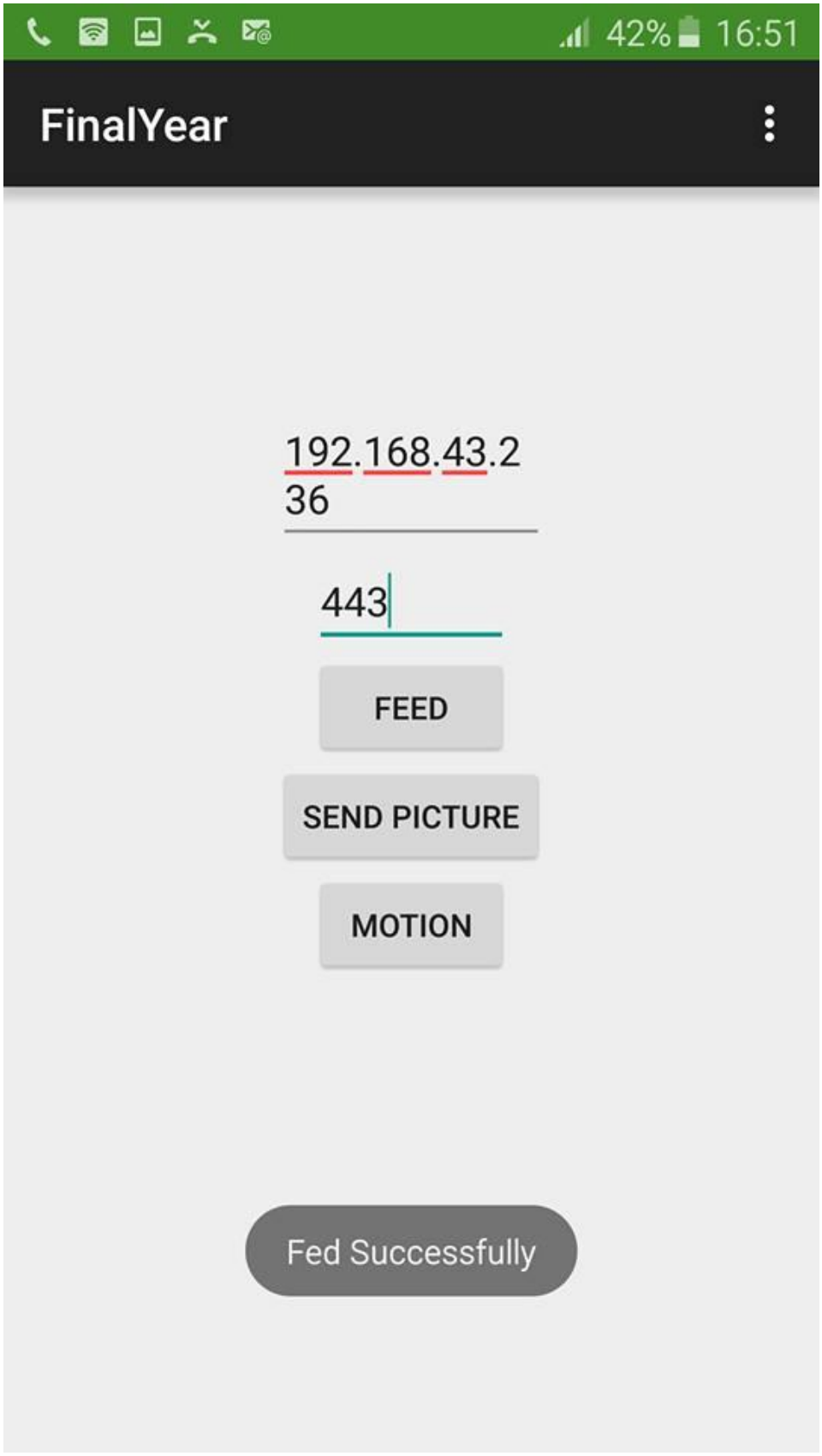
Οι από κάτω τρεις επιλογές είναι οι τρεις λειτουργίες που έχουμε αναλύσει παραπάνω.

Αν για παράδειγμα πατήσουμε το «feed» τότε θα φέρει στο terminal του raspberry το αποτέλεσμα που φαίνεται στην εικόνα.



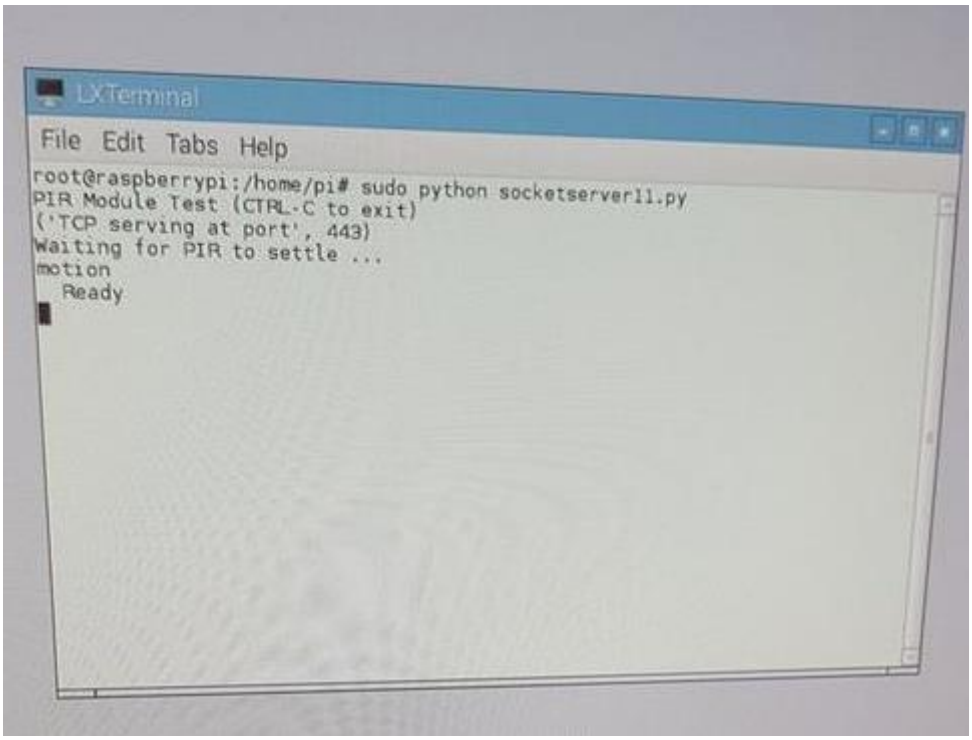
Εικόνα 5.4: Αποτελέσματα που εμφανίζονται στο τερματικό (1).

Και στην εφαρμογή θα πάρουμε για απάντηση ένα μήνυμα που θα λέει πως ολοκληρώθηκε με επιτυχία, βλέπε εικόνα 20.



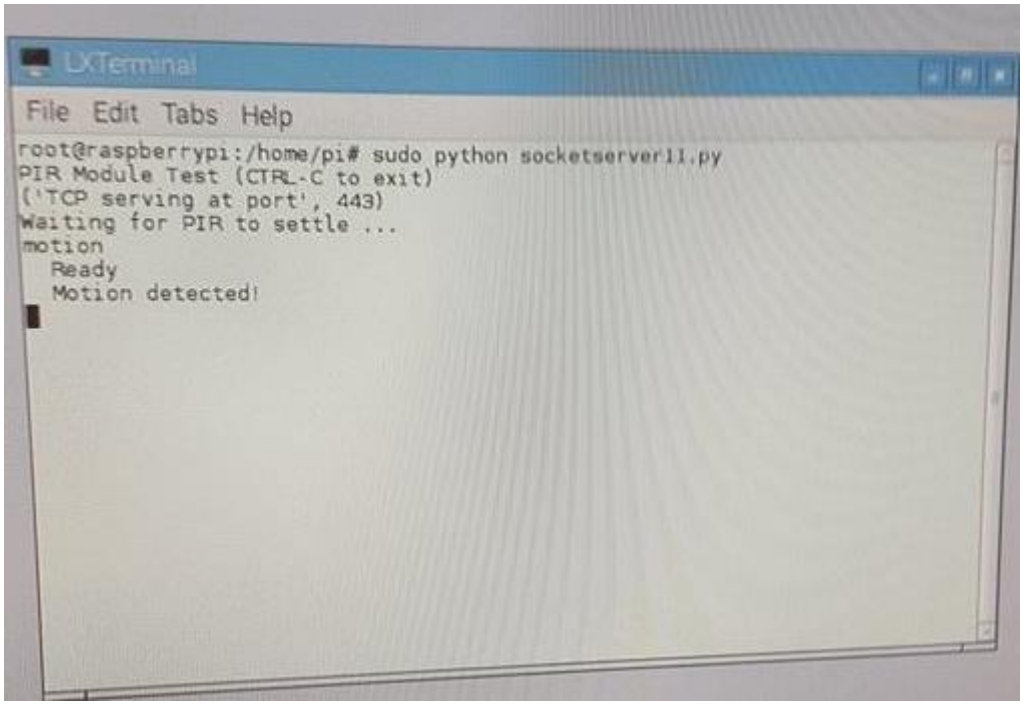
Εικόνα 5.5: Η ανάδραση που έχουμε στην εφαρμογή.

Αν πατήσουμε στην επιλογή «motion» τότε στο terminal του raspberry θα δούμε την παρακάτω εικόνα. Και θα την βλέπουμε μέχρις ότου να ανιχνεύσει κάποια κίνηση.



Εικόνα 5.6: Αποτελέσματα που εμφανίζονται στο τερματικό (2).

Όταν ανιχνεύσει κάποια κίνηση τότε η εικόνα του τερματικού θα γίνει όπως στην παρακάτω εικόνα.



Εικόνα 5.7: Αποτελέσματα που εμφανίζονται στο τερματικό (3).

ΚΕΦΑΛΑΙΟ 6

ΕΠΙΚΟΙΝΩΝΙΑ RASPBERRY ΚΑΙ ANDROID APPLICATION

6.1 Εισαγωγή

Το raspberry επικοινωνεί με το android application που έχουμε δημιουργήσει και δείξει στην εικόνα 18, μέσω TCP socket. Το TCP (Transmission Control Protocol) δηλαδή Πρωτόκολλο Ελέγχου Μεταφοράς, χρησιμοποιείται για την αποστολή και λήψη δεδομένων και την σωστή μεταφορά των δεδομένων χωρίς λάθη.

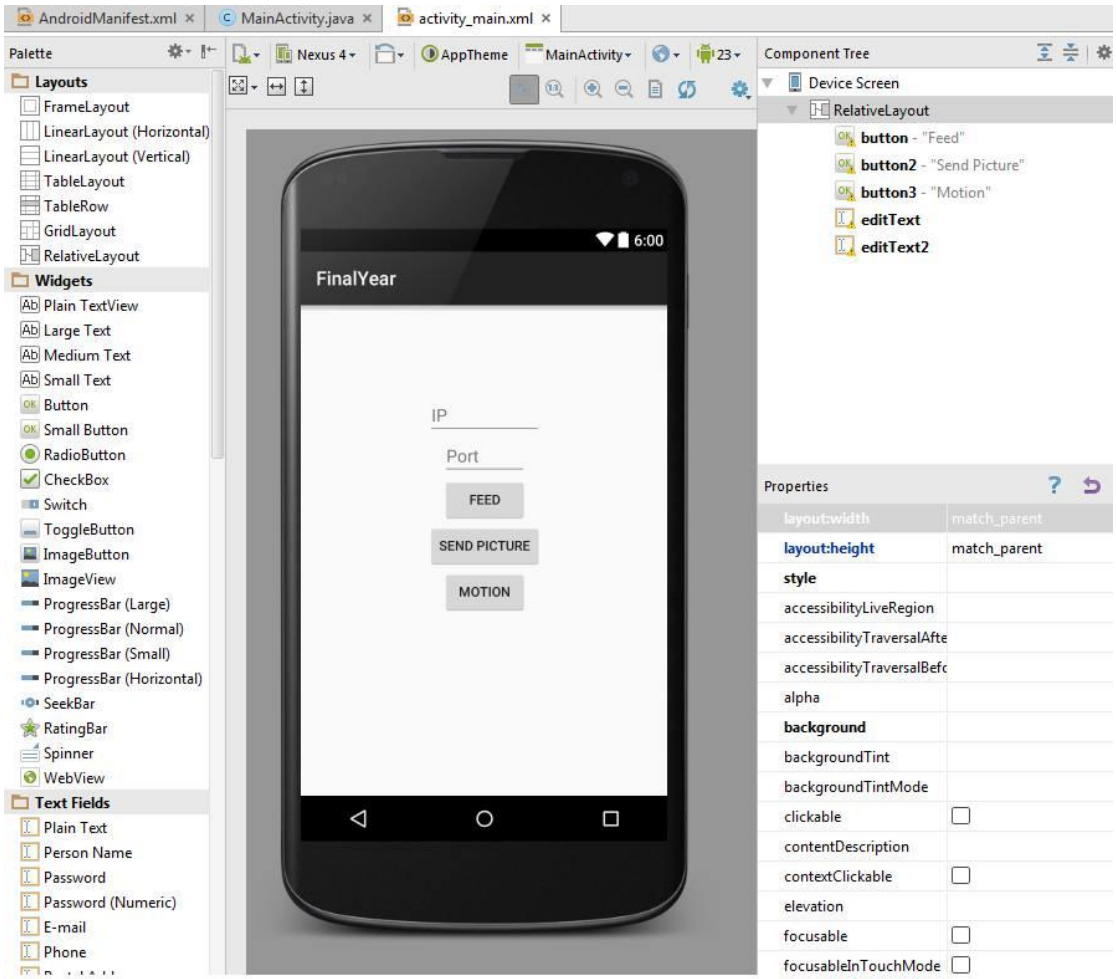
Το raspberry έχει τον ρόλο του server και η εφαρμογή τον ρόλο του client. Επομένως όταν το raspberry διαβάσει δεδομένα μέσω της πόρτας που έχουμε καθορίσει τότε κάνει τις ανάλογες ενέργειες.

Με λίγα λόγια όταν στην εφαρμογή μας εισάγουμε την IP του raspberry και την προκαθορισμένη πόρτα επικοινωνίας και επιλέξουμε μία από τις τρεις λειτουργίες τότε η εφαρμογή θα στείλει τα κατάλληλα δεδομένα προς το raspberry, και αυτό με τη σειρά του θα τα διαβάσει και αναλόγως ποια λειτουργία έχουμε επιλέξει θα την εκτελέσει. Η επικοινωνία αυτή, όπως γράφτηκε και παραπάνω, πραγματοποιείται μέσω TCP socket, πράγμα που σημαίνει πως άμα η IP ή/και η πόρτα δεν είναι σωστά τότε δεν θα γίνει τίποτα.

6.2 Ανάλυση του android studio και κώδικα εφαρμογής

Την εφαρμογή την αναπτύξαμε χρησιμοποιώντας το android studio.

Σε κάθε εφαρμογή υπάρχουν δύο μέρη. Το μπροστά και το πίσω. Το μπροστά μέρος είναι όπως φαίνεται και στην από κάτω εικόνα. Δηλαδή είναι αυτό που σχεδιάζουμε για να βλέπει ο χρήστης/χειριστής της εφαρμογής μας.



Εικόνα 6.1: Το μπροστά μέρος της εφαρμογής.

Το μπροστά μέρος δηλαδή είναι το interface.

Το πίσω μέρος είναι αυτό το οποίο δεν βλέπει ο χρήστης, αλλά μόνο ο προγραμματιστής. Είναι αυτό όπου εκεί εκτελούνται οι κατάλληλες εντολές έτσι ώστε να έχουμε το επιθυμητό αποτέλεσμα στον κώδικά μας.

Πιο συγκεκριμένα όπως είδαμε παραπάνω έχουμε τα εξής 5 πεδία που μπορεί να επιλέξει ο χρήστης.

Έχουμε :

- IP
- Port
- Feed
- Send Picture
- Motion

Οι δύο πρώτες ενέργειες, η εισαγωγή της IP και της πόρτας, δημιουργούνται στο πίσω μέρος με τον τρόπο που φαίνεται στην παρακάτω εικόνα.

```
<EditText
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/editText"
    android:layout_alignParentTop="true"
    android:layout_marginTop="78dp"
    android:hint="IP"
    android:layout_alignLeft="@+id/button2"
    android:layout_alignStart="@+id/button2"
    android:layout_alignRight="@+id/button2"
    android:layout_alignEnd="@+id/button2" />

<EditText
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/editText2"
    android:layout_below="@+id/editText"
    android:layout_alignLeft="@+id/button"
    android:layout_alignStart="@+id/button"
    android:layout_alignRight="@+id/button"
    android:layout_alignEnd="@+id/button"
    android:hint="Port" />
```

Εικόνα 6.2: Η υλοποίηση των δύο πεδίων Ip, port.

Οι άλλες 3 επιλογές, αντίστοιχα, υλοποιούνται όπως φαίνεται παρακάτω.

AndroidManifest.xml × | MainActivity.java × | activity_main.xml ×

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools" android:layout_width="match_parent"
    android:layout_height="match_parent" android:paddingLeft="16dp"
    android:paddingRight="16dp"
    android:paddingTop="16dp"
    android:paddingBottom="16dp" tools:context=".MainActivity">


    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Feed"
        android:id="@+id/button"
        android:layout_below="@+id/editText2"
        android:layout_centerHorizontal="true" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Send Picture"
        android:id="@+id/button2"
        android:layout_below="@+id/button"
        android:layout_centerHorizontal="true" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Motion"
        android:id="@+id/button3"
        android:layout_below="@+id/button2"
        android:layout_centerHorizontal="true" />
```

Εικόνα 6.3: Η υλοποίηση των τριών κουμπιών.

6.3 Λίγα λόγια για το android studio

Το android studio είναι ένα περιβάλλον στο οποίο μπορούμε να αναπτύξουμε εφαρμογές για συσκευές οι οποίες τρέχουν κάποια έκδοση του λογισμικού android.

Τα προγράμματα που γράφουμε είναι σε γλώσσα java και στο αρχείο αυτό είναι που γίνονται οι πράξεις που θέλουμε να κάνουμε. Όλες οι λειτουργίες δηλαδή υλοποιούνται εκεί. Με άλλα λόγια, το .java αρχείο είναι το καθαρό πίσω μέρος της όποιας εφαρμογής.

Παρακάτω φαίνεται ο κώδικας που έχουμε γράψει για την εφαρμογή που έχουμε αναπτύξει.

6.4 Κώδικας

```
package com.balafoutas.myapplication;

import android.os.AsyncTask;
import android.os.Environment;
import android.support.v7.app.ActionBarActivity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

import java.io.ByteArrayOutputStream;
import java.io.DataOutputStream;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.io.OutputStream;
import java.net.Socket;
```

```
public class MainActivity extends ActionBarActivity {


    Button btn;
    Button btn2;
    Button btn3;
    EditText ip;
    EditText port;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);


        ip = (EditText) findViewById(R.id.editText);
        port = (EditText) findViewById(R.id.editText2);
        btn = (Button) findViewById(R.id.button);
        btn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if(ip.getText().toString().equalsIgnoreCase("")                    ||
port.getText().toString().equalsIgnoreCase("")) {
                    Toast.makeText(getApplicationContext(),  "Fill  in  all  the  info",
Toast.LENGTH_SHORT).show();
                }
                else
                    new SendAction("test").execute("");
            }


        });
```

```
        btn2 = (Button) findViewById(R.id.button2);
        btn2.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if(ip.getText().toString().equalsIgnoreCase("") ||
port.getText().toString().equalsIgnoreCase("")) {
                    Toast.makeText(getApplicationContext(), "Fill in all the info",
Toast.LENGTH_SHORT).show();
                }
                else
                    new SendAction("stop").execute("");
            }

        });

        btn3 = (Button) findViewById(R.id.button3);
        btn3.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if(ip.getText().toString().equalsIgnoreCase("") ||
port.getText().toString().equalsIgnoreCase("")) {
                    Toast.makeText(getApplicationContext(), "Fill in all the info",
Toast.LENGTH_SHORT).show();
                }
                else
                    new SendAction("motion").execute("");
            }
```

```
        });
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.menu_main, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar will
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.xml.
        int id = item.getItemId();

        //noinspection SimplifiableIfStatement
        if (id == R.id.action_settings) {
            return true;
        }

        return super.onOptionsItemSelected(item);
    }

    public class SendAction extends AsyncTask<String, Integer, String> {
        String response = "";
```

```
Socket socket;
ObjectOutputStream oos;
ObjectInputStream ois;
String testString = "";
int count = 0;

public SendAction(String action)
{
   testString = action;
}

@Override
protected String doInBackground(String... params) {

   InputStream inputStream = null;
   OutputStream outputStream = null;



   //establish socket connection to server


   try {
      socket = new Socket(ip.getText().toString(),
Integer.parseInt(port.getText().toString()));


      System.out.println("GetPostCount");
      DataOutputStream writeOut = new
DataOutputStream(socket.getOutputStream());
```

```
            socket.getOutputStream().write(testString.getBytes());
            writeOut.flush();

            ByteArrayOutputStream writeBuffer = new
ByteArrayOutputStream(1024);
            byte[] buffer = new byte[1024];
            int bytesRead;
            InputStream writeIn = socket.getInputStream();

            while((bytesRead = writeIn.read(buffer)) != -1) {
               writeBuffer.write(buffer,0,bytesRead);
               response += writeBuffer.toString("UTF-8");
            }

            socket.close();
         } catch (IOException e) {
            e.printStackTrace();
         }
         return "Executed";
      }

      @Override
      protected void onPostExecute(String result) {

            Toast.makeText(getApplicationContext(), response,
Toast.LENGTH_SHORT).show();

      }
```

```
        @Override
        protected void onPreExecute() {


        }
    }
}
```

6.5 Σχολιασμός κώδικα

Σε αυτό το σημείο θα προσπαθήσουμε να εξηγήσουμε την δομή και κάποια δύσκολα κομμάτια του κώδικά που υπάρχει στις προηγούμενες έξι (6) σελίδες.

Σελίδα 1.

Σε αυτό το σημείο όπως βλέπουμε διαλέγουμε τις βιβλιοθήκες που θέλουμε και τις κάνουμε import.

Σελίδα 2.

Σε αυτό το σημείο βλέπουμε πως δημιουργούμε την κύρια κλάση μας στην οποία μέσα θα έχουμε τις κατάλληλες δηλώσεις μεταβλητών και κλάσεων.

Όπως μπορούμε να δούμε δηλώνουμε τρία (3) btn τα οποία το καθένα είναι για την κάθε μία λειτουργία (feed, send picture, motion).

Σελίδα 3.

Όπως γράφτηκε και παραπάνω τα τρία (3) btn είναι ένα για κάθε λειτουργία που μπορούμε να κάνουμε μέσω της εφαρμογής μας. Στην προηγούμενη σελίδα φάνηκε η υλοποίηση του πρώτου μόνο, στην παρούσα σελίδα φαίνεται και των άλλων δύο.

Όπως μπορούμε να δούμε και οι τρεις υλοποιήσεις των btn είναι σχεδόν ίδιες, με μοναδική διαφορά την παράμετρο που στέλνουμε στην κλάση SendAction.

Σελίδα 4.

Σε αυτή τη σελίδα βλέπουμε τρεις κλάσεις. Οι δύο πρώτες είναι κλάσεις γενικού σκοπού. Η τρίτη κλάση, SendAction, είναι και η πιο σημαντική στο πρόγραμμά μας. Μέσα σε αυτή γίνεται η επικοινωνία με tcp socket που αναλύσαμε και παραπάνω.

Σελίδα 5.

Εδώ έχουμε την συνέχεια της SendAction, και πιο συγκεκριμένα βλέπουμε την δημιουργία socket και την αποστολή της μεταβλητής που προσδιορίζει ποια λειτουργία θέλουμε να εκτελέσουμε, προς την IP που έχουμε ήδη εισάγει και την πόρτα.

Σελίδα 6.

Σε αυτή την σελίδα ολοκληρώνεται το πρόγραμμά μας.

6.5 Ανάλυση κώδικα του raspberry

Σε αυτό το σημείο θα παραθέσουμε τον κώδικα που τρέχει το raspberry ο οποίος έχει γραφτεί σε γλώσσα python, και θα εξηγήσουμε το πώς εκτελείται.

Θυμίζουμε πως έχουμε 3 λειτουργίες, την feed, send picture και motion. Για την λειτουργία feed η εφαρμογή στέλνει στο raspberry την μεταβλητή “test”, για την send picture την “stop” και τέλος για την motion την “motion”.

```
from imapclient import IMAPClient


import socket
import threading
import SocketServer
import RPi.GPIO as GPIO
import time
import smtplib
import picamera
from email.MIMEMultipart import MIMEMultipart
from email.MIMEBase import MIMEBase
from email.MIMEText import MIMEText
from email import Encoders
import os

GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)
GREEN_LED = 18
RED_LED = 23
GPIO.setup(GREEN_LED, GPIO.OUT)
GPIO.setup(RED_LED, GPIO.OUT)
GPIO.setup(17, GPIO.OUT)
pwm = GPIO.PWM(18, 180)
pwm1 = GPIO.PWM(17, 180)
# Define GPIO to use on Pi
GPIO_PIR = 7
```

```
print "PIR Module Test (CTRL-C to exit)"


# Set pin as input
GPIO.setup(GPIO_PIR,GPIO.IN)      # Echo


def take_a_selfie():
      with picamera.PiCamera() as camera:
          camera.start_preview()
          time.sleep(2)
          camera.capture('test.jpg')


def send_a_picture():
   gmail_user = "gianisbalafoutas@gmail.com"
   gmail_pwd = "ptyxiaki1"


   def mail(to, subject, text, attach):
         msg = MIMEMultipart()
         msg['From'] = gmail_user
         msg['To'] = to
         msg['Subject'] = subject
         msg.attach(MIMEText(text))
         part = MIMEBase('application', 'octet-stream')
         part.set_payload(open(attach, 'rb').read())
         Encoders.encode_base64(part)
         part.add_header('Content-Disposition',
               'attachment; filename="%s"' % os.path.basename(attach))
         msg.attach(part)
```

```
        mailServer = smtplib.SMTP("smtp.gmail.com", 587)
        mailServer.ehlo()
        mailServer.starttls()
        mailServer.ehlo()
        mailServer.login(gmail_user, gmail_pwd)
        mailServer.sendmail(gmail_user, to, msg.as_string())
        mailServer.close()

    mail("gianbalafoutas@gmail.com",
        "Geia apo thn ptuxiakh sou!",
        "Oxi ontws. na exei kai photo.",
        "test.jpg")

class ThreadedTCPRequestHandler(SocketServer.BaseRequestHandler):



    def handle(self):
        data2 = self.request.recv(1024)

        response = data2
        if "test" in response:
            pwm.start(15)
            pwm1.start(15)
            for cnt in range (0,15,1):
                for dc in range (0,51,5):
```

```
                pwm1.ChangeDutyCycle(dc)
                time.sleep(0.1)
            for dc in range (50,-1,-5):
                pwm1.ChangeDutyCycle(dc)
                time.sleep(0.1)

        time.sleep(1)
        pwm.ChangeDutyCycle(31)
        print(response)
        self.request.sendall("Fed Successfully")


    if "stop" in response:

        take_a_selfie()
        send_a_picture()
        print(response)
        self.request.sendall("Image sent successfully")

    if "motion" in response:
        Current_State  = 0
        Previous_State = 0
        try:
            print "Waiting for PIR to settle ..."
            print(response)
            self.request.sendall("Motion listener activated")
            while GPIO.input(GPIO_PIR)==1:
```

```
            Current_State  = 0
        print "  Ready"
       # Loop until users quits with CTRL-C
        while True :
        # Read PIR state
            Current_State = GPIO.input(GPIO_PIR)

            if Current_State==1 and Previous_State==0:
                print "  Motion detected!"
                take_a_selfie()
                send_a_picture()
                Previous_State=1
                break

            elif Current_State==0 and Previous_State==1:
                print "  Ready"
                Previous_State=0

        # Wait for 10 milliseconds
            time.sleep(0.01)

    except KeyboardInterrupt:
        print "  Quit"
# Reset GPIO settings
        GPIO.cleanup()
```

```
class ThreadedTCPServer(SocketServer.ThreadingMixIn, SocketServer.TCPServer):
    pass


if __name__ == "__main__":
    # Port 0 means to select an arbitrary unused port
    HOST, PORT = "0.0.0.0", 444


    tcpserver = ThreadedTCPServer((HOST, PORT-1), ThreadedTCPRequestHandler)
    server_thread = threading.Thread(target=tcpserver.serve_forever)
    server_thread.daemon = True
    server_thread.start()
    print("TCP serving at port", PORT-1)


    while True:
        pass
    tcpserver.shutdown()
```

Τώρα θα προσπαθήσουμε να σχολιάσουμε και τον κώδικα όπως κάναμε και προηγουμένως. Ο κώδικάς μας έχει έκταση έξι (6) σελίδες.

Σελίδα 1.

Όπως και στον προηγούμενο κώδικα που αναλύσαμε παραπάνω, έτσι κι εδώ, η πρώτη αυτή σελίδα είναι τα imports που θα χρειαστούμε και οι κατάλληλες δηλώσεις για εξόδος/εισόδους.

Σελίδα 2.

Εδώ βλέπουμε τις δύο συναρτήσεις, take_a_selfie() και send_a_picture(). Η πρώτη βγάζει την φωτογραφία μέσω της κάμερας που έχουμε συνδέσει στο raspberry μας και την αποθηκεύει προσωρινά με όνομα «test.jpg», ενώ η δεύτερη κάνει την σύνδεση με τον mail server και στέλνει την φωτογραφία που αποθηκεύσαμε πριν.

Σελίδα 3.

Στην συνάρτηση mail περνάμε τέσσερις (4) παραμέτρους οι οποίες είναι για το email του παραλήπτη, το θέμα και το περιεχόμενο του email καθώς φυσικά και την φωτογραφία. Επίσης βλέπουμε την κλάση όπου μέσα σε αυτή χειριζόμαστε τα ληφθέντα δεδομένα από την εφαρμογή. Όπως βλέπουμε, έχουμε το πρώτο if το οποίο ελέγχει αν είμαστε στην λειτουργία feed. Αν είμαστε, τότε στέλνει παλμό στον έναν σερβοκινητήρα (τον εξωτερικό) έτσι ώστε να ανοίξει το πορτάκι και μετά αρχίζει να στέλνεις παλμούς ανά 100msec στον δεύτερο σερβοκινητήρα (τον εσωτερικό). Και όταν τελειώσει η κίνηση του εσωτερικού κινητήρα μας, τότε στέλνει έναν παλμό στον εξωτερικό κινητήρα για να κλείσει το πορτάκι.

Σελίδα 4.

Εδώ βλέπουμε και τα άλλα δύο if που είναι για τις άλλες δύο λειτουργίες. Η λειτουργία send picture που εμείς την χειριζόμαστε σαν stop, βγάζει απλά την φωτογραφία και την στέλνει στο email που έχουμε δώσει παραπάνω. Η λειτουργία motion ενεργοποιεί τον αισθητήρα κίνησης και περιμένει μέχρις ότου να ανιχνεύσει κάποια κίνηση, και όταν την ανιχνεύσει τότε βγάζει φωτογραφία και την στέλνει στο email που έχουμε δώσει.

Σελίδα 5.

Σε αυτή τη σελίδα έχουμε την συνέχεια της λειτουργίας motion, η οποία εξηγήθηκε παραπάνω.

Σελίδα 6.

Σε αυτό το σημείο γίνεται η δήλωση της σύνδεσης μεταξύ της εφαρμογής του κινητού μας και του raspberry με χρήση TCP socket όπως έχουμε εξηγήσει σε προηγούμενα κεφάλαια.