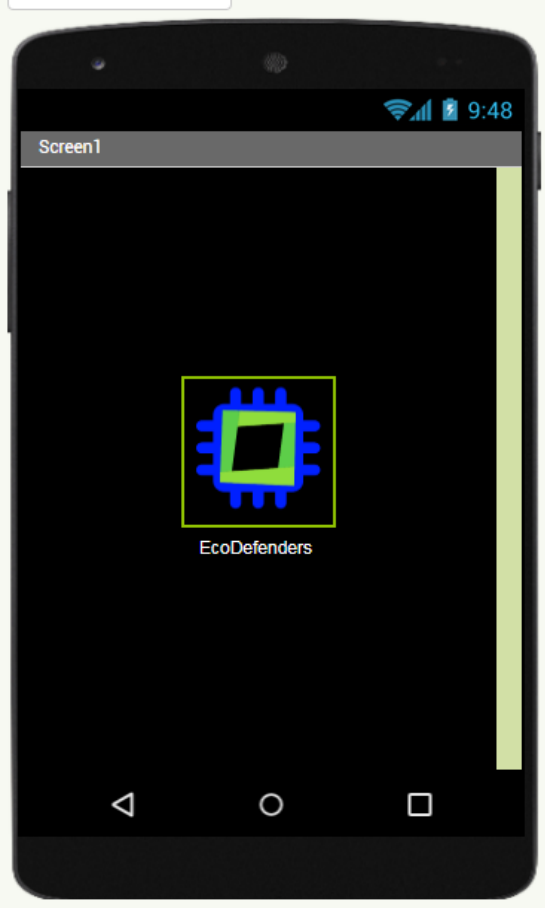
MIT APP INVENTOR

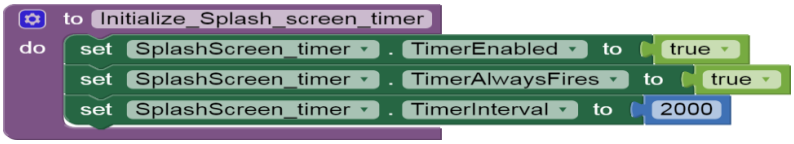
Η εφαρμογή αποτελείται από δύο ξεχωριστές οθόνες (screens). Η πρώτη (screen1) αποτελεί την «splash screen» την αρχική οθόνη, δηλαδή όπου παρουσιάζεται, για λίγα δευτερόλεπτα, το logo και το όνομα της ομάδας μας. Η δεύτερη (app) περιέχει τον κώδικα της εφαρμογής.

Screen: Screen1

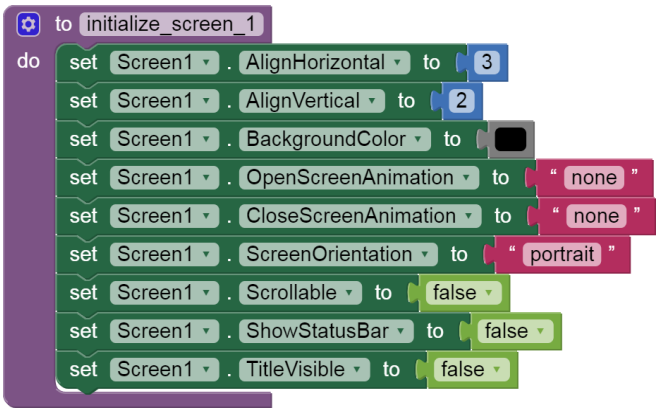


Διαδικασίες (procedures) αρχικοποίησης:

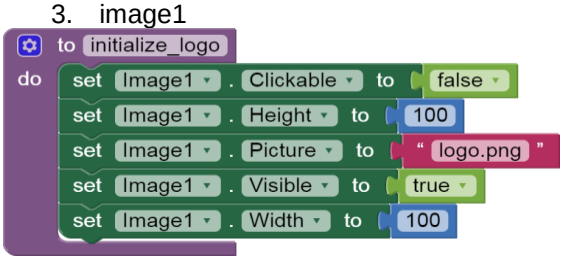
1. Clock1



2. Screen1

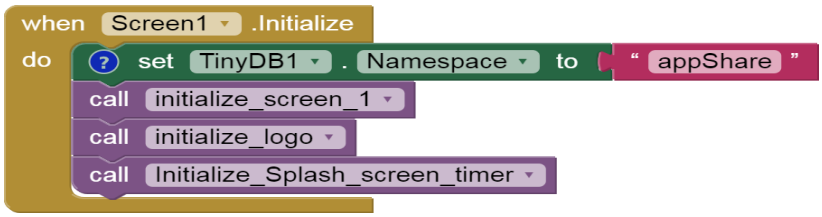


3. image1



Εικόνα 1 Μορφοποίηση οθόνης(screen1)

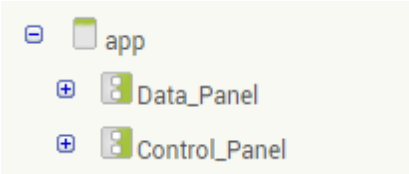
Οι διαδικασίες που δημιουργήθηκαν, τοποθετούνται στο control block «screen initialize» και ορίζεται ο χώρος όπου θα αποθηκεύονται οι τιμές του εργαλείου tinyDB1.



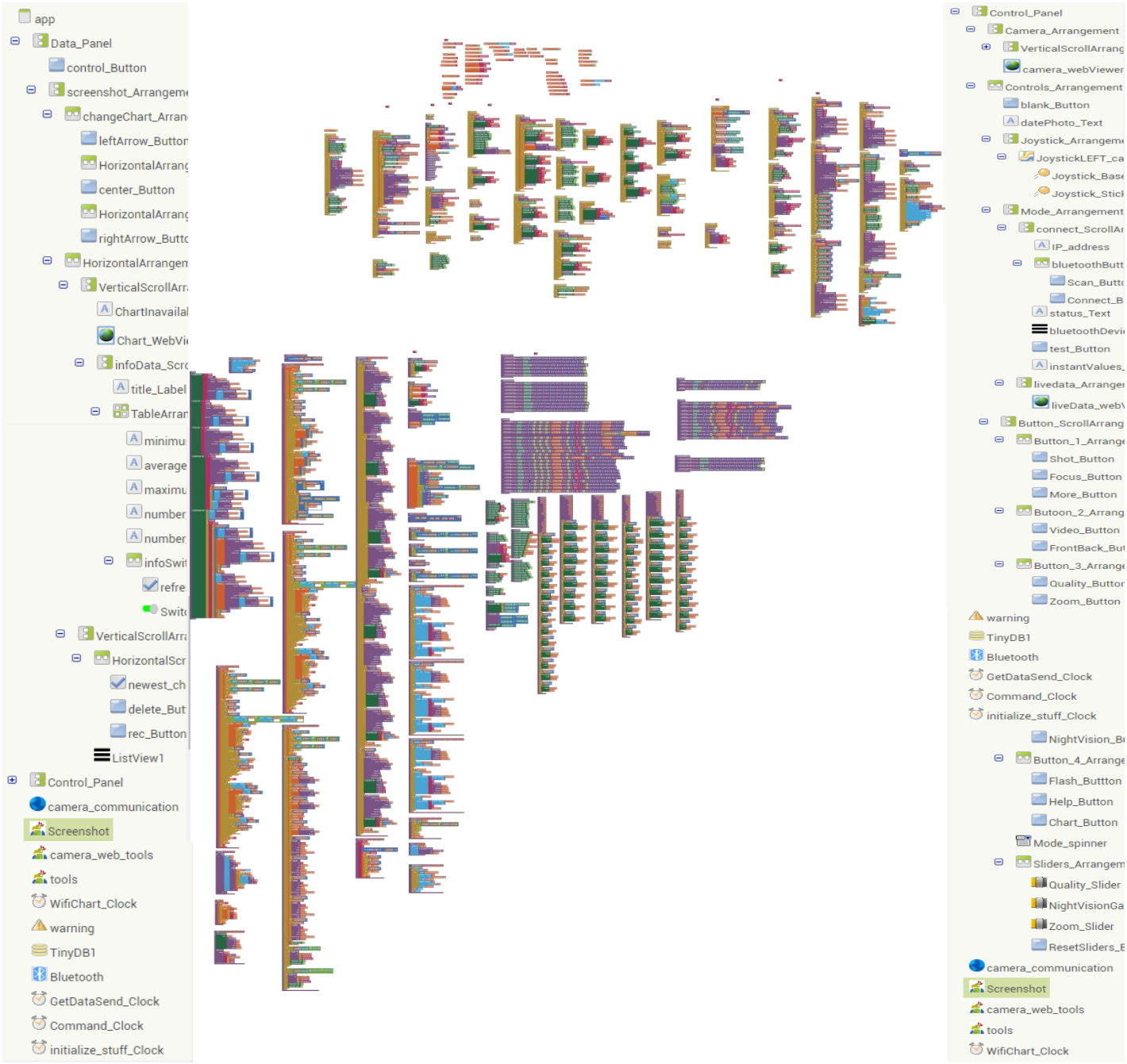
Λογική σκέψη:
Η εικόνα του λογότυπου και το όνομα είναι ορατά από την στιγμή που ανοίξει η οθόνη(screen1). Μόλις περάσει ένα ορισμένο χρονικό περιθώριο (SplashScreen_timer), ανοίγει η δεύτερη οθόνη (app), δηλαδή η βασική οθόνη της εφαρμογής.

Screen: App

Σε αυτή την οθόνη, περιέχεται όλος ο κώδικας της εφαρμογής και για λόγους οργάνωσης και ευκολίας, η περιγραφή του θα είναι πιο συνοπτική. Σε σημεία όπου η λογική του κώδικα είναι προϊόν της δημιουργικότητας των παιδιών, θα υπάρξει παράρτημα όπου θα γίνεται πιο διεξοδική ανάλυση των «blocks κώδικα». Η οθόνη, αν και φαίνεται σαν μία, περιέχει δύο υπό-οθόνες (Control_Panel και Chart_Panel), δηλαδή δύο διαφορετικές διατάξεις εργαλείων, τις οποίες ο χρήστης μπορεί να εναλλάσσει με το πάτημα ενός κουμπιού (Στο Control_Panel υπάρχει το chart_Button και στο Chart_Panel υπάρχει το control_Button).

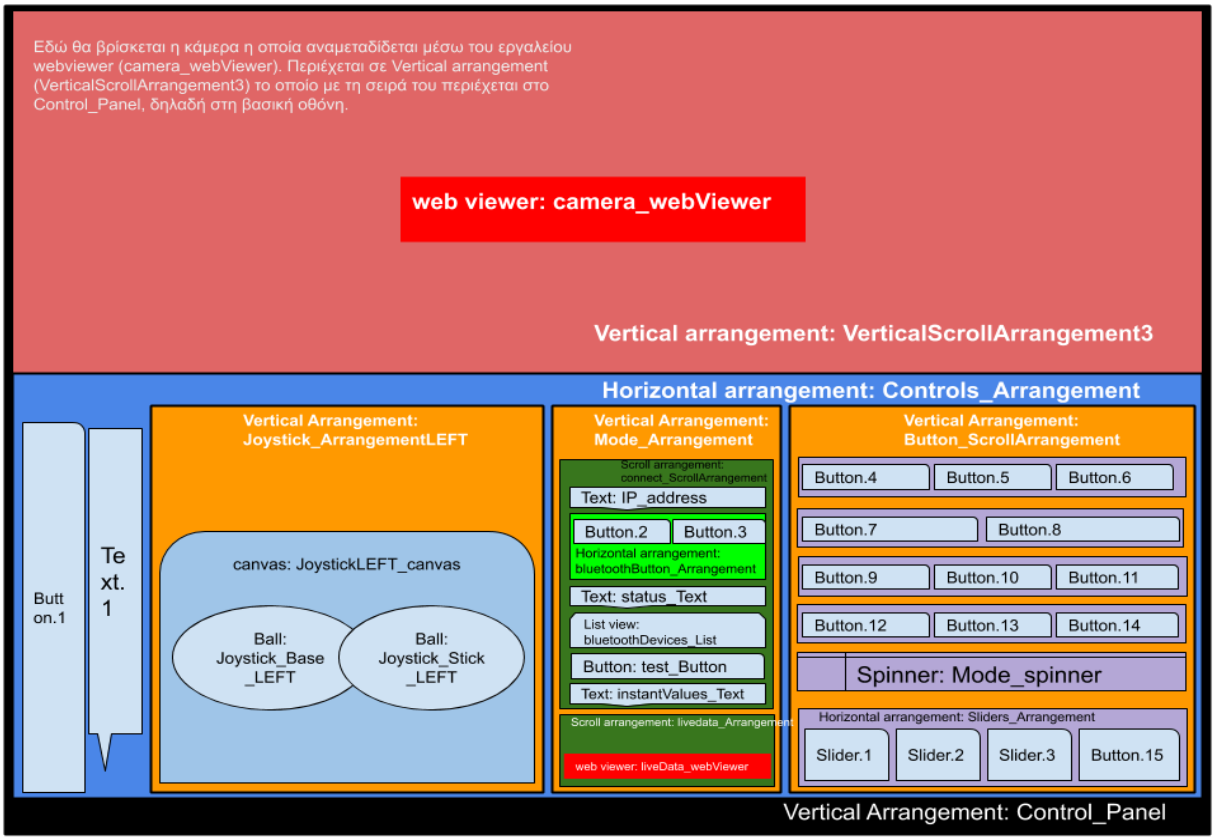


Εικόνα 1ΙΙ Οι δύο υπό-οθόνες



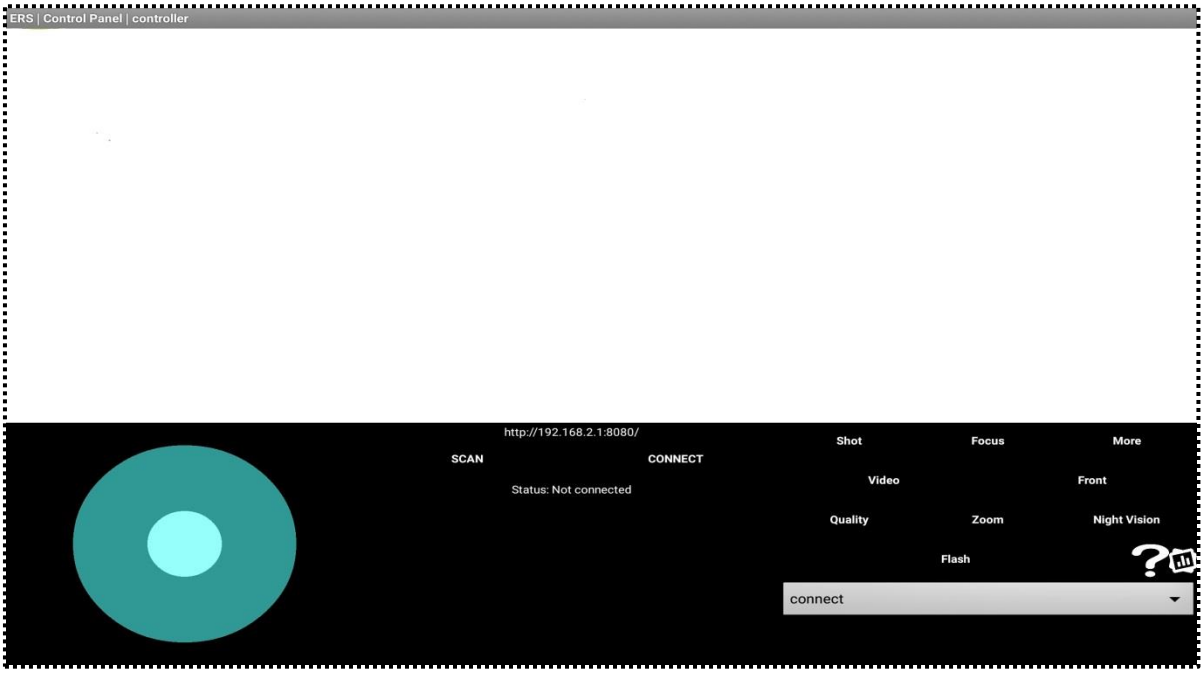
Εικόνα 2 Όλος ο κώδικας της οθόνης *app*

ΜΟΡΦΟΠΟΙΗΣΗ ΤΗΣ ΒΑΣΙΚΗΣ ΟΘΟΝΗΣ CONTROL_PANEL.



Buttons:

1. blank_Button
2. Scan_Button
3. Connect_Button
4. Shot_Button
5. Focus_Button
6. More_Button
7. Video_Button
8. FrontBack_Button
9. Quality_Button
10. Zoom_Button
11. NightVision_Button
12. Flash_Buttton
13. Help_Button
14. Chart_Button
15. ResetSliders_Button

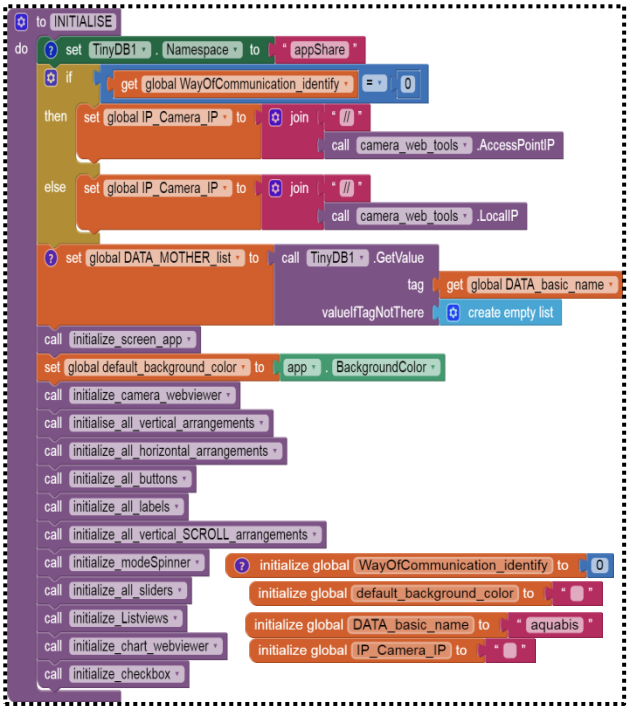


Εικόνα 3 Η υπό-οθόνη Control_Panel.

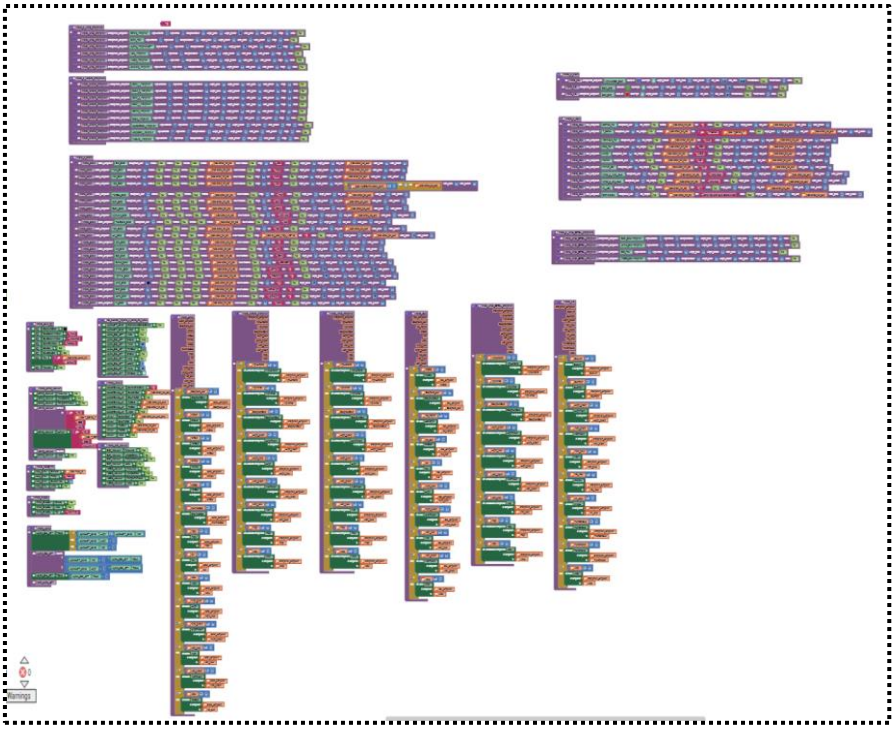
Εικόνα 5 Τμήμα κώδικα για το ξεκίνημα της εφαρμογής.

Εκκίνηση οθόνης

Όταν ανοίξει η δεύτερη οθόνη, με το όνομα app, εκτελούνται με τη σειρά ορισμένες διαδικασίες απαραίτητες, είτε για την λειτουργία των εργαλείων (κουμπιά, λίστες, χρονόμετρα, ετικέτες κ.α.), είτε για την διαμόρφωση της οθόνης (χρώμα παρασκηνίου, χρώμα και μέγεθος γραμμάτων, σχήμα κουμπιών κ.α.). Αναλυτικότερα, πρώτα, τίθεται ο χώρος στον οποίο αποθηκεύονται οι τιμές και καθορίζεται, αν πρόκειται το android να συνδεθεί με την δική του κάμερα (στην περίπτωση δοκιμής, ώστε να μην χρησιμοποιείται δεύτερο android) ή με αυτή κάποιας άλλης συσκευής. Έπειτα, ανακαλούνται, αν υπάρχουν, δεδομένα των τιμών από το σκληρό δίσκο του κινητού, ορίζεται το background της οθόνης και αρχικοποιούνται όλα τα εργαλεία: οθόνη, web viewer, arrangements, buttons, labels, mode spinners, sliders, list views, checkboxes.



Εικόνα 4 Βασική διαδικασία όπου περιέχονται όλες οι ενέργειες για την σωστή αρχικοποίηση της εφαρμογής.

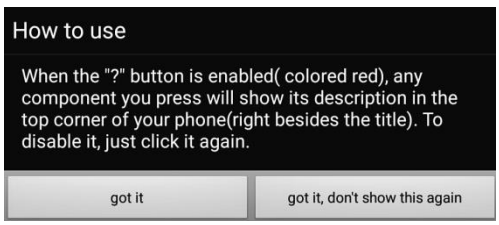


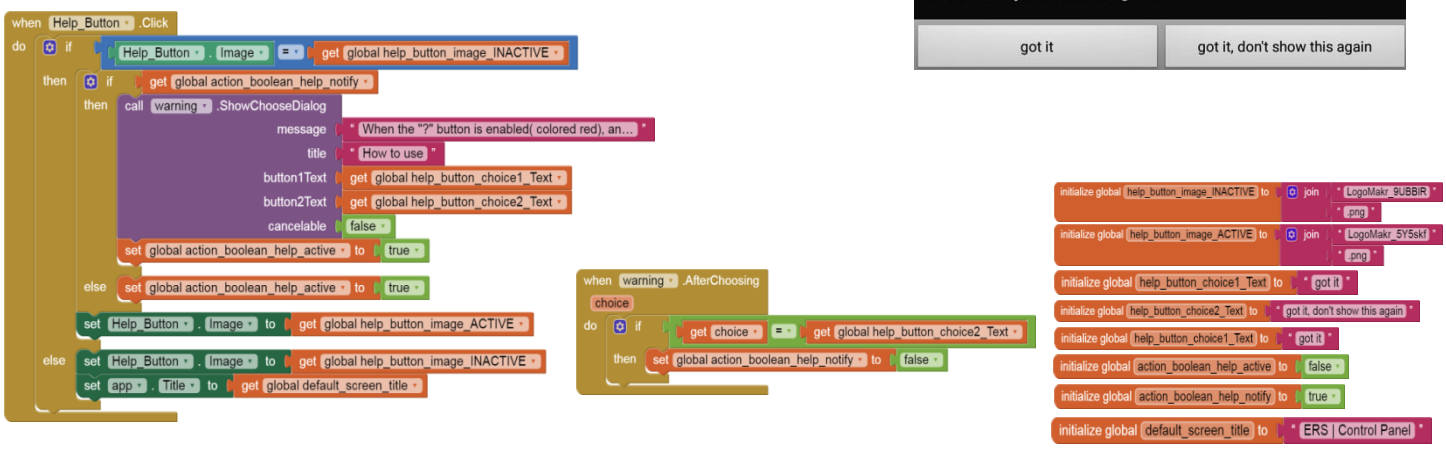
Εικόνα 6 Διαδικασίες αρχικοποίησης όλων των εργαλείων.

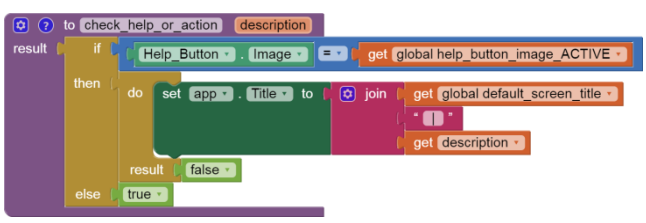
ΠΕΡΙΓΡΑΦΗ ΥΠΟ-ΟΘΟΝΗΣ CONTROL_PANEL

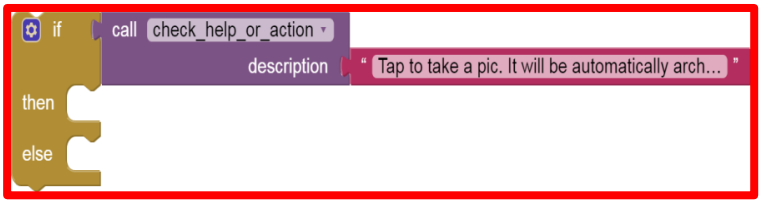
Οδηγίες χρήσης

Η εφαρμογή μπορεί έχει πολλές λειτουργίες, με αποτέλεσμα οποισδήποτε ενδιαφερόμενος να δυσκολευτεί να την αξιοποιήσει πλήρως. Το πρόβλημα αυτό, λύθηκε με την δημιουργία ενός βοηθητικού κουμπιού, το οποίο, αφού ενεργοποιηθεί, δίνει διευκρινίσεις για το ρόλο κάθε εικονικού εργαλείου (κουμπί, joystick, dropdown menu, listview) το οποίο αγγίζει ο χρήστης, δίχως να εκτελείται η αντίστοιχη ενέργεια (εκτός του joystck).





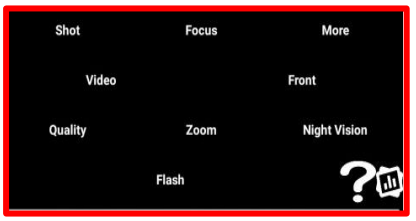




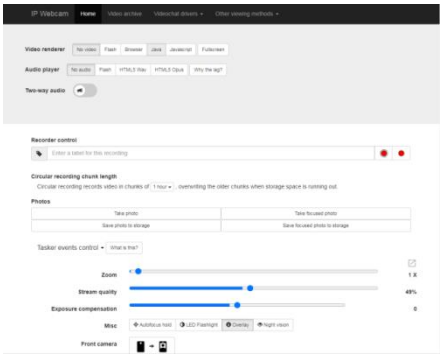
Εικόνα 7 Κώδικας για τις οδηγίες χρήσης

Κάμερα

Η ζωντανή μεταφορά οπτικού υλικού από ένα οποιοδήποτε android, στην συσκευή που κρατά ο χρήστης, στηρίζεται εκτός από το περιβάλλον προγραμματισμού MIT App Inventor και σε μία εμπορική εφαρμογή ανοιχτής χρήσης, την «ip webcam». Κι αυτό γιατί ενώ το MIT App Inventor υποστηρίζει τη χρήση κάμερας, δεν υποστηρίζει τη δυνατότητα εκτεταμένου ελέγχου της. Μάλιστα, η υλοποίηση αυτού του τμήματος κώδικα ήταν χρονοβόρο επιχείρημα, διότι δεν υπήρξε η αντίστοιχη βιβλιογραφία στο διαδίκτυο ούτε για τον τρόπο με τον οποίο αυτό θα μπορούσε να επιτευχθεί.

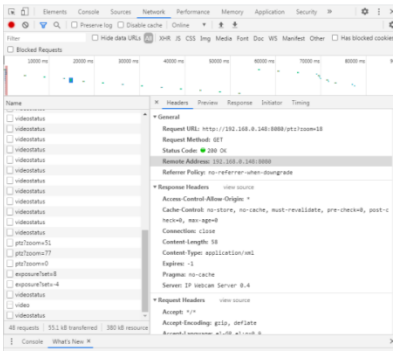


Η εφαρμογή ip webcam, δημιουργεί web server στο android, στον οποίο μπορεί να έχει πρόσβαση, οποιοσδήποτε κατέχει τη διεύθυνση διαδικτυακού πρωτοκόλλου (ip address). Με τη χρήση του εργαλείου web viewer και της επέκτασης TaifunWiFi αναγνωρίζεται η διεύθυνση διαδικτυακού πρωτοκόλλου και επιτυγχάνεται σύνδεση με τον web server που δημιούργησε η ip webcam. Η εφαρμογή αυτή, μετατρέπει το τηλέφωνο σε κάμερα δικτύου με πολλές επιλογές προβολής. Προβάλει την κάμερα σε οποιαδήποτε πλατφόρμα με πρόγραμμα αναπαραγωγής VLC ή πρόγραμμα περιήγησης ιστού. Μπορεί να μεταδίδει βίντεο σε δίκτυο WiFi χωρίς πρόσβαση στο διαδίκτυο. Χρησιμοποιούμε την εφαρμογή ip camera συνδέοντας την κάμερα του κινητού πάνω στον Aquabi αφού επιτρέψουμε στο android τηλέφωνο να λειτουργήσει ως σημείο πρόσβασης WiFi (Hotspot) με το android και το MIT app invertor. για να μπορούμε να κάνουμε zoom in- zoom out, να τραβάμε βίντεο (ακόμη και προγραμματισμένα με επιλογή ώρας και διάρκειας), φωτογραφίες, χρήση flash, νυκτερινή όραση και χρήση άλλων τρόπων επεξεργασίας της εικόνας (modes).

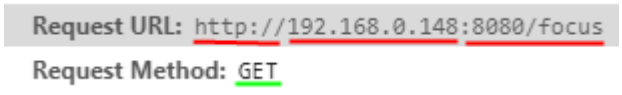


Αποστολή εντολών για τον έλεγχο της κάμερας

Αφού έγινε έρευνα, με το firebug, στην ιστοσελίδα που αντιστοιχούσε ο web server, μπόρεσε να γίνει κατανοητό το πρωτόκολλο επικοινωνίας και καταγράφηκαν οι εντολές (GET) που χρησιμοποιούνταν για τον έλεγχο των ρυθμίσεων της κάμερας. Ουσιαστικά, αξιοποιούμε το οπτικό-ακουστικό υλικό και το γράφημα δεδομένων που μας προσφέρει η ιστοσελίδα της ip camera, αλλά ταυτόχρονα στέλνουμε όποτε θέλουμε τις εντολές για τον έλεγχο της κάμερας, πατώντας κουμπιά της δικής μας εφαρμογής- στα οποία έχουν αντιστοιχηθεί οι κατάλληλες εντολές- και όχι της έτοιμης ιστοσελίδας της ip camera. Στην ουσία χρησιμοποιούμε την πλατφόρμα με την οποία συνδέεται η εφαρμογή ip camera για να έχουμε τα οφέλη ελέγχου της κάμερας. Οι εντολές αποστέλλονται με http requests, μέσω MIT inventor στην πλατφόρμα της εφαρμογής και έτσι έχουμε πρόσβαση στην κάμερα του κινητού που βρίσκεται στον Aquabi για τις χρήσεις που αναφέρθηκαν παραπάνω. Επίσης με τον ίδιο τρόπο έχουμε πρόσβαση στους αισθητήρες του κινητού όπως το GPS, το γυροσκόπιο, το μαγνητόμετρο κ.λ.π.

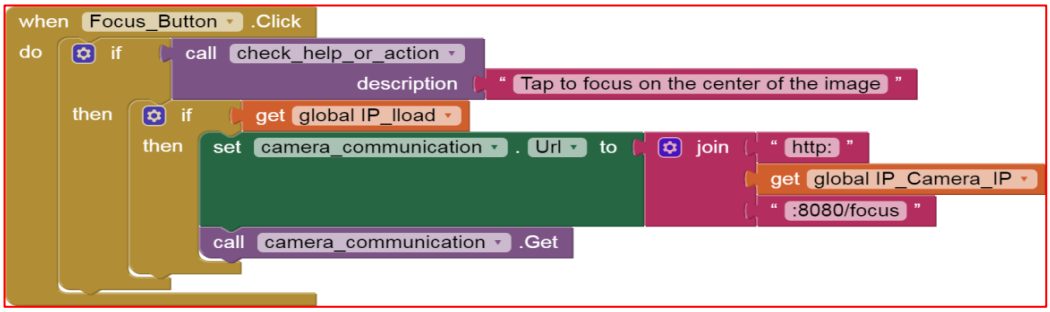


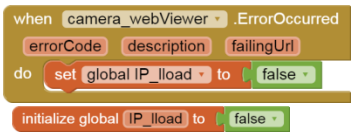
Εικόνα 8 Επέκταση Firebug, ανοίγει πατώντας το F12.



Δείγμα κώδικα

Πρώτα εκτελείται η υπό-συνθήκη που ελέγχει αν πρέπει να δοθεί επεξήγηση (ενεργοποιημένες οι οδηγίες χρήσεις) ή να εκτελεσθεί ο κώδικας που περιέχει. Έπειτα, εκτελείται η υπό-συνθήκη που ελέγχει αν υπάρχει σύνδεση με τον web server. Τέλος, αποστέλλεται μία http request πρωτοκόλλου GET που περιέχει, φυσικά, το κατάλληλο αίτημα.



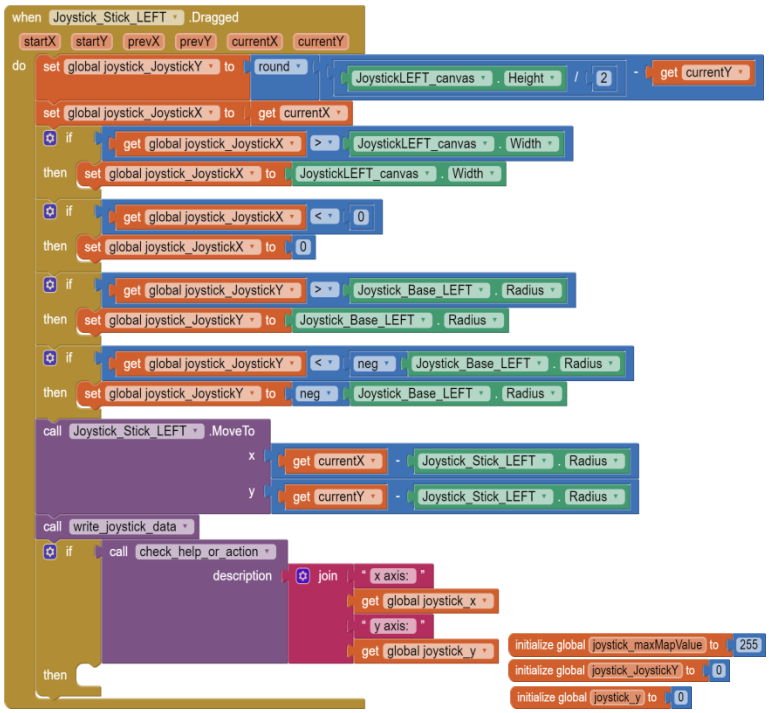


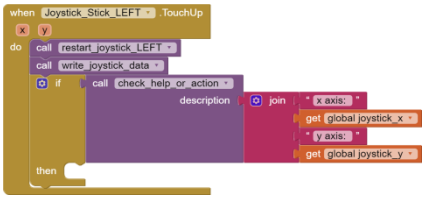
Τα blocks κώδικα που σχετίζονται με τη μεταφορά οπτικού υλικού, βρίσκονται σε πράσινο πλαίσιο στην εικόνα.

Μοχλός-Joystick

Το Joystick δημιουργήθηκε με το εργαλείο Canvas του MIT App Inventor για πιο εύκολο χειρισμό του πλωτού οχήματος. Δημιουργήθηκε δηλαδή με γραφική σχεδίαση, προγραμματισμό και χρήση μαθηματικών πράξεων ώστε να αντιστοιχούν οι τιμές θέσης του Joystick με τις τιμές ταχύτητας και κατεύθυνσης του Aquabi.

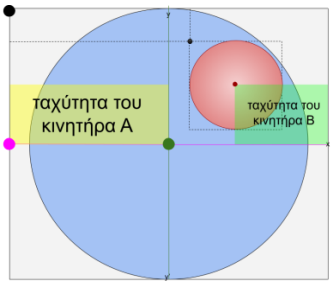
Πώς λειτουργεί





Το canvas είναι ένα ορθοκανονικό σύστημα αξόνων, με μονάδα το ένα pixel, στο οποίο ενσωματώθηκαν δύο δισδιάστατες μπάλες (ball), των οποίων η θέση χαρακτηρίζεται από το ακριανό σημείο, πάνω-αριστερά. Η αρχή του ορθοκανονικού συστήματος βρίσκεται, στην γωνία κάτω-αριστερά.

Η πρώτη δυσδιάστατη μπάλα παίζει διακοσμητικό ρόλο, παριστάνει, δηλαδή, την βάση του μοχλού, οπότε είναι στατική. Η δεύτερη, είναι το κινούμενο τμήμα του joystick, με το οποίο ελέγχεται η κίνηση του aquabi. Δηλαδή, καθορίζονται η ταχύτητα και το ποσοστό διαμοιρασμού αυτής μεταξύ των δύο κινητήρων. Η ταχύτητα ρυθμίζεται μετακινώντας κάθετα, δηλαδή, κατά μήκος του άξονα y, τον μοχλό ενώ το ποσοστό αυτής της ταχύτητας που θα αναπτύξει ο κάθε κινητήρας ορίζεται από την οριζόντια μετακίνηση του μοχλού. Για την κάθετη μετακίνηση, η θέση του μοχλού πρέπει να υπολογίζεται σε σχέση με το κέντρο και όχι την προεπιλεγμένη αρχή του canvas, ώστε η ουδέτερη θέση που αντιστοιχεί σε μηδενική ταχύτητα, να είναι στο κέντρο. Οπότε, $y' = \frac{H}{2} - y$ όπου Η είναι το ύψος του canvas και y είναι η θέση στον άξονα y'y. Για την οριζόντια μετακίνηση, η θέση του μοχλού υπολογίζεται από την προεπιλεγμένη αρχή των αξόνων, δηλαδή, άκρη αριστερά. Άρα, $x' = x$.

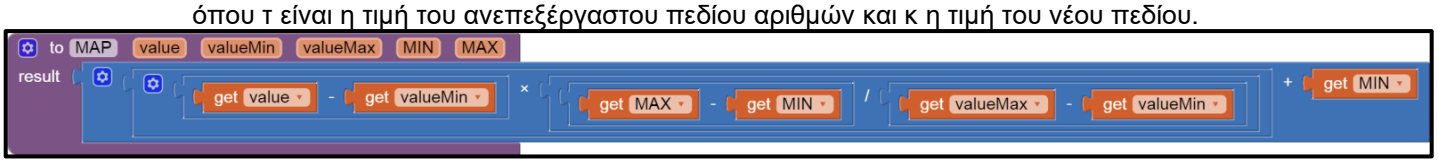


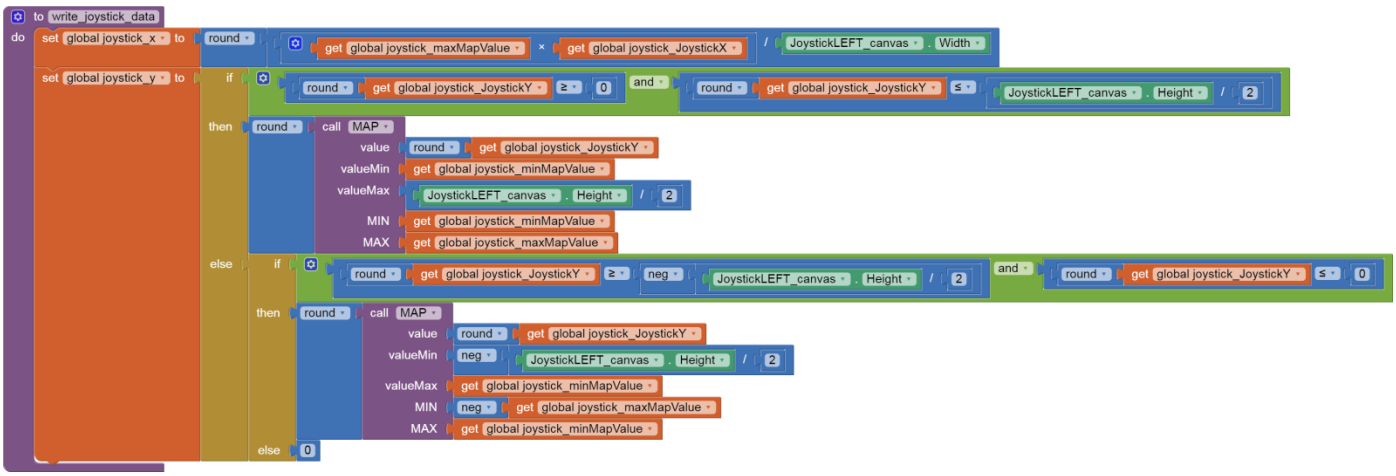
Επίσης, υπάρχουν όρια τα οποία αν ξεπεράσουν οι τιμές των δύο κατηγοριών, παίρνουν, αντίστοιχα, την μέγιστη ή ελάχιστη τιμή.

Τέλος, γίνεται η αντιστοίχιση του αριθμητικού πεδίου που προέκυψε από τις διαστάσεις του canvas, στο επιθυμητό πεδίο που θα αναγνωρίζει το arduino. Αυτό γίνεται με την διαδικασία map η οποία χρησιμοποιεί τον παρακάτω τύπο:

$$\frac{(\tau - \min(\tau)) * (\max(\kappa) - \min(\kappa))}{\max(\tau) - \min(\tau)} + \min(\kappa),$$

όπου τ είναι η τιμή του ανεπεξέργαστου πεδίου αριθμών και κ η τιμή του νέου πεδίου.





Μεταφορά δεδομένων μέσω Bluetooth

Η μεταφορά δεδομένων για τους δύο κινητήρες και από τους αισθητήρες θερμοκρασίας, ph και θολότητας βασίζεται στο bluetooth του κινητού και της επέκτασης του arduino.

Καταρχάς, η σύνδεση πραγμτοποιείται με δύο κουμπιά. Το Scan_Button και το connect_Button. Το scan_Button ελέγχει αν είναι ανοιχτή η λειτουργία bluetooth στη συσκευή και εμφανίζει μια λίστα από ονόματα τυπικών συσκευών bluetooth (High energy bluetooth devices). Το παρατεταμένο άγγιγμα του Scan_Button ακυρώνει την αναζήτηση συσκευών, αποσυνδέει την συσκευή αν είναι συνδεδεμένη με άλλη και εξαφανίζει την λίστα ονομάτων bluetooth συσκευών. Το connect_Button ελέγχει αν έχει επιλέξει ο χρήσης ένα από τα ονόματα που αναγράφονται στην λίστα και συνδέεται σε αυτό. Στην περίπτωση που ηπάρχει ήδη μία συσκευή συνδεδεμένη ,το connect_Button χρησιμοποιείται για να αποσυνδεθεί από αυτή. Μετά την σύνδεση, ένας τρόπος να διαπιστώσει ο χρήστης ότι πράγματι υπάρχει επικοινωνία με την άλλη συσκευή. Μπορεί να χρησιμοποιήσει το test_Button, το οποίο στέλνει εντολή στο arduino να ανταποκριθεί.

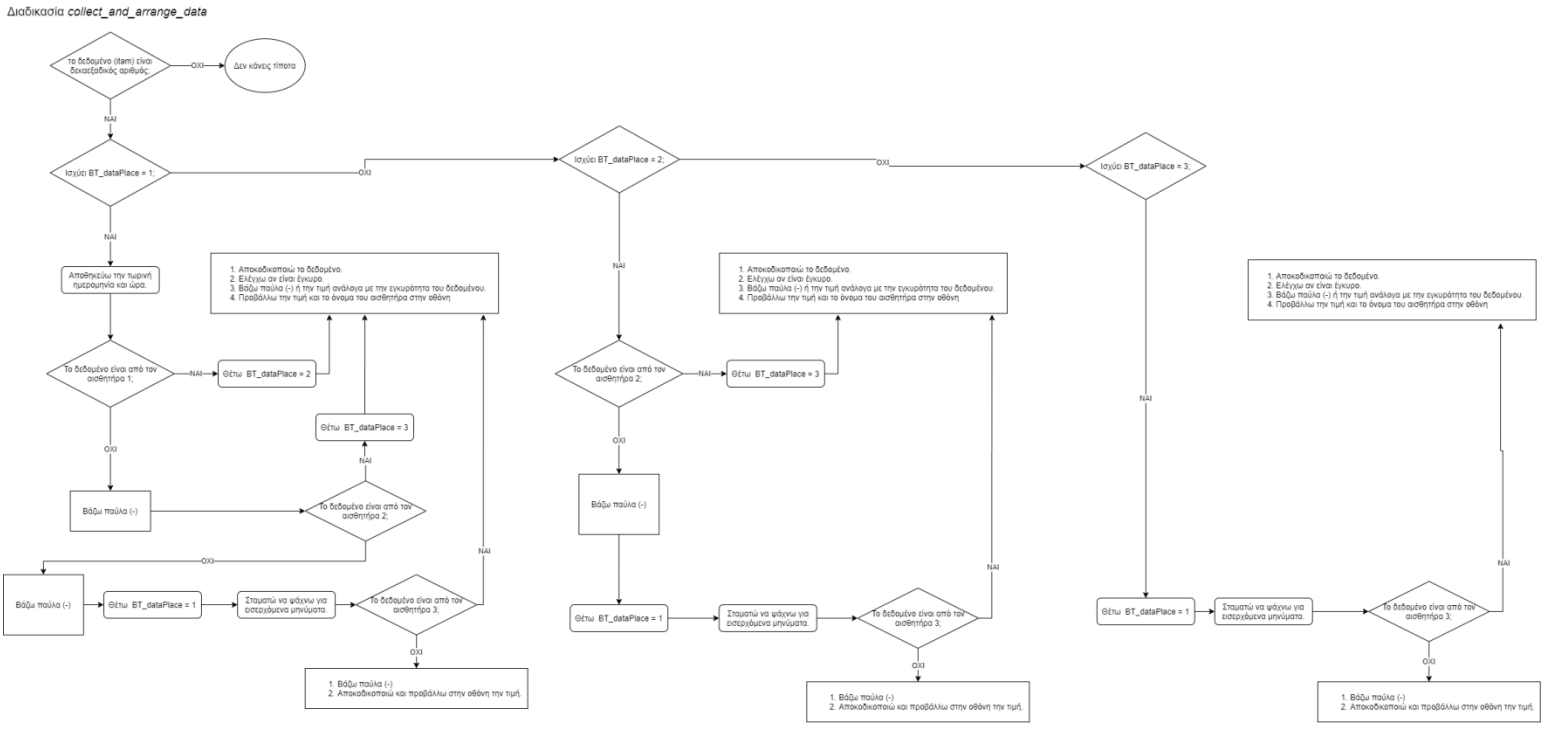
Τα δεδομένα που αποστέλονται προς το arduino είναι οι δύο μεταβλητές joystick_x και joytick_y, όπου έχουν πεδίο ορισμού joystick_x, joytick _y ε[0,255]. Για να τις ξεχωρίζει το arduino, πριν σταλθούν προς αυτό, στην κάθε μία από αυτές, προστίθεται ένας γνωστός αριθμός ο οποίος δίνει διαφορετικό πεδίο ορισμού.

Λίγο περισσότερο περίπλοκη διαδικασία είναι η απολαβή των μετρήσεων από το arduino. Οι μετρήσεις που λαβάνονται πρέπει να οργανωθούν σε λίστες, να ελεγθεί η εγκυρότητά τους και το σημαντικότερο να μπορεί να διαπιστωθεί πότε μία τιμή δεν έφτασε ποτέ στον προορισμό της. Επίσης, πρέπει να τονιστεί πως το πρόγραμμα δεν αναζητά συνεχώς για αποστέλλουσες τιμές και ούτε το arduino στέλνει συνεχώς ό,τι λαμβάνει από τους αισθητήρες του, διότι αυτό οδηγεί σε μεγάλη κατανάλωση ενέργειας. Για να γίνεται η μεταφορά πληροφοριών πιο οικονομικά, το πρόγραμμα στέλνει πρώτα εντολή στο Arduino (BT_data_sendData_Command) για να στείλει τις μετρήσεις από τους αισθητήρες και τότε, μόνο, εκτελεί αναζήτηση για τις τιμές.

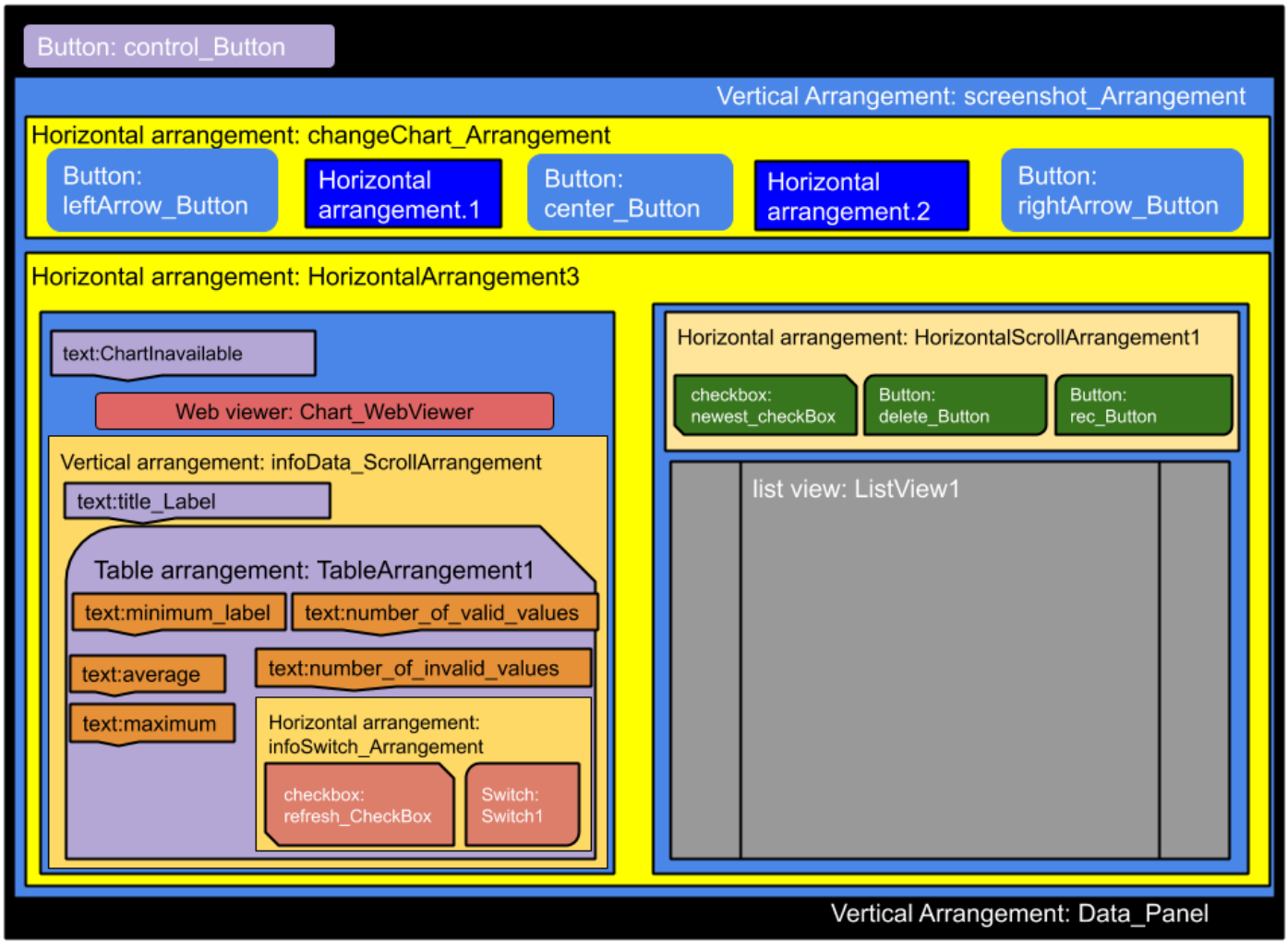
Αν αποστέλλονται δεδομένα, το πρόγραμμα της εφαρμογής συλλέγει το πακέτο δεδομένων και ελέγχει πρώτα αν είναι η προγραμματισμένη ανταπόκριση του arduino στην εντολή του test_Button (test_Command) ή στην εντολή BT_data_sendData_Command και προβάλει στην οθόνη την ανάλογη ενημέρωση στο instantValues_Text, στην οθόνη του android ειδάλως προβάλει τις έγκυρες τιμές που λαμβάνει. Έπειτα, στην περίπτωση που ο χρήστης έχει ενεργοποιήσει την καταγραφή δεδομένων από το rec_Button, συλλέγονται τα δεδομένα και αποθυκεύονται στην λίστα DATA_MOTHER_list.

Συλλογή των δεδομένων

Η συλλογή δεδομένων έχει τη συγκεκριμένη δομή, ώστε να μπορεί να διαπιστωθεί πότε δεν λαμβάνεται μία τιμή. Πρώτα από όλα, τα δεδομένα εισέρχονται στην συσκευή με μία χαρακτηριστική επεξεργασία. Πολλαπλασιάζονται με το εκατό για να κρατήσουν τα τρία δεκαδικά τους ψηφία, αθροίζονται με γνωστό αριθμό που χαακτηρίζει σε ποιο αισθητήρα ανήκουν και μετατρέπονται στην μορφή του δεκαεξαδικού συστήματος αρίθμησης ώστε να τα ξεχωρίζει το πρόγραμμα από τυχόν παρεμβολές άλλων συσκευών που λειτουργούν στο ίδιο φάσμα ραδιοσυχνοτήτων.



ΜΟΡΦΟΠΟΙΗΣΗ ΤΗΣ ΒΑΣΙΚΗΣ ΟΘΟΝΗΣ CONTROL_PANEL.



Arrangements:

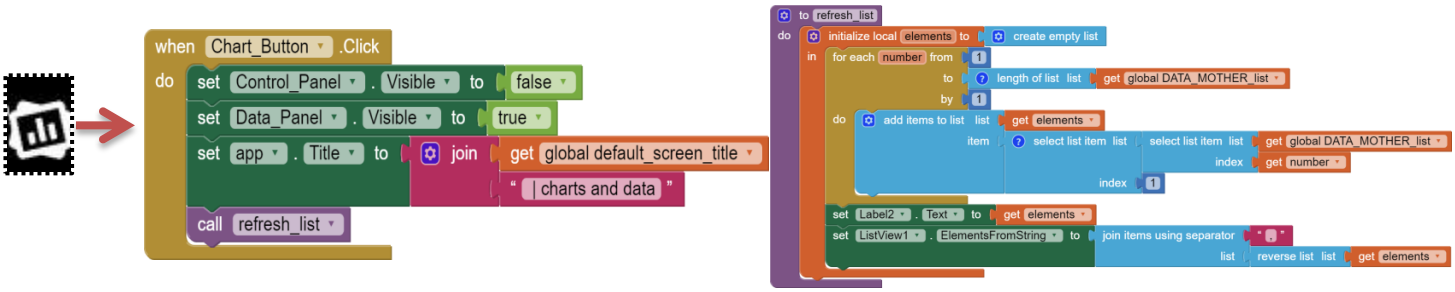
1. HorizontalArrangement2
2. HorizontalArrangement1 ← Ναι, πράγματι είναι ανάποδα η αρίθμηση.

ΠΕΡΙΓΡΑΦΗ ΥΠΟ-ΟΘΟΝΗΣ CONTROL_PANEL

Οι μετρήσεις από τους αισθητήρες του Arduino, καταγράφονται και αποθηκεύονται κατά ημέρα και ώρα, σε μια βάση δεδομένων και βάσει αυτών δημιουργούνται γραφήματα που, μετά από εντολή καταγραφής από τον χρήστη, αποθηκεύονται αυτόματα και διαγράφονται από το χρήστη. Δημιουργείται ένα γράφημα που περιλαμβάνει όλες τις μετρήσιμες παραμέτρους αλλά και ξεχωριστό γράφημα για κάθε μία από αυτές, δηλαδή τη θερμοκρασία, τη θολερότητα και το PH. Συνδυάστηκε δωρεάν επέκταση της Google για τη δημιουργία των γραφημάτων.
Όλες οι μετρήσεις αποθηκεύονται αυτόματα στο android και είναι διαθέσιμες για επεξεργασία και έκδοση συμπερασμάτων.
Επίσης όταν γίνεται η καταγραφή των τιμών πραγματοποιείται και κάποιας μορφής επεξεργασία. Μπορούμε να ξέρουμε πόσες τιμές πήραμε, εκ των οποίων, ποιες είναι άκυρες ή δεν λήφθηκαν, ποια είναι η μέγιστη και η ελάχιστη τιμή και ποιος ο μέσος όρος της χρονική περιόδου που καταγράφαμε το συγκεκριμένο σύνολο τιμών.
Οι μετρήσεις από τους αισθητήρες του Arduino, καταγράφονται και αποθηκεύονται κατά ημέρα και ώρα, σε μια βάση δεδομένων και βάση αυτών δημιουργούνται γραφήματα που αποθηκεύονται με επιλογή του χρήστη. Δημιουργείται ένα γράφημα που περιλαμβάνει όλες τις μετρήσιμες παραμέτρους αλλά και ξεχωριστά γραφήματα για κάθε μία από αυτές όπως τη θερμοκρασία, τη θολερότητα και το PH. Η εφαρμογή αυτή δημιουργήθηκε στο MIT App Inventor σε συνδυασμό με μια δωρεάν επέκταση της Google. Οι μετρήσεις αφού αποθηκευτούν στη Βάση Δεδομένων στο android αποστέλλονται με http requests και δημιουργούνται τα γραφήματα.
Μπορούμε να συνεργαστούμε με το Φορέα Διαχείρισης Λίμνης Κερκίνης και να μοιραστούμε τις μετρήσεις που θα συλλέξουμε.

Δημιουργία και επεξεργασία γραφημάτων

Αρχικά για να μεταφερθούμε σε αυτή την υπό-οθόνη, χρειάζετα να πατήσουμε το κουμπί chart_Button.



Το είδος των γραφημάτων αλλάζει όταν πατάμε το αριστερό ή δεξί βέλος, όπου αναλόγως, προστίθεται ή αφαιρείται μία μονάδα από μία μεταβλητή η οποία χαρακτηρίζει την σειρά των γραφημάτων. Μετά την ενέργειά μας, η διαδικασία chartType_picker ανασύρει δεδομένα από την λίστα DATA_MOTHER_list τα οποία αντιστοιχούν στον συγκεκριμένο αισθητήρα βάσει του index της λίστας, το οποίο καθορίζει η μεταβλητή Chart_file_list_data_select.
Η διαδικασία, ελέγχει αν μπορεί να ανασύρει τα επιθυμητά δεδομένα, αναγνωρίζει τον τρόπο που πρέπει να προβληθούν και μέσω της δευτερεύουσας διαδικασίας ShowChart, αξιοποιεί τα html αρχεία της google, chart.html και grafik.html και δημιουργεί γράφημα με την χρήση της τριτεύουσας διαδικασίας MakeWebString, στέλνοντας τον τρόπο αναπαράστασής τους, τον τίτλο, την λίστα που περιέχει τα συγκεκριμένα δεδομένα και την αντίστοιχη λίστα που περιέχει τις ημερομηνίες.

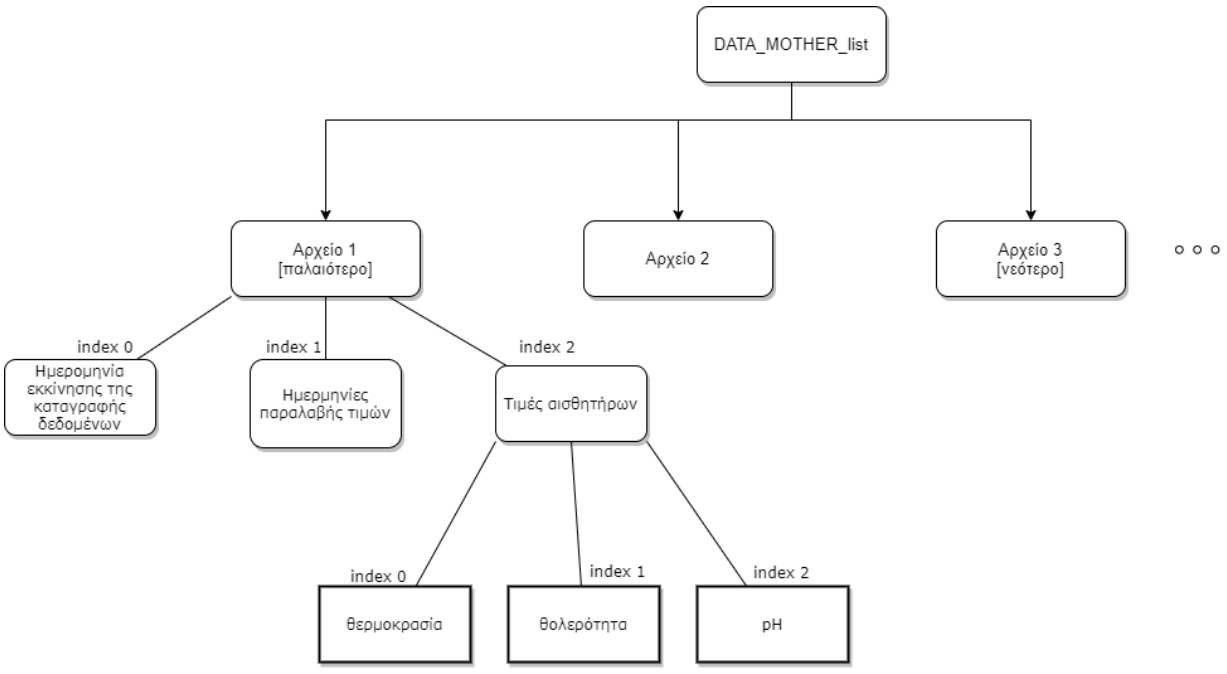
Σε συνέχεια της διαδικασίας chartType_picker, ακολουθεί η διαδικασία graph_data η οποία είναι υπεύθυνη για την επεξεργασία των τιμών και την προβολή, κάτω του γραφήματος, επιπλέον δεδομένων (ακρότατα, μέση τιμή, έγκυρες τιμές εκ του συνόλου και άκυρες τιμές). Μέσα σε αυτή τη διαδικασία χρησιμοποιούνται τρεις δευτερεύουσες διαδικασίες(minimum, maximum και average_or_number_of_valid_values), εκ των οποίων οι δύο (minimum και maximum) έχουν παρόμοια λειτουργία. Αρχικά, η κάθε μία από τις δύο, ελέγχει αν μπορεί να ανασύρει τα δεδομένα, όπως ακριβώς συμβαίνει και στην διαδικασία chartType_picker. Μετά, ελέγχει την λίστα των δεδομένων για να βρει έναν πραγματικό αριθμό, το οποίο καθιστά ελάχιστο ή μέγιστο (ανάλογα με την διαδικασία) και αμέσως, κάθε τιμή της λίστας δεδομένων, συγκρίνεται με το ελάχιστο ή μέγιστο και στην περίπτωση που είναι ανάλογα μικρότερη ή μεγαλύτερη παίρνει την θέση του και συνεχίζεται η σύγκριση με το υπόλοιπο της λίστας.

Το αποτέλεσμα της σύγκρισης θα είναι το επιθυμητό ακρότατο του γραφήματος. Φυσικά, η διαδικασία είναι ικανή να παραλείψει τις άκυρες ή απούσες τιμές (-). Η εύρεση της αρχικής ελάχιστης και σύγκριση των τιμών, εμφανίζονται δύο φορές σε κάθε διακασία διότι καλύπτουν την περίπτωση ανάλυσης μεμονωμένου αισθητήρα και την περίπτωση ανάλυσης όλων των αισθητήρων μαζί, στις οποίες διαφέρει η λίστα δεδομένων που χρησιμοποιείται. Επίσης, αν δεν υπάρχουν έγκυρες τιμές του αποτέλεσμα είναι μία παύλα(-). Όσον αφορά την διαδικασία average_or_number_of_valid_values, η λειτουργία λίγο διαφέρει από τις άλλες δύο και έχει ένα ιδιαίτερο χαρακτηριστικό, είναι αναδρομική διαδικασία (recursive function), δηλαδή μέσα στη διαδικασία, καλείται να εκτελεθεί η ίδια η διαδικασία.

Πρώτα, εκτελείται ο ίδιος έλεγχος γα την χρήση των δεδομένων όπως και στις προηγούμενες διαδικασίες. Μετά, χρησιμοποιούνται δύο μεταβλητές για τον έλεγχο της λίστας δεδομένων. Κατά τη διάρκεια του ελέγχου, κάθε δεδομένο της επιλεγμένης λίστας δεδομένων, αν είναι πραγματικός αριθμός, προστίθεται ως μονάδα στην μεταβλητή number_of_valid_values και ως αριθμιτική τιμή στην μεταβλητή sum. Έτσι τελικά, καθορίζεται το σύνολο των τιμών και το άθροισμά τους, από τα οποία θα προκύψει με μία διαίρεση, ο μέσος όρος. Το τελευταίο βήμα σε αυτή τη διαδικασία, είναι να διαπιστωθεί αν ζητείται το σύνολο των τιμών ή ο μέσος όρος τους, και να δοθεί το αντίστοιχο αποτέλεσμα. Αν ζητηθεί ο μέσος όρος, επειδή είναι πιθανό το άθροισμα να μην είναι αριθμός (-), πρέπει να γίνει ένας τελευταίος έλεγχος όπου μη αριθμοί να μετατρέπονται σε 0 ώστε να μπορεί να γίνει η απαραίτητη αριθμητική πράξη. Συνολικά, με αυτές τις τρεις δευτερεύουσες διαδικασίες προβάλονται τα επιπλέον δεδομένα από τα γραφήματα.

Το switch1 καθορίζει αν θα χρησιμοποιηθεί, για την αναπαράσταση τον δεδομένων, το αρχείο chart.html ή grafik.html τα οποία κάνουν το θέμα του γραφήματος μαύρο ή άσπρο αντίστοιχα. Παράλληλα ανανεώνεται και το γράφημα.
Η lisrtView1 καθορίζει το του index της λίστας στο οποίο αντιστοιχούν τα δεδομένα κάθε αισθητήρα. Επίσης, ρυθμίζει τα κουμπιά newest_checkBox και refresh_CheckBox και ανανεώνει το γράφημα και τα δεδομένα που το συνοδεύουν. Σε συνδιασμό με το delete_Button, διαγράφεται η λίστα δεδομένων που ανήκει στο index της λίστας DATA_MOTHER_list, ανανεώνεται η λίστα και ορίζεται το index ως 1, δηλαδή προεπιλέγεται το νεότερο αρχείο. Το newest_checkBox ορίζει, επίσης, το index της λίστας ως 1 και το rec_Button, είτε εισάγει νέα υπο-λίστα δεδομένων στην DATA_MOTHER_list, σε καινούριο index, είτε στην περίπτωση που δεν υπάρχει η λίστα DATA_MOTHER_list, την δημιουργεί.

Διάγραμμα δομής της λίστας DATA_MOTHER_list



Arduino

#include <SoftwareSerial.h>	**Βιβλιοθήκη για την μεταφορά δεδομένων μέσω BT.**
#include <DallasTemperature.h>	Βιβλιοθήκες για την μέτρηση της θερμοκρασίας από τον αισθητήρα μοντέλου DS18B20
#include <OneWire.h>	
#define ONE_WIRE_BUS A0	Θέτω όπου εμφανίζεται ο τίτλος την τιμή A0
SoftwareSerial Arduino(9, 10);	Ορίζω αντικείμενα και τις ιδιότητές τους, ώστε να μπορέσω να αξιοποιήσω τις παραπάνω βιβλιοθήκες
OneWire oneWire(ONE_WIRE_BUS);	
DallasTemperature sensors(&oneWire);	
#define power 13	Θέτω όπου εμφανίζεται power την τιμή 13
#define sensor3 A1	Θέτω όπου εμφανίζεται sensor3 την τιμή A1
#define sensor2 A2	Θέτω όπου εμφανίζεται sensor2 την τιμή A2
String answer;	Το σύνολο των απεσταλμένων δεδομένων
bool send_Command = false;	Πληροφορεί αν έχει σταλεί εντολή από το android για να στείλει τις μετρήσεις από τους αισθητήρες
int data;	Η μετατροπή του συνόλου δεδομένων σε αριθμό για πράξεις
float sensorData1;	Μετρήσεις από τον αισθητήρα θερμοκρασίας
float sensorData2;	Μετρήσεις από τον αισθητήρα θολερότητας
float sensorData3;	Μετρήσεις από τον αισθητήρα pH
int RecievedValue2 = 0;	Η τιμή που αντιστοιχεί στο joystick_x
int RecievedValue3 = 0;	Η τιμή που αντιστοιχεί στο joytick_y
unsigned long int avgValue;	Μεταβλητές για την εύρεση του pH
int buf[10], temp;	
int enA = 6;	Έξοδος για την ταχύτητα του αριστερού κινητήρα
int in1 = 7;	Μεταβλητές που καθορίζουν την φορά του αριστερού κινητήρα
int in2 = 5;	
int enB = 3;	Έξοδος για την ταχύτητα του δεξιού κινητήρα
int in3 = 2;	Μεταβλητές που καθορίζουν την φορά του αριστερού κινητήρα
int in4 = 4;	
int Speed;	Μέγιστη ταχύτητα η οποία θα μοιραστεί στους δύο κινητήρες
void setup() {	**Βασική διαδικασία εκκίνησης**
Serial.begin(9600);	Εκκίνηση επικοινωνίας μέσω του bluetooth, των αισθητήρων και του monitor
Arduino.begin(9600);	
sensors.begin();	
pinMode(power, OUTPUT);	Ορίζω αυτά τα pin ως εξόδους
pinMode(enA, OUTPUT);	
pinMode(enB, OUTPUT);	
pinMode(in1, OUTPUT);	
pinMode(in2, OUTPUT);	
pinMode(in3, OUTPUT);	
pinMode(in4, OUTPUT);	
digitalWrite(power, LOW);	Μηδενίζω, αρχικά, την τάση που λαμβάνουν οι αισθητήρες
}	

void loop() {	**Βασική διαδικασία βρόγχου**
get_SpeedDataAndCommands_Phone ();	Διαδικασία η οποία λαμβάνει και αναγνωρίζει τα δεδομένα για τους κινητήρες και τις εντολές
if (send_Command) {	Αν έχει ληφθεί εντολή αποστολής δεδομένων,
digitalWrite(power, HIGH);	Ανοίγω την τάση για τους αισθητήρες,

delay(40);	Περιμένω 40 χιλιοστά του δευτερολέπτου,
get_sensorData_Raw_And_Send_Phone();	Παίρνω τις μετρήσεις από τους αισθητήρες θερμοκρασίας, θολερότητας και turbidity
delay(10);	Περιμένω 10 χιλιοστά του δευτερολέπτου
digitalWrite(power, LOW); }	Μηδενίζω την τάση,
send_ControllerData_SpeedControllers();	Στέλνω τα δεδομένα για τους κινητήρες, στους κινητήρες
delay(100); }	Περιμένω 1/10 του λεπτού
void get_SpeedDataAndCommands_Phone () {	**Διαδικασία για την απολαβή δεδομένων από τη συσκευή android**
while (Arduino.available()) {	Όσο ανιχνεύω εισερχόμενα δεδομένα,
delay(10);	Περιμένω 10 χιλιοστά του δευτερολέπτου
if (Arduino.available() > 0) {	Αν τα δεδομένα αντιστοιχούν σε κάποιο χαρακτήρα
answer += char(Arduino.read());	Προσθέτω τον χαρακτήρα στην μεταβλητή για να φτιάξω μία λέξη
//Serial.print(answer);	Προβάλλω στο monitor την λέξη
}	
}	
if (answer != "") {	Αν η λέξη που έφτιαξα έχει περιεχόμενο,
data = answer.toInt();	Μετατρέπω την λέξη αυτή σε αριθμό (εφόσον γνωρίζω πως αυτό που λαμβάνω θα είναι πάντα στη μορφή αριθμού)
//Serial.println(data);	Προβάλλω στο monitor τον αριθμό
if (data == 5) {	Αν ο αριθμός ισούται με 5, σημαίνει πως ο χρήστης τεστάρει την επικοινωνία, οπότε
Arduino.println("fine"); **//Serial.println("Fine!");**	προβάλλω στο monitor την λέξη «Fine» και στέλνω στο arduino την λέξη «fine».
}	
delay(20);	Περιμένω 20 χιλιοστά του δευτερολέπτου
if (data == 1000) {	Αν ο αριθμός ισούται με 1000, σημαίνει πως πρέπει να στείλω τις μετρήσεις από τους αισθητήρες, οπότε
send_Command = true;	το σηματοδοτώ ρυθμίζοντας την μεταβλητή send_Command
//Serial.print(send_Command);	Προβάλλω στο monitor τον αριθμό που αποτελεί την εντολή send_command, «1000».
Serial.println("ok");	Προβάλλω στο monitor τη λέξη «ok»
Arduino.println("ok");	Στέλνω στο arduino την λέξη «ok».
}	
else {	Αλλιώς, σημαίνει πως δεν χρειάζεται να στείλω τις μετρήσεις από τους αισθητήρες, οπότε
send_Command = false;	το σηματοδοτώ ρυθμίζοντας την μεταβλητή send_Command
//Serial.println("nope");	προβάλλω τη λέξη «nope»
delay(10);	και περιμένω 10 χιλιοστά του δευτερολέπτου.
}	
if (data >= 2000 && data <= 2255) {	Αν ο αριθμός είναι μεταξύ του 2000 και του 2255, σημαίνει πως είναι η μεταβλητή που αντιστοιχεί στο joystick_x στη συσκευή android, όμως χρειάζεται πρώτα αποκωδικοποίηση, επομένως
RecievedValue2 = data - 2000;	αφαιρώ 2000 από την τιμή,
delay(20);	και περιμένω 20 χιλιοστά του δευτερολέπτου.
}	
else if (data >= 3000 && data <= 3510) {	Αλλιώς, αν η τιμή είναι μεταξύ του 3000 και του 3510, σημαίνει πως είναι η μεταβλητή που αντιστοιχεί στο joystick_y στη συσκευή android, όμως χρειάζεται πρώτα αποκωδικοποίηση, επομένως

RecievedValue3 = data - 3255;	αφαιρώ το 3255 (ώστε να θέσω μηδενική τιμή, το μέσο αυτού του πεδίου)
delay(20);	και περιμένω 20 χιλιοστά του δευτερολέπτου.
}	
answer = "";	Διαγράφω την απάντηση.
}	
}	
void get_sensorData_Raw_And_Send_Phone() {	**Διαδικασία για την απολαβή των μετρήσεων και την αποστολή τους στη συσκευή android**
do {	Παίρνω 5 συνεχόμενες μετρήσεις (με 10 χιλιοστά του δευτερολέπτου διαφορά) από τον αισθητήρα θερμοκρασίας, τις αθροίζω, στην μεταβλητή sensorData1 και βγάζω τον μέσο όρο. Όσο ο μέσος όρος δεν είναι αριθμός (δηλαδή έγινε σφάλμα), επαναλαμβάνω αυτή τη διαδικασία.
sensorData1 = 0;	
for (int i = 1; i <= 5; i++) {	
sensors.requestTemperatures();	
sensorData1 = sensorData1 + sensors.getTempCByIndex(0);	
delay(10);	
}	
sensorData1 = sensorData1 / 5.0;	
} while (isnan(sensorData1));	
sensorData1 = sensorData1 + 1000.0; //temperature	Προσθέτω σε αυτήν γνωστό αριθμό, 1000, ώστε να αλλάξει το πεδίο ορισμού του αισθητήρα και το πρόγραμμα android να την ξεχωρίζει.
Serial.print(" --> "); Serial.println (sensorData1 - 1000.0);	Προβάλλω την μέτρηση από τον αισθητήρα θερμοκρασίας στο monitor
delay(30);	Περιμένω 30 χιλιοστά του δευτερολέπτου.
Arduino.print (String(sensorData1 * 100.0, HEX));	Πολλαπλασιάζω την μέτρηση με το 100 ώστε να κρατήσω 2 δεκαδικά ψηφία, την μετατρέπω σε λέξη και την κωδικοποιώ στο δεκαεξαδικό σύστημα.
do {	Παίρνω 5 συνεχόμενες μετρήσεις (με 10 χιλιοστά του δευτερολέπτου διαφορά) από τον αισθητήρα θολερότητας, τις αθροίζω, στην μεταβλητή sensorData2 και βγάζω τον μέσο όρο. Όσο ο μέσος όρος δεν είναι αριθμός (δηλαδή έγινε σφάλμα), επαναλαμβάνω αυτή τη διαδικασία.
sensorData2 = 0;	
for (int i = 1; i <= 10; i++) {	
sensorData2 = sensorData2 + analogRead(sensor2);	
delay(50);	
}	
sensorData2 = sensorData2 / 10.0;	
} while (isnan(sensorData2));	
delay(500);	Περιμένω μισό δευτερόλεπτο.
sensorData2 = sensorData2 * (5.0 / 1024.0) + 2000.0; //turbidity	Ορίζω ως μονάδα μέτρησης της θολερότητας το 1V (volt) και προσθέτω 2000, ώστε να αλλάξει το πεδίο ορισμού του αισθητήρα και το πρόγραμμα android να την ξεχωρίζει.
Serial.print(" !--> "); Serial.println (sensorData2-2000.0);	Προβάλλω την μέτρηση από τον αισθητήρα θολερότητας στο monitor.
Arduino.print (String(sensorData2 * 100.0, HEX));	Πολλαπλασιάζω την μέτρηση με το 100 ώστε να κρατήσω 2 δεκαδικά ψηφία, την μετατρέπω σε λέξη και την κωδικοποιώ στο δεκαεξαδικό σύστημα.
delay(500);	Περιμένω μισό δευτερόλεπτο.
sensorData3 = find_pH(sensor3) + 3000.0; //pH	Εκτελώ τη διαδικασία find_pH για να πάρω μέτρηση από τον αισθητήρα pH και προσθέτω σε αυτήν γνωστό αριθμό, 1000, ώστε να αλλάξει το πεδίο ορισμού του αισθητήρα και το

	πρόγραμμα android να την ξεχωρίζει.
Serial.print(" ->-> "); Serial.println (sensorData3 - 3000.0);	Προβάλλω την μέτρηση από τον αισθητήρα pH στο monitor.
Arduino.print (String(sensorData3 * 100.0, HEX));	
delay(400);	Περιμένω 4/10 του δευτερολέπτου.
}	
void send_ControllerData_SpeedControllers() {	**Διαδικασία για τη κίνηση των κινητήρων**
int motorA;	Τιμή ταχύτητας αριστερού κινητήρα.
int motorB;	Τιμή ταχύτητα δεξιού κινητήρα.
boolean rotation;	Αν η κίνηση γίνεται προς τα πίσω (false) ή προς τα μπροστά (true).
if (isnan(RecievedValue2)) {	Αν η τιμή δεν είναι αριθμός (δηλαδή, έχει γίνει κάποιο σφάλμα),
RecievedValue2 = 0;	Την θέτω 0
}	
if (isnan(RecievedValue3)) {	Αν η τιμή δεν είναι αριθμός (δηλαδή, έχει γίνει κάποιο σφάλμα),
RecievedValue3 = 0;	Την θέτω 0
}	
if (RecievedValue3 <= 0) {	Αν το joystick_y είναι μη θετικός αριθμός,
Speed = RecievedValue3 * (-1);	θέτω την γενική ταχύτητα με τον αντίθετό του,
digitalWrite(in1, HIGH);	ρυθμίζω τους κινητήρες έτσι ώστε να κινούν το όχημα προς τα πίσω και
digitalWrite(in2, LOW);	
digitalWrite(in3, HIGH);	
digitalWrite(in4, LOW);	
rotation = false;	Ορίζω πως η κίνηση γίνεται προς τα πίσω.
}	
if (RecievedValue3 > 0) {	Αν το joystick_y είναι θετικός αριθμός,
Speed = RecievedValue3;	θέτω την γενική ταχύτητα ίση με αυτόν,
digitalWrite(in1, LOW);	ρυθμίζω τους κινητήρες έτσι ώστε να κινούν το όχημα προς τα μπροστά και
digitalWrite(in2, HIGH);	
digitalWrite(in3, LOW);	
digitalWrite(in4, HIGH);	
rotation = true;	ορίζω πως η κίνηση γίνεται προς τα μπροστά.
}	
if (RecievedValue2 >= 128) {	Αν το joystick_x είναι ίσο ή μεγαλύτερο του μισού του πεδίου ορισμού του (128), τότε
motorA = map (128 * Speed , 0 , 128 * 255 /*32640*/, 0 , 255);	Θέτω την ταχύτητα του αριστερού κινητήρα με ποσοστό διανομής 100% της γενικής ταχύτητας και την το αντιστοιχώ σε πεδίο τιμών από 0 μέχρι 255.
}	
else if (RecievedValue2 < 128) {	Αλλιώς, αν το joystick_x είναι μικρότερο από το μισό του πεδίου ορισμού του (128), τότε
motorA = map (RecievedValue2 * Speed , 0 , 128 * 255 /*32640*/ , 0 , 255);	Θέτω την ταχύτητα του αριστερού κινητήρα με ποσοστό διανομής που ορίζει η ReceivedValue2 της γενικής ταχύτητας και την αντιστοιχώ σε πεδίο τιμών από 0 μέχρι 255.
}	
if (255 - RecievedValue2 >= 128) {	Αν το υπόλοιπο του joystick_x είναι ίσο ή μεγαλύτερο του

	μισού του πεδίου ορισμού του (128), τότε
motorB = map (128 * Speed , 0 , 128 * 255 /*32640*/ , 0 , 255);	Θέτω την ταχύτητα του αριστερού κινητήρα με ποσοστό διανομής 100% της γενικής ταχύτητας και την το αντιστοιχώ σε πεδίο τιμών από 0 μέχρι 255.
}	
else if (255 - RecievedValue2 < 128) {	Αλλιώς, αν το υπόλοιπο του joystick_x είναι μικρότερο από το μισό του πεδίου ορισμού του (128), τότε
motorB = map ((255 - RecievedValue2) * Speed , 0 , 128 * 255 /*32640*/ , 0 , 255);	Θέτω την ταχύτητα του αριστερού κινητήρα με το υπόλοιπο ποσοστό διανομής που ορίζει η ReceivedValue2 της γενικής ταχύτητας και την αντιστοιχώ σε πεδίο τιμών από 0 μέχρι 255.
}	
analogWrite(enA, motorA);	Στέλνω την ταχύτητα του αριστερού κινητήρα στην έξοδο για τον αριστερό κινητήρα.
analogWrite(enB, motorB);	Στέλνω την ταχύτητα του δεξιού κινητήρα στην έξοδο για τον δεξιό κινητήρα.
Serial.print(motorA); Serial.print(" A || "); Serial.print(motorB); Serial.print(" B | ");	Προβάλλω στο monitor, την ταχύτητα του κάθε κινητήρα και αν κινούνται προς τα μπροστά ή προς τα πίσω (κάτι που αρχικά πρέπει να ορίσω εγώ, όταν συνδέω τα καλώδια των κινητήρων στην πλακέτα).
if (!rotation) {	
Serial.println("Backwards");	
} else if (rotation) {	
Serial.println("Forward");	
}	
}	
float find_pH (int sensor_pin) {	Διαδικασία για να βρεθεί μία ακριβή μέτρηση από τον αισθητήρα pH.
for (int i = 0; i < 10; i++)	Παίρνω 10 μετρήσεις από τον αισθητήρα pH με διαφορά 10 χιλιοστών του δευτερολέπτου και τις αποθηκεύω στη λίστα buf.
{	
buf[i] = analogRead(sensor_pin);	
delay(10);	
}	
for (int i = 0; i < 9; i++)	
{	
for (int j = i + 1; j < 10; j++)	
{	
if (buf[i] > buf[j])	
{	
temp = buf[i];	
buf[i] = buf[j];	
buf[j] = temp;	
}	
}	
}	
avgValue = 0;	
for (int i = 2; i < 8; i++)	
avgValue += buf[i];	
float pHVol = (float)avgValue * 5.0 / 1024 / 6;	Αντιστοιχώ το μέσο όρο σε πεδίο τιμών 0-12, επιτρέποντας δεκαδικά ψηφία.
float phValue = -5.70 * pHVol + 21.34;	Ρυθμίζω την τιμή βάσει των σφαλμάτων του αισθητήρα.
delay(20);	Περιμένω 20 χιλιοστά του δευτερολέπτου.
return phValue;	Επιστρέφω την, τελικώς, επεξεργασμένη μέτρηση.
}	