**Τ. Ε. Ι. ΣΤΕΡΕΑΣ ΕΛΛΑΔΑΣ**

**ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ**

**Τμήμα Πληροφορικής και Τεχνολογίας Υπολογιστών**

**ΑΝΑΛΟΓΙΚΕΣ ΜΕΤΡΗΣΕΙΣ ΚΑΙ ΕΠΕΞΕΡΓΑΣΙΑ ΑΥΤΩΝ ΣΕ ΛΕΙΤΟΥΡΓΙΚΟ ΣΥΣΤΗΜΑ lINUX**

**ΔΡΟΥΜΠΑΛΗΣ ΓΕΩΡΓΙΟΣ**

**ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ**

**Επιβλέπων**

**ΠΑΝΑΓΙΩΤΗΣ ΠΑΠΑΖΟΓΛΟΥ**

Αναπληρωτής Καθηγητής

Λαμία 2018

ΕΥΧΑΡΙΣΤΙΕΣ

Η ολοκλήρωση αυτής της πτυχιακής υλοποιήθηκε με την υποστήριξη ενός αριθμού ανθρώπων στους οποίους θα ήθελα να εκφράσω τις θερμότερες ευχαριστίες μου. Πρώτα από όλους θα ήθελα να ευχαριστήσω τον καθηγητή μου Παναγιώτη Παπάζογλου για την πολύτιμη βοήθεια του και στήριξη που μου παρείχε σε όλη την διάρκεια της πτυχιακής εργασίας , και για τις πολύτιμες γνώσεις που μου μεταβίβασε σε όλο αυτό το διάστημα που εργαστήκαμε μαζί. Τέλος , θα ήθελα να ευχαριστήσω όλη την οικογένειά μου και τους κοντινούς μου ανθρώπους που με στήριξαν στην διάρκεια των σπουδών μου.

Γεώργιος Δρούμπαλης

Οκτώβριος 2018

ΠΕΡΙΛΗΨΗ

Το αντικείμενο της παρούσας πτυχιακής είναι η δημιουργία ενός ενσωματωμένου συστήματος (Embedded system) για την μέτρηση αναλογικών μετρήσεων μέσω αισθητήρων και την απεικόνιση και επεξεργασία αυτών σε λειτουργικό σύστημα Linux με χρήση της προγραμματιστικής γλώσσας Python.

Θα μελετηθεί η ζωντανή και άμεση απεικόνιση των μετρήσεων από τον αισθητήρα και τον μικροελεγκτή στον ηλεκτρονικό υπολογιστή (Live plotting) , η δειγματοληψία αναλογικών μετρήσεων (Sampling) καθώς και η ζωντανή παρακολούθηση και η εκτέλεση συγκεκριμένων λειτουργιών όταν η μέτρηση φτάσει σε οριζόμενα από τον χρήστη όρια (Handlers).

Θα γίνει αναλυτική εξήγηση του υλικού που θα χρησιμοποιηθεί , ο προγραμματισμός του αλλά και ο προγραμματισμός του απαραίτητου λογισμικού για την απεικόνιση των δεδομένων και η επεξεργασία αυτών.

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ:ATMEGA16A,PYTHON35 , PYQT5,PYQTGRAPH,LM35Z,FEDORA,RASPBERRY PI 3,NUMPY,PYSERIAL,QTDESIGNER,PYCHARM,AVRDUDE

περιεχομενα

[1.1 Εισαγωγή 6](#_Toc534620271)

[1.2 Ενσωματωμένα Συστήματα ( Embedded systems) 7](#_Toc534620272)

[1.3 Raspberry Pi 9](#_Toc534620276)

[1.4 Python 11](#_Toc534620277)

[1.4.1 Ιστορία της Python 12](#_Toc534620278)

[2.1 Εισαγωγή 14](#_Toc534620279)

[2.2 Μικροελεγκτής AVR 14](#_Toc534620280)

[2.2.1 Ιστορία AVR μικροελεγκτών 15](#_Toc534620281)

[2.2.2 ATMega16a 15](#_Toc534620282)

[2.3 Αισθητήρας LM35Z 17](#_Toc534620283)

[2.4 Διεπαφή σειριακής επικοινωνίας FT232 19](#_Toc534620284)

[2.4.1 Ανατομία FT232 19](#_Toc534620285)

[2.5 Προγραμματιστής Μικροελεγκτή - USBASP 21](#_Toc534620286)

[2.6 Τελικό σχήμα διασύνδεσης 23](#_Toc534620287)

[2.7 Πρότυπο ασύγχρονης επικοινωνίας UART 24](#_Toc534620288)

[2.7.1 Λειτουργία UART 24](#_Toc534620289)

[2.8 Απαιτούμενο λογισμικό προγραμματισμού 25](#_Toc534620290)

[2.8.1 Raspbian 25](#_Toc534620291)

[2.8.2 GCC – AVR GCC 26](#_Toc534620292)

[2.8.2.1 Γλώσσα προγραμματισμού C 26](#_Toc534620293)

[2.8.3 AVRDude 26](#_Toc534620294)

[2.8.4 Fedora OS 27](#_Toc534620295)

[2.8.5 Pycharm IDE 27](#_Toc534620296)

[2.8.6 Python 3.5 28](#_Toc534620297)

[2.8.7 PyQTGraph 29](#_Toc534620298)

[2.8.8 PySerial 32](#_Toc534620299)

[2.8.9 Numpy 32](#_Toc534620300)

[2.8.10 PyQt 5 32](#_Toc534620301)

[2.8.11 Qt Designer 34](#_Toc534620302)

[2.8.12 Git 35](#_Toc534620303)

[3.1 Εισαγωγή 36](#_Toc534620304)

[3.2 Εφαρμογή της πτυχιακής εργασίας 36](#_Toc534620305)

[3.3 Επίπεδο Megaman 38](#_Toc534620306)

[3.4 Λειτουργία ADC 39](#_Toc534620307)

[3.5 Προγραμματισμός AVR – Firmware Megaman 41](#_Toc534620308)

[3.5.1 Μεταγλώττιση και Εγγραφή του Firmware 49](#_Toc534620309)

[4.1 Εισαγωγή 51](#_Toc534620310)

[4.2 Λειτουργίες Electro 51](#_Toc534620311)

[4.2.1 Επεξήγηση βοηθητικών εργαλείων Electro 53](#_Toc534620312)

[4.2.2 Επεξήγηση Live Plotting 54](#_Toc534620313)

[4.2.3 Επεξήγηση Sampling 57](#_Toc534620314)

[4.2.4 Επεξήγηση Handlers 59](#_Toc534620315)

[4.3 Ανάπτυξη λογισμικού Electro 62](#_Toc534620316)

[4.3.1 Σχεδίαση και προγραμματισμός GUI 66](#_Toc534620317)

[4.3.2 Εκκίνηση MRenderer 71](#_Toc534620318)

[4.3.3 Λειτουργία MRenderer 74](#_Toc534620319)

[5.1 Συμπεράσματα 78](#_Toc534620320)

[5.2 Πιθανές επεκτάσεις 78](#_Toc534620321)

1. ο κεφαλαιο

Εισαγωγη στα Ενσωματωμενα συστηματα και στο raspberry pi

# Εισαγωγή

Όπως αναφέρθηκε η συγκεκριμένη εργασία θα εκπονήσει την χρήση ενός ενσωματωμένου συστήματος (embedded system) για την λήψη αναλογικών μετρήσεων και την άμεση απεικόνισή τους και επεξεργασίας τους από έναν ηλεκτρονικό υπολογιστή.

Θα εξεταστούν μεθοδικά τα δύο δομικά υποσυστήματα (MegaMan, Electro) του συστήματος που έχει δημιουργηθεί για την λήψη και απεικόνιση των αναλογικών μετρήσεων.

Θα παρουσιαστεί το υλικό που θα χρειαστεί , ο τρόπος σύνδεσης που απαιτείται καθώς και ο προγραμματισμός του υποσυστήματος MegaMan που αποτελεί το low level κομμάτι του συστήματος και συμπεριλαμβάνει τον μικροελεγκτή , τον αισθητήρα καθώς και το firmware που είναι απαραίτητα για την λήψη των αναλογικών μετρήσεων.

Τέλος θα παρουσιαστεί ο προγραμματισμός και όλες οι τεχνολογίες που απαιτούνται για την δημιουργία του λογισμικού (Electro) που εκμεταλλεύεται το συγκεκριμένο υλικό καθώς όλες οι λειτουργίες του που δείχνουν πόσο χρήσιμα μπορούν να γίνουν τα ενσωματωμένα συστήματα και πως μπορούν να βελτιώσουν όλους τους τομείς μια σύγχρονης κοινωνίας.

## Ενσωματωμένα Συστήματα ( Embedded systems)

Ένα ενσωματωμένο σύστημα αποτελεί μέρος ενός μεγαλύτερου συστήματος που συνήθως είναι ένας υπολογιστής με ειδική λειτουργία που αναλαμβάνει να λύσει συγκεκριμένα προβλήματα.

Ένα ενσωματωμένο σύστημα αποτελεί συνδυασμό υλικού και λογισμικού. Χρησιμοποιεί ,όπως όλα τα υπολογιστικά συστήματα, κάποιον επεξεργαστή ή μικροελεγκτή ,μνήμη ram και rom , και μέσω σειριακών θυρών συνδέεται με αισθητήρες όπου μπορεί να παρακολουθεί διάφορες μετρήσεις, να δώσει αναφορά στον χρήστη μέσω ηχητικών μηνυμάτων ή οθονών και να δεχθεί οδηγίες από κάποιο πληκτρολόγιο ή οθόνη αφής.

Εικόνα 1.1‑1 Δομή Μικροελεγκτή

Τέτοια συστήματα χρησιμοποιούνται κατά κόρων σε πολλούς τομείς όπως:

1. Αυτοκινητοβιομηχανία. (Συστήματα ABS,ESP, Cruise Controls)
2. IoT – Internet of Things (Home automations, Smart devices, Smart Homes)
3. Επιστημονικές μελέτες (Ζωντανές μετρήσεις , Μετρήσεις φαινομένων)
4. Ιατρική (Αυτοματοποιημένα χειρουργικά ρομπότ, Αισθητήρες παρακολούθησης, Συσκευές πρόσληψης φαρμακευτικών ουσιών)
5. Ρομποτική
6. Ψυχαγωγία

Εικόνα 1.1‑2 Χρήσεις ενσωματωμένων συστημάτων

Κάνοντας χρήση αυτών των συστημάτων πολλοί επιστημονικοί και εμπορικοί τομείς λύνουν προβλήματα ασφαλείας , ιατρικά προβλήματα , παραγωγικά προβλήματα και προσφέρουν ανέσεις κάνοντας την ζωή των ανθρώπων πιο εύκολη.

Προσπαθώντας να γίνει όλη αυτή η τεχνολογία προσβάσιμη από μεγαλύτερο εύρος ανθρώπων δημιουργήθηκε το Raspberry Pi. Ένας μικρός υπολογιστής αρχιτεκτονικής ARM με πάρα πολύ μικρό κόστος, που μπορεί να γίνει βάση για πολλές από τις παραπάνω χρήσεις που αναφέρθηκαν ενσωματώνοντας ένα τεράστιο εύρος δυνατοτήτων, που δίνουν εύκολη πρόσβαση σε τεχνολογίες που απαιτούνται ,για την ανάπτυξη λύσεων σε πολλά από τα παραπάνω προβλήματα.



# Raspberry Pi

To Raspberry Pi αποτελεί μια σειρά μικρών υπολογιστών που έχουν αναπτυχθεί στο Ηνωμένο βασίλειο από το Raspberry Pi Foundation και πρώτη ημερομηνία κυκλοφορίας στις 29 Φεβρουαρίου 2012 με εισαγωγική τιμή πώλησης τα 35 δολάρια.

Εικόνα 1.3‑1Raspberry PI

Το κόστος των συγκεκριμένων υπολογιστών παραμένει αρκετά μικρό καθώς αποτελούν ιδανικό εκπαιδευτικό εργαλείο για όλες τις αναπτυσσόμενες χώρες.

Έκτοτε το Raspberry Pi έχει εξελιχθεί και έχει ένα τεράστιο αριθμό καινούριων μοντέλων με πολύ μεγάλες βελτιώσεις που το καθιστούν κατάλληλο εργαλείο για την εκπόνηση της συγκεκριμένης εργασίας. Το μοντέλου που χρησιμοποιούμε είναι το Raspberry Pi 3 Model B με τα παρακάτω τεχνικά χαρακτηριστικά.

1. Αρχιτεκτονική ARMv8-A (64/32-bit)
2. Επεξεργαστή 4× Cortex-A53 1.2 GHz
3. 1GB Ram μοιραζόμενη με την GPU
4. 4 θύρες usb
5. Θύρα HDMI
6. Θύρα ethernet για σύνδεση σε δίκτυο.
7. GPIO Θύρα για διασύνδεση με εξωτερικούς αισθητήρες.

A close up of a device

Description automatically generatedΤο παραπάνω υπολογιστικό σύστημα αποτελεί μια ιδανική πλατφόρμα για πληθώρα χρήσεων. Το raspberry Pi θα μπορούσε να χρησιμοποιηθεί στον έλεγχο ενός έξυπνου σπιτιού με χρήση διαφόρων αισθητήρων όπως θερμικοί αισθητήρες για έλεγχο θερμοκρασίας του κλίματος και του νερού ώστε να γίνει αυτοματοποίηση των κλιματιστικών μονάδων ενός κτιρίου και της θέρμανσης του νερού , αισθητήρες φωτός για την αυτοματοποίηση των φωτιστικών σωμάτων για μείωση του ενεργειακού κόστους, αισθητήρες κίνησης για αυτοματοποίηση εισόδων και εξόδων ενός κτιρίου.

Εικόνα 1.3‑2Raspberry Pi με LCD απεικονίζοντας καιρικές συνθήκες

Με την χρήση των ίδιων αισθητήρων θα μπορούσε να χρησιμοποιηθεί για επιστημονικές έρευνες όπως στην δημιουργία ενός μετεωρολογικού σταθμού για συλλογή πληροφοριών και παρατήρηση φυσικών φαινομένων μέχρι και την ζωντανή παρακολούθηση ασθενών με μικρό κόστος.

Το Raspberry Pi καθίσταται ιδανικό εκπαιδευτικό εργαλείο και τέλος πληθώρα ανθρώπων το χρησιμοποιούν σαν μονάδα ψυχαγωγίας είτε για κατανάλωση ψυχαγωγικού περιεχόμενου είτε ως παιχνιδοκονσόλα με χρήση του RetroPie.

Στην συγκεκριμένη εργασία θα χρησιμοποιηθεί ως ο υπολογιστής που θα λαμβάνει το αναλογικό σήμα αφού μετατραπεί από το ADC (analog to digital converter) του μικροελεγκτή και μέσω του ειδικού λογισμικού θα μπορεί να το απεικονίζει ζωντανά , να αποθηκεύει τις μετρήσεις και να εκτελεί κάποιες ενέργειες βάση συγκεκριμένων ορίων που θέτει ο χρήστης.

# Python

Για την δημιουργία του απαραίτητου λογισμικού που τρέχει στο Raspberry Pi και εκμεταλλεύεται το υποσύστημα MegaMan θα χρειαστούμε μια γλώσσα προγραμματισμού που έχει πληθώρα βιβλιοθηκών για την επικοινωνία αλλά και την ζωντανή απεικόνιση των αναλογικών μετρήσεων. Επίσης απαιτείται η συγκεκριμένη γλώσσα να έχει ένα σταθερό και ευέλικτο ui framework που θα μας επιτρέψει εύκολα και γρήγορα να δημιουργήσουμε το δεύτερο υποσύστημα , το Electro , που ουσιαστικά αποτελεί το UI του συστήματος δίνοντας εύκολη πρόσβαση των λειτουργιών του στον χρήστη. Τέλος θα ήταν πολύ χρήσιμο η γλώσσα αυτή να έχει την δυνατότητα να τρέξει σε όλα τα μεγάλα λειτουργικά συστήματα ( Windows , Linux , Mac OS).

Μια από τις γλώσσες που τηρεί όλες αυτές τις προϋποθέσεις είναι η Python , και αυτή η γλώσσα επιλέχθηκε για την υλοποίηση του απαραίτητου κώδικα.

## Ιστορία της Python

H Python είναι μια interpreted γλώσσα υψηλού επιπέδου που μπαίνει κάτω από την κατηγορία τον Γλωσσών προγραμματισμού γενικού σκοπού. Για να χρησιμοποιηθεί θα πρέπει να γίνει εγκατάσταση ,στο λειτουργικό σύστημα , ο απαραίτητος διερμηνέας της γλώσσας (interpreter) ώστε να μπορέσει να μεταφραστεί ο κώδικας της Python σε κώδικα μηχανής.

Εικόνα 1.4‑1Το λογότυπο της Python

Δημιουργός της γλώσσας είναι ο Guido van Rossum και η πρώτη έκδοση της γλώσσας κυκλοφόρησε το 1991.

Κύρια φιλοσοφία της γλώσσας είναι ο καλά δομημένος κώδικας που επιτρέποντας την εύκολη ανάγνωσή του. Το dynamic type σύστημα της Python δίνει την δυνατότητα στον προγραμματιστή να εστιάσει στα πιο σημαντικά κομμάτια του προγραμματισμού αφήνοντας την διαχείριση των τύπων των μεταβλητών στην ίδια την γλώσσα. Η διαχείριση της μνήμης του προγράμματος γίνεται επίσης από την ίδια την γλώσσα προσφέροντας ακόμα μεγαλύτερη ευκολία και ασφάλεια στων προγραμματισμό αφού η μνήμη του προγράμματος δεν είναι ποτέ εκτεθειμένη στον προγραμματιστή. Αυτό αποτρέπει τα σφάλματα μνήμης.

Η Python υποστηρίζει πολλών ειδών τύπους προγραμματισμού όπως ο αντικειμενοστραφής και διαδικαστικός προγραμματισμός. Στην υλοποίηση που θα παρουσιαστεί έχει ακολουθηθεί ο αντικειμενοστραφής προγραμματισμός.

Κύριο πλεονέκτημα της γλώσσας είναι ότι χτίστηκε με στόχο να είναι εύκολα επεκτάσιμη. Αυτό είναι απόρροια της επιλογής των δημιουργών της να μην συμπεριλάβουν στον πυρήνα της γλώσσας όλα τα απαιτούμενα χαρακτηριστικά αλλά δίνοντας τα ως βιβλιοθήκες. Αυτή η επιλογή δίνει την δυνατότητα στην κοινότητα της Python να γράψουν πολλές μικρές βιβλιοθήκες που ο τελικός χρήστης μπορεί πολύ εύκολα να χρησιμοποιήσει στον κώδικα του. Στις τελευταίες εκδόσεις της python με την χρήση του PIP , που αποτελεί το πρόγραμμα διαχείρισης των πακέτων της Python, ο χρήστης μπορεί πολύ εύκολα να κατεβάσει τα απαιτούμενα πακέτα και να τα χρησιμοποιήσει άμεσα.

1. ο κεφαλαιο

τεχνολογίες διασυνδεσησ

# Εισαγωγή

Στο συγκεκριμένο κεφάλαιο θα αναλυθεί το πρώτο υποσύστημα της πτυχιακής , τον Megaman. Επίσης θα αναλυθούν όλα τα δομικά του κομμάτια και ο προγραμματισμός του.

Το συγκεκριμένο υποσύστημα αποτελεί το low level του συστήματος που είναι υπεύθυνο για την διασύνδεση του μικροελεγκτή με τον αισθητήρα θερμότητας , το firmware για την επικοινωνία και την λειτουργία των παραπάνω, την μετατροπή του αναλογικού σήματος σε ψηφιακό καθώς και την διασύνδεση και την αποστολή των μετρήσεων στο υπολογιστικό σύστημα.

# Μικροελεγκτής AVR

Ο μικροελεγτής όπως αναφέρθηκε αποτελεί την καρδιά τους ενσωματωμένου συστήματος. Αν δούμε τα δομικά στοιχεία ενός μικροελεγκτή θα δούμε ότι αποτελείται από:

1. Έναν επεξεργαστή
2. Ένα ρολόι συστήματος
3. Μνήμη
4. Καθώς και κάποια άλλα περιφερειακά

Πολύ εύκολα λοιπόν μπορεί να ειπωθεί ότι ένας μικροελεγκτής είναι ουσιαστικά ένας υπολογιστής.

Εδώ χρησιμοποιείται ένας μικροελεγκτής τύπου AVR , πιο συγκεκριμένα τον ATMega16A.

## Ιστορία AVR μικροελεγκτών

Οι AVR αποτελούν μια οικογένεια μικροελεγκτών που αναπτύσονται από το 1996 από την εταιρεία Atmel. Μετέπειτα η τεχνολογία αγοράστηκε από την Microchip Technology το 2016.

Πρόκειται για παραμετροποιημένα τσιπ των 8bit της Harvard αρχιτεκτονικής RISC.

Η αρχιτεκτονική AVR πρωτοήλθε σε σύλληψη από δύο φοιτήτες του Νορβηγικού ινστιτούτου τεχνολογίας , τον Vegard Wollan και τον Alf-Egil Bogen. Αν και δεν είναι επιβεβαιωμένο είναι κοινώς αποδεκτό ότι το AVR σημαίνει Alf and Vegard's RISC processor από τα ονόματα των φοιτητών που είχαν την αρχική ιδέα.

## ATMega16a

Ο συγκεκριμένος μικροελεγκτής αποτελεί ένα χαμηλής κατανάλωσης τσιπ 8/16 bit. Παρακάτω φαίνονται τα χαρακτηριστικά του σύμφωνα με την Microchip.

Ο συγκεκριμένος μικροελεγκτής έχει 40 pins χωρισμένα σε 4 βασικές θύρες ονοματισμένες Α,Β,C,D. Από τις τέσσερις θύρες αυτή που χρήζει ιδιαίτερης προσοχής είναι η A. Στην συγκεκριμένη θύρα του μικροελεγκτή είναι υλοποιημένο το ADC σύστημα που επιτρέπει μετατροπή των αναλογικών σημάτων σε ψηφιακών. Αναλυτικότερα τα pin 33-40 αφορούν την θύρα A και χρησιμοποιούνται μεταξύ άλλων στην είσοδο του αναλογικού συστήματος και στην μετατροπή του σε ψηφιακό. Σε αυτό το σημείο θα πρέπει να συνδεθεί οποιοσδήποτε αισθητήρας ώστε να μπορέσει να ψηφιοποιηθεί το σήμα του και να αποσταλεί εκεί που είναι επιθυμητό.

Εικόνα 2.2‑1 Χαρακτηριστικά ATMega16a

Στην εικόνα 2.2-2 φαίνονται αναλυτικά τα pins του μικροελεγκτή καθώς και η θύρα Α με τα pins που θα χρησιμοποιηθούν για την λήψη και την μετατροπή του αναλογικού σήματος.



Εικόνα 2.2‑2 Δομή του ATMega16a

Πιο αναλυτικά ο αισθητήρας θερμότητας που θα χρησιμοποιηθεί θα συνδεθεί στο pin PA0 (ADC0) και αυτό παίζει σημαντικό ρόλο στην υλοποίηση του λογισμικού του μικροελεγκτή.

# Αισθητήρας LM35Z

Η εργασία θα επικεντρωθεί στην μέτρηση θερμοκρασίας ως παράδειγμα αναλογικού σήματος. Για αυτό τον σκοπό θα χρησιμοποιηθεί ο αισθητήρας θερμοκρασίας LM35z. Ο συγκεκριμένος πρόκειται για έναν αισθητήρα ακριβείας , με έξοδο τάσης γραμμικά ανάλογη με την θερμοκρασία κελσίου.

Ο αισθητήρας αυτός έχει ένα τεράστιο πλεονέκτημα έναντι των αισθητήρων που είναι βαθμονομημένοι στην κλίμακα Kelvin καθώς ο χρήστης δεν χρειάζεται να αφαιρεί μεγάλο σταθερό ποσό τάσης από την έξοδο το αισθητήρα ώστε να πάρει μετρήσεις στην κλίμακα κελσίου.

Ο LM35z δεν απαιτεί εξωτερική βαθμονόμηση ώστε να παρέχει αποδεκτή ακρίβεια σε θερμοκρασίες δωματίου με εύρος τους -55℃ εώς τους 150℃.

Εικόνα 2.3‑1 Διάγραμμα αισθητήρα

Το συγκεκριμένο όργανο μέτρησης φέρει 3 pins για την διασύνδεσή του με τον μικροελεγκτή. Το ένα pin θα συνδεθεί στην πηγή ρεύματος, το δεύτερό , που είναι και η έξοδος της τάσης, στην θύρα Α του μικροελεγκτή σε pin της επιλογής του χρήστη και το τελευταίο pin στην γείωση του κυκλώματος.

Εικόνα 2.3‑2 3D απεικόνιση του LM35

Στο σχήμα 2.2-4 φαίνονται αναλυτική τα τρία pins του συγκεκριμένου αισθητήρα. Για να αποφεύγονται τα λάθη ο αισθητήρας έχει ένα πολύ χαρακτηριστικό σχήμα. Έτσι η επίπεδη πλευρά του θεωρείται πάντα η όψη του αισθητήρα.

# Διεπαφή σειριακής επικοινωνίας FT232

Έχοντας καλύψει την λήψη και την επεξεργασία/μετατροπή του σήματος θα πρέπει στην συνέχεια το επεξεργασμένο σήμα να γίνει serialization και να αποσταλεί στο υπολογιστικό σύστημα για να γίνει η απεικόνιση από το Electro, το λογισμικό απεικόνισης και επεξεργασίας του σήματος.

Σε αυτό το σημείο θα μπορούσε να χρησιμοποιηθεί το GPIO του Raspberry Pi. Αυτό όμως θα περιόριζε σημαντικά την ευελιξία του συστήματος καθώς ο μικροελεγκτής θα μπορούσε να γίνει χρήσει μόνο από το Raspberry Pi. Για να αποφευχθεί αυτό και να δοθεί μεγαλύτερη ευελιξία στο σύστημα , το serialization του σήματος θα το αναλάβει ένα FTDI FT232.

## Ανατομία FT232

To FT232 πρόκειται για μια διεπαφή σειριακή επικοινωνίας από USB θύρα σε σειριακή UART. Επιτρέπει στην μεταφορά του σήματος από την TX(Trasmit) θύρα του μικροελεγκτή στην RX(Receive) θύρα του FT232 και στην συνέχεια στην USB θύρα του υπολογιστικού συστήματος που θέλουμε να κάνουμε επεξεργασία το σήμα.

Εικόνα 2.4‑1 FT232

Στην εικόνα 2.2-5 φαίνεται το FT232 που θα χρησιμοποιηθεί για την αποστολή του σήματος στο Raspberry PI. Η συγκεκριμένη συσκευή διεπαφής αποτελείται από:

1. 6 pins
   1. VCCIO για την παροχή ρεύματος
   2. GND – Γείωση
   3. TXD – Transmit σήματος από το FT232
   4. RXD – Receive σήματος προς το FT232
   5. RTS - Request to Send Control Output / Handshake Signal.
   6. CTS - Input Clear To Send Control Input / Handshake Signal.
2. Το τσιπ που είναι υπεύθυνο για την λειτουργία του
3. 3 ενδεικτικά led λήψης σήματος ,αποστολής σήματος και παροχής ενέργειας
4. Την έξοδο USB
5. Και έναν επιλογέα τάσης 5V ή 3.3V. Στην συγκεκριμένη υλοποίηση έχει επιλεχθεί η λειτουργία 5V

Στο σχήμα Σχήμα 2.2 6 φαίνεται αναλυτικά το διάγραμμα του FT232



Σχήμα 2.4‑2 Αναλυση του κυκλώματος του FT232

# Προγραμματιστής Μικροελεγκτή - USBASP

Έχοντας αναλύσει τα δομικά κομμάτια του υποσυστήματος Megaman μένει να αναλυθεί η συσκευή που απαιτείται για τον προγραμματισμό του μικροελεγκτή.

Για τον σκοπό αυτόν θα χρησιμοποιηθεί το USBASP v2.0 που είναι η κατάλληλη συσκευή ώστε να γίνει η μεταφορά του απαραίτητου κώδικα στον μικροελεγκτή.



Εικόνα 2.5‑1 USB ASP version 2.0 και το καλώδιο σύνδεσης

Η συσκευή αποτελείται συνήθως από ένα ATMega8 καθώς και κάποια παθητικά εξαρτήματα. Ο συγκεκριμένος προγραμματιστής δεν χρειάζεται ειδικούς drivers για την λειτουργία του. Σε συνεργασία με το ειδικό λογισμικό AVRDude μπορεί να γίνει η εγγραφή του firmware του μικροελεγκτή και να ξεκινήσει η λειτουργία του συστήματος. Ο συγκεκριμένο προγραμματιστής συνδέεται με τον μικροελεγκτή με χρήση μιας ειδικής καλωδιοταινίας. Η συγκεκριμένη καλωδιοταινία απότελείται από 10 υποδοχές που ταιριάζουν με την έξοδο του USBASP. Στο σχήμα 2.2-8 φαίνονται αναλυτικά οι υποδοχές του USBASP και οι υποδοχές της καλωδιοταινίας. Το συγκεκριμένο σχήμα θα γίνει οδηγός για την σωστή διασύνδεση του USBASP και του μικροελεγκτή

# Τελικό σχήμα διασύνδεσης

Εικόνα 2.5‑1 Υποδοχές USBASP και καλωδιοταινίας

Ολοκληρώνοντας την ανάλυση όλων των απαραίτητων εξαρτημάτων μένει να γίνει σχεδιασμός όλου του κυκλώματος του υποσυστήματος του Megaman. Στο παρακάτω σχήμα φαίνεται όλο το υποσύστημα συμπεριλαμβάνοντας όλα τα παθητικά μέρη καθώς και η διασύνδεση του προγραμματιστή του μικροελεγκτή.



Εικόνα 2.6‑2 Βασικό διάγραμμα υποσυστήματος Megaman

# Πρότυπο ασύγχρονης επικοινωνίας UART

To UART αποτελεί ένα κύκλωμα που επιτρέπει την ασύγχρονη σειριακή επικοινωνία υπολογιστών με μικροελεγκτές. Η λέξη UART σημαίνει universal asynchronous receiver/transmitter. Στην συγκεκριμένη υλοποίηση επειδή δεν χρησιμοποιείται εξωτερικός κρύσταλλος για αλλαγή του ρολογιού του μικροελεγκτή προτιμήθηκε να χρησιμοποιηθεί το εσωτερικό ενσωματωμένο ρολόι του μικροελεγκτή, αυτό έχει ως αποτέλεσμα η επικοινωνία μέσω UART να γίνεται στα 1200 baud rate.

## Λειτουργία UART

Το UART αποστέλλει bytes δεδομένων σε σειριακή μορφή ως bits. Στην πλευρά της συσκευής που δέχεται τα δεδομένα χρησιμοποιείται άλλο ένα UART που συλλέγει τα bits και αναδημιουργεί τα bytes της αρχικής πληροφορίας. Κάθε ένα UART περιέχει έναν καταχωρητή ολίσθησης για την μετατροπή σειριακής μορφής σε παράλληλης. Η συγκεκριμένη συσκευή αποστέλλει πλαίσια πληροφορίας των 10 bits από τα οποία το πρώτο είναι το bit εκκίνησης και το τελευταίο το bit διακοπής.

Εικόνα 2.7‑1 Πλαίσιο επικοινωνίας UART

# Απαιτούμενο λογισμικό προγραμματισμού

Για να γίνει η ανάπτυξη του απαραίτητου λογισμικού και για την λειτουργία του συστήματος θα χρειαστεί μια πληθώρα λογισμικού. Απαιτείται λογισμικό για την λειτουργία του Raspberry Pi ,για την συγγραφή του Firmware του μικροελεγκτή, για το compilation του firmware και την μετατροπή του σε αρχείο .hex καθώς και για την εγγραφή του στον μικροελεγκτή.

Τέλος θα χρειαστεί λογισμικό για την συγγραφή του Electro , του λογισμικού που κάνει λήψη και επεξεργασία των αναλογικών μετρήσεων. Παρακάτω θα παρουσιαστεί το απαραίτητο λογισμικό.

## Raspbian

Το Raspbian είναι το λειτουργικό που χρησιμοποιείται στο Raspberry Pi. Είναι ένα λειτουργικό βασισμένο στο Debian και από το 2015 προσφέρεται επίσημα από τον δημιουργό του Raspberry ως το κύριο λειτουργικό σύστημα.

Εικόνα 16 Raspbian on action

## GCC – AVR GCC

Ο GCC αποτελεί τον μεταγλωττιστή επιλογής για την μεταγλώττιση του πηγαίου κώδικα του firmware. Περιέχει όλα τα απαραίτητα εργαλεία για την μετατροπή του firmware στην απαραίτητη μόρφη για την εγγραφή του στον μικροελεγκτή καθώς και όλες τις βιβλιοθήκες που απαιτούνται για την συγγραφή του firmware.

## Γλώσσα προγραμματισμού C

Κατ’ επέκταση της παραγράφου 2.8.2 η γλώσσα που θα χρησιμοποιηθεί για την ανάπτυξη του firmware είναι η γλώσσα C.

Για την συγκεκριμένη γλώσσα προγραμματισμού δεν χρειάζονται ιδιαίτερες συστάσεις. Είναι μια γλώσσα προγραμματισμού γενικού σκοπού , με στατικό σύστημα τύπων που υποστηρίζει διαδικαστικό προγραμματισμό.

Η γλώσσα προγραμματισμού C είναι παιδί του Dennis Ritchie που αναπτύχθηκε στο διάστημα 1969-1973 στα Bell Labs και χρησιμοποιήθηκε για να γίνει ανάπτυξη του Unix, του πρώτου Posix λειτουργικού και πατέρα κατά μία έννοια του Linux.

## AVRDude

Το AVRDude αποτελεί το λογισμικό που θα χρησιμοποιηθεί για την εγγραφή του firmware στον μικροελεγκτή μετά τις απαραίτητες μετατροπές του πηγαίου κώδικα σε .hex τύπο αρχείου. Το avrdude υποστηρίζει πληθώρα συσκευών προγραμματισμού για την εγγραφή του firmware και είναι διαθέσιμο και στο λειτουργικό Windows αλλά και στο Linux.

## Fedora OS

Για την διευκόλυνση του προγραμματισμού και επειδή το Raspbian είναι ένα λειτουργικό για επεξεργαστές βασισμένους στην αρχιτεκτονική ARM όλος ο προγραμματισμός θα γίνει στο λειτουργικό Fedora με εκδόσεις των βιβλιοθηκών που υπάρχουν και στα repositories του Raspbian για να αποφευχθεί οποιαδήποτε μη συμβατότητα. Αυτό θα γίνει γιατί τα βοηθητικά εργαλεία που θα χρησιμοποιούν (IDE,Code Editors) δεν έχουν υποστήριξη για την αρχιτεκτονική ARM, αυτά τα εργαλεία δεν είναι υποχρεωτικά για τον προγραμματισμό , αλλά σίγουρα προσφέρουν τεράστια ευκολία και μειώνουν πολύ τον χρόνο υλοποίησης.

## Pycharm IDE

To Pycharm αποτελεί ένα IDE γραμμένο από την Jetbrains, εταιρεία που έχει πολύ μεγάλη ιστορία στην δημιουργία IDEs για πολλές γλώσσες προγραμματισμού συμπεριλαμβανομένου του IntelliJ που πρόκειται για ένα από τα IDE της Java. Έρχεται σε δύο εκδόσεις , την Community που αναπτύσσεται από την κοινότητα και την Professional που αναπτύσσεται από την Jetbrains και απαιτεί χρηματικό ποσό για την άδεια χρήσης του.

To Pycharm έχει πολλά βοηθητικά εργαλεία όπως autocomplete που βοηθάει την αυτόματη συμπλήρωση του κώδικα και την εύκολη εύρεση μεθόδων που εμπεριέχονται στις βιβλιοθήκες της Python , debug δυνατότητες που βοηθούν στην εύρεση προβλημάτων του κώδικα και πολλά εργαλεία refactoring του κώδικα που βοηθούν στην βελτίωσή του.



Εικόνα 17 PyCharm IDE

## Python 3.5

Όπως αναφέρθηκε η Python είναι η high level γλώσσα προγραμματισμού που επιλέχθηκε για την ανάπτυξη του κώδικα του Electro. Για να αποφύγουμε τις ασυμβατότητες με το Raspbian και επειδή η τελευταία έκδοση που υποστηρίζεται επίσημα από τα repositories του Raspbian είναι η Python 3.5 , θα χρησιμοποιηθεί η παραπάνω έκδοση και στο Fedora. Με αυτόν τον τρόπο μπορεί ,κατά κάποιο τρόπο , να συγχρονιστεί ο κώδικας και οι απαιτήσεις του. Η συγκεκριμένη έκδοση της Python είναι πλήρως συμβατή και με το Pycharm που θα χρησιμοποιηθεί για την ανάπτυξη του κώδικα.

## PyQTGraph

Το Pyqtgraph είναι μια βιβλιοθήκη ,γραμμένη εξολοκλήρου στην Python βασισμένη στα PyQt4 / Pyside και Numpy. Είναι ένα εργαλείο προγραμματισμού γραφικών διεπαφών που προσανατολίζεται για μαθηματικές/ επιστημονικές και μηχανολογικές χρήσεις.

Παρόλο που το PyQtGraph είναι γραμμένο σε Python είναι εξαιρετικά γρήγορο λόγω την εκτεταμένης και έξυπνης διαχείρισης μεγάλων αριθμών με το Numpy. Αυτό δίνει ένα εξαιρετικό πλεονέκτημα στην ζωντανή απεικόνιση των αναλογικών δεδομένων που λαμβάνονται από τον Megaman. Σε αρχικές προσεγγίσεις έγιναν δοκιμές και με το MatPlotLib. Μια ακόμα βιβλιοθήκη στην κατηγορία του Pyqtgraph.

Παρόλο που στο Fedora και σε αρχιτεκτονικές υπολογιστών με αρχιτεκτονικές 86x\64x επεξεργαστής τα αποτελέσματα ήταν πολύ καλά έως εξαιρετικά πολύ γρήγορα έγινε εμφανές ότι στο Raspberry Pi , με ιδιαίτερα πιο αδύναμό hardware , ότι η ζωντανή απεικόνιση των δεδομένων φέρουν ιδιαίτερες και εμφανείς καθυστερήσεις που κάνουν το λογισμικό του Electro μη λειτουργικό για τον τελικό χρήστη.

Στην προσπάθεια να γίνει βελτίωση τον χρόνων αυτών έγιναν προσπάθειες για optimization του κώδικα όπως:

1. Blit του γραφήματος
2. Chaching των δομικών στοιχείων του γραφήματος
3. Chaching των δυναμικών λιστών ζωντανών δεδομένων
4. Εκτεταμένη χρήση numpy και number crunching

Εικόνα 18 Παραδείγματα Χρησης του Pyqtgraph

Παρόλη την παραπάνω προσπάθεια και τα εκτεταμένα optimizations η εφαρμογή παρέμενε κατά μεγάλα ποσοστά έξω των αποδεκτών ορίων για χρήση ζωντανής απεικόνισης των αναλογικών δεδομένων στο Raspberry Pi.

Σε αυτή την χρονική στιγμή έγινε εμφανές ότι το MatplotLib δεν είναι η κατάλληλη βιβλιοθήκη για την επίλυση αυτού του προβλήματος και αντί αυτού επιλέχθηκε το PyQtGraph.

Εικόνα 19 Απεικόνιση δεδομένων με MatPlotLib

Σε πρώιμες δοκιμές με το PyQtGraph έγινες ξεκάθαρο ότι είναι το κατάλληλο εργαλείο για την ζωντανή απεικόνιση των δεδομένων με υπερβολικά μικρές καθυστερήσεις μη αντιληπτές από τον τελικό χρήστη.

## PySerial

Η λήψη των δεδομένων από τον Electro θα γίνει με χρήση του PySerial. Μιας βιβλιοθήκης της Python που προσφέρει τα κατάλληλα εργαλεία και τρέχει σε όλα τα διαδεδομένα λειτουργικά όπως Windows,Linux,Mac OS

Η συγκεκριμένη βιβλιοθήκη θα μας επιτρέψει εύκολα να ανοίξουμε τις απαραίτητες πόρτες επικοινωνίας επιτρέποντας επίσης την ρύθμιση του απαιτούμενο baud rate ώστε να γίνεται η σωστή μετατροπή των αναλογικών δεδομένων που στέλνει μέσω UART το ενσωματωμένο σύστημα.

Εδώ αξίζει να σημειωθεί ότι η συγκεκριμένη βιβλιοθήκη επιτρέπει την ρύθμιση του διαστήματος λήψης των αναλογικών μετρήσεων που επιτρέπει την κατάλληλη λειτουργία του συστήματος χωρίς εμφανή σημάδια καθυστερήσεων και φαινομένων υπερβολικής και άσκοπης δειγματοληψίας. Αυτό έχει ως αποτέλεσμα μείωσης του συνόλου των επεξεργαστικών κύκλων που χρειάζονται κατά την ζωντανή απεικόνιση των δεδομένων από το Raspberry Pi.

## Numpy

Το Numpy αποτελεί ένα ευρέως διαδομένο πακέτο επιστημονικής ανάλυσης της Python που αποτελεί βάση για πολλές από τις παραπάνω βιβλιοθήκες που αναφέρθηκαν.

Χρήση του Numpy γίνεται και στον κώδικα του renderer για την απεικόνιση των δεδομένων του ενσωματωμένου συστήματος.

## PyQt 5

Το PyQt αποτελεί ένα σύνολο bindings ,του γνωστού Framework Qt , για την Python. Μέσω αυτού του πακέτου μπορεί να γίνει χρήση του Qt που είναι γραμμένο εξολοκλήρου σε C++ και είναι το κύριο GUI framework για την C++.

Το PyQt είναι ένα crossplatform πακέτο για όλα τα μεγάλα λειτουργικά όπως Windows , Linux , Mac OS ακόμα και του Android. Το συγκεκριμένο πακέτο αποτελείται περίπου από 1000 κλάσεις που προσφέρουν όλα τα απαραίτητα εργαλεία για την δημιουργία ενός γραφικού περιβάλλοντος στην Python.

Το PyQt δεν έχει μόνο ικανότητες δημιουργίας GUI αλλά περιέχει εργαλεία και abstractions για τα παρακάτω:

1. GUI widgets
2. a multimedia framework
3. a help system
4. a fully functional web browser
5. XML
6. OpenGL
7. SVG
8. SQL databases
9. regular expressions
10. Unicode
11. threads
12. network sockets

Αποτελεί μια πλήρης λύση για την δημιουργία μιας πληρέστατης εφαρμογής με ιδιαιτέρως εκλεπτυσμένα χαρακτηριστικά.

Το PyQt περιέχει επίσης τον Qt Designer καθώς και εργαλεία μετατροπής του XML κώδικα που παράγεται από τον Qt Designer σε κώδικα Python ώστε να μπορεί η Python να εμφανίσει το γραφικό περιβάλλον.

Το Qt και κατ’ επέκταση το PyQt είναι framework βασισμένη στην αρχιτεκτονική γεγονότων ( event based ) και φέρει ένα σύστημα σημάτων/υποδοχών (signal/slot) όπου επιτρέπουν την διασύνδεση των UI components. Το σύστημα αυτό θα αναλυθεί παρακάτω

## Qt Designer

Το Qt Designer αποτελεί ένα εργαλείο σχεδιασμού γραφικών εφαρμογών χρησιμοποιώντας τα Qt Widgets του Qt. Ο χρήστης μπορεί να δημιουργήσει γραφικά παράθυρα εφαρμογών , custom widgets και διαλόγους στην λογική σχεδιασμού what-you-see-is-what-you-get (WYSIWYG). Το WYSIWYG( προφέρεται γουίζιγουιγκ) είναι ένα εύκολος τρόπος σχεδιασμού μια παραθυρικής εφαρμογής χρησιμοποιώντας drag n drop και προεπισκόπηση της εφαρμογής σε ζωντανό χρόνο. Αυτό έχει ως αποτέλεσμα ο προγραμματιστής να γλιτώνει πολύ χρόνο στην δημιουργία GUI αφού όλος ο κώδικας παράγεται αυτόματα από τον designer σε μορφή xml και μπορεί εύκολα να αναγνωστεί από τις εσωτερικές υποδομές του Qt. Παρόλα αυτά ο xml κώδικας της εφαρμογής μπορεί πάρα πολύ εύκολα να μετατραπεί σε κώδικα python μέσω εργαλείων που έρχονται μαζί με το PyQt.

Ο Qt Designer έχει επίσης ένα σύστημα διαχείρισης όλων των απαιτούμενων πόρων όπως εικόνες και μεταφράσης.

## Git

Για την ασφάλεια του κώδικα και των πόρων του συστήματος θα πρέπει ο κώδικας να μπει κάτω από κάποιο Source Control σύστημα για να γίνεται πάντα παρακολούθηση της προόδου , των αλλαγών αλλά και της φύσης των αλλαγών. Για αυτό τον σκοπό επιλέχθηκε το Git.

Το git αποτελεί ένα version-control σύστημα που παρακολουθεί τις αλλαγές που γίνονται στον κώδικα και κρατάει αντίγραφα σε όλη την περίοδο ανάπτυξης.

Το git δημιουργήθηκε από τον Linus Torvalds για την ανάπτυξη του πυρήνα του Linux και την εύκολη διαμοίραση και διαχείριση του κώδικα με μεγάλο αριθμό ατόμων που δεν βρίσκονται απαραίτητα μέσα στο ίδιο δίκτυο.

1. ο ΚΕΦΑΛΑΙΟ

αναπτυξη ενσωματωμενου συστηματοσ

# Εισαγωγή

Σε αυτό το κεφάλαιο θα αναλυθεί ο τρόπος που δημιουργήθηκε το ενσωματωμένο σύστημα. Θα παρουσιαστούν τα διαγράμματα του οι συνδέσεις καθώς και ο τελικός προγραμματισμός του firmware του ενσωματωμένου συστήματος. Θα παρουσιαστεί επίσης ο τρόπος που το firmware μετατρέπεται από κώδικα C σε κώδικα μηχανής και πως το παραγόμενο αρχείο μετατρέπεται σε αρχείο .hex για να γραφτεί στον μικροελεγκτή. Όλα τα παραπάνω θα γίνουν με το λογισμικό που αναφέρθηκε στο τέλος του δεύτερου κεφαλαίου.

# Εφαρμογή της πτυχιακής εργασίας

Όπως αναφέρθηκε η συγκεκριμένη πτυχιακή αφόρα την υλοποίηση και τον προγραμματισμό ενός συστήματος που αποτελείται από έναν ενσωματωμένο σύστημα που λαμβάνει αναλογικές μετρήσεις , τις μετατρέπει σε ψηφιακό σήμα και τις στέλνει σε ένα λογισμικό για επεξεργασία. Το συγκεκριμένο σύστημα αποτελείται από δύο βασικά επίπεδα/υποσυστήματα. Το πρώτο επίπεδο αποτελείται από το ενσωματωμένο σύστημα που λαμβάνει και αποστέλλει το σήμα μέσω ενός FT232 στον υπολογιστή , σε αυτό το σημείο αναλαμβάνει το δεύτερο υποσύστημα που αποτελείται από το λογισμικό που λαμβάνει το σήμα και προσφέρει διαφόρων ειδών επεξεργασίες στο συγκεκριμένο σήμα.

Το επίκεντρο της εργασίας είναι η μέτρηση της θερμοκρασίας. Αυτό όμως δεν σημαίνει ότι μπορεί να περιοριστεί μόνο σε αυτό. Με άλλους αισθητήρες μπορούν να μετρηθούν τα επίπεδα του φωτός , αποστάσεις , γυροσκοπικές συντεταγμένες , πίεση και πολλά άλλα φυσικά μεγέθη.

Εικόνα 20 Τα επίπεδα του συστήματος

Πιο συγκεκριμένα σε αυτό το κεφάλαιο θα αναλυθεί το πρώτο επίπεδο του συστήματος από το υλικό του μέχρι και τον προγραμματισμό του firmware που διαβάζει την θερμοκρασία την μετατρέπει και την αποστέλλει μέσω UART στο δεύτερο επίπεδο του συστήματος.

# Επίπεδο Megaman

Το κατώτατο επίπεδο του συστήματος αποτελείται από έναν ATMega16a έναν αισθητήρα θερμότητας LM35 που είναι υπεύθυνα για την μέτρηση της θερμοκρασίας και την μετατροπή της από αναλογικό σε ψηφιακό σήμα. Αφού γίνει η μετατροπή του σήματος από αναλογικό σε ψηφιακό τότε το firmware αναλαμβάνει την μετατροπή των τιμών σε ένα αλφαριθμητικό σε συγκεκριμένη μορφή που αναμένει το δεύτερο επίπεδο του συστήματος. Η μορφή αυτή αποτελείται από την τιμή της θερμοκρασίας σε βαθμούς κελσίου σε αλφαριθμητικό και έναν ειδικό χαρακτήρα νέας γραμμής στο τέλος.

Λαμβάνοντας τις τιμές με την παραπάνω μορφή , το δεύτερο επίπεδο του υποσυστήματος ξέρει ανά κύκλο μετρήσεως την τιμή της θερμοκρασίας αλλά και ότι έληξε η μέτρηση.

Στην Εικόνα 21 φαίνεται αναλυτικά η σύνδεση του υποσυστήματος λήψης των αναλογικών μετρήσεων. Αφού γίνει η λήψη της θερμοκρασίας από τον αισθητήρα το σήμα στέλνεται στην Πόρτα Α του μικροελεγκτή. Σε αυτό το σημείο αναλαμβάνει το ADC του μικροελεγκτή για την λήψη του σήματος και την μετατροπή του σε ψηφιακό σήμα. Στον κώδικα του firmware γίνεται επεξεργασία του σήματος σε βαθμούς κελσίου με τον παρακάτω τύπο.

Το firmware σε αυτό το στάδιο της επεξεργασίας μπορεί να μετατρέψει την τιμή αυτή σε αλφαριθμητικό σύμφωνα με τον τύπο αλφαριθμητικών που αναφέρθηκε στην προηγούμενη παράγραφο. Μέσω του FT232 γίνεται αποστολή των αλφαριθμητικών σε σειριακή μορφή στο υπολογιστικό σύστημα του Raspberry Pi μέσω θύρας USB.



Εικόνα 21 Σχηματικό διάγραμμα

# Λειτουργία ADC

Ένας μικροελεγκτής εσωτερικά μπορεί να διαχειριστεί μόνο ψηφιακό σήμα , παρόλα αυτά όμως όλοι οι αισθητήρες μπορούν να αποστείλουν μόνο πληροφορίες μέσω ποσοστού τάσης που μπορείς να κυμαίνεται από 0-3.3v ή από 0-5v , ανάλογα με τον τρόπο λειτουργίας του συστήματος. Όπως γίνεται ξεκάθαρο από το παραπάνω γεγονός φαίνεται να παρουσιάζεται ένα σημαντικό πρόβλημα διασύνδεσης ενός μικροελεγκτή και ενός αισθητήρα.

Για να ξεπεραστεί το πρόβλημα αυτό ένα μικροελεγκτής AVR φέρει ένα ADC ( Analog to Digital Conversion ) υπεύθυνο για την μετατροπή του σήματος ενός αισθητήρα σε ψηφιακό σήμα, ικανό να επεξεργαστεί από τον μικροελεγκτή. Το ADC έχει δύο τρόπους λειτουργίας (Operation Values).

|  |  |
| --- | --- |
| **Operation Values** | |
| Bit | Operation Value |
| 8 Bit | 255 |
| 10 Bit | 1024 |

Ο τρόπος λειτουργίας εξαρτάται από το μέγεθος της τάσης που δουλεύει το σύστημα. Αν το σύστημα δουλεύει στα 3.3v τότε δουλεύει με 8bit ενώ αν δουλεύει στα 5v τότε δουλεύει με 10 bit.

Ένα ADC μετατρέπει ένα σήμα συνεχή χρόνου και συνεχούς τάσης σε ένα σήμα διακεκριμένου χρόνου και διακεκριμένης τάσης. Η μετατροπή γίνεται μέσω κβαντοποίησης του σήματος , αυτό προσθέτει αυτόματα κάποιο ποσοστό θορύβου στο σήμα.

Η ακρίβεια της μετατροπής μπορεί να καθοριστεί από τον παρακάτω μαθηματικό τύπο

Παίρνοντας τον παραπάνω τύπο μπορούμε εύκολα να βρούμε την ακρίβεια του ADC με παραδείγματα. Αυτό σημαίνει ότι αν θεωρητικά το ADC δουλεύει σε λειτουργία 5v τότε το αποτέλεσμα είναι το παρακάτω:

# Προγραμματισμός AVR – Firmware Megaman

Ο προγραμματισμός του μικροελεγκτή θα πραγματοποιηθεί όπως προαναφέρθηκε στην γλώσσα προγραμματισμού C και θα γίνει χρήση της βιβλιοθήκης **avr/io.h** που περιέχει όλα τα απαραίτητα εργαλεία για τον προγραμματισμό ενός μικροελεγκτή από την γλώσσα C.

Θα παρουσιαστούν αναλυτικά ο κώδικας που είναι υπεύθυνος για την λειτουργία του μικροελεγκτή , την μετατροπή των αναλογικών μετρήσεων και την αποστολή τους μέσω UART στο δεύτερο επίπεδο του συστήματος. Ο κώδικας αυτός αναλαμβάνει να ρυθμίσει σωστά τις πόρτες του μικροελεγκτή καθώς και τα pins που χρησιμοποιούνται για την λήψη των μετρήσεων από τον αισθητήρα θερμότητας , την ενεργοποίηση και ρύθμιση του κυκλώματος UART του μικροελεγκτή καθώς και την μετατροπή των μετρήσεων και την αποστολή τους μέσω UART.



Εικόνα 22 Διάγραμμα ροής firmware firmware

Το πρώτο στάδιο της ροής του κώδικας είναι να αρχικοποιήσει την κατεύθυνση της πόρτας A που όπως έχει αναφερθεί είναι η πόρτα που φέρει το ADC. Για αυτόν τον σκοπό θα χρειαστεί να αρχικοποιηθεί ο DDRA καταχωρητής με την τιμή 0x00. Αναλυτικότερα η μορφή 0x00 μπορεί να γραφτεί ως 0b0000000.

Το 0b σημαίνει για την γλώσσα προγραμματισμού C ότι καταχωρούμε σε δυαδικό σύστημα σε μια μεταβλητή. Για τον προγραμματισμό ενός AVR η θέση που τοποθετούμε το 1 σε αυτή την αλληλουχία υποδηλώνει πιο pin της πόρτας θα έχει κατεύθυνση εξόδου. Η αντιστοιχία των pin με την δυαδική αλληλουχία γίνεται με κατεύθυνση από δεξιά προς αριστερά. Έτσι αν η αλληλουχία είχε την 0b00000001 τότε θα σήμαινε ότι το pin 1 της συγκεκριμένης πόρτας λειτουργεί ως έξοδος.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Pin** | - | 7 | 6 | 5 | 4 | 3 | 2 | 1 |
| **Δυαδική Αλληλουχία** | 0b | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Με τον παραπάνω πίνακα δίνεται το οπτικό ερέθισμα για την κατανόηση της έννοιας που εξηγήθηκε για την χρήση της δυαδικής αλληλουχίας στην αρχικοποίηση του καταχωρητή DDRΑ. Επειδή θα χρησιμοποιηθεί μόνο το πρώτο pin της συγκεκριμένης πόρτας και επειδή τα υπόλοιπα pin της δεν χρήζουν ιδιαίτερης προσοχής στην παρούσα φάση θα αρχικοποιηθούν όλα τα pins της πόρτας ως pins εισόδου. Αυτό μεταφράζεται στον παρακάτω κώδικα της γλώσσας C.

DDRA = 0x00;

Με την παραπάνω γραμμή κώδικα όλα τα pins του καταχωρητή έχουν ρυθμιστεί ως pins εισόδου. To 0x00 είναι η δεκαεξαδική μορφή της δυαδικής αλληλουχίας και χρησιμοποιείται κυρίως για συντομία στον κώδικα και είναι και η μορφή που προτιμήθηκε να χρησιμοποιηθεί σε ολόκληρο τον κώδικα.

Το επόμενο βήμα του κώδικα είναι η ενεργοποίηση του ADC και της συχνότητας του. Αυτό θα γίνει με τις παρακάτω εντολές:

ADCSRA = 0x87; /\* Enable ADC, with freq/128 \*/

ADMUX = 0x40; /\* Vref: Avcc, ADC channel: 0 \*/

Το ADC Control and Status Register ή αλλιώς ADCSRA αναλαμβάνει να ενεργοποιήσει το ADC του μικροελεγκτή.

Οι παραπάνω εντολές μπορούν να ενταχθούν σε μια μέθοδο με κύριο σκοπό την αρχικοποίησει του μικροελεγκτή.

void ADC\_Init(){

    DDRA = 0x00;     /\* Make ADC port as input \*/

    ADCSRA = 0x87; /\* Enable ADC, with freq/128 \*/

    ADMUX = 0x40; /\* Vref: Avcc, ADC channel: 0 \*/

}

Το επόμενο βήμα του firmware είναι να ενεργοποιήσει το USART του μικροελεγκτή ώστε να έχει την δυνατότητα αποστολής των τιμών. Αυτό απαιτεί την ρύθμιση του ρολογιού στο σωστό BaudRate , που στην συγκεκριμένη υλοποίηση είναι 1200 , και την ενεργοποίηση του USART ανοίγοντας τα TX και RX pins του μικροελεγκτή. Αυτό θα επιτευχθεί με τον παρακάτω κώδικα

void usart\_init(){

    UBRRH = 0;          /\* BaudRate - Clock Set \*/

    UBRRL =51;          /\* BaudRate - Clock Set \*/

    UCSRB|= (1<<RXEN)|(1<<TXEN); /\* Enable Tx and Rx\*/

    UCSRC |= (1 << URSEL)|(3<<UCSZ0);

}

Με τις δύο αυτές μεθόδους έχει επιτευχθεί η αρχικοποίηση του ADC του μικροελεγκτή καθώς και η αρχικοποίηση του UART. Η κύρια επαναληπτική διαδικασία του υποσυστήματος μπορεί πλέον να ξεκινήσει να δέχεται τις τιμές από το ADC να τις μετατρέπει σε βαθμούς κελσίου και στην συνέχεια να τις αποστέλει. Η μορφή της επαναληπτικής διαδικασίας είναι η παρκάτω:

    while(1)    {

     celsius = (ADC\_Read(0)\*4.88); /\*Reading Value and converting to Celsius\*/

     celsius = (celsius/10.00);   /\*Reading Value and converting to Celsius\*/

     itoa(celsius,Temperature,10);/\*Converts to String\*/

usart\_string\_transmit(Temperature);

usart\_string\_transmit("\n");

        /\*Transmits string to PC\*/

        usart\_data\_transmit(0x0d);

     \_delay\_ms(15);

     memset(Temperature,0,10);

    }

Η συγκεκριμένη διαδικασία επαναλαμβάνεται επ’ αόριστον στέλνοντας συνεχόμενα μετρήσεις από τον αισθητήρα στο UART. Όπως φαίνεται η παραπάνω διαδικασία αποτελείται από πολλές μικρότερες μεθόδους που η κάθε μία αναλαμβάνει μια πολύ συγκεκριμένη εργασία μέσα στον κώδικα. Παρακάτω θα αναλυθούν μία προς μία καθώς και ο τρόπος λειτουργίας τους.

Η μέδοθος **ADC\_Read** είναι υπεύθυνη για την λήψη της τιμής από το ADC παίρνοντας ως όρισμα το κανάλι εισαγωγής που στην συγκεκριμένη υλοποίηση είναι 0.

int ADC\_Read(char channel){

    ADMUX = 0x40 | (channel & 0x07);/\* set input channel to read \*/

    ADCSRA |= (1<<ADSC); /\* Start ADC conversion \*/

    while (!(ADCSRA & (1<<ADIF))); /\* Wait until end of conversion by polling ADC interrupt flag \*/

    ADCSRA |= (1<<ADIF); /\* Clear interrupt flag \*/

    \_delay\_ms(1); /\* Wait a little bit \*/

    return ADCW;/\* Return ADC word \*/

}

Η μέθοδος **itoa** είναι καθαρά κομμάτι της γλώσσας C και δεν πρόκειται για κάποια εξειδικευμένα μέθοδος για τον προγραμματισμός του μικροελεγκτή και σκοπός της είναι η μετατροπή από ακέραιο σε αλφαριθμητικό.

Η μέθοδος **usart\_string\_transmit** αναλαμβάνει την προσπέλαση του αλφαριθμητικού που έχει παραχθεί από την μετατροπή της μέτρησης και αποστέλλει κάθε χαρακτήρα του αλφαριθμητικού αυτού στην μέθοδο **usart\_data\_transmit.**

void usart\_string\_transmit(char \*string){

    while(\*string)    {

        usart\_data\_transmit(\*string++);

    }

}

Η **usart\_data\_transmit** είναι υπεύθυνη για την αποστολή των δεδομένων

void usart\_data\_transmit(unsigned char data ){

    while ( !( UCSRA & (1<<UDRE)) );

    UDR = data;

}

Στην συνέχεια ο κώδικας αποστέλλει το “\n” , το σύμβολο για το κλείσιμο της γραμμής , κάτι που είναι απαραίτητο για να γίνεται γνωστό στο σημείο λήψης ότι η αποστολή της μέτρησης τελείωσε.

Ο ολοκληρωμένος κώδικας του firmware είναι ο παρακάτω:

#include <avr/io.h>

#include <util/delay.h>

#include <string.h>

#include <stdio.h>

#include <stdlib.h>

void ADC\_Init(){

    DDRA = 0x00;     /\* Make ADC port as input \*/

    ADCSRA = 0x87; /\* Enable ADC, with freq/128 \*/

    ADMUX = 0x40; /\* Vref: Avcc, ADC channel: 0 \*/

}

void usart\_init(){

    UBRRH = 0;          /\* BaudRate - Clock Set \*/

    UBRRL =51;          /\* BaudRate - Clock Set \*/

    UCSRB|= (1<<RXEN)|(1<<TXEN);

    UCSRC |= (1 << URSEL)|(3<<UCSZ0);

}

int ADC\_Read(char channel){

    ADMUX = 0x40 | (channel & 0x07); /\* set input channel to read \*/

    ADCSRA |= (1<<ADSC); /\* Start ADC conversion \*/

    while (!(ADCSRA & (1<<ADIF))); /\* Wait until end of conversion by polling ADC interrupt flag \*/

    ADCSRA |= (1<<ADIF); /\* Clear interrupt flag \*/

    \_delay\_ms(1); /\* Wait a little bit \*/

    return ADCW; /\* Return ADC word \*/

}

void usart\_data\_transmit(unsigned char data ){

    while ( !( UCSRA & (1<<UDRE)) );

    UDR = data;

}

void usart\_string\_transmit(char \*string){

    while(\*string)    {

        usart\_data\_transmit(\*string++);

    }

}

int main(){

    char Temperature[10];

    float celsius;

char buffer[20];

    ADC\_Init(); /\* initialize ADC\*/

    usart\_init();

    while(1)    {

     celsius = (ADC\_Read(0)\*4.88); /\*Reading Value and converting to Celsius\*/

     celsius = (celsius/10.00);   /\*Reading Value and converting to Celsius\*/

     itoa(celsius,Temperature,10);/\*Converts to String\*/

usart\_string\_transmit(Temperature);

usart\_string\_transmit("\n");

        /\*Transmits string to PC\*/

        usart\_data\_transmit(0x0d);

     \_delay\_ms(15);

     memset(Temperature,0,10);

    }

}

# Μεταγλώττιση και Εγγραφή του Firmware

Μετά την ολοκλήρωση του κώδικα θα πρέπει να ακολουθηθούν κάποια συγκεκριμένα βήματα για την μεταγλώττιση του κώδικα και την μετατροπή του σε .hex αρχείο ώστε να γίνει η εγγραφή του μέσω USBASP στον μικροελεγκτή. Για τον σκοπό αυτό θα χρησιμοποιηθεί η παρακάτω εντολή:

**avr-gcc -Wall -g -Os -mmcu=atmega16 -o firmware.bin firmware.c**

Με την παραπάνω εντολή εκτελείται το πρώτο βήμα στην μεταγλώττιση του αρχείου. Αυτό έχει ως αποτέλεσμα να παραχθεί ,για τον μικροελεγκτή atmega16, ένα .bin από το αρχείου του κώδικα. Αυτό το .bin αρχείο είναι ο παραγόμενος κώδικας μηχανής από τον μεταγλωττιστή. Το δεύτερο στάδιο απαιτεί την μετατροπή του .bin αρχείου σε αρχείο .hex. Αυτό γίνεται να για μπορέσει να γίνει η εγγραφή του από τον **USBASP** στον μικροελεγκτή. Αυτό θα πραγματοποιηθεί με την παρακάτω εντολή:

**avr-objcopy -j .text -j .data -O ihex readtemp.bin readtemp.hex**

Σε αυτό το σημείο ο κώδικας έχει μετατραπεί κατάλληλα και είναι έτοιμος για εγγραφή στον μικροελεγκτή.

Χρησιμοποιώντας τα σχήματα 2.2-8 και 2.2-9 του κεφαλαίου 2 επιτυγχάνεται η σύνδεση του USBASP με τον μικροελεγκτή.

Είναι εφικτή σε αυτό το σημείο η εγγραφή του του firmware στον μικροελεγκτή για να ξεκινήσει η λειτουργία του υποσυστήματος Megaman. Αυτό θα πραγματοποιηθεί με την παρακάτω εντολή:

Εικόνα 23 Διασύνδεση μικροελεγκτή - USBASP

**avrdude -p atmega16 -c usbasp -U flash:w:readtemp.hex:i -F -P usb**

Με την χρήση του προγράμματος avrdude η εγγραφή του firmware γίνεται επιτυχημένα μέσω της συσκευής USBASP.

Το υποσύστημα του Megaman είναι πλέον ολοκληρωμένο και μπορεί να ξεκινήσει την συλλογή των μετρήσεων , την μετατροπή τους και την αποστολή τους στο δεύτερο επίπεδο του συστήματος.

1. ο κεφαλαιο

Αναπτυξη λογισμικου επεξεργασιασ αναλογικων μετρησεων

# Εισαγωγή

Σε αυτό το κεφάλαιο θα αναλυθεί η ανάπτυξη του δεύτερου επιπέδου του συστήματος , του λογισμικού λήψης και επεξεργασίας των αναλογικών μετρήσεων Electro. Θα παρουσιαστούν κάποια δομικά κομμάτια κώδικα του λογισμικού , ο τρόπος λειτουργίας του καθώς και οι χρήσεις που παρουσιάζει σε πραγματικά σενάρια. Θα γίνει παρουσίαση του λογισμικού που απαιτήθηκε για την υλοποίηση του Electro καθώς και η παραμετροποίηση του λογισμικού για την εύκολη ανάπτυξη του κώδικα. Λόγω του μεγέθους του κώδικα είναι πρακτικά αδύνατο να παρουσιαστεί ολόκληρος ο κώδικας και να αναλυθεί σε γραπτό λόγο.

# Λειτουργίες Electro

Το λογισμικό Electro αποτελεί το γραφικό περιβάλλον επεξεργασίας των αναλογικών μετρήσεων που στέλνει το πρώτο επίπεδο του συστήματος. Προσφέρει τριών ειδών επεξεργασίας των αναλογικών μετρήσεων:

1. Live Plotting – Ζωντανή απεικόνιση των αναλογικών μετρήσεων
2. Sampling – Δειγματοληψία αναλογικών μετρήσεων σε συγκεκριμένο εύρος βημάτων
3. Handlers – Εκτέλεση Shell Script ή Electro Action όταν η αναλογική μέτρηση φτάσει σε οριζόμενο από τον χρήστη μέγεθος.
4. Άνοιγμα αρχείων που έχουν αποθηκευτεί από την δειγματολειψία.

Το κάθε ένα είδος επεξεργασίας θα αναλυθεί σε επόμενες παραγράφους αναλυτικά. Το συγκεκριμένο λογισμικό προσφέρει επίσης την επιλογή της συσκευής καθώς και την ρύθμιση του BaudRate για την λήψη των μετρήσεων.

Ο Εlectro δίνει στον τελικό χρήστη έναν εύκολο τρόπο αλληλεπίδρασης με το επίπεδο του Megaman χωρίς χρήση επίπονων εντολών σε περιβάλλον command line και αποτελεί ένα παράδειγμα για την εμπορική μορφή του συστήματος καθώς και την δυνατότητα να γίνει εκμετάλλευση ενός ενσωματωμένου συστήματος με εύκολο τρόπο.

Το λογισμικό έχει υλοποιηθεί με χρήση του PyQt5 και είναι γραμμένο στην γλώσσα προγραμματισμού Python. Εσωτερικά γίνεται χρήση των βιβλιοθηκών που αναλύθηκε στην παράγραφο 2.8 για την λήψη των μετρήσεων (PySerial) καθώς και την ζωντανή απεικόνιση των μετρήσεων (PyQtGraph)

O Electro αποτελείται από τρεις βασικές καρτέλες που ξεχωρίζουν νοηματικά και οπτικά τους τρόπους επεξεργασίας των αναλογικών μετρήσεων. Στην κορυφή του λογισμικού και εκτός όλων των καρτελών βρίσκονται η επιλογή της συσκευής καθώς και ο επιλογέας του baudrate.



Εικόνα 24 Γραφικό περιβάλλον Electro

# Επεξήγηση βοηθητικών εργαλείων Electro

Στα αριστερά του παραθύρου βρίσκεται ένα toolbar με βοηθητικά εργαλεία του Electro που μπορούν εύκολα και γρήγορα να ξεκινήσουν τις βασικές διαδικασίες του προγράμματος. Αναλυτικότερα:

1.  Άνοιγμα αρχείων δειγματολειψίας. Το συγκεκριμένο πλήκτρο του λογισμικού μπορεί να κάνει άνοιγμα των αρχείων δειγματολειψίας που έχει παραχθεί και να κάνει γραφική απεικόνιση των δεδομένων.
2.  Εκκίνηση επιλεγμένης διαδικασίας. Όπως παρουσιάστηκε το συγκεκριμένο λογισμικού έχεις τρεις βασικές καρτέλες επεξεργασίας. Όταν έχει επιλεχθεί μια από την τρεις καρτέλες το συγκεκριμένο πλήκτρο κάνει εκκίνηση της διαδικασίας της συγκεκριμένης καρτέλας
3.  Ανανέωση των αναγνωρισμένων συσκευών. Το συγκεκριμένο πλήκτρο κάνει ανανέωση των αναγνωρισμένων συσκευών λήψης αναλογικών μετρήσεων. Γίνεται χρήση του συγκεκριμένου πλήκτρου σε περιπτώσεις όπου έχουν προστεθεί ή αφαιρεθεί από τις θύρες του υπολογιστικού συστήματος , συσκευές λήψης αναλογικών μετρήσεων
4.  Εκκαθάριση λίστας αναγνωρισμένων συσκευών. Το πλήκτρο αυτό κάνει εκκαθάριση των αναγνωρισμένων συσκευών.
5.  Επαναφορά προεπιλεγμένων ρυθμίσεων. Με το πλήκτρο αυτό ο Electro επαναφέρει όλες του τις καρτέλες στις προεπιλεγμένες τιμές καθαρίζοντας τα μονοπάτια αποθήκευσης.
6.  Επαναφορά προεπιλεγμένων ρυθμίσεων σε επιλεγμένη καρτέλα. Ο χρήστης έχοντας επιλέξει μια συγκεκριμένη καρτέλα του Electro μπορεί να επαναφέρει την καρτέλα αυτή στις προεπιλεγμένες της ρυθμίσεις με την χρήση του συγκεκριμένου πλήκτρου.

# Επεξήγηση Live Plotting

Μια από τις κεντρικές λειτουργίες του Electro είναι η λήψη των αναλογικών μετρήσεων και η ζωντανή τους απεικόνιση (Live Plotting) με χρήση του PyQtGraph.



Εικόνα 25 Live plotting Electro

Στην Εικόνα 25 φαίνεται η καρτέλα της λειτουργίας του Live Plotting. Στην συγκεκριμένη καρτέλα ο χρήστης μπορεί να θέσει παραμετρικά αν θέλει κατά την διάρκεια του Live Plotting να γίνεται καταγραφή των τιμών στην κονσόλα του plotting έχοντας όχι μόνο το γράφημα αλλά και την δυνατότητα προβολής των αναλογικών μετρήσεων σε μορφή κειμένου. Αυτή η λειτουργία ενεργοποιείται με την ενεργοποίηση του **Logging.**

****

Εικόνα 26 Logging των μετρήσεων στην κονσόλα

Ο χρήστης μπορεί να επιλέξει για το αν θέλει να κάνει καταγραφεί των αναλογικών μετρήσεων κατά την διάρκεια του Live Plotting. Η επιλογή που ενεργοποιεί την συγκεκριμένη λειτουργεία είναι η File Recording. Με αυτή την επιλογή ο χρήστης ενεργοποιεί την καταγραφή των αναλογικών μετρήσεων και μπορεί να αποθηκεύσει το παραγόμενο αρχείο σε μονοπάτι της επιλογής του. Σε αυτό το σημείο έχει την δυνατότητα παραμετροποίησης του ονόματος του αρχείου που θα αποθηκευτεί. Σε περίπτωση που δεν ενεργοποιηθεί το Custom Name τότε το αρχείο θα πάρει ένα αυτόματα παραγόμενο όνομα.



Εικόνα 27 File recording των μετρήσεων

Πατώντας το πλήκτρο εκκίνησης μετά την κατάλληλη παραμετροποίηση ο χρήστης ξεκινάει την διαδικασία του live plotting.

Εικόνα 28 Live Plotting σε εξέλιξη

Στην Εικόνα 28 φαίνεται το παράθυρο εμφάνισης των αναλογικών μετρήσεων και αποτελείται από ένα παράθυρο που παράγεται από το pyqtgraph , την βιβλιοθήκη που χρησιμοποιεί εσωτερικά ο Electro για να κάνει render τις αναλογικές μετρήσεις. Σε αυτό το παράθυρο γίνονται εμφανή δύο γραφήματα , το αριστερό γράφημα αναφέρεται στην θερμοκρασία που δέχεται live από το ενσωματωμένο σύστημα και το δεξί γράφημα είναι η μέση «ιστορική» θερμοκρασία από την εκκίνηση της διαδικασίας.

# Επεξήγηση Sampling

Η δεύτερη βασική λειτουργία του Electro είναι η δειγματοληψία αναλογικών μετρήσεων και η offline απεικόνιση τους. Αυτό σημαίνει ότι ο χρήστης μπορεί να επιλέξει εύρος βημάτων για την εκτελέσει την δειγματοληψία.

Εικόνα 29 Sampling - Καρτέλα δειγματολειψίας

Στην συνέχεια μπορεί να επιλέξει όπως και στο live plotting το μονοπάτι αποθήκευσης του αρχείου που θα παραχθεί , αν θέλει να έχει παραμετροποιημένο όνομα και όχι το αυτόματα παραγόμενο και αν θέλει στο τέλος της δειγματοληψίας αυτόματα να γίνει render η δειγματολειψία.

Ο ορισμός του εύρους της δειγματοληψίας γίνεται επιλέγοντας στα spinners της εφαρμογής τις επιθυμητές τιμές σε μονάδα μέτρησης βήματος.



Εικόνα 30 Ορισμός εύρους δειγματοληψίας

Το βήμα είναι μία θεωρητική μονάδα μέτρησης και σημαίνει πότε ολοκληρώνεται ένας κύκλος μέτρησης , δηλαδή η λήψη της αναλογικής μέτρησης η μετατροπή της , η αποστολή της μέσω Uarτ και τέλος η λήψης της και καταγραφή της από τον Electro, αποτελεί ένα βήμα. Στην Εικόνα 31 γίνεται οπτική αναπαράσταση της συγκεκριμένης έννοιας



Εικόνα 31 Πλήρης κύκλος μέτρησης - Step/Βήμα

Η επιλογή του ονόματος και του μονοπατιού αποθήκευσης γίνεται με παρόμοιο τρόπο όπως και στην καρτέλα του live plotting.

# Επεξήγηση Handlers

Οι Handlers αποτελούν ένα εργαλείο του Electro που μπορούν κάτω από συγκεκριμένες προϋποθέσεις να εκτελέσουν μια ενέργεια όταν η μέτρηση φτάσει σε οριζόμενα από τον χρήστη όρια. Αυτό σημαίνει ότι όταν ο χρήστης επιλέξει να συμβεί μια ενέργεια στην περίπτωση που η θερμοκρασία ξεπεράσει π.χ τους 100 βαθμούς κελσίου , τότε ο Electro όταν αναγνωρίσει ότι η θερμοκρασία έφτασε σε αυτό το σημείο θα εκτελέσει αυτόματα μια ενέργεια που έχει οριστεί από τον χρήστη.

Ο Electro την παρούσα φάση υποστηρίζει τριών ειδών ενέργειες που μπορούν να εκτελεστούν όταν η θερμοκρασία ξεπεράσει ένα όριο τιμής ή όταν η θερμοκρασία βρίσκεται κάτω από ένα συγκεκριμένο όριο τιμής. Στην δεύτερη περίπτωση η ενέργεια θα εκτελείται σε κάθε ένα βήμα μέτρησης.



Εικόνα 32Καρτέλα Handlers του Electro

Οι τρεις τύποι ενεργειών που υποστηρίζονται αυτή την στιγμή από το λογισμικό είναι η εκτέλεση εμφάνισης μηνύματος , ο αυτόματος τερματισμός της μέτρησης καθώς και η εκτέλεση ενός shell script. Η τελευταία επιλογή ανοίγει τον δρόμο σε άπειρες εφαρμογές καθώς σε αυτό το σημείο το λογισμικό έχει την ικανότητα να εκτελέσει ένα custom python script που μπορεί να ξεκινήσει μια πολύπλοκη διαδικασία υπολογισμού.



Εικόνα 33Επιλογή Action εκτέλεσης

Στην Εικόνα 33 φαίνεται τα drop downs επιλογές των διαθέσιμών Actions του λογισμικού. Στην συγκεκριμένη εικόνα ο Electro έχει οριστεί να εκτελέσει την εμφάνιση ενός μηνύματος όταν η θερμοκρασία έχει ξεπεράσει τους 50 βαθμούς κελσίου. Όπως φαίνεται στην Εικόνα 34 ο χρήστης μπορεί να επιλέξει πότε θα εκτελεστεί η επιλεγμένη ενέργεια.



Εικόνα 34 Επιλογή ορίου εκτέλεσης action

Κατά την διάρκεια εκτέλεσης του handling ο Electro ενημερώνει τον χρήστη εμφανίζοντας ένα label



Εικόνα 35 Monitoring execution

Όπως φαίνεται στην Εικόνα 35 το λογισμικό έχει ρυθμιστεί να εκτελέσει ένα python script με το όνομα QSortAlgo περνώντας όρισμα –F όταν η θερμοκρασία που παρακολουθείται γίνει μεγαλύτερη των 89 βαθμών κελσίου. Στην περίπτωση που ο χρήστης θέλει να σταματήσει την παρακολούθηση μπορεί να πατήσει το πλήκτρο Stop όπου θα αναγκάσει τον Electro να ακυρώσει την παρακολούθηση της θερμοκρασίας.



Εικόνα 36 Πλήκτρο ακύρωση του Handling

# Ανάπτυξη λογισμικού Electro

Ο Electro αποτελείται από δύο επίπεδα κώδικα που αναλαμβάνουν την λήψη των αναλογικών μετρήσεων από το υποσύστημα του Megaman, το rendering των τιμών και την διαχείριση του γραφικού περιβάλλοντος. Μπορεί να γίνει διάκριση των δύο επιπέδων σε επίπεδο GUI και σε επίπεδο Renderer. Το επίπεδο του GUI αναλαμβάνει την έκθεση των λειτουργιών του συστήματος με εύκολο οπτικό τρόπο προσφέροντας γρήγορα αλληλεπίδραση μεταξύ χρήστη και συστήματος. Το επίπεδο Renderer αποτελεί το επίπεδο όπου λαμβάνει τις παραμετροποιήσεις που θέτει ο χρήστης στο GUI και ξεκινάει τις απαραίτητες ενέργειας ανάλογα με την διαδικασία που έχει επιλεχθεί. Ο Renderer είναι υπεύθυνος για το άνοιγμα της σύνδεσης με το επίπεδο του Megaman καθώς και για την λήψη και επεξεργασία των αναλογικών μετρήσεων. Όταν ο χρήστης έχει ορίσει στο GUI την επιλεγμένη συσκευή και το κατάλληλο baudrate το GUI εκκινεί τον renderer με χρήση της Popen της Python περνώντας της την επιλεγμένη συσκευή και το baudrate. Με αυτόν τον τρόπο επιτυγχάνεται η απελευθέρωση του γραφικού περιβάλλοντος και επιτρέπουν στην χρήση ακόμα και όταν ο Renderer εκτελεί κάποια διαδικασία. Ο Renderer του Electro λέγεται MRenderer και παράγεται από το Megaman + Renderer.



Εικόνα 37 Διάγραμμα επιπέδων Electro

Η εκκίνηση του λογισμικού γίνεται μέσω της κονσόλας από το κεντρικό φάκελο του project πληκτρολογώντας την παρακάτω εντολή.

**python3 init.py rasp**

Το όρισμα **rasp** απαιτείται μόνο στην περίπτωση που ο Electro τρέχει στο Raspberry Pi και δίνει στον Electro την απαραίτητη πληροφορία ώστε να ανοίξει το παράθυρο σε μικρότερο μέγεθος και εκκινώντας τον renderer στην σωστή έκδοση της Python που είναι διαθέσιμη στο Raspberry Pi.

Το αρχείο init.py αποτελεί το σημείο εκκίνησης του Electro και σε αυτό το στάδιο η κεντρική κλάση του υποσυστήματος αρχικοποιείται με τις σωστές ρυθμίσεις. Το GUI του Electro μπορεί πλέον να φανεί στην οθόνη του χρήστη και η επεξεργασίας των αναλογικών μετρήσεων μπορεί να προχωρήσει. Όταν ο χρήστης προχωρήσει στην επιθυμητή παραμετροποίηση και πατήσει το πλήκτρο εκκίνησης της διαδικασίας το GUI αναλαμβάνει την αποστολή των παραμέτρων στον MRenderer για να γίνει η εκτέλεση της επιθυμητής ενέργειας.

Εδώ αξίζει να σημειωθεί ότι ο MRenderer κάνει εκκίνηση των λειτουργιών μέσω της χρήσης του wrapper RenderingThreadLooper. Ο συγκεκριμένος wrapper κάνει εσωτερικά χρήση του Python thread και δίνει δυνατότητα εκτέλεσης ενός κώδικα παράλληλα από την εκτέλεση του κύριου thread του MRenderer προσφέροντας την δυνατότητα του Wait αλλά και την εκτέλεση μέσα σε ένα συγκεκριμένο εύρος βημάτων , ιδιαίτερα χρήσιμη δυνατότητα για την λειτουργία του Sampling. Παρακάτω βλέπουμε τον κώδικα του wrapper

class RenderingThreadLooper:

def \_\_init\_\_(self, target, timeout=-1, name="Processing\_Thread" , onfinishexecution = None):

self.executing = True

self.runnablemethod = target

self.timeout = timeout

self.threadname = name

self.renderthread = None

self.onfinishexec = onfinishexecution

def executiontarget(self):

if self.timeout == -1:

while self.executing:

self.runnablemethod()

else:

for i in range(0, self.timeout):

self.runnablemethod()

print(i)

def finishexecution(self):

self.executing = False

if self.onfinishexec is not None:

self.onfinishexec()

print(self.renderthread.name + " finished")

def run(self):

self.renderthread = Thread(target=self.executiontarget)

self.renderthread.setName(self.threadname)

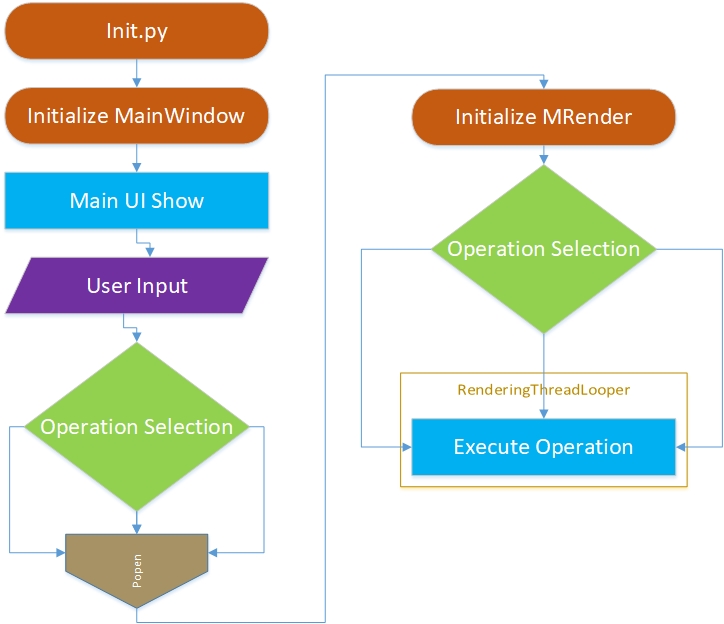
self.renderthread.start()

print(self.renderthread.name + " started")

def wait(self):

self.renderthread.join()

Στην εικόνα 38 βλέπουμε το διάγραμμα του Electro από την εκκίνηση του λογισμικό , την αρχικοποίηση του και την επιλογή της επιθυμητής λειτουργίας μέχρι και την αποστολή της παραμετροποίησης στον MRenderer για εκτέλεση της λειτουργίας μέσω του RenderingThreadLooper.



Εικόνα 38 Electro Flow

# Σχεδίαση και προγραμματισμός GUI

Για την ανάπτυξη του Electro έχει χρησιμοποιηθεί το Pycharm IDE για τον προγραμματισμό στην γλώσσα Python και ο Qt Designer για την σχεδίαση του GUI και την ενσωμάτωση των resources (π.χ εικονίδια εφαρμογής).

Η διαδικασία σχεδίασης του GUI περιλαμβάνει τον σχεδιασμό του στον Qt Designer την ενσωμάτωση των απαιτούμενων πόρων , και στην συνέχεια την μετατροπή του παραγόμενου .xml κώδικα από τον Qt Designer σε κώδικα Python με το εργαλείο pyuic5. Το συγκεκριμένο εργαλείο έρχεται μαζί με το πακέτο PyQt5 και έχει την δυνατότητα να μετατρέπει το αρχείο που παράγει ο Qt Designer σε κώδικα Python ώστε να γίνεται εύκολα η ενσωμάτωση του UI μέσω κώδικα Python. Αυτό επιτυγχάνεται χρησιμοποιώντας την παρακάτω εντολή:

**Pyuic5 UI.ui -o UI.py**

Επειδή η ανάπτυξη του UI γίνεται παράλληλα με την ανάπτυξη του κώδικα και θα χρειαστεί να καλείται συνέχεια η διαδικασία μετατροπής του UI , θα οριστεί ένα external tool στο PyCharm ώστε κάθε φορά που θα εκτελείται μια ενημερωμένη έκδοση του electro θα ενημερώνεται ταυτόχρονα και ο κώδικας του UI. Αυτό θα επιτευχθεί πηγαίνοντας στις ρυθμίσεις του project του Pycharm από το menu File -> Settings -> Extarnal Tools. Σε αυτό το σημείο μπορεί ο χρήστης πατώντας το + να προσθέσει ένα εξωτερικό εργαλείο για τον pycharm με αυτόν τον τρόπο είναι σχετικά εύκολο να ενημερώνεται το python αρχείο με όλες τις αλλαγές που έχουν γίνει από τον Qt Designer. Στην εικόνα 39 φαίνεται ένα παράδειγμα για τον τρόπο παραμετροποίησης του pyuic5 ως external tool στο Pycharm.



Εικόνα 39 Ρύθμιση external tool python

Στην παρακάτω εικόνα φαίνεται το γραφικό περιβάλλον του Qt Designer που χρησιμοποιείται για την σχεδίαση του UI του Electro.



Εικόνα 40 Qt Designer UI

Κατά τον σχεδιασμό του UI στον Qt Designer ο χρήστης πρέπει να ορίσει τα απαραίτητα ονόματα των αντικειμένων (objectName) στον property toolbox του Qt Designer για να υπάρχει πρόσβαση του αντικειμένου από τον κώδικα της Python.



Εικόνα 41Property Editor of Qt Designer

Με αυτόν τον τρόπο κατά την ενσωμάτωση του κώδικα του UI στον κεντρικό κώδικα της python ο χρήστης έχει την δυνατότητα διαχείρισης του αντικειμένου. Παρακάτω φαίνεται στον κώδικα πως επιτυγχάνεται αυτό.

class MainUI(QMainWindow):

RunsOnRaspberry = False

def \_\_init\_\_(self, rasp, initfilepath):

super().\_\_init\_\_()

self.initfilepath = initfilepath

self.monitorthread = None

self.ui = UI.mainui.Ui\_MainWindow()

MainUI.RunsOnRaspberry = rasp

self.ui.setupUi(self)

self.initializeMainWindowSize()

self.initializeelectro()

self.connectuicomponetstosignal()

Στις παρακάτω γραμμές κώδικα φαίνεται στην \_\_init\_\_ μέθοδο της κεντρικής κλάσης η αρχικοποίηση του κώδικα του UI καθώς και η ανάθεση του ως κεντρικό παράθυρο για την εφαρμογή

self.ui = UI.mainui.Ui\_MainWindow()

self.ui.setupUi(self)

Η έννοια της του objectName γίνεται ορατή σε πολλά σημεία του κώδικα οι παρακάτω γραμμές κώδικα όμως αποτελούν χαρακτηριστικό παράδειγμα

def initializeliveplottingtab(self):

self.ui.filecheckbox.setChecked(False)



Εικόνα 42 Ανάθεση ονόματος στο filecheckbox

Επιλέγοντας στον designer το επιθυμητό widget μπορεί ο χρήστης να θέσει objectname στο αντικείμενο αυτό καθώς και οποιαδήποτε άλλη παραμετροποίηση αυτός επιθυμεί. Φαίνεται λοιπόν ότι θέτοντας στον designer στο widget του File Recording objectname filecheckbox από τον κώδικα της python μπορεί να γίνει εκμετάλλευση του αντικειμένου αυτού καλώντας το με ως self.ui.filecheckbox. Με αυτόν τον τρόπο ο χρήστης μπορεί στο runtime να αλλάξει properties των αντικειμένων δυναμικά.

# Εκκίνηση MRenderer

Η εκκίνηση του MRenderer γίνεται από μια κεντρική μέθοδο στην κύρια κλάση του Electro που ονομάζεται startmainproc. Η συγκεκριμένη μέθοδος είναι υπεύθυνη για την κατανόηση της ενέργειας που θέλει να εκτελέσει ο χρήστης και της εκκίνησης του MRenderer με τα κατάλληλα ορίσματα. Παρακάτω βλέπουμε την κεντρική μέθοδο:

def startmainproc(self):

if self.ui.selecteddevicecombobox.findText("None"):

if self.ui.monitoringlabel.isVisible():

return

selectedtab = self.ui.tabWidget.currentWidget()

if selectedtab is self.ui.liveplottingtab:

self.startliveplotting()

elif selectedtab is self.ui.samplingtab:

self.startsampling()

elif selectedtab is self.ui.handlerstab:

self.startmonitoring()

else:

print("unknown tab selected")

else:

self.showmessagebox("There is no proper device selected")

Η συγκεκριμένη μέθοδος αναγνωρίσει μέσω της επιλεγμένης καρτέλα για το ποια λειτουργία θα εκκινήσει και τα σωστά ορίσματα που θα στείλει στον renderer.

Στην περίπτωση που ο χρήστης έχει επιλέξει την καρτέλα του live plotting τότε ο κώδικας αναλαμβάνει να εκκινήσει τον MRenderer με αυτή την λειτουργία και να του περάσει όλη την παραμετροποίηση που ο χρήστης έχει ορίσει στην καρτέλα αυτή, παρακάτω φαίνεται ο κώδικας του που εκκινεί τον renderer στην λειτουργία live plotting.

def startliveplotting(self):

if self.ui.liveplottingcheckbox.isChecked()

or self.ui.loggingcheckbox.isChecked()

or self.ui.filecheckbox.isChecked():

print(\_\_file\_\_)

Popen([self.getpythonversion(),

os.path.join(self.initfilepath, "Renderer/MRenderer.py"),

str(RendererOperationsType.LivePlotting.value),

self.ui.selecteddevicecombobox.currentText(),

self.ui.speedspinbox.text(),

self.getcompbinedfilename(), "None",

str(self.ui.loggingcheckbox.isChecked()),

str(self.ui.filecheckbox.isChecked()), "None"])

Ο συγκεκριμένος κώδικας εκκινεί τον MRenderer σε λειτουργία live plotting και του περνάει την επιλεγμένη συσκευή , το σωστό baudrate , το αν θα γίνει logging και file recording και σε ποια περιοχή θα αποθηκεύσει τα αρχεία. Στην περίπτωση που κάποια λειτουργία δεν υποστηρίζει ή δεν χρειάζεται κάποια ενέργεια του renderer τότε σε αυτό το όρισμα περνιέται “None” και μεταφράζεται από τον renderer σε default τιμή αρχικοποίησης.

Έτσι όταν ο χρήστης θέλει να εκτελέσει δειγματοληψία και διαλέξει την καρτέλα του sampling τότε ο renderer ξεκινάει μέσω του παρακάτω κώδικα.

def startsampling(self):

print(\_\_file\_\_)

check\_call([self.getpythonversion(),

os.path.join(self.initfilepath,"Renderer/MRenderer.py"),

str(RendererOperationsType.Sampling.value),

self.ui.selecteddevicecombobox.currentText(),

self.ui.speedspinbox.text(),

self.getcompbinedfilename2(),

self.ui.tospinbox.text(),

"None", "True",

str(self.ui.autoopenfilecheckbox.isChecked())])

Μια από τις κύριες διαφορές που χρήζει ιδιαίτερης προσοχής είναι η χρήση της check\_call και όχι της Popen. Ο λόγος που καλείται ο MRenderer μέσω αυτή της διαδικασίας είναι ότι κατά την διάρκεια του Sampling είναι επιθυμητό να κλειδώσει η εφαρμογή μέχρι να ολοκληρωθεί η διαδικασία. Στην περίπτωση όμως που ο χρήστης επιθυμεί να ενεργοποιήσει το handling τότε ο renderer ξεκινάει μέσω του RenderingThreadLooper. Αυτό επιτρέπει στο UI να λειτουργεί χωρίς να μπλοκάρεται από την διαδικασία του handling σε περίπτωση που ο χρήστης θέλει να ακυρώσει την διαδικασία του handling. Αυτό γίνεται γιατί ο σχεδιασμός της εφαρμογής προβλέπει handling για πολύ μεγάλα χρονικά διαστήματα κάτι που εμφανίζει την ανάγκη για την δυνατότητα ακύρωσης της συγκεκριμένης λειτουργίας. Κατά την διάρκεια του handling εμφανίζεται ένα label που αναγράφει “Monitoring”. Αυτό για να ενημερωθεί ο χρήστης ότι εκείνη την στιγμή η εφαρμογή εκτελεί handling. Το συγκεκριμένο label όμως θα πρέπει να γίνει μη ορατό όταν η διαδικασία ακυρωθεί ή ολοκληρωθεί. Σε αυτή την περίπτωση γίνεται χρήση του ορίσματος onfinishexecution του RenderingThreadLooper όπου κατά την ολοκλήρωση της λειτουργία ο RenderingThreadLooper αναλαμβάνει να εκτελέσει οποιαδήποτε μέθοδο περαστεί στο όρισμα αυτό.

def finishexecution(self):

self.executing = False

if self.onfinishexec is not None:

self.onfinishexec()

print(self.renderthread.name + " finished")

Στο παραπάνω κομμάτι κώδικα που βρίσκεται στην κλάση του RenderingThreadLooper βλέπουμε την υλοποίηση αυτής της λειτουργίας.

# Λειτουργία MRenderer

Η λειτουργία του MRenderer αποτελείται από 3 βασικά βήματα , την λήψη των παραμέτρων από τον Electro ,την αρχικοποίηση και την εκτέλεση της επιθυμητής λειτουργίας. Κατά την αρχικοποίηση του γίνεται και η επιλογή των παραμέτρων σε περίπτωση που περαστεί κάποιο όρισμα ως “None”, αυτό φαίνεται στο παρακάτω παράδειγμα

def GetTerminalLogging(argv):

if argv[6] == "None":

return False

else:

if argv[6] == "True":

return True

else:

return False

Ο κώδικας που φαίνεται παραπάνω αναλαμβάνει την αρχικοποίηση της λειτουργίας εμφάνισης των αναλογικών μετρήσεων στην κονσόλα κατά το runtime. Σε περίπτωση που ο κώδικας αυτός καλεστεί με όρισμα None τότε η default λειτουργία έχει οριστεί ως απενεργοποιημένη. Με ανάλογο τρόπο γίνεται η αρχικοποίηση και των υπόλοιπων ενεργειών του renderer. Δηλαδή με χρήση μεμονωμένων μεθόδων που αναλαμβάνουν να αρχικοποιήσουν τον renderer σύμφωνα με τα ορίσματα που έχουν ληφθεί από τον Electro. Τέλος ο renderer είναι έτοιμος να εκκινήσει την επιθυμητή λειτουργία.

Μια από τις βασικές ευθύνες του MRenderer είναι η μετατόπιση των αναλογικών μετρήσεων κατά την διαδικασία του rendering. Αυτό συμβαίνει γιατί ο τρόπος που λειτουργεί η βιβλιοθήκη του pyqtgraph απαιτεί την χρήση ενός linspace του numpy. Αν δεν γίνει μετατόπιση των δεδομένων που περιέχονται στον linspace , το pyqtgraph θα αναγκαστεί να δημιουργήσει γράφημα από το βήμα 0 μέχρι το βήμα Ν που αυτό σημαίνει ότι το γράφημα που δημιουργείται εκείνη την στιγμή θα συνεχίζει να μικραίνει με αποτέλεσμα την δυσκολία ανάγνωσής του. Η μετατόπιση των δεδομένων δεν ισχύει στην περίπτωση του Offline Rendering με άνοιγμα αρχείου δειγματοληψίας για να μπορεί ο χρήστης να κάνει επεξεργασία του γραφήματος από το βήμα 0.

Η μετατόπιση των linspace γίνεται με την τεχνική του array slicing της Python και φαίνεται στο παρακάτω κώδικα.

Xm[:-1] = Xm[1:]# shift data

Am[:-1] = Am[1:]

Το Xm[:-1] , που αφορά το linspace των αναλογικών μετρήσεων , σημαίνει ότι ο πίνακας αυτός από την αρχή του μέχρι το προτελευταίο του στοιχείο θα αρχικοποιηθεί με το κομμάτι του ίδιου πίνακα από το δεύτερο στοιχείο του μέχρι το τελευταίο του.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ΧΜ | 1 | 2 | 3 | 4 | 5 | 6 |
| ΧΜ[:-1] | 1 | 2 | 3 | 4 | 5 | Χ |
| ΧΜ[1:] | Χ | 2 | 3 | 4 | 5 | 6 |
| After Shifting | 2 | 3 | 4 | 5 | 6 | Empty |

Μετά την μετατόπιση η τελευταία θέση του linspace παραμένει κενή για να γεμίσει με την καινούρια τιμή που έχει γίνει λήψη από τον Megaman. Το εύρος των linspaces έχει οριστεί στις 500 τιμές.

Το τελικό rendering των τιμών γίνεται με το παρακάτω κομμάτι κώδικα:

curve.setData(Xm) # set the curve with this data

curve.setPos(ptr, 1) # set x position in the graph to 0

curve2.setData(Am)

curve2.setPos(ptr, 1)

QtGui.QApplication.processEvents() # you MUST process the plot now

1. ο ΚΕΦΑΛΑΙΟ

Συμπερασματα επεκτασεις

# Συμπεράσματα

Στην συγκεκριμένη πτυχιακή εργασία παρουσιάστηκε ο τρόπος σύνδεσης όλων των απαραίτητων υλικών του ενσωματωμένου συστήματος καθώς και o προγραμματισμός του στην γλώσσα προγραμματισμού C. Παρουσιάστηκε η σημαντικότητα αυτών των συστημάτων σε διάφορους τομείς καθώς και η εκμετάλλευσή τους σε επίπεδο λογισμικού. Ο συνδυασμός τέτοιων συστημάτων και λογισμικού ανοίγει άπειρες πιθανότητες για επίλυση καθημερινών προβλημάτων. Η μέτρηση και παρατήρηση φυσικών φαινομένων και η αυτοματοποίηση γίνονται πιο εύκολα μέσα από μια συνύπαρξη λογισμικού και υλικού. Γράφοντας την συγκεκριμένη πτυχιακή εργασία εξηγήθηκαν πολλά συστήματα της καθημερινότητας καθώς και ο τρόπος λειτουργίας τους όπως πολλά συστήματα ασφαλείας που φέρουν τα σύγχρονα αυτοκίνητα που βασίζονται εξολοκλήρου σε ενσωματωμένα συστήματα και λογισμικά αλληλεπίδρασης με τον τελικό χρήστη.

# Πιθανές επεκτάσεις

Κατά την διάρκεια δημιουργίας του συστήματος που παρουσιάστηκε εμφανίστηκαν διάφορες δημιουργικές ιδέες για το μέλλον του συγκεκριμένου συστήματος. Μία σημαντική επέκταση θα ήταν η πιο γενική αποστολή μετρήσεων από το υποσύστημα του Megaman. Αυτή την στιγμή οι μετρήσεις στέλνονται ήδη σε βαθμούς κελσίου. Αυτό σημαίνει αυτόματα ότι το υπάρχον σύστημα περιορίζεται μόνο σε μετρήσεις θερμοκρασίες. Μια πιθανή λύση είναι να στέλνονται από το μικροελεγκτή οι θερμοκρασίες χωρίς κάποια μετατροπή. Αυτό θα σήμαινε ότι ο χρήστης θα μπορούσε να θέσει την μετατροπή που απαιτείται στο επίπεδο του Electro. Η συγκεκριμένη τεχνική θα έδινε μεγάλη ευελιξία στο σύστημα καθώς θα μπορούσε να δεχτεί οποιαδήποτε μέτρηση και να την μετατρέψει στο σημείο λήψης. Μια ακόμα βελτίωση στο επίπεδο Megaman είναι η δυνατότητα λήψης τιμών από τον Electro. Αυτό θα επέτρεπε την παραμετροποίηση του Megaman σε ζωντανό χρόνο που θα έδινε δυνατότητες παύσης της μέτρησης.

Στο επίπεδο του Electro υπάρχουν πολλές σκέψεις και ιδέες σχετικά με βελτιώσεις που θα μπορούσαν να γίνουν. Μια πολύ σημαντική βελτίωση είναι η προσθήκη πολλαπλών handlers κατά την διαδικασία του handling. Πολλαπλές ενέργειες θα μπορούσαν αν εκτελεστούν όταν οι μετρήσεις έχουν φτάσει σε ένα συγκεκριμένο όριο που έχει θέσει ο χρήστης. Αύξηση του πλήθους των ενεργειών που δίνει το πρόγραμμα στην εκτέλεση του handling θα ήταν επίσης μια πολύ σημαντική βελτίωση του λογισμικού. Επίσης πολλές βελτιώσεις μπορούν να γίνουν στον renderer του λογισμικού , βελτιώσεις όπως η προσθήκη της δυνατότητας παραμετροποίησης των γραφημάτων προσθήκης γραφημάτων και αφαίρεσης γραφημάτων. Όλες αυτές οι πληροφορίες και ρυθμίσεις θα μπορούσαν να αποθηκεύονται σε ένα custom αρχείου και να εξάγονται , στην συνέχεια θα μπορούσαν να εισαχθούν στον renderer ώστε να ξεκινήσει η διαδικασία.

Τέλος αναγνωρίζοντας την σημαντικότητα του IoT η δημιουργία ενός Web API όπου θα στέλνει την μετρήσεις θα μπορούσε να ανοίξει δρόμους απομακρυσμένης διαχείρισης του συστήματος καθώς και την δημιουργία mobile εφαρμογών. Ο χρήστης με αυτό τον τρόπο θα είχε την δυνατότητα λήψης των μετρήσεων στην έξυπνη συσκευή του και θα μπορούσε να διαχειριστή το σύστημα απομακρυσμένα , λαμβάνοντας πληροφορίες και μετρήσεις και θέτοντας ενέργειες που πρέπει να εκτελεστούν όταν οι μετρήσεις φτάσουν σε συγκεκριμένες τιμές.

Η δημιουργία εφαρμογών για τα υπόλοιπα λειτουργικά συστήματα φαίνεται σε θεωρητικό επίπεδο αρκετά εύκολη καθώς τεχνολογίες όπως η Python , το Qt και η γλώσσα C βρίσκονται ήδη στα περισσότερα λειτουργικά , οπότε μικρές αλλαγές και προσθήκες στον υπάρχον codebase θα έδιναν στην δυνατότητα στο σύστημα να τρέξει και στα υπόλοιπα λειτουργικά.

ΒΙΒΛΙΟΓΡΑΦΙΑ

1. https://www.efxkits.us/classification-of-embedded-systems/
2. https://en.wikipedia.org/wiki/Python\_(programming\_language)
3. https://en.wikipedia.org/wiki/Embedded\_system
4. http://www.ti.com/lit/ds/symlink/lm35.pdf
5. https://el.wikipedia.org/wiki/UART
6. https://en.wikipedia.org/wiki/Universal\_asynchronous\_receiver-transmitter
7. https://en.wikipedia.org/wiki/Raspbian
8. https://en.wikipedia.org/wiki/C\_(programming\_language)
9. http://www.pyqtgraph.org/
10. http://www.numpy.org/
11. https://pythonhosted.org/pyserial/index.html
12. http://doc.qt.io/qt-5/qtdesigner-manual.html
13. https://riverbankcomputing.com/software/pyqt/intro
14. https://en.wikipedia.org/wiki/Git
15. https://newbiehack.com/
16. Patrick Hood-Daniel - https://www.youtube.com/channel/UCC7ifdmN7ebFo-eXBUZkeiw
17. https://en.wikipedia.org/wiki/Analog-to-digital\_converter
18. <http://www.porlidas.gr/AVR/EmbInstrGr.htm>
19. Αρχιτεκτονική & Προγραμματισμός μικροεπεξεργαστών – Π. Παπαζογλου