**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](#_Toc532559247)

[1.2 Ενσωματωμένα Συστήματα ( Embedded systems) 7](#_Toc532559248)

[1.3 Raspberry Pi 9](#_Toc532559252)

[1.4 Python 11](#_Toc532559253)

[1.4.1 Ιστορία της Python 12](#_Toc532559254)

[2.1 Εισαγωγή 14](#_Toc532559255)

[2.2 Μικροελεγκτής AVR 14](#_Toc532559256)

[2.2.1 Ιστορία AVR μικροελεγκτών 15](#_Toc532559257)

[2.2.2 ATMega16a 15](#_Toc532559258)

[2.3 Αισθητήρας LM35Z 17](#_Toc532559259)

[2.4 Διεπαφή σειριακής επικοινωνίας FT232 19](#_Toc532559260)

[2.4.1 Ανατομία FT232 19](#_Toc532559261)

[2.5 Προγραμματιστής Μικροελεγκτή - USBASP 21](#_Toc532559262)

[2.6 Τελικό σχήμα διασύνδεσης 24](#_Toc532559263)

[2.7 Πρότυπο ασύγχρονης επικοινωνίας UART 26](#_Toc532559264)

[2.7.1 Λειτουργία UART 26](#_Toc532559265)

[2.8 Απαιτούμενο λογισμικό προγραμματισμού 27](#_Toc532559266)

[2.8.1 Raspbian 27](#_Toc532559267)

[2.8.2 GCC – AVR GCC 28](#_Toc532559268)

[2.8.2.1 Γλώσσα προγραμματισμού C 28](#_Toc532559269)

[2.8.3 AVRDude 28](#_Toc532559270)

[2.8.4 Fedora OS 29](#_Toc532559271)

[2.8.5 Pycharm IDE 29](#_Toc532559272)

[2.8.6 Python 3.5 30](#_Toc532559273)

[2.8.7 PyQTGraph 31](#_Toc532559274)

[2.8.8 PySerial 34](#_Toc532559275)

[2.8.9 Numpy 34](#_Toc532559276)

[2.8.10 PyQt 5 34](#_Toc532559277)

[2.8.11 Qt Designer 36](#_Toc532559278)

[2.8.12 Git 37](#_Toc532559279)

[3.1 Εισαγωγή 38](#_Toc532559280)

[3.2 Εφαρμογή της πτυχιακής εργασίας 38](#_Toc532559281)

[3.3 Επίπεδο Megaman 40](#_Toc532559282)

[3.4 Λειτουργία ADC 41](#_Toc532559283)

[3.5 Προγραμματισμός AVR – Firmware Megaman 43](#_Toc532559284)

[3.5.1 Μεταγλώττιση και Εγγραφή του Firmware 50](#_Toc532559285)

[4.1 Εισαγωγή 52](#_Toc532559286)

[4.2[Πατήστε και γράψτε την επικεφαλίδα] 52](#_Toc532559287)

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 2.2‑3 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 2.2‑4 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 2.2‑5 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 2.2‑6 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 2.2‑7 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 2.2‑8 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 2.2‑9 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 2.2‑10 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>

#define degree\_sysmbol 0xdf

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

[Πατήστε και γράψτε το κείμενο σας]

# [Πατήστε και γράψτε την επικεφαλίδα]

[Πατήστε και γράψτε το κείμενο σας]

ΒΙΒΛΙΟΓΡΑΦΙΑ

* 1. ……………
  2. ……………
  3. ……………
  4. ……………
  5. ……………
  6. ……………
  7. ……………
  8. ……………
  9. ……………
  10. ……………

**Μορφή βιβλιογραφικών αναφορών**

**Συνέδριο**

Nasser, N., & Hassanein, H.S. (2004). Bandwidth Reservation Policy for Multimedia Wireless Networks and its Analysis. *IEEE Internernational Conference on Communications* (pp. 3030-3034). Paris, France.

**Περιοδικό**

Oliveria, C., Kim, J.B., & Suda, T. (1998). An adaptive bandwidth reservation scheme for high-speed multimedia wireless networks. *IEEE J Select Areas Comm*, *16*(6), 858-874.

**Βιβλίο**

Lee, W.C.Y. (2006). *Wireless and Cellular Telecommunications*. New York: McGraw-Hill.

**Παράδειγμα αναφοράς πηγής στο κείμενο**

(Oliveria et al. 1998; Lee 2006)

(1) Επίσης, θα πρέπει να υπάρχει και μία λίστα αναφορών στο τέλος κάθε κεφαλαίου

ΠΑΡΑΡΤΗΜΑΤΑ