ΑΛΕΞΑΝΔΡΕΙΟ ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ
ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΝΙΚΩΝ ΜΗΧΑΝΙΚΩΝ ΤΕ

ΜΕΤΑΠΤΥΧΙΑΚΟ ΠΡΟΓΡΑΜΜΑ

« ΕΦΑΡΜΟΣΜΕΝΑ ΗΛΕΚΤΡΟΝΙΚΑ ΣΥΣΤΗΜΑΤΑ»

«Μελέτη και κατασκευή συσκευής παρακολούθησης αθλητών με αποστολή μετρήσεων σε πραγματικό χρόνο»

Κωδικός: 1822

Διπλωματική Εργασία
του
Κοκκινίδη Σάββα

Επιβλέπων καθηγητής
Γιακουμής Άγγελος

Θεσσαλονίκη, 2019

Περίληψη

Σκοπός της παρούσας εργασίας είναι ο σχεδιασμός, κατασκευή και προγραμματισμός μιας φορητής συσκευής βασιζόμενη σε μικροελεγκτή, που μπορεί να μετρήσει δύναμη, επιτάχυνση και γωνίες προσανατολισμού για την μελέτη των κινήσεων των αθλητών. Η συσκευή αποστέλλει ασύρματα μέσω Bluetooth και σε πραγματικό χρόνο τις μετρήσεις σε μια Android εφαρμογή και έχει όσο το δυνατόν μικρότερες διαστάσεις και βάρος ώστε να μην επηρεάζει την απόδοση των αθλητών.

Λέξεις Κλειδιά

Μετρήσεις, μικροελεγκτής, επιταχυνσιόμετρο, γυροσκόπιο, πραγματικού χρόνου, Bluetooth, Kalman φίλτρο, Android εφαρμογή

Abstract

"Design and implementation of athletes tracking device with real-time measurements"

The aim of this thesis is to design, implement and program a portable device based on a microcontroller that can measure strength, acceleration and orientation angles in order to study the movements of athletes. The device sends wirelessly using Bluetooth the real-time measurements to an Android application and has small dimensions and weight as possible so as not to affect the performance of the athletes.

Key Words

Measurements, microcontroller, accelerometer, gyroscope, real-time, Bluetooth, Kalman filter, Android application

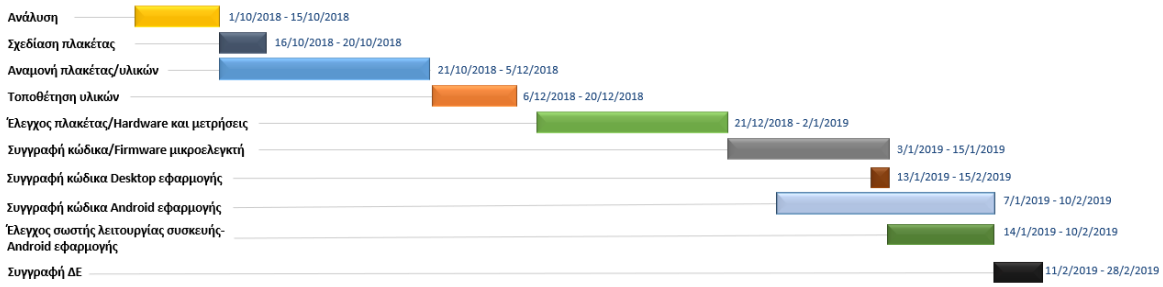
1. Εισαγωγή

Στο πλαίσιο της παρούσας εργασίας έγινε σχεδιασμός, κατασκευή και προγραμματισμός μιας φορητής συσκευής βασιζόμενη σε μικροελεγκτή, που θα μπορεί να μετρά διάφορα μεγέθη όπως δύναμη, επιτάχυνση και γωνιών προσανατολισμού για την μελέτη των κινήσεων των αθλητών. Η συσκευή αποστέλλει ασύρματα μέσω Bluetooth και σε πραγματικό χρόνο τις μετρήσεις σε μια Android εφαρμογή και έχει όσο το δυνατόν μικρότερες διαστάσεις και βάρος ώστε να μην επηρεάζει την απόδοση των αθλητών. Κύρια αισθητήρια για τη συλλογή των παραπάνω δεδομένων είναι ένα επιταχυνσιόμετρο και ένα γυροσκόπιο τριών αξόνων, από το συνδυασμό τους και τη χρήση Kalman φίλτρου που είναι ένας βέλτιστος παρατηρητής, βέλτιστος ως προς την αποθορυβοποίηση, που είναι απαραίτητος για να ελεγχθούν πολλά πραγματικά συστήματα. Με το συνδυασμό των παραπάνω αισθητηρίων και ενός Kalman φίλτρου, είναι εφικτός ο υπολογισμός των γωνιών Pitch και Roll της συσκευής. Η συσκευή μπορεί επίσης να μετρήσει τη δύναμη (παραμόρφωση) που ασκείται σε συγκεκριμένο σημείο με τη χρήση αισθητηρίων Strain Gauge και αντίστοιχου ολοκληρωμένου για Conditioning. Οι υπόλοιπες μετρήσεις όπως η στάθμη της μπαταρίας και η θερμοκρασία της πλακέτας, αποστέλλονται σαν επιπλέον πληροφορία για ενημέρωση του χρήστη. Όλες οι παραπάνω μετρήσεις απεικονίζονται σε γραφήματα πραγματικού χρόνου σε μία Android εφαρμογή. Η Android εφαρμογή μπορεί και αποθηκεύει τα γραφήματα δημιουργώντας έτσι ένα ιστορικό, όπου ο χρήστης έχει τη δυνατότητα απεικόνισης γραφημάτων/μετρήσεων που έγιναν κατά το παρελθόν. Για την παραμετροποίηση της συσκευής υλοποιήθηκε και μία Desktop εφαρμογή όπου μέσω USB καλωδίου ο χρήστης μπορεί να διαβάσει ή να αλλάξει βασικές παραμέτρους. Τέλος, η συσκευή περιλαμβάνει μια επαναφορτιζόμενη μπαταρία λιθίου και κύκλωμα φόρτισης μέσω USB που της δίνει αυτονομία έως δύο ώρες.

Περιεχόμενα

2. Χρονοδιάγραμμα

Ακολουθεί το χρονοδιάγραμμα (Εικόνα 1) για κάθε στάδιο μέχρι και την τελική υλοποίηση της διπλωματικής εργασίας:

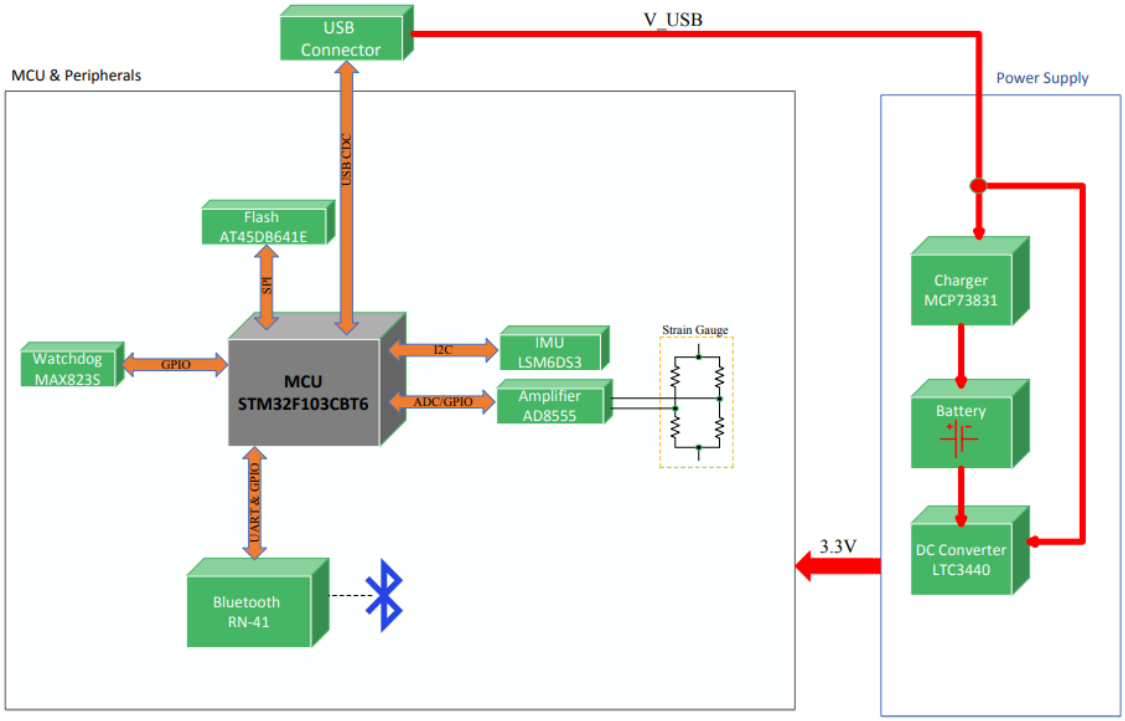


Εικόνα 1: Χρονοδιάγραμμα διπλωματικής εργασίας

Αξίζει να σημειωθεί πως μεγάλο χρονικό διάστημα αφιερώθηκε κατά την αναμονή της πλακέτας και των υλικών, όπως και κατά την τοποθέτηση.

3. Υλικό/Hardware

Στο κεφάλαιο αυτό, θα γίνει περιγραφή του υλικού της συσκευής και η λογική με την οποία επιλέχθηκαν τα συγκεκριμένα υλικά. Κύρια κριτήρια κατά την επιλογή των υλικών ήταν το κόστος, αξιοπιστία, διαθεσιμότητα, ευκολία στην τοποθέτηση-κόλληση και τέλος σημαντικό ρόλο έπαιξε η εμπειρία με ορισμένα από αυτά τα υλικά.



Διάγραμμα 1: Το μπλοκ διάγραμμα της συσκευής

3.1 Μικροελεγκτής

Ένας μικροελεγκτής είναι ένα υπολογιστικό κύκλωμα, σχεδιασμένο σε ένα και μόνο ολοκληρωμένο κύκλωμα υψηλής κλίμακας ολοκλήρωσης. Όπως κάθε υπολογιστικό κύκλωμα, περιέχει κεντρική μονάδα επεξεργασίας, έναν αριθμό καταχωρητών, κυκλώματα μνήμης και κυκλώματα ελέγχου περιφερειακών συσκευών. Κάθε μικροελεγκτής είναι ικανός να ανταλλάξει σήματα με το εξωτερικό περιβάλλον, να εκτελέσει αριθμητικές πράξεις ανάμεσα σε μεταβλητές και να καταχωρήσει κάποιες τιμές στη μνήμη RAM που διαθέτει [1]. Κάθε μικροελεγκτής περιέχει μέσα σε ένα και μοναδικό ολοκληρωμένο κύκλωμα τα παρακάτω στοιχεία:

- έναν αριθμό από καταχωρητές ειδικού σκοπού (συσσωρευτή, καταχωρητή κατάστασης, μετρητή προγράμματος, καταχωρητή εντολών, καταχωρητή δείκτη)
- εσωτερικούς χρονιστές - απαριθμητές
- αριθμητική και λογική μονάδα (ALU)
- μονάδα αποκωδικοποίησης εντολών

Βασικά στοιχεία ενός μικροελεγκτή αποτελούν:

- η μνήμη προγράμματος (ROM ή EPROM)
- η μνήμη καταχωρητών / μεταβλητών (RAM)

Η αρχιτεκτονική που χρησιμοποιούν είναι Harvard για την επικοινωνία της CPU και της μνήμης. Οι κατηγορίες που χωρίζονται οι μικροελεγκτές είναι δύο με βάση τον τρόπο που χρησιμοποιούν τις εντολές για τον υπολογισμό μιας εργασίας ανεξάρτητου μεγέθους. Οι κατηγορίες αυτές είναι:

- Η αρχιτεκτονική RISC
- Η αρχιτεκτονική CISC

Ο προγραμματισμός των μικροελεγκτών μπορεί να γίνει είτε από γλώσσες υψηλού επιπέδου είτε σε γλώσσα κατανοητή από τον μικροελεγκτή(π.χ. Assembly). Η πιο διαδεδομένη γλώσσα προγραμματισμού είναι η C.

3.1.1 Αρχιτεκτονική ARM

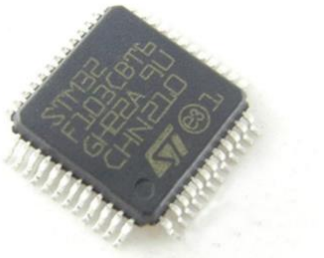
Η ARM είναι μια αρχιτεκτονική συνόλου εντολών RISC των 32-bit, η οποία έχει αναπτυχθεί από την ARM Holdings. Τα αρχικά σημαίνουν Προχωρημένη Μηχανή RISC (Advanced RISC Machine), ενώ παλαιότερα σήμαιναν Μηχανή RISC Acorn (Acorn RISC Machine). Η αρχιτεκτονική ARM είναι η πιο συχνά χρησιμοποιούμενη αρχιτεκτονική συνόλου εντολών 32-bit όσον αφορά τους επεξεργαστές που παράγονται. Η αρχιτεκτονική δημιουργήθηκε αρχικά από την Acorn Computers για χρήση στους προσωπικούς υπολογιστές της, με τα πρώτα προϊόντα που βασίζονταν στην αρχιτεκτονική ARM να είναι η σειρά Acorn Archimedes που εμφανίστηκε το 1987 [2].

Οι επεξεργαστές ARM είναι σχετικά απλοί, κάτι που τους κάνει κατάλληλους για εφαρμογές χαμηλής ισχύος. Αυτό έχει ως αποτέλεσμα να έχουν υπερισχύσει στις αγορές των κινητών και των ενσωματωμένων συστημάτων, σαν μικροί και σχετικά χαμηλού κόστους μικροεπεξεργαστές και μικροελεγκτές. Το 2005, περίπου το 98% των πάνω από ένα δισεκατομμύριο κινητών τηλεφώνων που πωλούνται κάθε χρόνο είχαν τουλάχιστον έναν επεξεργαστή ARM. Το 2009 οι επεξεργαστές ARM αντιστοιχούσαν περίπου στο 90% όλων των ενσωματωμένων επεξεργαστών RISC 32-bit και χρησιμοποιούνται σε μεγάλο βαθμό σε

καταναλωτικά ηλεκτρονικά προϊόντα, συμπεριλαμβανομένων των προσωπικών ψηφιακών βοηθών (personal digital assistants, PDAs), των κινητών τηλεφώνων, των συσκευών ψηφιακής μουσικής και πολυμέσων, των φορητών κονσολών βιντεοπαιχνιδιών, των αριθμομηχανών και περιφερειακών υπολογιστών όπως οι σκληροί δίσκοι και οι δρομολογητές.

3.1.2 Μικροελεγκτής - STM32F103CBT6

Έχοντας ασχοληθεί με τον συγκεκριμένο μικροελεγκτή αλλά και τη συγκεκριμένη σειρά της εταιρείας STMicroelectronics, επέλεξα τον STM32F103CBT6 της σειράς F103xx. Ο συγκεκριμένος μικροελεγκτής κατασκευάζεται με βάση την αρχιτεκτονική ARM και διαθέτει ως πυρήνα τον Cortex-M3 32-bit RISC. Διαθέτει μνήμη Flash 128Kbytes και SRAM 20Kbytes. Ο συγκεκριμένος μικροελεγκτής ανήκει στη κατηγορία «Mainstream Performance line», με μέγιστη συχνότητα τα 72MHz και πληθώρα περιφερειακών συνδεσιμότητας [3].



Εικόνα 2: Ο μικροελεγκτής STM32F103CBT6

Τα περιφερειακά του STM32F103CBT6 περιλαμβάνουν μεταξύ άλλων δύο I2C's (Inter-Integrated Circuit), δύο SPI's (Serial Peripheral Interface), δύο ADC's (Analog To Digital Converter) των 12 bits, έναν Advanced Control Timer, έναν timer για Pulse Width Modulation (PWM) και τρείς timers γενικού σκοπού των 16 bits. Επίσης, περιλαμβάνονται και θύρες USART (Universal Synchronous/Asynchronous Receiver Transmitter),CAN (Controller Area Network) και USB(Universal Serial Bus).

Κύρια χαρακτηριστικά του STM32F103CBxx:

- Μέγιστη συχνότητα 72MHz
- Τάση λειτουργίας 2-3.6V
- 64 Kbytes Flash memory / 20 Kbytes SRAM
- Χαμηλή κατανάλωση
- Ενσωματωμένοι oscillators για RTC
- A/D μετατροπείς 12 bit
- DMA με 7 κανάλια και υποστήριξη σχεδόν όλα τα περιφερειακά (timers, SPIs, ADC, USARTs, I2Cs)
- I/O θύρες με δυνατότητα αντιστοίχισης σε 16 external interrupt vectors και σχεδόν όλα συμβατά με 5V στάθμες (5V tolerant)
- 2 τρόπους για Debug (SWD, JTAG)
- 4 timers εκ των οποίων 3 γενικού σκοπού και 1 ειδικού ελέγχου (λειτουργία PWM)
- Διεπαφές επικοινωνίας:
    - 3 x USART
    - 2 x SPI
    - 2 x I2 C

Όλα τα παραπάνω σε συνδυασμό με το χαμηλό κόστος (περίπου 4.1 ευρώ) και τη δωρεάν διάθεση Software εργαλείου (STM32CubeMX) από τον κατασκευαστή για αρχικοποίηση του μικροελεγκτή και των περιφερειακών του, καθιστούν τον STM32F103CBT6 μια εξαιρετική λύση για τη συγκεκριμένη εφαρμογή. Να σημειωθεί πως το χαμηλό ποσοστό χρήσης των μνημών SRAM/Flash αλλά και η χρήση ελάχιστων (σε σχέση πάντα με τις δυνατότητες του μικροελεγκτή) περιφερειακών, κάνει σχεδόν οποιαδήποτε αλλαγή ή προσθήκη στο Firmware/Hardware εφικτή.

Ποσοστό χρήσης μνήμης SRAM	55.4 %
Ποσοστό χρήσης μνήμης Flash	33.3 %

Πίνακας 1: Ποσοστό χρήσης μνημών του μικροελεγκτή

```
 43 920 bytes of readonly  code memory
    854 bytes of readonly  data memory
 11 349 bytes of readwrite data memory
```

Εικόνα 3: Χρήση μνημών SRAM/Flash (από αρχείο .map)

3.2 Bluetooth

Το Bluetooth είναι ένα πρωτόκολλο επικοινωνίας σχεδιασμένο για επικοινωνίες μικρών αποστάσεων, χαμηλού bandwidth και μεταξύ δύο συζευγμένων μερών (peer-to peer). Είναι μια τεχνολογία ασύρματη που αρχικά υλοποιήθηκε για να αποτελέσει μια εναλλακτική λύση στα RS-232 καλώδια δεδομένων. Δημιουργήθηκε από την εταιρία Ericson το 1994. Αυτή τη στιγμή το πρωτόκολλο διαχειρίζεται από τον οργανισμό Bluetooth Special Interest Group ο οποίος αριθμεί πάνω από 25,000 εταιρίες. Η IEEE έχει πατεντάρει το πρωτόκολλο ως το IEEE 802.15.1. Το πρωτόκολλο χρησιμοποιεί τις συχνότητες από 2400MHz έως 2480MHz. Διαχωρίζει τα μεταδιδόμενα δεδομένα σε πακέτα και τα μεταδίδει σε ένα από τα 79 του κανάλια. Κάθε κανάλι έχει bandwidth περίπου 1MHz. Γενικά πρόκειται για ένα πρωτόκολλο βασισμένο στα πακέτα με σαφή master-slave αρχιτεκτονική. Κάθε master συσκευή μπορεί να επικοινωνήσει μέχρι με 7 slaves σε ένα piconet (ένα δίκτυο δηλαδή από συσκευές συνδεμένες με αυτό το πρωτόκολλο), ενώ παρέχεται και η δυνατότητα για εναλλαγή των ρόλων του master με τον slave. Το Bluetooth είναι ένα πρωτόκολλο που σχεδιάστηκε για να αντικαταστήσει την επικοινωνία μέσω καλωδίων με σαφή μείωση της ενεργειακής κατανάλωσης. Οι συσκευές χρησιμοποιούν τα RF κανάλια ,έτσι δεν χρειάζεται να βρίσκονται σε οπτική επαφή. Όμως η απόσταση μέχρι την οποία είναι δυνατόν να εκπέμψουν ποικίλει ανάλογα με την κλάση του υλικού που έχει κάθε συσκευή [4].

Class	***Max Power permitted (mW)***	***Typical Range (m)***
1	100	~100
2	2.5	~10
3	1	~1

Πίνακας 2: Κλάσεις και μέγιστη απόσταση

3.2.1 Serial Port Profile (SPP)

Για την επικοινωνία και την αποστολή των δεδομένων μεταξύ της συσκευής και της Android εφαρμογής, έπρεπε να επιλέξω κάποιο πρωτόκολλο ασύρματης δικτύωσης που να παρέχει κάλυψη αρκετών μέτρων (~ 30 μέτρα), να υποστηρίζει μεγάλο μέγεθος πακέτων και συγχρόνως να υποστηρίζεται σε κάθε κινητό ή tablet ώστε ο χρήστης να μην επιβαρύνεται με επιπλέον έξοδα. Με βάση τα παραπάνω, η επιλογή του Bluetooth ως πρωτόκολλο επικοινωνίας ήταν μονόδρομος.

Σημαντικό γνώρισμα της τεχνολογίας Bluetooth είναι η δυνατότητα αναβάθμισης και επέκτασής που έχει, ώστε να μπορεί να ενσωματωθεί σε νέα προϊόντα και να χρησιμοποιηθεί με πολλούς και διαφορετικούς τρόπους. Αυτές οι επεκτάσεις ονομάζονται «Προφίλ» (Profiles). Για να μπορέσει μια συσκευή να συνδεθεί με κάποια άλλη, θα πρέπει και οι δύο να υποστηρίζουν το ίδιο προφίλ. Παρόλα αυτά, κάθε συσκευή μπορεί να ενσωματώνει περισσότερα από ένα προφίλ. Το πιο διαδεδομένο Profile και με τη μεγαλύτερη χρήση σε ενσωματωμένα συστήματα, είναι το SPP (Serial Port Profile). Το συγκεκριμένο Profile λειτουργεί όπως και μια κοινή ενσύρματη σειριακή θύρα και είναι πλήρως συμβατό με τις εφαρμογές μεταφοράς δεδομένων RS232.

3.2.2 Bluetooth Module - RN41

Για τη χρήση του SPP Profile και τη διασύνδεση της συσκευής με την Android εφαρμογή, έπρεπε να χρησιμοποιηθεί κάποια έτοιμη πλακέτα-Module με ενσωματωμένη κεραία (On-Chip). Το Module που επέλεξα να χρησιμοποιήσω είναι το RN41 της εταιρίας Microchip. Το συγκεκριμένο Module έχει πολύ μικρές διαστάσεις που το κάνει ιδανικό για τη πλακέτα της συσκευής αλλά και επίσης χαμηλή κατανάλωση που αυξάνει την αυτονομία της συσκευής. Ο κατασκευαστής αναφέρει σαν μέγιστη δυνατή απόσταση επικοινωνίας τα 100m αλλά στην πράξη και μετά από μετρήσεις, η μέγιστη απόσταση χωρίς διακοπές στην επικοινωνία είναι ~40 μέτρα χωρίς εμπόδια (με τη συγκεκριμένη κεραία που διαθέτει).

Εικόνα 4: RN41 Bluetooth module

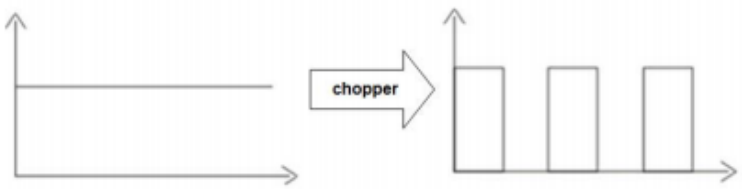
Κύρια χαρακτηριστικά του RN41:

- Πιστοποίηση Class 1 Bluetooth 2.1
- UART (SPP ή HCI) πρωτόκολλα
- Προγραμματιζόμενες λειτουργίες χαμηλής κατανάλωσης
- Ασφαλής επικοινωνία, 128-bit κρυπτογράφηση

- Διόρθωση λαθών για σωστή αποστολή πακέτων
- Auto-discovery/pairing χωρίς έλεγχο από το Firmware
- SMT pads για εύκολη τοποθέτηση και ενσωματωμένη κεραία (On-Chip)
- Μέγιστη απόσταση επικοινωνίας (100m line-of-sight)

3.3 DC/DC μετατροπέας

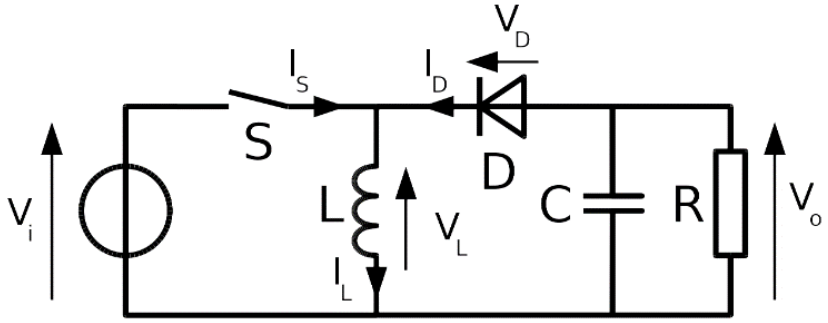
Οι DC-DC μετατροπείς ονομάζονται και κατατμητές (ή ψαλιδιστές-choppers). Πρόκειται κατά βάση για κυκλώματα που χρησιμοποιούν ηλεκτρονικούς δι ακόπτες μεγάλης συχνότητας ώστε να αποκόπτουν ή όχι μια DC τάση εισόδου από το φορτί ο, παράγοντας έτσι μια DC τάση εξόδου με διαφορετική μέση τιμή. Η απλουστευμένη γενική αρχή λειτουργίας απεικονίζεται στην εικόνα 5. Με χρήση κατάλληλων στοι-χείων και κυκλωματικών διατάξεων (αξιοποιώντας τις δυνατότητες αποθήκευσης ενέργ ειας παθητικών στοιχείων) είναι δυνατόν να παραχθούν και τάσης εξόδου με μεγαλύτερη μέση τιμή από αυτήν της εισόδου. Συνεπώς οι DC-DC μετατροπείς μπορούν να χρησιμοποι-ηθούν για τον υποβιβασμό και την ανύψωση μιας DC τάσης. Έτσι ανάλογα με την λειτου-ργία του μετατροπέα αυτός μπορεί να είναι υποβιβασμού (buck), ανύψωσης (boost) ή υπο-βιβασμού ανύψωσης (buck-boost) [5].



Εικόνα 5: Απλοποιημένη γενική αρχή λειτουργίας των κατατμητών

3.3.1 Buck-Boost μετατροπέας

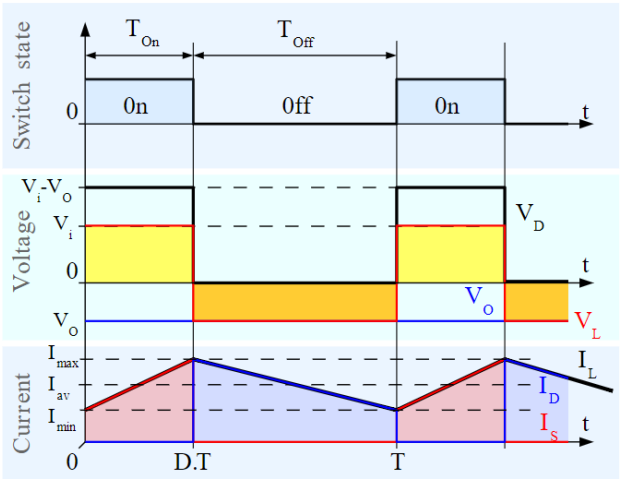
Ο μετατροπέας DC/DC υποβιβασμού/ανύψωσης τάσης (step-down/up ή Buck-Boost converter) παράγει μια συνεχή τάση εξόδου χαμηλότερη/υψηλότερη από τη συνεχή τάση εισόδου. Το κύκλωμα του μετατροπέα φαίνεται στην εικόνα 6.



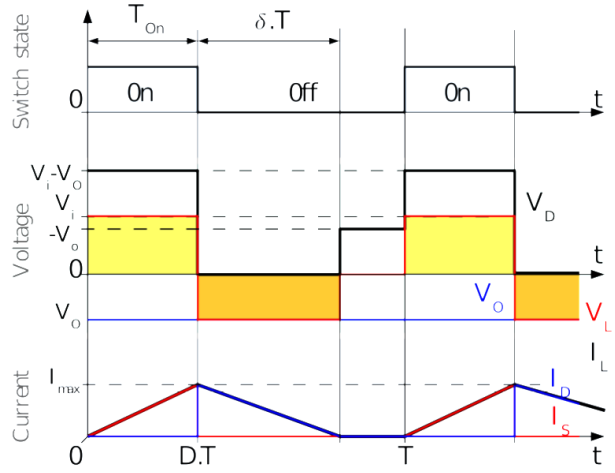
Εικόνα 6: Διάγραμμα του μετατροπέα Buck-Boost

Όταν το ημιαγωγικό στοιχείο άγει, η δίοδος D είναι πολωμένη ανάστροφα και παρατηρούνται δύο βρόχοι. Ο ένας είναι ο βρόχος εισόδου που φορτίζει το πηνίο μέσω της πηγής και ο άλλος είναι ο βρόχος εξόδου που τροφοδοτεί με ισχύ το φορτίο, μέσω του πυκνωτή εξόδου. Όταν ο διακόπτης οδηγηθεί στην αποκοπή, η τάση της επαγωγής αλλάζει πολικότητα και μόλις γίνει ίση με την τάση εξόδου, η δίοδος πολώνεται ορθά. Μέσω της διόδου, ένα μέρος της ενέργειας

ή όλη η ενέργεια η οποία έχει αποθηκευτεί στην επαγωγή, μεταφέρεται στην έξοδο. Η πηγή εισόδου προσφέρει ενέργεια στο κύκλωμα, καθώς ο διακόπτης είναι σε αποκοπή. Όπως σε όλους τους DC-DC μετατροπείς, έτσι και στον Buck-Boost, εμφανίζονται δύο περιοχές λειτουργίας, η περιοχή της συνεχούς και της ασυνεχούς αγωγής του ρεύματος. Ως συνεχής αγωγή (Continuous Conduction Mode, CCM) ορίζεται η κατάσταση λειτουργίας στην οποία το ρεύμα που διαρρέει το πηνίο εξομάλυνσης είναι πάντα μεγαλύτερο του μηδενός, ενώ ως ασυνεχής αγωγή (Discontinuous Conduction Mode, DCM) ορίζεται η κατάσταση λειτουργίας στην οποία το ρεύμα που διαρρέει το πηνίο εξομάλυνσης παρουσιάζει διαστήματα όπου μηδενίζεται. Οι κυματομορφές για τα στοιχεία του μετατροπέα και για τις δύο περιοχές λειτουργίας φαίνονται στα σχήματα 1 και 2:



Σχήμα 1: Continuous Conduction Mode



Σχήμα 2: Discontinuous Conduction Mode

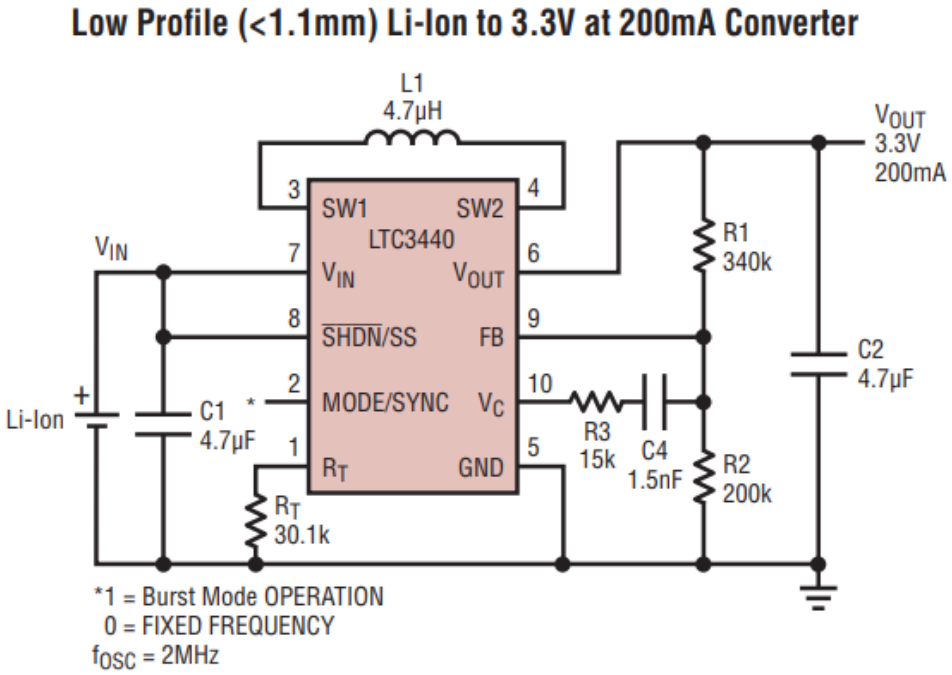
3.3.2 Buck/Boost μετατροπέας - LTC3440

Για να μπορεί η συσκευή να είναι αυτόνομη, υπάρχει μια μπαταρία ιόντων λιθίου με ονομαστική τάση 3.7V που μπορεί να κρατήσει τη συσκευή σε λειτουργία για 2 ώρες. Οι συγκεκριμένες μπαταρίες έχουν ένα εύρος τάσης (είτε κατά τη φόρτιση είτε κατά την εκφόρτιση) από περίπου 4.25V μέχρι περίπου 2.8V (να σημειωθεί ότι τα κυκλώματα

προστασίας περιορίζουν ακόμα περισσότερο τη τάση στα άκρα της μπαταρίας, συνήθως από 3V μέχρι 4.2V).

Όπως αναφέραμε παραπάνω, η τάση της μπαταρίας θα κυμαίνεται από περίπου 3V μέχρι 4.2V. Αυτό σημαίνει ότι για να παράγουμε τα 3.3V που χρειαζόμαστε για να λειτουργήσουν σωστά ο μικροελεγκτής και το Bluetooth module, θα πρέπει να επιλέξουμε κάποιον μετατροπέα Buck-Boost με μεγάλο δείκτη απόδοσης, ελάχιστα εξωτερικά υλικά και να καταλαμβάνουν ελάχιστο χώρο στη πλακέτα. Για όλους τους παραπάνω λόγους, ο Buck-Boost μετατροπέας LTC3440 της εταιρείας Analog Devices είναι μια πολύ καλή επιλογή. Ρυθμίζοντας τη διακοπτική συχνότητα του μετατροπέα στη μέγιστη τιμή (2MHz), τα εξωτερικά υλικά που χρειάζεται το ολοκληρωμένο και το μέγεθος τους, μειώνονται σε μεγάλο βαθμό. Τα βασικά χαρακτηριστικά του LTC3440 από το datasheet του κατασκευαστή:

- ***Τάση εισόδου:*** 2.5V – 5.5V
- ***Τάση εξόδου: :*** 2.5V – 5.5V
- ***Διακοπτική συχνότητα:*** 300KHz – 2MHz
- ***Μέγιστο ρεύμα εξόδου:*** 600mA



Εικόνα 7: Σχηματικό κυκλώματος με το LTC3440 και μπαταρία ιόντων λιθίου

3.4 Μπαταρία Li-ion

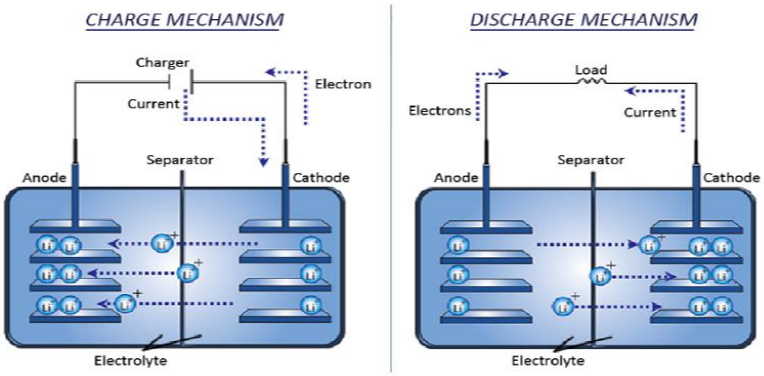
Η μπαταρία ιόντων λιθίου (γνωστή και ως Li-ion μπαταρία) ανήκει στην κατηγορία των επαναφορτιζόμενων μπαταριών. Σε αυτή τη μπαταρία, ιόντα λιθίου μετακινούνται από το αρνητικό ηλεκτρόδιο προς το θετικό κατά την εκφόρτιση και αντιστρόφως κατά τη φόρτιση. Οι μπαταρίες Li-ion είναι πολύ κοινές σε διάφορα ηλεκτρονικά που χρησιμοποιούνται στην καθημερινότητα. Είναι ένας από τους πιο δημοφιλείς τύπους επαναφορτιζόμενων μπαταριών για κινητές ηλεκτρονικές συσκευές, λόγω των εξής χαρακτηριστικών: υψηλή πυκνότητα ενέργειας, μικρή επίδραση μνήμης και μικρή απώλεια φορτίου όταν δεν χρησιμοποιούνται. Αξίζει να σημειωθεί πως οι μπαταρίες Li-ion αποκτούν μεγάλη φήμη και σε άλλες κατηγορίες

χρήσης, εκτός των καταναλωτικών ηλεκτρονικών, όπως: στρατιωτικές εφαρμογές, ηλεκτρικά αυτοκίνητα και διαστημικές εφαρμογές. Σημαντικά προτερήματα αποτελούν τα προαναφερθέντα χαρακτηριστικά καθώς και το μειωμένο βάρος των μπαταριών αυτών σε σχέση με μπαταρίες άλλων τύπων για τις ίδιες απαιτήσεις ενέργειας [6]. Οι μπαταρίες ιόντων λιθίου χωρίζονται σε έξι βασικές κατηγορίες ανάλογα με τη χημική τους σύσταση:

- Lithium Cobalt Oxide (LiCoO2)
- Lithium Manganese Oxide (LiMn2O4)
- Lithium Nickel Manganese Cobalt Oxide (LiNiMnCoO2)
- Lithium Iron Phosphate (LiFePO4)
- Lithium Nickel Cobalt Aluminum Oxide (LiNiCoAlO2)
- Lithium Titanate (Li4Ti5O12)

3.4.1 Περιγραφή λειτουργίας

Τα τρία κύρια στοιχεία που απαρτίζουν λειτουργικά τουλάχιστον μία μπαταρία ιόντων λιθίου είναι το θετικό ηλεκτρόδιο, το αρνητικό ηλεκτρόδιο και ο ηλεκτρολύτης. Στις μπαταρίες ιόντων λιθίου (Li-ion) έχουμε εισαγωγή ιόντων εντός του κρυσταλλικού πλέγματος του ηλεκτροδίου χωρίς αλλαγή της κρυσταλλικής του δομής. Τα ηλεκτρόδια έχουν τις εξής βασικές ιδιότητες: ανοιχτές κρυσταλλικές δομές επιτρέπουν την εισαγωγή ή την αφαίρεση των ιόντων λιθίου, και, την ικανότητα να δεχθούν ηλεκτρόνια συγχρόνως. Τονίζεται ξανά επίσης ότι λόγω της αντιδραστικότητας του καθαρού λιθίου ο ηλεκτρολύτης πρέπει να αποτελείται από μη υδατικά οργανικά άλατα.



Εικόνα 8: Σχηματική αναπαράσταση φόρτισης και εκφόρτισης μπαταρίας Li-ion

Στην εικόνα 8 παρατηρούμε σχηματικά τον τρόπο λειτουργίας μίας μπαταρίας Li-ion κατά την φόρτιση και κατά την εκφόρτιση. Κατά τη διάρκεια της εκφόρτισης μίας μπαταρίας ιόντων λιθίου, η άνοδος (ακροδέκτης (-) του σχήματος) οξειδώνεται ηλεκτροχημικά, και αυτό προκαλεί την απελευθέρωση ιόντων λιθίου μέσα στον ηλεκτρολύτη. Ταυτόχρονα τα ιόντα λιθίου ταξιδεύουν στον ηλεκτρολύτη και «αποζημιώνουν» για το αρνητικό φορτίο που ρέει μέσα στην κάθοδο (ακροδέκτης (+) του σχήματος) από το εξωτερικό κύκλωμα. Έτσι, τα ιόντα λιθίου απορροφώνται από την κάθοδο. Η αντίστροφη διαδικασία ακολουθείται κατά την διαδικασία της φόρτισης, όπου συνεχές ρεύμα από την πηγή φόρτισης (charger) ρέει όπως φαίνεται στο παραπάνω σχήμα, φορτίζοντας έτσι τη μπαταρία. Στο παραπάνω σχήμα διακρίνονται καθαρά ο θετικός ακροδέκτης, ο αρνητικός ακροδέκτης, ο διαχωριστής και ο ηλεκτρολύτης που αποτελούν βασικά συστατικά της μπαταρίας. Αυτό που αλλάζει μεταξύ φόρτισης και εκφόρτισης είναι η φορά κίνησης των ηλεκτρονίων, η φορά ροής του ρεύματος

και η φορά προς την οποία μετακινούνται τα ιόντα λιθίου. Παρατηρείται επίσης, ότι κατά τη διαδικασία της φόρτισης παρέχουμε ηλεκτρική ενέργεια στην μπαταρία, η οποία αποθηκεύεται με τη μορφή χημικής ενέργειας, ενώ κατά την εκφόρτιση η χημική ενέργεια μετατρέπεται σε ηλεκτρική παρέχοντας την απαραίτητη ισχύ στο φορτίο.

3.4.2 Πλεονεκτήματα μπαταριών Li-ion

Παρ' όλο που όπως αναφέρθηκε υπάρχουν αρκετοί διαφορετικοί τύποι μπαταριών Li-ion, θα περιγράψουμε κάποια από τα γενικά πλεονεκτήματα που αυτές παρουσιάζουν. Υπάρχουν αρκετά πλεονεκτήματα στη χρήση μπαταριών ιόντων λιθίου έναντι σε άλλους τύπους μπαταριών. Αυτά είναι:

- ***Υψηλή πυκνότητα ενέργειας:*** Η αρκετά υψηλότερη πυκνότητα ενέργειας είναι ένα από τα κύρια προτερήματα των μπαταριών ιόντων λιθίου. Οι σύγχρονες ηλεκτρονικές συσκευές απαιτούν μεγάλη διάρκεια λειτουργίας μεταξύ των φορτίσεων, ενώ ταυτόχρονα αυξάνεται η απαίτηση σε κατανάλωση ισχύος. Επομένως, υπάρχει πάντοτε η ανάγκη να υφίστανται μπαταρίες με μεγάλη ενεργειακή πυκνότητα
- ***Διατήρηση του φορτίου:*** Ένα μεγάλο πρόβλημα στην τεχνολογία των μπαταριών είναι η εκφόρτισή τους λόγω μη χρήσης. Οι μπαταρίες ιόντων λιθίου πλεονεκτούν και σε αυτόν τον τομέα έναντι άλλων τύπων μπαταριών, έχοντας αρκετά μικρό ρυθμό εκφόρτισης λόγω μη χρήσης (self-discharge). Χαρακτηριστικό παράδειγμα είναι ότι οι μπαταρίες ιόντων λιθίου χάνουν περίπου 5% του φορτίου τους ανά μήνα, ενώ οι μπαταρίες NiMH έχουν αντίστοιχο ρυθμό εκφόρτισης 20%
- ***Χαμηλή συντήρηση:*** Ένα πολύ σημαντικό προτέρημα των μπαταριών ιόντων λιθίου είναι ότι δεν απαιτούν συντήρηση για να διασφαλίσουν ποιοτική λειτουργία. Άλλοι τύποι μπαταριών απαιτούν περιοδική εκφόρτιση προκειμένου να διασφαλίσουν πως δεν υπάρχει επίδραση φαινομένων μνήμης. Όπως περιγράφεται στη συνέχεια, οι Li-ion μπαταρίες δεν έχουν αντίστοιχο πρόβλημα, οπότε δεν απαιτούν ανάλογη συντήρηση
- ***Μειωμένο βάρος:*** Άμεση συνέπεια της υψηλής ενεργειακής πυκνότητας, είναι πως για καθορισμένες ενεργειακές απαιτήσεις, μία μπαταρία ιόντων λιθίου είναι πολύ ελαφρύτερη από αντίστοιχες μπαταρίες άλλου τύπου. Οι μπαταρίες Li-ion μπορεί να είναι έως και 40% ελαφρύτερες από τις μπαταρίες νικελίου
- ***Υψηλότερη απόδοση:*** Οι μπαταρίες ιόντων λιθίου περιέχουν ποιοτικότερα χημικά συστατικά σε σχέση με άλλους τύπους μπαταριών. Αυτό μεταφράζεται σε καλύτερη ηλεκτρική αγωγιμότητα, και άρα μικρότερη εσωτερική αντίσταση της μπαταρίας. Επομένως, όπως γίνεται εύκολα κατανοητό έχουμε μικρότερη κατανάλωση ισχύος στην εσωτερική αντίσταση (μικρότερες απώλειες) και άρα μεγαλύτερο βαθμό απόδοσης
- ***Μηδενική επίδραση μνήμης:*** Αυτό μεταφράζεται ως η δυνατότητα να φορτίζουμε και να εκφορτίζουμε την μπαταρία πολλές φορές χωρίς να μειώνεται η χωρητικότητά της. Ακόμη, δεν απαιτείται κάθε φορά πλήρης εκφόρτιση και μετά φόρτιση όπως σε μπαταρίες τύπου νικελίου, προκειμένου να μην δημιουργηθούν κρύσταλλοι στη μπαταρία. Άρα μπορούν να χρησιμοποιηθούν για εκατοντάδες κύκλους φόρτισης και εκφόρτισης

3.4.3 Μπαταρία Li-ion - 3.7V/150mAh

Όπως αναφέρθηκε και παραπάνω, μια μπαταρία ιόντων λιθίου είναι ιδανική για την εφαρμογή μας. Σύμφωνα με τις μετρήσεις της κατανάλωσης της συσκευής, μια μπαταρία με χωρητικότητα μεγαλύτερη ή ίση από 3.7V/150mAh είναι ικανή να κρατήσει τη συσκευή μας σε λειτουργία για περίπου δύο ώρες. Η τελική μπαταρία που επιλέχθηκε (εικόνα 9) με κατασκευαστή την εταιρεία Cellevia Power, έχει μικρές διαστάσεις και βάρος ώστε να μην επιβαρύνει τον αθλητή με επιπρόσθετο βάρος και να μπορεί να τοποθετηθεί οπουδήποτε. Τα χαρακτηριστικά της μπαταρίας είναι:

- ***Τύπος Μπαταρίας***: Li-ion
- ***Τάση Μπαταρίας*:** 3.7Volt
- ***Μέγιστο ρεύμα φόρτισης/ εκφόρτισης:*** 150mA
- ***Χωρητικότητα:*** 150mAh
- ***Διάσταση X:*** 22mm
- ***Διάσταση Y:*** 15mm
- ***Διάσταση Z:*** 7mm

Εικόνα 9: Η μπαταρία της συσκευής

3.5 Σύστημα Αδρανειακής Εκτίμησης (IMU)

Το σύστημα IMU (Inertia Measurment Unit), το οποίο σε ελεύθερη ελληνική μετάφραση σημαίνει “Σύστημα Αδρανειακής Εκτίμησης”, είναι ένα ηλεκτρονικό εξάρτημα που χρησιμοποιείται κυρίως σε μη επανδρωμένα αεροσκάφη καθώς και στα εξελιγμένα κινητά τηλέφωνα. Αποτελείται από αισθητήρες όπως γυροσκόπια (Gyroscope) και επιταχυνσιόμετρα (Accelerometer) τα οποία παρέχουν δεδομένα πραγματικού χρόνου με σκοπό την εκτίμηση του προσανατολισμού ή και της θέσης του στον τρισδιάστατο χώρο. Η τεχνική αυτή ονομάζεται αδρανειακή πλοήγηση (INS). Η αδρανειακή πλοήγηση είναι μία αυτόνομη τεχνική πλοήγησης κατά την οποία μετρήσεις που παρέχονται από επιταχυνσιόμετρα και γυροσκόπια, χρησιμοποιούνται για την παρακολούθηση της θέσης και του προσανατολισμού ενός αντικειμένου σχετικών με την αρχική θέση, προσανατολισμό και ταχύτητα [7][8]. Τα Αδρανειακά Συστήματα Πλοήγησης (INSs) περιέχουν IMUs, που τυπικά

έχουν τρία γυροσκόπια και τρία επιταχυνσιόμετρα, που μετρούν τις τρεις συνιστώσες της γωνιακής ταχύτητας και τις τρεις συνιστώσες της γραμμικής (μη βαρυτικής) επιτάχυνσης αντίστοιχα. Με την επεξεργασία των σημάτων από αυτές τις συσκευές τα INS μπορούν να παρακολουθήσουν τη θέση και τον προσανατολισμό ενός αντικειμένου, όπως ένα UAV. Οι εφαρμογές της αδρανειακής πλοήγησης έχουν διευρυνθεί τα τελευταία χρόνια, λόγω της κατασκευής των MEMS (Microelectromechanical systems), που έκαναν δυνατή την κατασκευή αρκετά μικρών και ελαφρών INS.

3.5.1 Επιταχυνσιόμετρο

Το επιταχυνσιόμετρο είναι ένας αισθητήρας ο οποίος καταγράφει όλες τις δυνάμεις που ασκούνται πάνω του, όπως και την επιτάχυνση σύμφωνα με την βαρύτητα της γης. Είναι πολύ ευαίσθητο στις δυνάμεις που δέχεται και αυτό το καθιστά καλή επιλογή για πολλών ειδών εφαρμογές. Ειδικότερα, χρησιμοποιείται για την ανίχνευση ενός πατήματος της οθόνης, την ανίχνευση της θέσης της οθόνης του κινητού και την ανίχνευση βημάτων. Η ευαισθησία του στις δυνάμεις που του ασκούνται το καθιστά πολύ θορυβώδες για κάθε είδους εφαρμογές που μελετούν την αλλαγή θέσης. Στις περιπτώσεις αυτές το επιταχυνσιόμετρο μπορεί να είναι αξιόπιστο μόνο σε βάθος χρόνου και μόνο αν χρησιμοποιηθεί κάποιου είδους χαμηλοπερατού φίλτρου στα δεδομένα. Αξίζει να σημειωθεί ότι δεν μπορεί να καταγράψει τις μεταβολές της κίνησης της στροφής αριστερά και δεξιά, γιατί η βαρύτητα δεν επιδρά σε αυτόν τον άξονα.

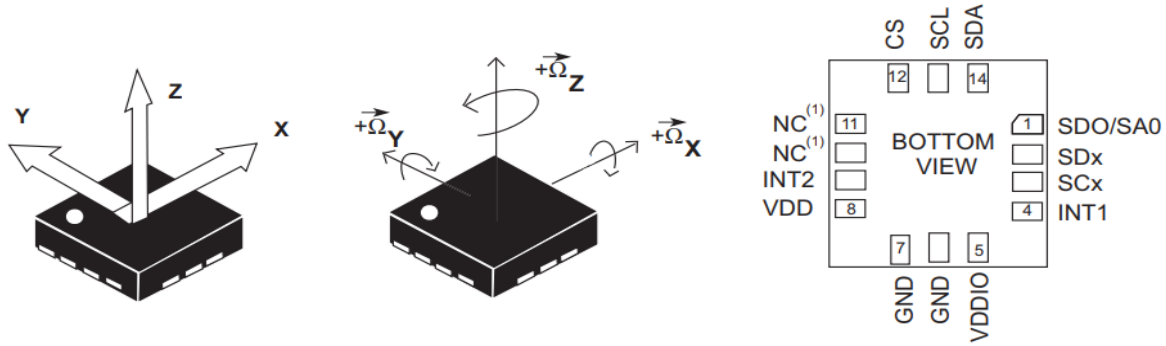
3.5.2 Γυροσκόπιο

Τα γυροσκόπια είναι συσκευές οι οποίες χρησιμοποιούν τις αρχές της αδρανείας και της διατήρησης της στροφορμής για να διατηρήσουν σταθερό προσανατολισμό τους σε σχέση με μία αρχική καθορισμένη διεύθυνση. Έτσι είναι ιδανικές συσκευές για την μέτρηση των γωνιακών περιστρόφων και κατ' επέκταση τον προσδιορισμό των διορθώσεων που πρέπει να επιβληθούν στους φορείς τους ώστε οι τελευταίοι να διατηρήσουν σταθερή διεύθυνση. Τα γυροσκόπια μπορούν να ταξινομηθούν με βάση την αρχή λειτουργίας και τα κατασκευαστικά χαρακτηριστικά τους σε τρεις μεγάλες κατηγορίες τα μηχανικά, οπτικά και τα μικρο-ηλεκτρομηχανικά.

3.5.3 IMU - LSM6DS3

Για τη συγκεκριμένη εργασία επιλέχθηκε το LSM6DS3 της εταιρίας STMicroelectronics. Τα κύρια κριτήρια της επιλογής αυτής ήταν το μικρό package, η μεγάλη συχνότητα δειγματοληψίας, το χαμηλό κόστος και η δυνατότητα προσθήκης μαγνητόμετρου για μελλοντική χρήση/αναβάθμιση. Τα βασικά χαρακτηριστικά του LSM6DS3 είναι:

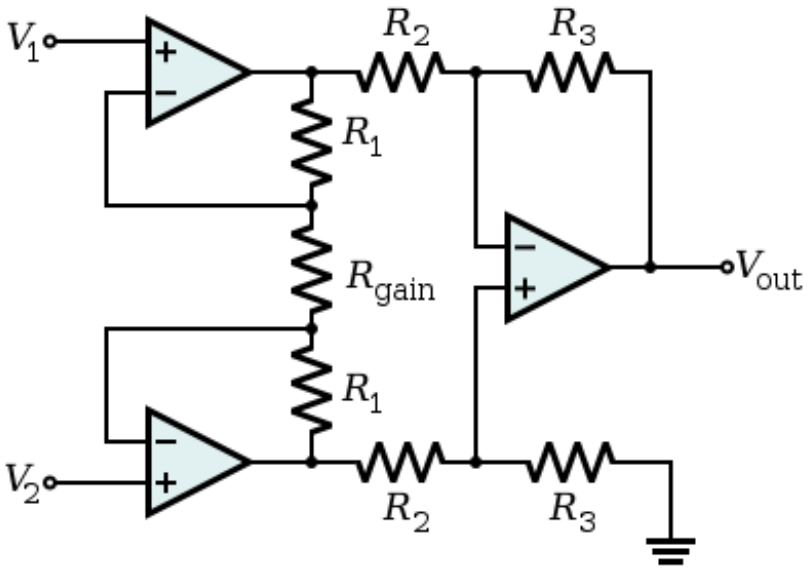
- ***Τάση τροφοδοσίας:*** 1.7V-3.6V
- ***Μέγιστη συχνότητα δειγματοληψίας επιταχυνσιόμετρου***: 6.66KHz
- ***Μέγιστη συχνότητα δειγματοληψίας γυροσκοπίου***: 1.66KHz
- ***Μέγιστη κατανάλωση:*** 1.25mA
- ***Πρωτόκολλα επικοινωνίας:*** SPI,I2C
- ***Διάσταση X:*** 2.5mm
- ***Διάσταση Y:*** 3mm
- ***Διάσταση Z:*** 0.86mm



Εικόνα 10: IMU - LSM6DS3

3.6 Ενισχυτής οργανολογίας

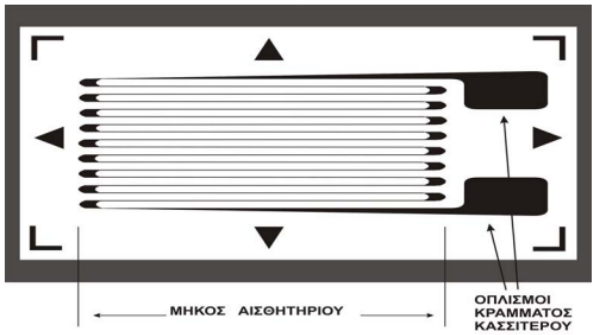
Ο ενισχυτής οργανολογίας είναι ένας τύπος διαφορικού ενισχυτή ο οποίος έχει εξοπλιστεί με δύο τελεστικούς ενισχυτές (buffers), οι οποίοι εξαλείφουν την ανάγκη για αντίσταση εισόδου και έτσι κάνουν τον ενισχυτή ικανό για χρήση σε μετρήσεις και εξοπλισμό δοκιμών. Επιπρόσθετα χαρακτηριστικά είναι ότι περιλαμβάνει πολύ χαμηλή τάση εκτροπής (DC offset), μικρή μετατόπιση, χαμηλό θόρυβο, πολύ υψηλό κέρδος ανοιχτού βρόγχου, πολύ υψηλό λόγο απόρριψης κοινού σήματος και πολύ υψηλή εσωτερική αντίσταση [9]. Αυτοί οι ενισχυτές χρησιμοποιούνται εκεί που απαιτείται μεγάλη ακρίβεια και σταθερότητα στο κύκλωμα τόσο βραχυπρόθεσμα όσο και μακροπρόθεσμα. Αν και οι ενισχυτές μέτρησης οργάνου απεικονίζονται συνήθως σχηματικά όμοια με έναν ιδανικό τελεστικό ενισχυτή, ο ενισχυτής οργανολογίας σχεδόν πάντα αποτελείται εσωτερικά από 3 τελεστικούς ενισχυτές. Αυτοί είναι τοποθετημένοι έτσι ώστε να υπάρχει ένας τελεστικός ενισχυτής για να ρυθμίζει την κάθε είσοδο (+/-) και ένας για να παράγει το επιθυμητό αποτέλεσμα με την κατάλληλη αντίσταση για τη λειτουργία. Ο πιο συχνά χρησιμοποιούμενος ενισχυτής οργανολογίας είναι ο ακόλουθος:



Εικόνα 11: Διάγραμμα ενισχυτή οργανολογίας

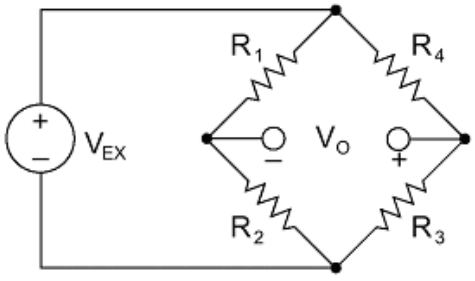
3.6.1 Αισθητήρια Παραμόρφωσης (Strain Gauges)

Η παραμόρφωση είναι ένα φυσικό μέγεθος που την τιμή του την μετράμε με αισθητήρια παραμόρφωσης. Καθώς η πλειονότητα των αισθητηρίων παραμόρφωσης αποτελούνται από πιεζοηλεκτρικά στοιχεία ή από ημιαγωγούς, είναι στο σύνολό τους παθητικά ωμικά στοιχεία κατασκευασμένα με την εναπόθεση (deposing) ή την χαλκογράφηση (etching) ενός αγωγού ή ενός ευαίσθητου πλέγματος ενισχυμένο από λεπτό φύλλο μετάλλου σε ένα υπόστρωμα γνωστό ως περίβλημα φορέα ημιαγωγού (carrier matrix). Η εικόνα 12 παρουσιάζει σε τομή ένα τυπικό είδος αισθητηρίου πίεσης που χρησιμοποιείται ευρύτατα:



Εικόνα 12: Τυπικό είδος αισθητηρίου πίεσης

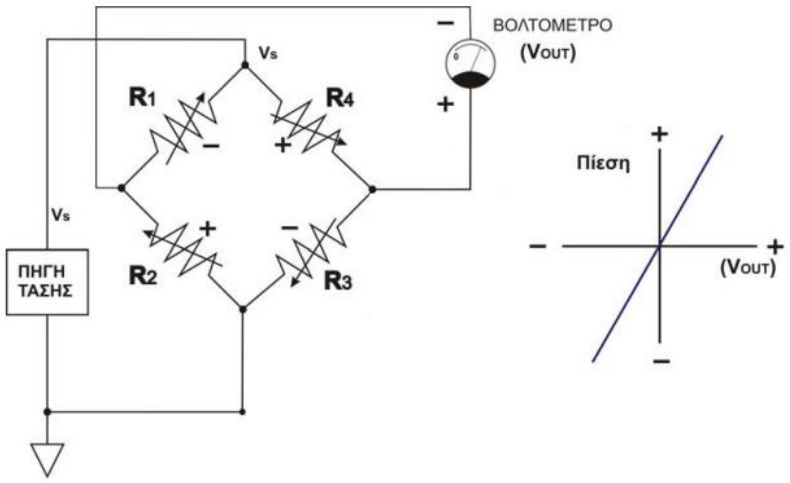
Τα αισθητήρια αυτά είναι κατασκευασμένα από ένα πλέγμα ή από πολλαπλές στρώσεις πλεγμάτων. Επιπλέον πολλές στρώσεις πλεγμάτων μπορούν να παραταχθούν στο περίβλημα του φορέα έτσι ώστε να έχουμε την δυνατότητα να μπορούμε να μετρήσουμε πιέσεις που ασκούνται από διάφορες κατευθύνσεις πάνω στην επιφάνια του αισθητηρίου. Συνήθως τα αισθητήρια παραμόρφωσης είναι προσαρτημένα σε όργανα δοκιμής με ένα είδος σύνδεσης που επιτρέπει την μετάδοση της δύναμης από το αντικείμενο στο αισθητήριο. Παράλληλα απομονώνει την ασκούμενη δύναμη έτσι ώστε να μετρηθεί χωρίς τυχών εξωτερικές αλλοιώσεις άλλων παραγόντων και τέλος εξαλείφει την μεταφορά θερμότητας από το αισθητήριο στο όργανο δοκιμής. Μια προστατευτική επίστρωση μπορεί επίσης να τοποθετηθεί στο εξωτερικό περίβλημα του αισθητηρίου έτσι ώστε να το σφραγίζει και να το προστατεύει από τους εξωτερικούς ηλεκτρικούς θορύβους του περιβάλλοντος. Ωμικά αισθητήρια παραμόρφωσης χρησιμοποιούνται σε συνδεσμολογίες με γέφυρα Wheatstone. Μια τέτοια συνδεσμολογία φαίνεται στην εικόνα 13:



Εικόνα 13: Τυπική συνδεσμολογία γέφυρας Wheatstone

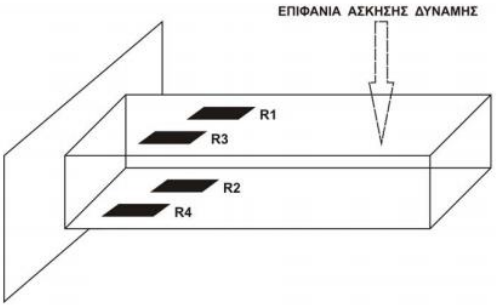
Είναι γνωστό ότι το χαρακτηριστικό της γέφυρας Wheatstone είναι ότι δρα αποτελεσματικά στην ανίχνευση οποιασδήποτε κατάστασης μη ισορροπίας ανάμεσα στις

αντιστάσεις των κλάδων της. Μια τυπική γέφυρα Wheatstone μπορεί να περιέχει από ένα έως τέσσερα ενεργά αισθητήρια παραμόρφωσης. Στην εικόνα 14 βλέπουμε το σχηματικό διάγραμμα μιας τέτοιας γέφυρας που περιέχει τέσσερα τέτοια αισθητήρια, με την μορφή της ισοδύναμης μεταβλητής αντίστασής τους σε κάθε ένα από τους κλάδους της γέφυρας. Οι κοινές γέφυρες Wheatstone περιέχουν αντιστάσεις με τιμές 120Ω, 350Ω, 600Ω, 700Ω και 1000Ω. Όταν όμως η γέφυρα αυτή περιέχει αισθητήρια παραμόρφωσης οι τιμές των αντιστάσεων κυμαίνονται από 25-30Ω έως αρκετά ΚΩ. Σε μια δεδομένη τάση τροφοδοσίας η γέφυρα που διαθέτει υψηλότερης τιμής ωμικά στοιχεία θα παράγει μικρότερα ποσά θερμότητας από ότι μια γέφυρα με μικρής τιμής ωμικά στοιχεία. Παράλληλα σε γέφυρες με ωμικά στοιχεία μεγάλης τιμής, οι αντιστάσεις που έχουν οι αγωγοί που ενώνουν το αισθητήριο με το κύριο μέρος της γέφυρας, την πηγή τροφοδοσίας της καθώς και τα όργανα μέτρησης δεν θα επηρεάζουν πολύ την τιμή των μετρήσεων άρα και την ακρίβειά στις μετρήσεις μας.



Εικόνα 14: Γέφυρα Wheatstone με τέσσερα ενεργά αισθητήρια παραμόρφωσης

Τα τέσσερα αισθητήρια σε μια γέφυρα παράγουν την μέγιστη τιμή μη ισορροπίας άρα και μέγιστη τάση εξόδου (Vout) σε ένα συγκεκριμένο επίπεδο παραμόρφωσης. Σε γέφυρες που περιέχουν δυο ενεργά αισθητήρια, η Vout σε μια καθορισμένη τιμή παραμόρφωσης θα είναι σχεδόν η μισή της τιμής που θα είχε εάν η γέφυρα είχε τέσσερα ενεργά αισθητήρια. Με την ίδια λογική μια γέφυρα που περιέχει ένα ενεργό αισθητήρα σε μια καθορισμένη τιμή παραμόρφωσης θα έχει τιμή Vout ίση με το ένα τέταρτο (1/4) της τιμής που θα είχε με τέσσερα ενεργά αισθητήρια. Σε μια γέφυρα πλήρους διαμόρφωσης, τα δυο αισθητήρια παραμόρφωσης πρέπει να είναι προσαρτημένα στην επιφάνεια που ασκείται δύναμη στο όργανο, ενώ τα άλλα δυο στην αντίθετη πλευρά του οργάνου. Έτσι τα δυο πρώτα αισθητήρια αυξάνουν την τιμή της αντίστασης τους υπό την άσκηση δύναμης (πίεσης) ενώ τα άλλα δυο την μειώνουν. Η εικόνα 15 δείχνει την μορφή μιας τέτοιας κατασκευής:



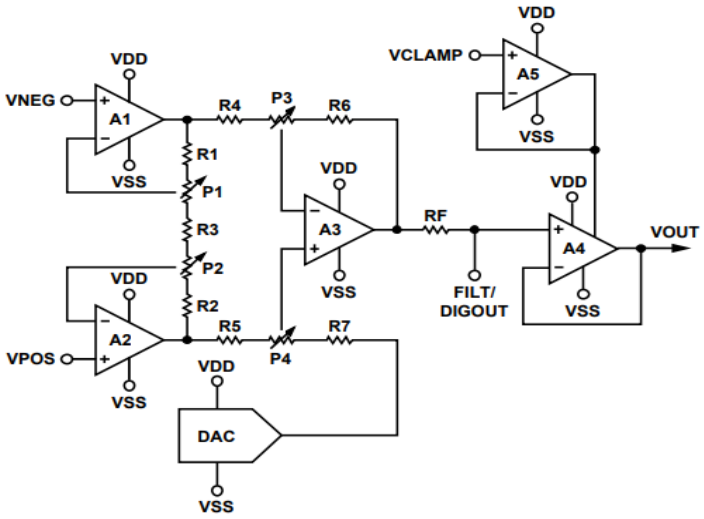
Εικόνα 15: Γέφυρα Wheatstone με τέσσερα ενεργά αισθητήρια παραμόρφωσης

Όπως και οι άλλοι τύποι αισθητήρων που έχουν ωμική συμπεριφορά, έτσι και στα αισθητήρια παραμόρφωσης η ακρίβεια τους μπορεί να επηρεαστεί από τον ηλεκτρικό θόρυβο, από τα θερμοηλεκτρικά ηλεκτρομαγνητικά πεδία (EMFs) και το φαινόμενο θέρμανσης Joule. Για τον λόγο αυτό τα αισθητήρια παραμόρφωσης πρέπει να χρησιμοποιούνται με την ανάλογη ηλεκτρομαγνητική θωράκιση και να λαμβάνουμε υπόψη τα μεταλλικά στοιχεία των επαφών, την τάση τροφοδοσίας και την θερμοκρασία του περιβάλλοντος που με την παρουσία τους μειώνουν την ακρίβεια των μετρήσεων μας. Εδώ πρέπει να σημειωθεί όπως θα δούμε σε επόμενη παράγραφο ότι οι γέφυρες μισής διαμόρφωσης και διαμόρφωσης ενός τετάρτου, είναι περισσότερο ευαίσθητες σε σφάλματα μη γραμμικότητας όταν λειτουργούν σε περιοχές μη ισορροπίας από ότι η γέφυρες πλήρους διαμόρφωσης. Επίσης τα δυο πρώτα είδη διαμορφώσεων επηρεάζονται περισσότερο από σφάλματα που προκύπτουν με επίδραση της αντίστασης των ενδιάμεσων αγωγών ανάμεσα στην γέφυρα και τα αισθητήρια παραμόρφωσης. Με την χρήση της διαμόρφωσης πλήρους γέφυρας όλα τα προαναφερθέντα σφάλματα μπορούν να μειωθούν στο ελάχιστο ή ακόμα και να εξαλειφθούν

3.6.2 Προγραμματιζόμενος Ενισχυτής – AD8555

Όπως αναφέρθηκε και παραπάνω, η έξοδος της γέφυρα θα πρέπει να ενισχυθεί σε επίπεδα τέτοια που να μπορεί ο ADC του μικροελεγκτή να μετρήσει το σήμα εξόδου. Για ελάχιστο αριθμό εξαρτημάτων στη πλακέτα αλλά και για να έχουμε προγραμματιζόμενα κέρδη απολαβής, επιλέχθηκε το ολοκληρωμένο εξάρτημα AD8555 της εταιρείας Analog Devices. Ο AD8555 είναι ένας ενισχυτής με ψηφιακά προγραμματιζόμενη απολαβή και αντιστάθμιση. Η ενίσχυση (Gain) ρυθμίζεται από το 70 έως το 1280 και η αντιστάθμιση (Offset) από 0 έως VCC. Επιπλέον, παρουσιάζει μηδενική ολίσθηση (Zero-Drift). Τα κύρια χαρακτηριστικά του είναι:

- ***Τάση τροφοδοσίας:*** 2.7V-5.5V
- ***Τάση ασσυμετρίας εισόδου:*** 10μV
- ***Λόγος απόρριψης κοινού σήματος CMRR:*** 96dB
- ***Μέγιστη κατανάλωση:*** 2.5mA



Εικόνα 16: Σχηματικό διάγραμμα του AD8555

Το μετρούμενο σήμα εισέρχεται στον ολοκληρωμένο διαφορικό ενισχυτή από τους ακροδέκτες VPOS και VNEG. Τα στοιχεία A1, A2, R1, R2, R3, P1 και P2 σχηματίζουν το πρώτο στάδιο απολαβής του διαφορικού ενισχυτή. Οι τελεστικοί ενισχυτές P1 και P2 είναι αυτό-μηδενιζόμενοι μειώνοντας το σφάλμα αντιστάθμισης στην είσοδο. Τα ποτενσιόμετρα P1 και P2 προγραμματίζονται ψηφιακά προσφέροντας το πρώτο στάδιο της απολαβής από 4 έως 6.4 με ανάλυση 7-bit. Η απολαβή πρώτου σταδίου GAIN1, περιγράφεται από την εξίσωση:

$$GAIN1 \approx 4 \times \left(\frac{6.4}{4}\right)^{\left(\frac{Code}{127}\right)} \qquad (1)$$

Η παράμετρος G1 προγραμματίζεται στο ολοκληρωμένο από τον μικροελεγκτή. Τα στοιχεία A3, R4, R5, R6, R7, P3 και P4 σχηματίζουν το δεύτερο στάδιο απολαβής του διαφορικού ενισχυτή. Ο τελεστικός ενισχυτής P3 είναι επίσης αυτό-μηδενιζόμενος. Τα ποτενσιόμετρα P3 και P4 προγραμματίζονται ψηφιακά προσφέροντας το δεύτερο στάδιο της απολαβής από 17,5 έως 200 με ανάλυση 3bit. Η απολαβή δευτέρου σταδίου GAIN2 σαν συνάρτηση της προγραμματιζόμενης παραμέτρου G2 δίνεται στον Πίνακα 3:

Second Stage Gain Code	Second Stage Gain
0	17.5
1	25
2	35
3	50
4	70
5	100
6	140
7	200

Πίνακας 3: Απολαβή δευτέρου σταδίου

Για την παραγωγή μεταβλητής τάσης αντιστάθμισης στην έξοδο του ενισχυτή χρησιμοποιείται ένας μετατροπέας ψηφιακού σε αναλογικό σήμα (DAC). Η τάση εξόδου δίνεται από την εξίσωση 2:

$$VDAC \approx \left(\frac{Code + 0.5}{256} \right) (VDD - VSS) + VSS \quad (2)$$

Η αντίσταση RF του Σχήματος 1 σε συνδυασμό με έναν εξωτερικό πυκνωτή 10nF που είναι συνδεδεμένος στην έξοδο "FILT" του ολοκληρωμένου σχηματίζουν ένα χαμηλοπερατό φίλτρο με κατώφλι το 1kHz. Το φιλτραρισμένο σήμα διέρχεται από το Buffer A4 για να μειωθεί η εμπέδιση του και να περιοριστεί το πλάτος του στην τιμή της τάσης αναφοράς VCLAM. Το σήμα VOUT που εξέρχεται από την VOUT είναι ενισχυμένο, φιλτραρισμένο και θα οδηγηθεί στο ADC του μικροελεγκτή. Η συνολική ενίσχυση είναι το γινόμενο της απολαβής πρώτου και δεύτερου σταδίου. Τελικά, η τάση εξόδου, VOUT, εκφράζεται συναρτήσει της τάσης εισόδου VPOS-VNEG, των δυο συντελεστών απολαβής, GAIN1 και GAIN2, και της τάσης αντιστάθμισης VDAC μέσω της εξίσωσης 3:

$$VOUT = GAIN(VPOS - VNEG) + VDAC \quad (3)$$

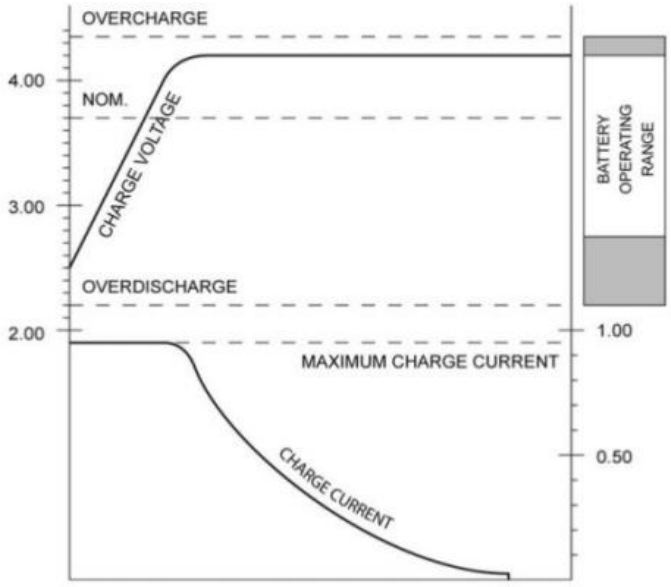
3.7 Φορτιστής μπαταριών Ιόντων Λιθίου

Οι μπαταρίες ιόντων λιθίου έχουν τον υψηλότερο λόγο ενέργειας προς βάρος μπαταρίας και τον υψηλότερο λόγο ενέργειας προς μέγεθος σε σύγκριση με όλες τις άλλες μοντέρνες επαναφορτιζόμενες μπαταρίες. Επίσης είναι το ταχύτερα διαδιδόμενο σύστημα επαναφορτιζόμενων μπαταριών στην αγορά, με εφαρμογές που αφορούν στην αγορά των φορητών υπολογιστών, των κινητών τηλεφώνων κλπ. Συγκρινόμενη με τις παραδοσιακές επαναφορτιζόμενες μπαταρίες, οι Li-Ion μπαταρίες έχουν ιδιαίτερα χαμηλή εσωτερική αντίσταση, υψηλό κύκλο ζωής, γρήγορο χρόνο φόρτισης, χαμηλό ρυθμό αυτό-εκφόρτισης και χαμηλή τοξικότητα. Στην πράξη, υπάρχει μόνο ένας τρόπος να φορτιστεί μια μπαταρία ιόντων λιθίου. Συνήθως, οι κατασκευαστές φροντίζουν έτσι ώστε να δίνουν μαζί με το προϊόν και πολύ αυστηρούς κανόνες φόρτισης. Κατ' επέκτασιν λοιπόν και οι σχεδιαστές φορτιστών για Li-Ion μπαταρίες θα πρέπει να τηρούν κατά γράμμα αυτούς τους κανόνες.

Οι μπαταρίες Li-Ion φορτίζονται χρησιμοποιώντας μία σταθερή τάση, σε συνδυασμό με έναν περιοριστή ρεύματος έτσι ώστε να αποφεύγεται η υπερθέρμανση κατά τη διάρκεια της αρχικού σταδίου της διαδικασίας φόρτισης. Η φόρτιση τερματίζεται όταν η τιμή του ρεύματος φόρτισης πέφτει κάτω από ένα όριο το οποίο θέτει κάθε φορά ο εκάστοτε κατασκευαστής. Σημειώνεται πως η υπερφόρτιση μπορεί να οδηγήσει σε καταστροφή της μπαταρίας και για αυτό θα πρέπει να λαμβάνουν χώρα ειδικοί έλεγχοι έτσι ώστε να ανιχνεύεται η πλήρης φόρτιση της μπαταρίας και να τερματίζεται έγκαιρα. Σημειώνεται πως ο στατικός ηλεκτρισμός ή ένας χαλασμένος φορτιστής μπορούν να καταστρέψουν το κύκλωμα προστασίας της μπαταρίας και να τεθούν οι διακόπτες στερεής κατάστασης μονίμως σε ON. Αυτό μπορεί να συμβεί εν αγνοία του χρήστη. Μια μπαταρία χωρίς κύκλωμα προστασίας μπορεί να φαίνεται πως συμπεριφέρεται φυσιολογικά αλλά δεν παρέχει καμία προστασία κατά της κακής χρήσης της μπαταρίας. Θα πρέπει να σημειωθεί πως οι τυπικές μπαταρίες Li-Ion, δεν μπορούν και δεν πρέπει να φορτίζονται σε θερμοκρασίες χαμηλότερες των 0C. Αν φορτιστούν σε χαμηλές θερμοκρασίες, προκαλούνται χημικές αντιδράσεις στο εσωτερικό των κελιών της μπαταρίας οι οποίες μπορούν να προκαλέσουν μόνιμες καταστροφές και να καταστήσουν τη χρήση της μπαταρίας επικίνδυνη. Σημειώνεται επίσης πως οι μπαταρίες Li-Ion θα πρέπει να προστατεύονται από συγκρούσεις και από υψηλούς ρυθμούς φόρτισης καθώς

και αυτοί οι δύο παράγοντες μπορούν να προκαλέσουν βλάβες σε αυτόν τον τύπο επαναφορτιζόμενης μπαταρίας. Υπάρχουν δύο στάδια φόρτισης των μπαταριών ιόντων λιθίου:

Στάδιο σταθερού ρεύματος: Η φόρτιση μιας μπαταρίας ιόντων λιθίου ξεκινά με την εφαρμογή σταθερού ρεύματος στη μπαταρία. Το μέγεθος του ρεύματος εξαρτάται από την εκάστοτε μπαταρία και καθορίζεται ως προδιαγραφή από τον κατασκευαστή. Αυτό το στάδιο φόρτισης ολοκληρώνεται όταν η τάση της μπαταρίας φτάσει στο κατώφλι το οποίο επίσης δίνει ο κατασκευαστής της μπαταρίας.
Στάδιο σταθερής τάσης: Αφού η μπαταρία πιάσει το κατώφλι τάσης στο πρώτο στάδιο φόρτισης, πλέον ο φορτιστής τροφοδοτεί την μπαταρία με σταθερή τάση (ενώ το ρεύμα πλέον μεταβάλλεται). Αυτό το στάδιο φόρτισης ολοκληρώνεται μόλις το ρεύμα φόρτισης πέσει κάτω από ένα κατώφλι το οποίο καθορίζεται από τον εκάστοτε κατασκευαστή.

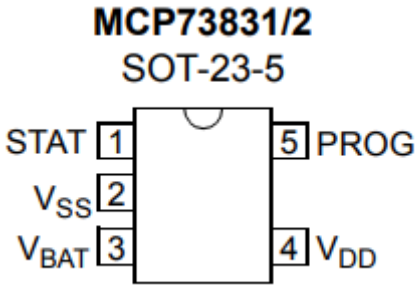


*Εικόνα 17: Γραφική χ*ρονικής εξέλιξης της τάσης-ρεύματος κατά τη διάρκεια φόρτισης

3.7.1 Φορτιστής MCP73831

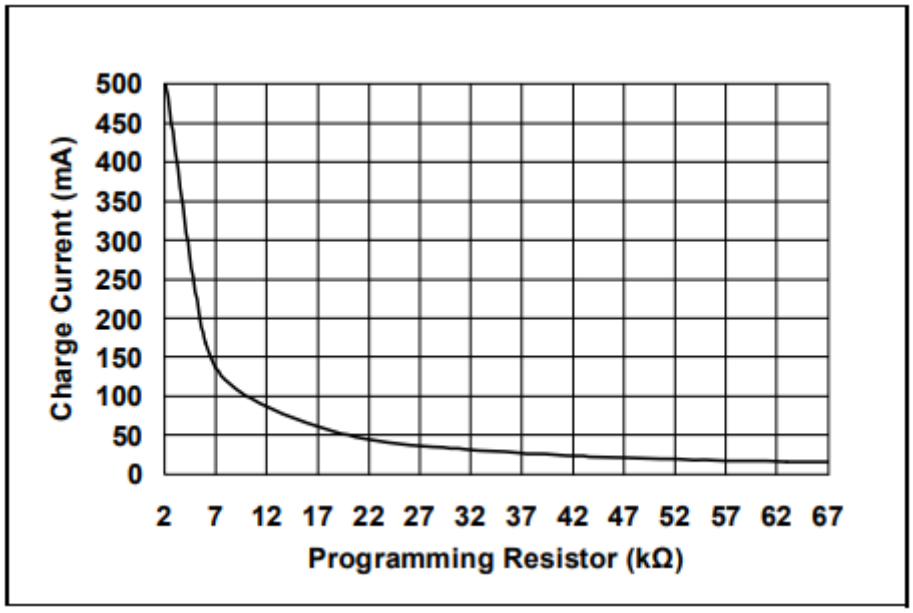
Το ολοκληρωμένο MCP73831 είναι ένας γραμμικός ελεγκτής φόρτισης που μπορεί να αναλάβει τη φόρτιση μπαταριών Li-Ion και Li-Po, εφόσον συμπίπτουν οι τεχνικές προδιαγραφές. Το κόστος αλλά και τα ελάχιστα εξωτερικά υλικά, ήταν τα κύρια κριτήρια για την επιλογή. Η πλήρης ονομασία του ολοκληρωμένου κυκλώματος είναι MCP73831T-2ACI/OT. Το τμήμα “MCP73831” της ονομασίας υποδηλώνει το μοντέλο της συσκευής, ενώ στη συνέχεια παρατίθενται ορισμένα ιδιαίτερα, πιο συγκεκριμένα χαρακτηριστικά. Ο χαρακτήρας “T” αναφέρεται στο Tape and Reel “πακετάρισμα” πολλαπλών μονάδων. Ο επόμενος αριθμός “2” σχετίζεται με τη μέγιστη τάση φόρτισης της μπαταρίας, δηλαδή τα 4.2V. Άλλες διαθέσιμες επιλογές είναι τα 4.35V, 4.4V, 4.5V, που δεν είναι κατάλληλες όμως για την περίπτωσή μας. Οι χαρακτήρες “AC” ορίζουν τις τιμές μίας πληθώρας χαρακτηριστικών που θα δούμε αναλυτικά παρακάτω. Τέλος, το γράμμα “I” δηλώνει την

αντοχή σε θερμοκρασίες βιομηχανικού περιβάλλοντος, ενώ η κατάληξη “OT” τη συσκευασία SOT-23.



Εικόνα 18: Διάταξη ακροδεκτών του MCP73831

STAT: Είναι pin τριών καταστάσεων (Low, High και Hi-Z). Θα δούμε πότε βρίσκεται στην κάθε κατάσταση στη συνέχεια. Στο μοντέλο MCP73832 το pin αυτό είναι δύο καταστάσεων (Low και Hi-Z), αφού καταλήγει στην ανοιχτοκυκλωμένη καταβόθρα (open-drain) ενός FET transistor και αποτελεί τη μόνη διαφορά μεταξύ των δύο μοντέλων.
VSS: Το pin αυτό συνδέεται στη γείωση του συστήματος.
VBAT: Το pin VBAT συνδέεται με τον θετικό ακροδέκτη της μπαταρίας μας. Θα συνδέσουμε ακόμη μεταξύ του pin αυτού και της γείωσης έναν πυκνωτή παράκαμψης (bypass capacitor) 10uF, για αύξηση της σταθερότητας του κυκλώματος. Είναι χρήσιμος κυρίως στο στάδιο της σταθερής τάσης (constant voltage) που θα δούμε παρακάτω, για την εξάλειψη φαινομένων AC. Πρακτικά οποιουδήποτε είδους πυκνωτής καλής ποιότητας μπορεί να χρησιμοποιηθεί σε αυτό το σημείο, αλλά μία γενικά καλή λογική είναι η χρήση κεραμικών πυκνωτών χαμηλής ισοδύναμης αντίστασης σειράς (ESR).
VDD: Το pin αυτό τροφοδοτείται με τάση από το USB καλώδιο. Μεταξύ αυτού του pin και της γείωσης πρέπει επίσης να συνδεθεί ένας bypass πυκνωτής μεγέθους 10uF, για εξομάλυνση της τάσεως κατά τα μεταβατικά φαινόμενα και απόσβεση των AC συνιστωσών.
PROG: Το PROG pin μας δίνει τη δυνατότητα να προγραμματίσουμε το ρεύμα φόρτισης, συνδέοντας μία αντίσταση μεταξύ αυτού και της γείωσης. Η αντίσταση προγραμματισμού προέρχεται από το datasheet (Εικόνα 19) του ολοκληρωμένου και επιλέχθηκε να είναι 15kΩ για ρεύμα φόρτισης περίπου ίσο με ~70mA



Εικόνα 19: Χαρακτηριστική καμπύλη ρεύματος φόρτισης και RPROG

Ο κύκλος φόρτισης της μπαταρίας αποτελείται συνολικά από 5 διαφορετικές καταστάσεις λειτουργίας του MCP73831. Στην περίπτωσή μας θα συναντήσουμε μόνο τις 4 από αυτές:

Shutdown Mode: Το MCP73831 έχει ενσωματωμένο ένα under-voltage lockout (UVLO) κύκλωμα. Όταν η τάση στο pin VDD είναι μικρότερη από περίπου 3.38V ή από την τάση της μπαταρίας προσαυξημένη κατά 50mV, το ολοκληρωμένο κύκλωμα παύει να λειτουργεί, έως ότου η τάση γίνει μεγαλύτερη των 3.45V ή της τάσεως της μπαταρίας προσαυξημένης κατά 150mV. Παρατηρούμε ότι και στις δύο περιπτώσεις, δημιουργείται ένας βρόχος υστέρησης, ο οποίος έχει εύρος μέχρι 100mV. Στην κατάσταση αυτή δεν υφίσταται φόρτιση της μπαταρίας και το ρεύμα διαρροής από την μπαταρία προς το MCP73831 είναι ελάχιστο – μικρότερο των 2uA. Το pin STAT βρίσκεται σε κατάσταση υψηλής αντίστασης (Hi-Z).

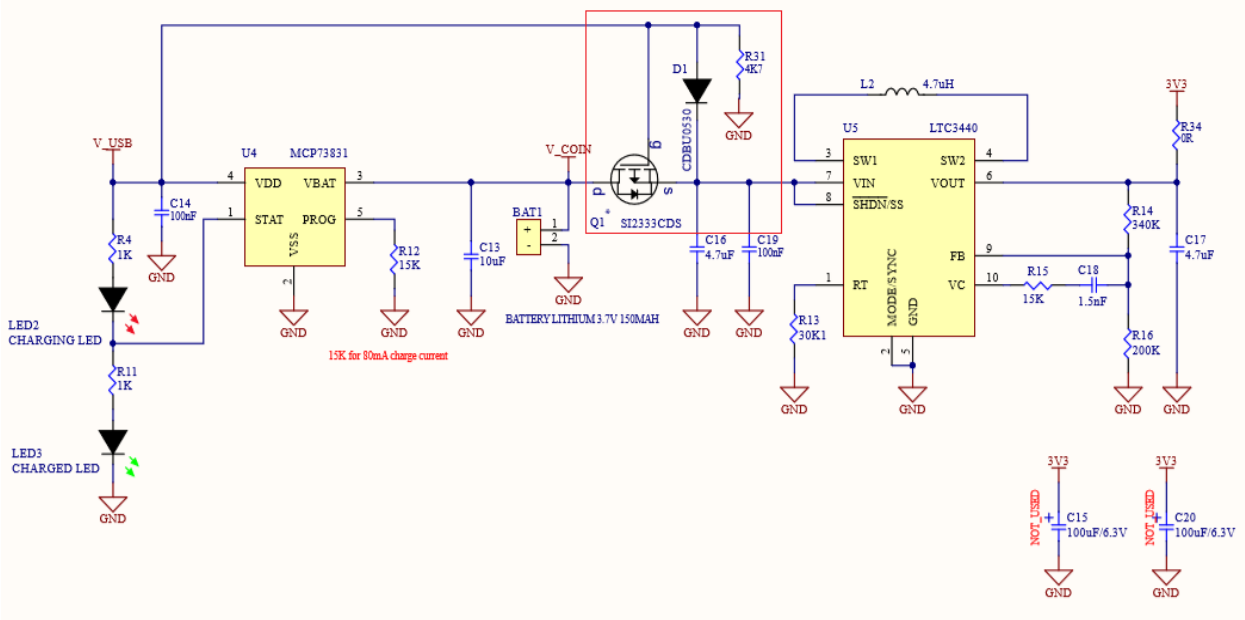
Preconditioning Mode: Για να εισέλθει ο φορτιστής σε αυτήν τη λειτουργία, πρέπει η τάση στο VDD να καλύπτει τις προϋποθέσεις της προηγούμενης παραγράφου και η τάση της μπαταρίας να είναι μικρότερη του 66% της μέγιστης τιμής της, δηλαδή των 2.77V περίπου. Καθώς όμως στην περίπτωσή μας η τάση της μπαταρίας δεν είναι δυνατόν να μειωθεί σε τιμή μικρότερη των 2.9V, αποκλείεται να βρεθούμε σε αυτή την κατάσταση. Το preconditioning προετοιμάζει ουσιαστικά μπαταρίες που έχουν εκφορτιστεί περισσότερο από όσο πρέπει, φορτίζοντάς τες μόνο με το 10% του ρεύματος IPROG (τα προηγούμενα ποσοστά μπορεί να διαφέρουν, αν στην ονομασία MCP73831T-2ACI/OT υπάρχουν άλλοι χαρακτήρες αντί των “AC”). Το pin STAT σε αυτήν την περίπτωση έχει χαμηλή στάθμη (Low).

Fast Charge/Constant-Current Mode: Μετά το preconditioning και μόλις η τάση του VBAT pin ξεπεράσει τα 2.77V, η φόρτιση συνεχίζεται με το ρεύμα IPROG. Και σε αυτή την περίπτωση η τιμή του STAT pin είναι Low.

Constant-Voltage Mode: Μόλις η τάση της μπαταρίας φτάσει στη μέγιστη τιμή της (με μία ανοχή 0.75%), παύει η φόρτιση σταθερού ρεύματος και το pin VBAT τροφοδοτεί με σταθερή τάση 4.2V. Επίσης, το pin STAT έχει στάθμη Low.

Charge Complete Mode: Η έξοδος από την προηγούμενη κατάσταση επέρχεται μόνο όταν το μέσο ρεύμα που ρέει από τον φορτιστή προς την μπαταρία γίνει μικρότερο του 7.5% του IREG, το οποίο μεταφράζεται σε **ITERM =0.075*IREG=11.25mA** . Το γεγονός αυτό υποδεικνύει ότι η φόρτιση ολοκληρώθηκε και το pin STAT γίνεται High. Για να αρχίσει ένας νέος κύκλος φόρτισης, πρέπει η τάση της μπαταρίας να μειωθεί σε λιγότερο από το 94% της μέγιστης τιμής, που ισούται με **VRTH=0.94·4.2V≃3.95V** . Τότε θα βρεθούμε πάλι στο Fast Charge Mode, εκτός και αν κάποια άλλη συνθήκη μας έχει ήδη οδηγήσει το κύκλωμα σε Shutdown Mode.

Ο φορτιστής τροφοδοτείται μέσω USB με 5V και θέλουμε όσο φορτίζεται η μπαταρία, το υπόλοιπο κύκλωμα να λειτουργεί κανονικά χωρίς να επηρεάζει τη διαδικασία φόρτισης. Όσο υπάρχει παρουσία φορτίου συνδεδεμένου στο κύκλωμα το οποίο καταναλώνει ρεύμα τουλάχιστον ίσο με το ITERM, η φόρτιση δεν ολοκληρώνεται. Βέβαια στην περίπτωσή μας και λόγω της υψηλής απαίτησης ρεύματος, το κύκλωμα μεταβαίνει πάλι στην κατάσταση Fast Charge αντί να παραμείνει μόνιμα σε Constant-Voltage Mode, λόγω εκφόρτισης της μπαταρίας, οπότε το πρόβλημα ανάγεται στο προηγούμενο.



Εικόνα 20: Σχηματικό κυκλώματος Load Share

Για το διαμοιρασμό του φορτίου απαιτούνται μόνο 3 επιπλέον ηλεκτρονικά στοιχεία. Το πρώτο στοιχείο και βασικότερο, είναι το P-MOSFET ισχύος SI2333CDS (Q1). Όπως φαίνεται και από το σχεδιάγραμμα, με την προσθήκη της Pull-Down αντίστασης R31 και όταν το κύκλωμα δεν τροφοδοτείται με 5V από το USB, η τάση πύλης του Q1 ισούται με 0V . Τότε, εξαιτίας της παρασιτικής διόδου του Q1, η ελάχιστη τάση πηγής του θα είναι VDSmin=VBATmin−VDSPAR =2.9V−0.7V=2.2V. Άρα, VGS=0V−2.2V= −2.2V οπότε το Q1 αρχίζει να άγει, παρακάμπτοντας την παρασιτική δίοδο. Ως αποτέλεσμα, το φορτίο του συστήματος τροφοδοτείται με την τάση της μπαταρίας μας, όπως ακριβώς θα συνέβαινε και στον αρχικό σχεδιασμό (θεωρούμε αμελητέα την πτώση τάσης VDS κατά μήκος του Q1). Να σημειώσουμε ότι η τάση VGS(th) του επιλεγμένου transistor δεν πρέπει είναι μικρότερη των 1V κατά απόλυτη τιμή, διαφορετικά αυτό δε θα άγει όταν η τάση VBAT είναι η ελάχιστη.

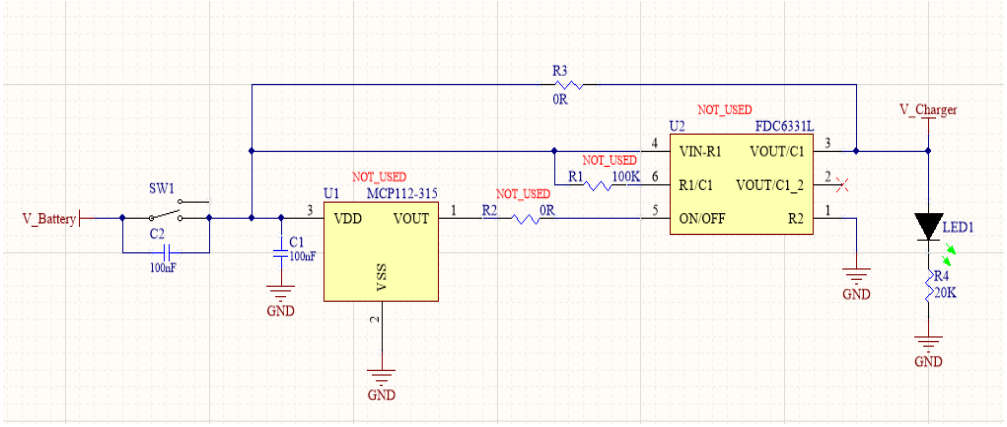
Πρέπει να προσέξουμε ώστε η τιμή της αντίστασης RDS(on) του Q1 να είναι χαμηλή, για να ελαχιστοποιηθούν οι απώλειες ισχύος και η πτώση τάσεως κατά μήκος της.

Για το MOSFET που επιλέξαμε, η μέγιστη RDS(on) είναι μόλις 60mΩ. Σε περίπτωση που συνδέσουμε το USB, η VGS ισούται με την πτώση τάσης της διόδου D1 που είναι πάντα θετική τιμή, οπότε το Q1 δεν άγει. Άρα η μπαταρία μας αποσυνδέεται από το φορτίο και η φόρτισή της δεν επηρεάζεται από τη λειτουργία του υπολοίπου συστήματος. Το φορτίο σε αυτή την περίπτωση τροφοδοτείται μέσω της διόδου D1, απευθείας από το τροφοδοτικό, χωρίς να περιορίζεται από το μέγιστο ρεύμα φόρτισης του MCP73831, όπως στον αρχικό σχεδιασμό. Η δίοδος CDBU0530 (D1) χρησιμεύει στο να αποτρέψει τη ροή ρεύματος από τη μπαταρία προς τον ακροδέκτη VDD, όταν το τροφοδοτικό δεν είναι συνδεδεμένο. Η D1 μπορεί να υποστηρίξει μέγιστο ρεύμα κανονικής πόλωσης μέχρι και 3A, το οποίο αρκεί για τις απαιτήσεις του φορτίου. Επιλέξαμε μία δίοδο τύπου Schottky, για να ελαχιστοποιηθούν οι απώλειες ισχύος, λόγω της μικρότερης πτώσης τάσεως που παρουσιάζει (η οποία είναι 0.2V στην περίπτωσή μας). Η επιλογή αυτή δεν είναι δεσμευτική βέβαια, αν και η μέγιστη πτώση τάσεως δεν πρέπει να ξεπερνά μία ορισμένη τιμή, έπειτα από την οποία ενδέχεται να υπάρχει πρόβλημα με την εσωτερική, παρασιτική δίοδο του Q1. Η ελάχιστη τάση τροφοδοσίας που μπορεί να δώσει το τροφοδοτικό στο pin VDD ώστε να λειτουργεί ο ελεγκτής φόρτισης, είναι 4.5V. Η παρασιτική δίοδος του Q1 σύμφωνα με το σχετικό datasheet, ενεργοποιείται όταν η τάση $V_{DS\_par} \geqslant 0.7V$. Αν υποθέσουμε ότι η μπαταρία μας έχει φορτιστεί στη μέγιστη τάση 4.2V, τότε η μέγιστη πτώση τάσεως της D1 επιτρέπεται να είναι $V_{D1} = V_{DD} - (V_{BAT} - V_{DS\_par}) = 5V - (4.2V - 0.7V) = 1.5V$.

3.7.2 Προστασία εκφόρτισης

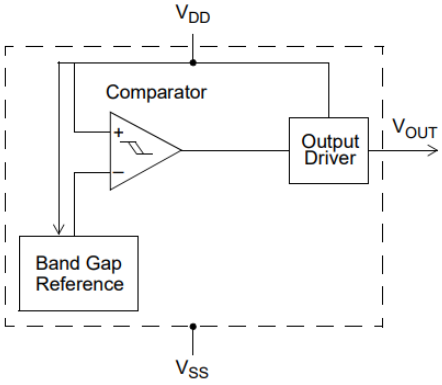
Για τη συγκεκριμένη εργασία, η μπαταρία που επιλέχθηκε (όπως και αρκετές άλλες μπαταρίες Li-ion) έχει ενσωματωμένη προστασία που αποκόπτει τη μπαταρία από το υπόλοιπο κύκλωμα όταν η τάση της είναι μικρότερη από ~2.9V με σκοπό να την προστατέψει. Για ορισμένες μπαταρίες που δεν έχουν τη συγκεκριμένη προστασία, υλοποιήσαμε ένα κύκλωμα που αποκόπτει τη μπαταρία όταν η τάση στα άκρα της γίνεται μικρότερη από 3.15V. Το κύκλωμα ουσιαστικά αποτελείται από το ολοκληρωμένο MCP112-315 (U1), το διακόπτη FDC6331L (U2) και την Pull-Up αντίσταση R1.

Να σημειωθεί πως στην τελική πλακέτα η αντίσταση R3 παρακάμπτει το κύκλωμα που αναφέρουμε μιας και έχει σχεδιαστεί για μπαταρίες που δεν έχουν αντίστοιχο κύκλωμα.



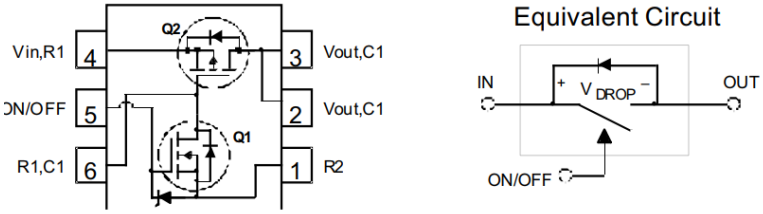
Εικόνα 21: Το κύκλωμα προστασίας από ολική εκφόρτιση

Το ολοκληρωμένο MCP112-315 (U1), συγκρίνει τη τάση της μπαταρίας και αναλόγως οδηγεί το FDC6331L. Τα κύρια κριτήρια για την επιλογή του ήταν το χαμηλός κόστος, τα ελάχιστα εξωτερικά υλικά, η χαμηλή κατανάλωση (μέγιστη 1.75μA) και η υστέρηση που μας προσφέρει. Για το συγκεκριμένο κωδικό (τα τελευταία 3 ψηφία στο τέλος του κωδικού, αναγράφουν την τάση αποκοπής), η υστέρηση είναι από 31mV έως 185mV στους 25C.



Εικόνα 22: Μπλοκ διάγραμμα του MCP112-315

Το FDC6331L της εταιρείας Fairchild που λειτουργεί ως διακόπτης, αποτελείται από δύο Mosfet (P και N) με RDSon μικρότερη από 70mΩ και μέγιστο ρεύμα 2.8A.



Εικόνα 23: Ισοδύναμο κύκλωμα του FDC6331L

3.8 Μνήμη Flash

Η μνήμη Flash είναι είδος μνήμης, το οποίο δεν απαιτεί την παροχή ηλεκτρικού ρεύματος προκειμένου να διατηρήσει τα δεδομένα που είναι αποθηκευμένα σε αυτήν. Γνωστή και με τον όρο ως non-volatile memory. Εφαρμόζοντας ηλεκτρικό ρεύμα μπορεί να διαγραφεί και να αναπρογραμματιστεί αρκετές φορές και χωρίζεται σε δυο κατηγορίες, NOR Flash και NAND Flash μνήμη, οι οποίες πήραν το όνομά τους από την μέθοδο που είναι δομημένες κατασκευαστικά. Σε αντίθεση με την μνήμη NAND, η μνήμη NOR μπορεί να διαβαστεί ανά byte συνεχόμενα και για αυτό το λόγο συχνά χρησιμοποιείται όταν ο πρωταρχικός σκοπός της μνήμης είναι η αποθήκευση και η εκτέλεση του κώδικα εκκίνησης (Bootloader) μιας ηλεκτρονικής συσκευής. Ενώ μέρος της μνήμης μπορεί επίσης να χρησιμοποιηθεί και για την αποθήκευση δεδομένων του χρήστη. Οι περισσότερες όμως φορητές συσκευές, όπως τα USB, οι σύγχρονες ψηφιακές κάμερες, τα Smartphones χρησιμοποιούν μνήμες NAND που τους επιτρέπουν την άμεση προσάρτηση αποθηκευτικού χώρου για την αποθήκευση δεδομένων.

Το χαρακτηριστικό της τυχαίας προσπέλασης που παρουσιάζουν οι μνήμες NOR Flash, δίνει την δυνατότητα υλοποίησης της λειτουργίας execute-in-place (XiP), η οποία

συχνά απαιτείται στα ενθυλακωμένα συστήματα. Παρόλα αυτά ένας μεγάλος αριθμός επεξεργαστών περιλαμβάνουν απευθείας διεπαφή για την χρήση NAND Flash μνήμης, προσφέροντας την δυνατότητα εκκίνησης (boot) της συσκευής μέσω μνήμης NAND Flash, χωρίς την χρήση NOR Flash. Οι συγκεκριμένοι επεξεργαστές αποτελούν μια πρακτική λύση όταν το κόστος, η χωροταξία και ο διαθέσιμος αποθηκευτικός χώρος της συσκευής, αποτελούν σημαντικοί παράγοντες [10].

3.8.1 NOR-Flash AT45DB641E

Για την αποθήκευση πακέτων και των παραμέτρων του συστήματος, επιλέχθηκε μία εξωτερική μνήμη NOR-Flash με χωρητικότητα 8MB. Η συγκεκριμένη μνήμη είναι η AT45DB641E της εταιρείας Adesto. Το κύριο κριτήριο κατά την επιλογή της ήταν η εμπειρία που είχα με παρόμοιες μνήμες της ίδιας οικογένειας και το πρωτόκολλο επικοινωνίας. Τα κύρια χαρακτηριστικά της είναι:

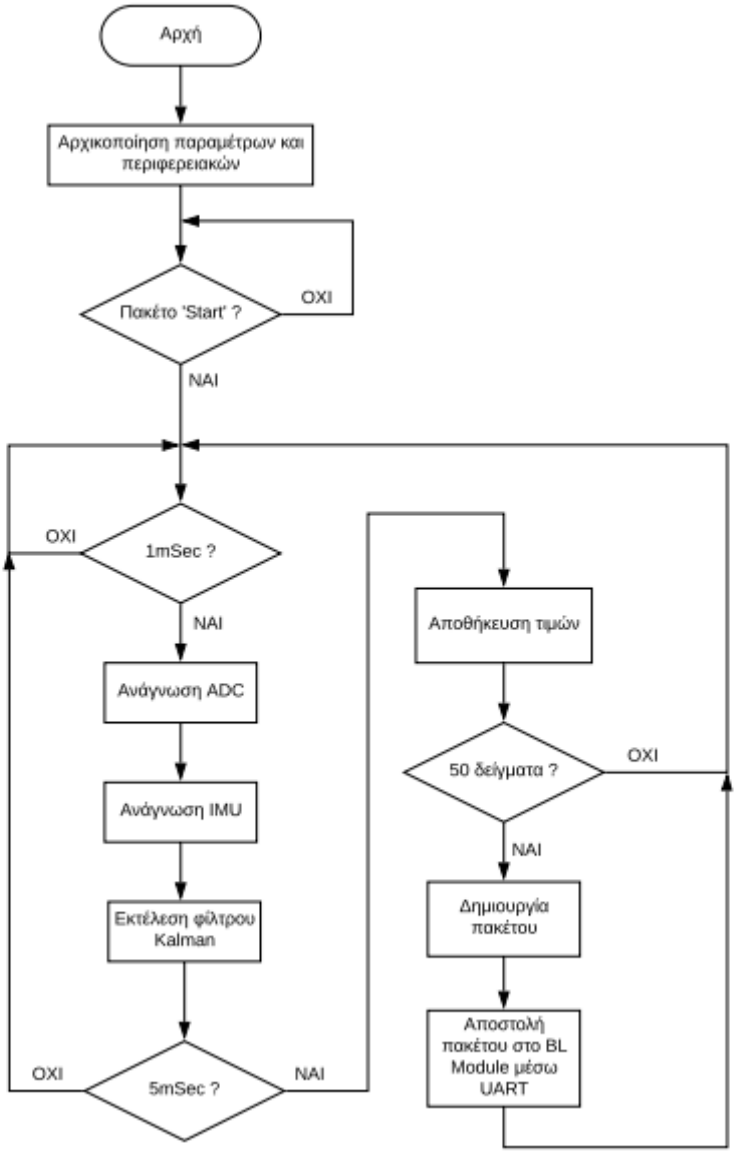
- ***Τάση τροφοδοσίας:*** 1.7V-3.6V
- ***Bytes ανά σελίδα***: 256 ή 264
- ***Χωρητικότητα***: 8MB (Mbytes)
- ***Πρωτόκολλο επικοινωνία*ς**: SPI
- ***Μέγιστη συχνότητα SCK:*** 85 MHz
- ***Τυπική κατανάλωση (συχνότητα 15MHz)***: 7mA
- ***Μέγιστος αριθμός Erase/Write Cycles ανά σελίδα:*** 100000

4. Κώδικας μικροελεγκτή/Firmware

Στο κεφάλαιο αυτό, θα γίνει περιγραφή του κώδικα του μικροελεγκτή (Firmware) όπως και της λογικής που ακολουθήθηκε. Επίσης, θα γίνει αναφορά στα εργαλεία που χρησιμοποιήθηκαν για τη συγγραφή του κώδικα και την υλοποίηση των απαραίτητων βιβλιοθηκών/αρχικοποιήσεων.

4.1 Διάγραμμα ροής main()

Στην εικόνα 24 φαίνεται το διάγραμμα ροής με τα βήματα που ακολουθούνται στο κυρίως πρόγραμμα.



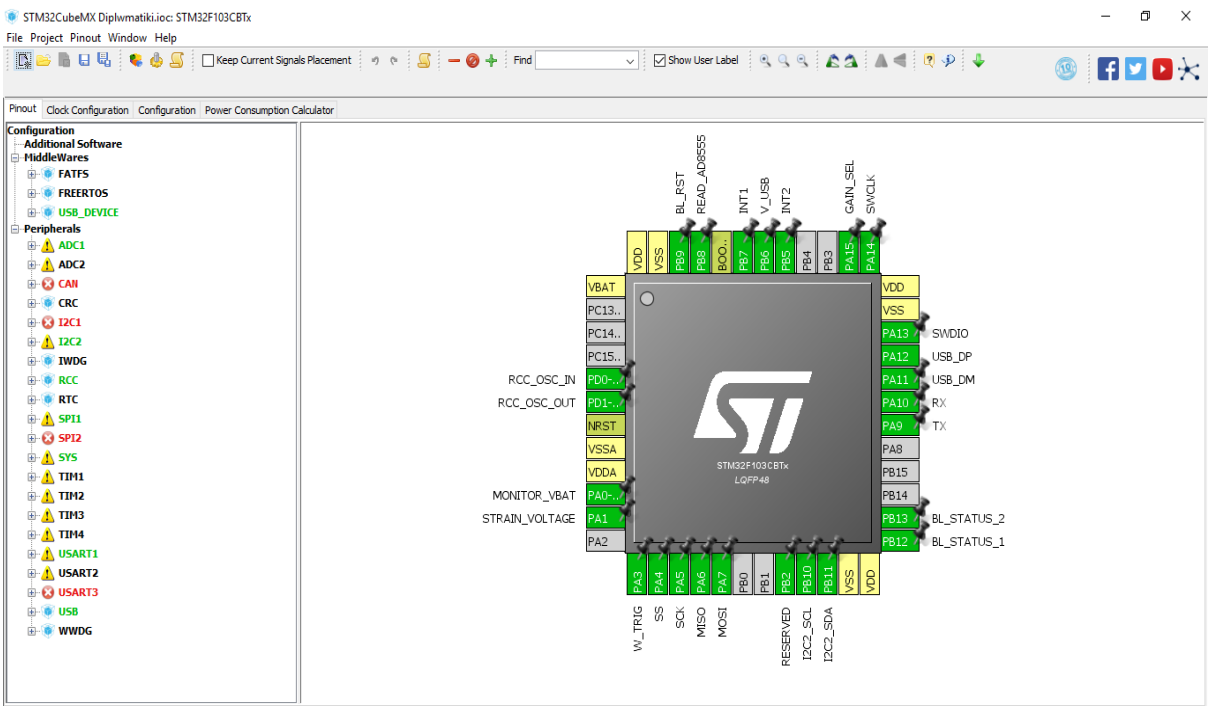
Εικόνα 24: Διάγραμμα ροής της main() συνάρτησης

4.2 EWARM της IAR

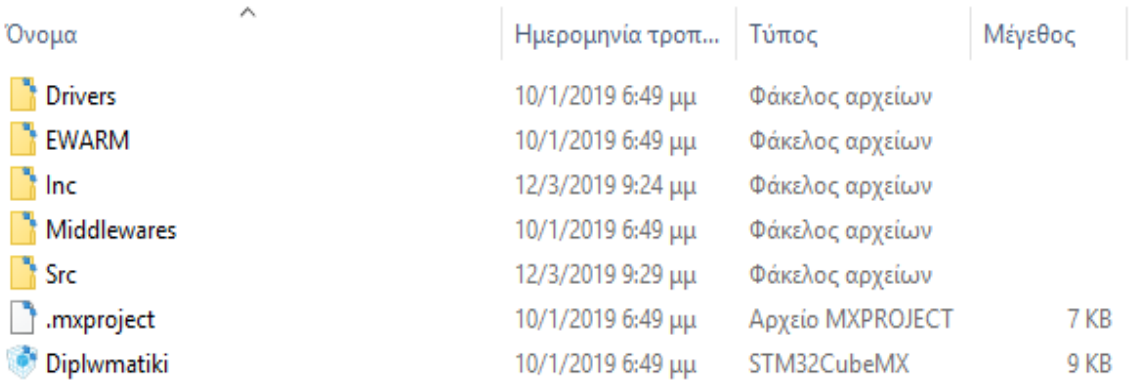
Ο compiler για προϊόντα ARM που διατίθεται από την IAR περιλαμβάνεται σε μία ολοκληρωμένη σουίτα εφαρμογών, με την οποία είναι δυνατή η βελτιστοποίηση της ανάπτυξης ενσωματωμένων εφαρμογών. Εκτός από την υποστήριξη στους παλιότερης τεχνολογίας επεξεργαστές ARM 7 και ARM 9 παρέχεται όπως ήταν φυσικό πλήρης υποστήριξη και για τον Cortex επεξεργαστή με τις προηγμένες λειτουργίες του. Το πακέτο αυτό προγραμμάτων ονομάζεται EWARM (Embedded Workbench for ARM) και μέσω αυτού είναι δυνατή η ανάπτυξη ενσωματωμένων εφαρμογών ARM. Με το EWARM δίνεται η δυνατότητα ανάπτυξης κώδικα με μέγεθος μέχρι και τέσσερα 4GBytes. Το κατέβασμα του κώδικα μετά την μετάφραση ώστε να ακολουθήσει η διαδικασία του Debug υποστηρίζεται από διάφορες συσκευές, όπως το IAR J-Trace το οποίο συνδέεται μέσω USB ή το ST-LINK της εταιρίας STMicroelectronics.

4.3 Εργαλείο STM32Cube

Το πρόγραμμα STM32Cube παρέχεται δωρεάν από την εταιρεία STMicroelectronics και υποστηρίζει σχεδόν όλους τους μικροελεγκτές της. Είναι ένα εργαλείο για την αρχικοποίηση των περιφερειακών και την προσθήκη όλων των απαραίτητων βιβλιοθηκών. Με τη χρήση του συγκεκριμένου εργαλείου μπορούμε να ορίσουμε/δηλώσουμε τα περιφερειακά με τις ρυθμίσεις που εμείς θέλουμε γρήγορα και αξιόπιστα. Στο τέλος της διαδικασίας το εργαλείο δημιουργεί ένα Project μόνο με τις απαραίτητες βιβλιοθήκες και αρχικοποιήσεις.



Εικόνα 25: Το περιβάλλον του STM32Cube και τα pin του STM32F103CB



Εικόνα 26: Οι φάκελοι από το STM32Cube που περιέχουν τα αρχεία .c και .h

4.4 Αρχικοποίηση περιφερειακών

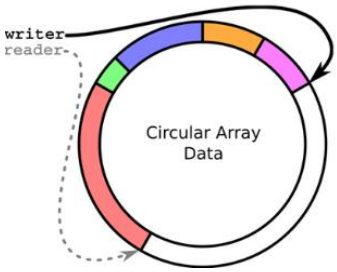
Κατά την εκκίνηση της συσκευής η πρώτη συνάρτηση που εκτελείται πριν οτιδήποτε άλλο είναι η **NOP_Delay(uint32_t mdelay)**. Η συγκεκριμένη συνάρτηση εισάγει ουσιαστικά μία χρονική καθυστέρηση πριν ξεκινήσει η εκτέλεση των υπόλοιπων συναρτήσεων ώστε να σταθεροποιηθεί η τροφοδοσία του μικροελεγκτή και των υπόλοιπων ολοκληρωμένων και να αποφευχθούν λάθη κατά τις αρχικοποιήσεις. Αξίζει να σημειωθεί πως επειδή τη συγκεκριμένη στιγμή δεν έχουν ακόμα αρχικοποιηθεί τα περιφερειακά (άρα και οι Timers), η καθυστέρηση γίνεται με διαδοχικές εκτελέσεις της εντολής **__NOP().** Ο χρόνος έχει ρυθμιστεί πειραματικά στα 200msec και αντιστοιχεί σε 50000 επαναλήψεις. Τυχόν αλλαγές (για παράδειγμα αλλαγή του επιπέδου βελτιστοποίησης) στις ρυθμίσεις του Compiler, μπορούν να επηρεάσουν το χρόνο καθυστέρησης ή ακόμα και να τον μηδενίσουν.

Στη συνέχεια εκτελούνται οι συναρτήσεις που δημιουργεί το εργαλείο STM32Cube και αφορούν τις αρχικοποιήσεις των GPIO, I2C, UART, DMA, USB CDC, SPI και Timers. Οι ρυθμίσεις για το εκάστοτε περιφερειακό θα αναλυθούν στις αντίστοιχες ενότητες.

4.5 Διαχείριση μνήμης Flash

Η επικοινωνία με την εξωτερική μνήμη Flash γίνεται μέσω SPI με ταχύτητα 2.25MBits/sec και υλοποιήθηκε αντίστοιχη βιβλιοθήκη σύμφωνα με τις απαιτήσεις μας. Να σημειωθεί πως ενώ η διαδικασία ανάγνωσης των δεδομένων από μία σελίδα της συγκεκριμένης Flash μπορεί να διαρκέσει μερικά μsec, η διαδικασία της εγγραφής μπορεί να διαρκέσει έως και 6msec, η περαιτέρω αύξηση της ταχύτητας επικοινωνίας δεν έχει νόημα μιας και ο χρόνος που αναφέρθηκε για την εγγραφή έχει να κάνει με εσωτερικές διαδικασίες της Flash και όχι με την ταχύτητα επικοινωνίας. Αυτός ο χρόνος ουσιαστικά σημαίνει πως εάν θέλουμε να κάνουμε εγγραφές στην Flash κατά την διάρκεια των μετρήσεων/αποστολών πακέτων, θα πρέπει η διαδικασία εγγραφής να γίνει με Non-Blocking (μηχανή πεπερασμένων καταστάσεων) λογική ώστε να μπορεί ‘παράλληλα’ να εκτελεί ή να εξυπηρετεί και άλλες συναρτήσεις/περιφερειακά. Η συνάρτηση **Flash_Write_Using_States()** εκτελείται μονίμως στη **main()** συνάρτηση και με βάση την τρέχουσα κατάσταση της μηχανής και άλλες παραμέτρους, αλλάζει κατάσταση.

Όπως αναφέραμε στα χαρακτηριστικά της μνήμης AT45DB641E, ο μέγιστος αριθμός εγγραφών ανά σελίδα είναι 100000. Αυτό ουσιαστικά σημαίνει πως εάν θέλουμε να αποθηκεύουμε τα πακέτα που αποστέλλουμε, θα πρέπει να προνοήσουμε ώστε να μην ξεπεραστεί αυτός ο αριθμός ανά σελίδα. Για να ξεπεράσουμε αυτό το πρόβλημα, η λύση ήταν να υλοποιηθεί μία κυκλική ουρά στην Flash ώστε κάθε νέα εγγραφή πακέτου να γίνεται σε επόμενες σελίδες.



Εικόνα 27: Κυκλική ουρά με χρήση δεικτών ανάγνωσης/εγγραφής

Φυσικά και με αυτή τη λογική κάποια στιγμή θα ξεπεραστεί αυτός ο μέγιστος αριθμός ανά σελίδα αλλά για να γίνει θα πρέπει η συσκευή να είναι σε κατάσταση αποστολής πακέτων για ένα τεράστιο χρονικό διάστημα. Η αποθήκευση του πακέτου γίνεται μόνο όταν η Android εφαρμογή δεν απαντήσει με acknowledge εντός του χρονικού ορίου που ορίσαμε. Συγκεκριμένα:

- Η μνήμη έχει 32768 σελίδες των 256bytes αλλά δεσμεύουμε για την ουρά τις 32658, αφήνοντας τις υπόλοιπες 120 σελίδες για την αποθήκευση των δεικτών της ουράς, των παραμέτρων του συστήματος και μερικές σελίδες για μελλοντική χρήση.
- Η δομή που συλλέγουμε τα δεδομένα ανά πακέτο έχει μέγεθος 708Bytes, δηλαδή για την εγγραφή ενός πακέτου θέλουμε 3 σελίδες. Προσθέσαμε άλλη μία σελίδα για μελλοντική χρήση σε περίπτωση που μεγαλώσουμε τη δομή των δεδομένων. Συνολικά πλέον χρειαζόμαστε 4 σελίδες ανά πακέτο.
- Με κάθε εγγραφή πακέτου στη Flash θα πρέπει να ανανεώνουμε και τους δείκτες ανάγνωσης και εγγραφής. Για το λόγο αυτόν δεσμεύουμε 100 σελίδες και με κάθε εγγραφή των δεικτών, αυξάνουμε τον μετρητή εγγραφών ανά σελίδα. Αυτό πρακτικά σημαίνει πως όταν μία σελίδα που κρατάει τους δείκτες φθάσει τις 90000 εγγραφές, τότε μεταβαίνουμε σε επόμενη σελίδα.
- Κατά την εκκίνηση του προγράμματος γίνεται έλεγχος των σελίδων ώστε να βρεθεί η σελίδα που πληρεί τα κριτήρια που θέσαμε. Τα κριτήρια για να είναι διαθέσιμη μία σελίδα δεικτών είναι ο αριθμός εγγραφών που πρέπει να είναι κάτω από 90000 φορές και το CRC (Cyclic Redundancy Check) που ανανεώνουμε σε κάθε εγγραφή
- Με βάση όλα τα παραπάνω και με δεδομένο ότι αποστέλλουμε πακέτα κάθε 250msec, η συσκευή μπορεί να αποθηκεύει πακέτα μέχρι να φθάσει τις μέγιστες εγγραφές για:

(100 σελίδες * 90000) / 4 / 3600 = 625 ώρες

Τα 10Bytes στην αρχή κάθε σελίδας δεικτών, φαίνονται στον πίνακα 4.

Θέσεις στη σελίδα	**Μέγεθος (Bytes**	**Λειτουργία**
[0:3]	4	Μετρητής εγγραφών
[4:5]	2	Δείκτης ανάγνωσης
[6:7]	2	Δείκτης εγγραφής
[8:9]	2	CRC

Πίνακας 4: Οι μεταβλητές που αποθηκεύονται στην αρχή κάθε σελίδες δεικτών

Οι παράμετροι του συστήματος είναι ουσιαστικά 2, η πρώτη είναι το όνομα του Bluetooth module που θα εμφανίζεται στην Android εφαρμογή και η δεύτερη είναι ο τετραψήφιος κωδικός πρόσβασης/Pairing. Κατά την εκκίνηση της συσκευής ελέγχεται ένα Flag που βρίσκεται στη θέση 0 της σελίδας παραμέτρων. Εάν το Flag έχει τιμή '1', τότε στέλνουμε τις παραμέτρους (όνομα και κωδικό) στο Bluetooth module. Ο λόγος που χρησιμοποιούμε το επιπλέον Flag είναι για να στέλνουμε τις παραμέτρους στο Bluetooth module μόνο όταν έχει αλλάξει κάποια παράμετρος, που είναι και λογικό. Επιπλέον και το Bluetooth module έχει κάποιο είδος Flash μνήμης όπου κρατάει τις παραμέτρους που στέλνουμε, ο αριθμός εγγραφών είναι πεπερασμένος και επειδή ο κατασκευαστής δεν αναφέρει κάποια πληροφορία, καλό είναι να προνοήσουμε εμείς. Το όνομα και ο κωδικός του Bluetooth, μπορούν να αλλάξουν μόνο με χρήση USB καλωδίου και της Desktop εφαρμογής που θα περιγράψουμε σε επόμενο κεφάλαιο.

4.6 **Χρονισμός**

Ο SysTick (System Timer) είναι ένας χρονιστής που υπάρχει σε όλους τους ARM μικροελεγκτές και ουσιαστικά παράγει συνεχόμενες διακοπές (Interrupts) με περίοδο που δηλώνεται από το χρήστη. Κάθε φορά που ο χρονιστής φθάσει στην τιμή που ορίσαμε, παράγει μία διακοπή και η εκτέλεση του κώδικα μεταβαίνει στο αντίστοιχο Interrupt Vector. Ο χρόνος που αφιερώνουμε μέσα στο Vector πρέπει να είναι όσο μικρότερος γίνεται, για το λόγο αυτό αποφεύγουμε χρονοβόρες εντολές/συναρτήσεις/πράξεις. Στο κώδικα μας, μέσα στο Vector απλά αυξάνουμε κάποιους μετρητές.

Η περίοδος που σχεδόν πάντα είναι προ-ρυθμισμένος ο χρονιστής SysTick είναι 1msec. Εμείς επειδή χρειαζόμασταν μικρότερο βήμα χρόνου, ρυθμίσαμε την περίοδο στα 50μSec. Έτσι το πρόγραμμα είναι πιο ευέλικτο για μελλοντικές τροποποιήσεις ή προσθήκες. Προσοχή χρειάζεται όταν αλλάζουμε την περίοδο του SysTick διότι πολλές από τις βιβλιοθήκες των περιφερειακών που παράγει το STM32Cube χρησιμοποιούν ως χρονιστή τον SysTick. Για παράδειγμα, η πιο κλασική συνάρτηση είναι η **HAL_Delay(uint32_t Delay)** που δημιουργεί μία καθυστέρηση (Delay) με παράμετρο μία τιμή σε msec. Με την αλλαγή της περιόδου από 1msec σε 50μsec, αυτή η συνάρτηση θα υλοποιούσε μια καθυστέρηση κατά 20 φορές μικρότερη από την επιθυμητή. Εμείς έπρεπε να διορθώσουμε τη συνάρτηση για να μετράει με σωστή βάση χρόνου:

```
__weak void HAL_Delay(uint32_t Delay)
{
        uint32_t tickstart = HAL_GetTick();
        uint32_t wait = Delay*20; // Διόρθωση λόγω της αλλαγής περιόδου SysTick

        /* Add a freq to guarantee minimum wait */
        if (wait < HAL_MAX_DELAY)
        {
          wait += (uint32_t)(uwTickFreq);
        }

        while ((HAL_GetTick() - tickstart) < wait)
        {
        }
}
```

Θα μπορούσαμε επίσης να μην αλλάξουμε την περίοδο του SysTick και να ρυθμίσουμε κάποιον άλλο Timer να παράγει διακοπές ανά 50μsec. Η λύση αυτή απορρίφθηκε γιατί με μελλοντική μετάβαση σε μεγαλύτερη/γρηγορότερη οικογένεια μικροελεγκτή θα έπρεπε να γίνουν επιπλέον αλλαγές.

4.7 IMU

Για την επικοινωνία με το LSM6DS3, χρησιμοποιούμε I2C με συχνότητα clock στα 400KHz. Υλοποιήθηκε αντίστοιχη βιβλιοθήκη ώστε να μπορούμε να ρυθμίζουμε ευκολότερα τις παραμέτρους που θέλουμε. Η αρχικοποίηση του IMU γίνεται μέσω της συνάρτησης **Accelerometer_Gyroscope()** και όπως φαίνεται και παρακάτω είναι ξεκάθαρο ποιες ρυθμίσεις επιλέξαμε το επιταχυνσιόμετρο και το γυροσκόπιο:

```
while(Accelerometer_Gyroscope(ACC_1666Hz,
                ACC_RANGE_16G, ACC_FILTER_400Hz,
                GYR_1666Hz, GYR_RANGE_2000dps,
                MAX_ACC_GYR_INIT_ERRORS) != OK)
{
        HAL_Delay(1000);
}
```

- Η πρώτη παράμετρος ορίζει τη συχνότητα δειγματοληψίας για το επιταχυνσιόμετρο και έχει επιλεγεί να είναι 1666Hz
- Η δεύτερη ορίζει το εύρος της επιτάχυνσης. Έτσι οι τιμές που θα διαβάζουμε θα αφορούν ένα εύρος από -16G έως+16G
- Η τρίτη ορίζει τη συχνότητα για το Anti-Aliasing φίλτρο στα 400Hz
- Η τέταρτης ορίζει τη συχνότητα δειγματοληψίας για το γυροσκόπιο και έχει επιλεγεί να είναι 1666Hz
- Η πέμπτη ορίζει το εύρος της ταχύτητας περιστροφής. Έτσι οι τιμές που θα διαβάζουμε θα αφορούν ένα εύρος από -2000dps έως 2000dps (degress per second)
- Η τελευταία ορίζει πόσες θα είναι οι επαναλήψεις σε πιθανό σφάλμα μέχρι η συνάρτηση να επιστρέψει λάθος τιμή. Πρακτικά δεν γίνεται ποτέ, ο μόνος λόγος για να υπάρξει σφάλμα είναι να **hardware** πρόβλημα στη τροφοδοσία ή στα σήματα από το IMU προς τον μικροελεγκτή

Από τη στιγμή που ο χρήστης θα ξεκινήσει ένα νέο κύκλο μετρήσεων (πατώντας το πλήκτρο Start στην εφαρμογή), η συνάρτηση **Get_IMU_Value()** καλείται ακριβώς κάθε 1msec διαβάζοντας τους καταχωρητές του IMU. Από τη στιγμή που διαβάσουμε χωρίς σφάλμα τους καταχωρητές θερμοκρασίας, επιτάχυνσης για κάθε άξονα και ταχύτητας περιστροφής για κάθε άξονα, μεταφέρουμε τις τιμές στη δομή που αποθηκεύονται οι μετρήσεις.

Για την ανάγνωση των καταχωρητών χρησιμοποιούμε την συνάρτηση **HAL_I2C_Mem_Read()** που τα βασικά ορίσματα της είναι η διεύθυνση της I2C συσκευής, η διεύθυνση του πρώτου καταχωρητή που θέλουμε να διαβάσουμε, τον πίνακα που στον οποίο θέλουμε να μας επιστρέψει τις τιμές και τα συνολικά bytes που θέλουμε να διαβάσουμε. Όπως φαίνεται και στο αρχείο **accelerometer_gyroscope.h**, οι καταχωρητές για τη θερμοκρασία, της επιτάχυνσης ανά άξονα και της ταχύτητας περιστροφής ανά άξονα είναι σε διαδοχικές θέσεις:

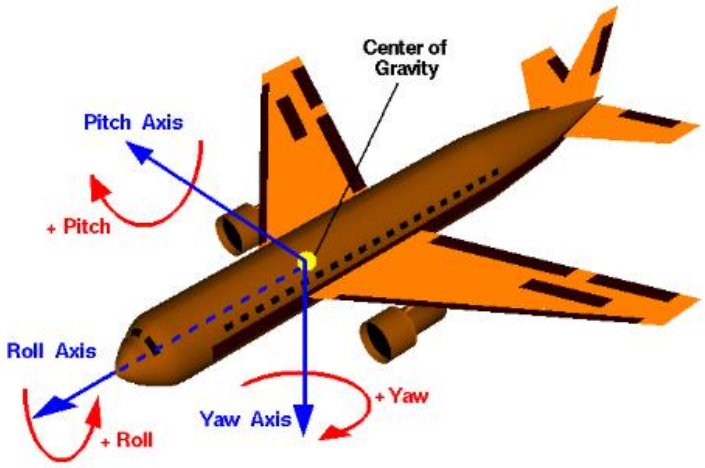
```
#define LSM6DS3_ACC_GYRO_OUT_TEMP_L         0x20
#define LSM6DS3_ACC_GYRO_OUT_TEMP_H         0x21
#define LSM6DS3_ACC_GYRO_OUTX_L_G           0x22
#define LSM6DS3_ACC_GYRO_OUTX_H_G           0x23
#define LSM6DS3_ACC_GYRO_OUTY_L_G           0x24
#define LSM6DS3_ACC_GYRO_OUTY_H_G           0x25
#define LSM6DS3_ACC_GYRO_OUTZ_L_G           0x26
#define LSM6DS3_ACC_GYRO_OUTZ_H_G           0x27
#define LSM6DS3_ACC_GYRO_OUTX_L_XL          0x28
#define LSM6DS3_ACC_GYRO_OUTX_H_XL          0x29
#define LSM6DS3_ACC_GYRO_OUTY_L_XL          0x2A
#define LSM6DS3_ACC_GYRO_OUTY_H_XL          0x2B
#define LSM6DS3_ACC_GYRO_OUTZ_L_XL          0x2C
#define LSM6DS3_ACC_GYRO_OUTZ_H_XL          0x2D
```

Αυτό σημαίνει ότι δεν χρειάζεται να ξανακαλέσουμε τη συνάρτηση **HAL_I2C_Mem_Read(),** απλά δηλώνουμε τη πρώτη διεύθυνση του καταχωρητή που θέλουμε να διαβάσουμε και το σωστό μέγεθος bytes. Παρακάτω φαίνεται ένα μέρος της συνάρτησης **Get_IMU_Value()**:

```
if(HAL_I2C_Mem_Read(&hi2c2, LSM6DS3_I2C_ADDRESS, LSM6DS3_ACC_GYRO_OUT_TEMP_L,
I2C_MEMADD_SIZE_8BIT, Read_Buffer_I2C, 14, 5000) == HAL_OK)
 {
    Measure.Temperature                 =  (int16_t) (Read_Buffer_I2C[1]<<8 | Read_Buffer_I2C[0]);
    Measure.GyrInfo[Measure.Index].X =  (int16_t) (Read_Buffer_I2C[3]<<8 | Read_Buffer_I2C[2]);
    Measure.GyrInfo[Measure.Index].Y =  (int16_t) (Read_Buffer_I2C[5]<<8 | Read_Buffer_I2C[4]);
    Measure.GyrInfo[Measure.Index].Z =  (int16_t) (Read_Buffer_I2C[7]<<8 | Read_Buffer_I2C[6]);
    Measure.GyrInfo[Measure.Index].Z *=  -1; // Το υλικό βρίσκεται στη Bottom πλευρά
    Measure.AccInfo[Measure.Index].X  =  (int16_t) (Read_Buffer_I2C[9]<<8 | Read_Buffer_I2C[8]);
    Measure.AccInfo[Measure.Index].Y  =  (int16_t) (Read_Buffer_I2C[11]<<8 | Read_Buffer_I2C[10]);
    Measure.AccInfo[Measure.Index].Z  =  (int16_t) (Read_Buffer_I2C[13]<<8 | Read_Buffer_I2C[12]);
    Measure.AccInfo[Measure.Index].Z  *=  -1;  // Το υλικό βρίσκεται στη Bottom πλευρά
 }
```

4.8 Γωνίες προσανατολισμού και φίλτρο Kalman

Για τον υπολογισμό των γωνιών προσανατολισμού (Pitch,Roll) θα πρέπει να χρησιμοποιηθούν τα δεδομένα από το επιταχυνσιόμετρο και από το γυροσκόπιο σε συνδυασμό με κάποιο φίλτρο. **Η γωνία Yaw χρειάζεται και μαγνητόμετρο για να μπορεί να υπολογισθεί σωστά!**



Εικόνα 28: Αναπαράσταση των γωνιών Pitch,Roll και Yaw

Γενικά τα προβλήματα/σφάλματα που θα συναντήσουμε στα δεδομένα που επιστρέφουν οι δύο αισθητήρες είναι:

- Τα δεδομένα από το επιταχυνσιόμετρο είναι θορυβώδεις. Ακόμη και με ένα φίλτρο χαμηλής διέλευσης θα εξακολουθούν να υπάρχουν αιχμές και θόρυβος
- Το κύριο μειονέκτημα των γυροσκοπίων, είναι το φαινόμενο της ολίσθησης ή drift. Αυτό ουσιαστικά σημαίνει πως μετά από κάποιο χρονικό διάστημα, η απόκλιση των μετρήσεων θα μεταβάλλει το αρχικά θεωρημένο σημείο μηδέν με αποτέλεσμα να μην υπάρχει το απαραίτητο set-point για τη διατήρηση της ευστάθειας. Επίσης ένα ακόμη μειονέκτημα που παρουσιάζουν τα γυροσκόπια είναι η ευαισθησία τους στις δονήσεις

Ένα από τα καλύτερα φίλτρα για τη περίπτωση μας, είναι το περίφημο Kalman Filter. Χρησιμοποιείται στα αεροσκάφη, πυραύλους και σε γεωστατικούς δορυφόρους. Θεωρείται ένα από τα μεγαλύτερα ευρήματα του περασμένου αιώνα, και δικαίως. Είναι σε θέση να υπολογίσει το σφάλμα κάθε μέτρησης από τις προηγούμενες μετρήσεις, και αφαιρώντας το να δώσει την πραγματική τιμή της γωνίας. Με λίγα λόγια είναι ένας αλγόριθμος που «μαθαίνει» σε κάθε επανάληψη. Βέβαια έχει δύο μειονεκτήματα, απαιτεί μεγάλη υπολογιστική ισχύ για την επεξεργασία των δεδομένων και είναι αρκετά πολύπλοκο στην κατανόηση του [11].

4.8.1 Μαθηματική ανάλυση

Για την καλύτερη κατανόηση του φίλτρου αρχικά παρουσιάζονται οι βασικές έννοιες και οι βασικές εξισώσεις που είναι απαραίτητες για την υλοποίηση του [12]. Η προηγούμενη εκτιμώμενη κατάσταση βασίζεται στην προηγούμενη κατάσταση και τις εκτιμήσεις των καταστάσεων πριν από αυτήν .

$$\hat{\boldsymbol{x}}_{k-1|k-1}$$

A priori είναι η εκτίμηση του πίνακα καταστάσεων στην παρούσα κατάσταση βασισμένη στις προηγούμενες καταστάσεις του συστήματος και στις εκτιμήσεις πριν από αυτήν:

$$\hat{\boldsymbol{x}}_{k|k-1}$$

A posteriori είναι η εκτίμηση της κατάστασης μίας δεδομένης στιγμής k με βάση τις παρατηρήσεις μέχρι εκείνη την στιγμή συμπεριλαμβανομένης και της στιγμής k:

$$\hat{\boldsymbol{x}}_{k|k}$$

Το πρόβλημα που δημιουργείται είναι ότι οι καταστάσεις του συστήματος πρέπει να μελετηθούν μέσω παρατηρήσεων z_k. Το σύστημα αυτό ονομάζεται Hidden Markov Model. Αυτό σημαίνει ότι μία κατάσταση θα βασίζεται στην κατάσταση στον χρόνο k και σε όλες τις προηγούμενες καταστάσεις. Επιπλέον αυτό έχει σαν αποτέλεσμα η εκτίμηση της κατάστασης να μην είναι έμπιστη πριν το Kalman σταθεροποιηθεί. Το $\hat{x}$ αναφέρεται στην εκτίμηση μίας κατάστασης ενώ το x στην κανονική κατάσταση. Η κατάσταση του συστήματος μία δεδομένη στιγμή k δίνεται από την σχέση :

$$\boldsymbol{x}_k = \boldsymbol{F} x_{k-1} + \boldsymbol{B} u_k + w_k$$

Όπου x_k είναι ο πίνακας καταστάσεων ο οποίος είναι :

$$\boldsymbol{x}_k = \begin{bmatrix} \theta \\ \dot{\theta}_b \end{bmatrix}_k$$

Η έξοδος του φίλτρου είναι η γωνία θ και επίσης η θ_b η οποία είναι η ποσότητα κατά την οποία το γυροσκόπιο δημιουργεί σφάλμα. Έτσι η πραγματική τιμή δίνεται αν αφαιρεθεί από την μέτρηση του γυροσκόπιου, το σφάλμα.

Ο πίνακας F είναι το μοντέλο μετάβασης το οποίο εφαρμόζεται στην κατάσταση x_{k-1}.

$$\boldsymbol{F} = \begin{bmatrix} 1 & -\Delta t \\ 0 & 1 \end{bmatrix}$$

Το u_k είναι η είσοδος ελέγχου που στην παρούσα περίπτωση είναι η μέτρηση του γυροσκοπίου σε μοίρες ανά δευτερόλεπτο. Αυτή η μέτρηση γράφεται ως θ_k. Έτσι η εξίσωση μπορεί να γραφτεί με αυτόν τον τρόπο:

$$\boldsymbol{x}_k = \boldsymbol{F}x_{k-1} + \boldsymbol{B}\dot{\theta}_k + w_k$$

Ο πίνακας B είναι το μοντέλο της εισόδου ελέγχου. Ορίζεται ως:

$$\boldsymbol{B} = \begin{bmatrix} \Delta t \\ 0 \end{bmatrix}$$

Οι τιμές είναι απόλυτα φυσιολογικές διότι πολλαπλασιάζοντας το Δt με το $\dot{\theta}$ παίρνουμε την γωνιακή θέση θ και επειδή το σφάλμα θ_b δεν υπολογίζεται απευθείας από το $\dot{\theta}$ το δεύτερο στοιχείο του πίνακα είναι μηδέν.

Ο θόρυβος w_k είναι γκαουσιανής κατανομής και έχει την μορφή:

$$\boldsymbol{w}_k \sim N\left(0, \boldsymbol{Q}_k\right)$$

Ο πίνακας Q_k περιέχει τις τιμές σφαλμάτων του επιταχυνσιόμετρου και του γυροσκοπίου και είναι της μορφής:

$$\boldsymbol{Q}_k = \begin{bmatrix} Q_\theta & 0 \\ 0 & Q_{\dot{\theta}_b} \end{bmatrix} \Delta t$$

Όπως φαίνεται ο πίνακας εξαρτάται από τον χρόνο k και η τιμή του επιταχυνσιόμετρου και το σφάλμα του γυροσκοπίου πολλαπλασιάζονται με το Δt. Αυτό έχει νόημα διότι όσο περνά ο χρόνος ο θόρυβος αυξάνεται.

Αυτές οι τιμές πρέπει να είναι γνωστές για να δουλέψει το φίλτρο και αναλόγως την τιμή τους το φίλτρο μπορεί να δουλεύει σωστά ή όχι. Για παράδειγμα αν οι εκτιμήσεις είναι αργές τότε εμπιστευόμαστε παραπάνω από όσο πρέπει το επιταχυνσιόμετρο και πρέπει να μειωθεί η τιμή του Q_θ για καλυτέρα αποτελέσματα.

Η μέτρηση του zk της κατάστασης xk είναι:

$$\boldsymbol{z}_k = \boldsymbol{H}x_k + v_k$$

Η τιμή της είναι ίση με την τωρινή κατάσταση πολλαπλασιασμένη με έναν πίνακα H συν την μέτρηση του θορύβου. Ο πίνακας H είναι το μοντέλο παρατήρησης. Αφού η μέτρηση προέρχεται μόνο από το επιταχυνσιόμετρο ο πίνακας είναι:

$$\boldsymbol{H} = \begin{bmatrix} 1 & 0 \end{bmatrix}$$

Ο θόρυβος uk της μέτρησης είναι γκαουσιανής κατανομής και γράφεται:

$$\boldsymbol{v}_k \sim N\left(0, \boldsymbol{R}\right)$$

Ο R δεν είναι πίνακας και είναι μόνο μία μεταβλητή και ορίζεται:

$$\boldsymbol{R} = E\left[v_k \quad {v_k}^T\right] = var(v_k)$$

Υποθέτουμε ότι ο θόρυβος είναι ίδιος και δεν μεταβάλλεται με τον χρόνο, άρα:

$$var(v_k) = var(v)$$

Αν η μεταβλητή είναι πολύ υψηλή το φίλτρο θα λειτουργεί με πολύ αργό ρυθμό αφού θα εμπιστεύεται νέες τιμές λιγότερο. Αντιθέτως αν είναι πολύ μικρή, οι μετρήσεις μπορεί να περιέχουν πολύ θόρυβο αφού θα εμπιστευόμαστε το επιταχυνσιόμετρο σε μεγάλο βαθμό.

Έτσι χρειαζόμαστε τις τιμές των Q_θ,$Q_{\dot{\theta}_b}$ και $var(v)$.

Παρακάτω φαίνονται οι εξισώσεις για τον υπολογισμό της κατάστασης του συστήματος $\hat{x}_k$ σε δύο φάσεις, την πρόβλεψη και το update.

Πρόβλεψη

Με τις δύο πρώτες εξισώσεις γίνεται προσπάθεια πρόβλεψης της τωρινής κατάστασης και του σφάλματος την στιγμή k. Αρχικά το φίλτρο υπολογίζει την τωρινή κατάσταση βασισμένο στις προηγούμενες καταστάσεις και στην μέτρηση του γυροσκοπίου.

$$\hat{\boldsymbol{x}}_{k|k-1} = \boldsymbol{F}\hat{x}_{k-1|k-1} + \boldsymbol{B}\dot{\theta}_k$$

Για αυτό το λόγο ονομάζεται είσοδος ελέγχου διότι χρησιμοποιείτε ως μία επιπλέον είσοδος για να εκτιμήσει την παρούσα κατάσταση που ονομάζεται a priori state $\hat{x}_{k|k-1}$ όπως έχει αναφερθεί. Στη συνέχεια υπολογίζεται το a priori σφάλμα $P_{k|k-1}$ βασισμένο στο προηγούμενο σφάλμα $P_{k-1|k-1}$, το οποίο ορίζεται ως:

$$\boldsymbol{P}_{k|k-1} = \boldsymbol{F}\boldsymbol{P}_{k-1|k-1}\boldsymbol{F}^T + \boldsymbol{Q}_k$$

Αυτός ο πίνακας χρησιμοποιείται για να γίνει αντιληπτό πόσο πολύ αξιόπιστες είναι οι τωρινές τιμές της εκτιμώμενης κατάστασης. Όσο μικρότερος είναι τόσο περισσότερο αξιόπιστη είναι

η τωρινή εκτιμώμενη κατάσταση. Είναι φανερό ότι το σφάλμα αυξάνεται από το προηγούμενο update αφού το σφάλμα πολλαπλασιάζεται με τους πίνακες F, F^T και προσθέτουμε τον θόρυβο Q_k της στιγμής k.

Ο πίνακας P στην συγκεκριμένη περίπτωση είναι ένας 2×2 πίνακας:

$$\boldsymbol{P} = \begin{bmatrix} P_{00} & P_{01} \\ P_{10} & P_{11} \end{bmatrix}$$

Update

Για το update αρχικά γίνεται ο υπολογισμός της διαφοράς ανάμεσα στην μέτρηση z_k και στην a priori κατάσταση $x_{k|k-1}$.

$$\tilde{\boldsymbol{y}}_k = \boldsymbol{z}_k - \boldsymbol{H}\hat{x}_{k|k-1}$$

Στη συνέχεια γίνεται έλεγχος για το κατά πόσο αξιόπιστες είναι οι μετρήσεις με βάση το a priori σφάλμα και τον πίνακα R. Όσο μεγαλύτερη είναι η τιμή της μέτρησης του θορύβου τόσο μεγαλύτερη είναι και η τιμή του S. Αυτό σημαίνει πως οι εισερχόμενες μετρήσεις δεν είναι αξιόπιστες.

Στη συνέχεια υπολογίζουμε την παρακάτω ποσότητα:

$$\boldsymbol{S}_k = \boldsymbol{H}\boldsymbol{P}_{k|k-1}\boldsymbol{H}^T + \boldsymbol{R}$$

Αυτή προσπαθεί να προβλέψει κατά πόσο μπορούμε να εμπιστευτούμε την μέτρηση η οποία είναι βασισμένη στον a priori πίνακα σφάλματος $P_{k|k-1}$ και στον πίνακα R. Όσο μεγαλύτερη είναι η μέτρηση του θορύβου τόσο μεγάλη είναι και η τιμή S και σημαίνει ότι δεν μπορούμε να εμπιστευτούμε την εισερχόμενη μέτρηση.

Στη συνέχεια γίνεται ο υπολογισμός του κέρδους του φίλτρου.

$$\boldsymbol{K}_k = \boldsymbol{P}_{k|k-1}\boldsymbol{H}^T\boldsymbol{S}_k^{-1}$$

Αν δεν γνωρίζουμε τις αρχικές τιμές του συστήματος τότε ο πίνακας P είναι:

$$\boldsymbol{P} = \begin{bmatrix} L & 0 \\ 0 & L \end{bmatrix}$$

Οπου *L* είναι ένας μεγάλος αριθμός.

Στην συγκεκριμένη εφαρμογή επειδή υποθέτουμε ότι το αρχικό σφάλμα του γυροσκοπίου είναι 0 και επειδή βρίσκουμε την αρχική γωνιακή θέση έχει τεθεί:

$$\boldsymbol{P} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

Σε αυτήν την περίπτωση το κέρδος του φίλτρου είναι ένας 2×1 πίνακας:

$$\boldsymbol{K} = \begin{bmatrix} K_0 \\ K_1 \end{bmatrix}$$

Μετά από τα παραπάνω μπορεί να γίνει το update της a posteriori εκτίμησης της τωρινής κατάστασης:

$$\hat{\boldsymbol{x}}_{k|k} = \hat{\boldsymbol{x}}_{k|k-1} + \boldsymbol{K}_k\, \tilde{\boldsymbol{y}}_k$$

Αυτό γίνεται προσθέτοντας της a priori κατάστασης $\hat{x}_{k|k-1}$ με το κέρδος του φίλτρου πολλαπλασιασμένο κατά $\tilde{y}_k$ το οποίο είναι η διαφορά της μέτρησης z_k και της εκτιμώμενης a priori κατάστασης $\hat{x}_{k|k-1}$. Άρα μπορεί να είναι είτε θετικό ή αρνητικό.

Τέλος ανανεώνεται ο a posteriori πίνακας σφάλματος:

$$\boldsymbol{P}_{k|k} = (\boldsymbol{I} - \boldsymbol{K}_k \boldsymbol{H})\boldsymbol{P}_{k|k-1}$$

Όπου *I* είναι ο πίνακας ταυτότητας και ορίζεται ως:

$$\boldsymbol{I} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Αυτό που κάνει το φίλτρο δηλαδή είναι ότι διορθώνει τον πίνακα σφαλμάτων με βάση τις διορθώσεις των εκτιμήσεων.

4.8.2 Μετατροπή δεδομένων σε γωνίες και χρονισμός

Ο κώδικας πρέπει να μετατρέψει τις τιμές από το επιταχυνσιόμετρο και το γυροσκόπιο σε γωνίες. Αρχικά πρέπει να διαβάσουμε τις τιμές που έχουν προηγουμένως διαβάσει από το IMU και είναι πλέον αποθηκευμένες στη δομή μας:

```
accX = (int16_t) Measure.AccInfo[Measure.Index].X;
accY = (int16_t) Measure.AccInfo[Measure.Index].Y;
accZ = (int16_t) Measure.AccInfo[Measure.Index].Z;
gyroX = (int16_t) Measure.GyrInfo[Measure.Index].X;
gyroY = (int16_t) Measure.GyrInfo[Measure.Index].Y;
gyroZ = (int16_t) Measure.GyrInfo[Measure.Index].Z;
```

Έτσι μπορούμε να υπολογίσουμε την γωνία του επιταχυνσιόμετρου:

```
double roll  = atan2(accY, accZ) * RAD_TO_DEG;
double pitch = atan(-accX / sqrt(accY * accY + accZ * accZ)) * RAD_TO_DEG;
```

Τα δεδομένα από το γυροσκόπιο θα πρέπει να μετατραπούν σε μοίρες/sec. Για να γίνει αυτό πρέπει να διαιρεθούν με την ευαισθησία του γυροσκοπίου. Το datasheet εμπεριέχει τον παρακάτω πινάκα:

G_So	Angular rate sensitivity	FS = ±125		4.375		mdps/LSB
		FS = ±250		8.75		
		FS = ±500		17.50		
		FS = ±1000		35		
		FS = ±2000		70		

Πίνακας 4: Η ευαισθησία του γυροσκοπίου σύμφωνα με το κατασκευαστή ανά κλίμακα

Οπότε για τη κλίμακα των +-250dps θα πρέπει να πολλαπλασιάζουμε με τη τιμή 0.00875:

```
double gyroXrate = gyroX * 0.007f; // Convert to deg/s
double gyroYrate = gyroY * 0.007f; // Convert to deg/s
```

Για τον υπολογισμό των γωνιών και την εκτέλεση του αλγόριθμου θα πρέπει να είναι γνωστό το χρονικό διάστημα (delta time) από την προηγούμενη μετατροπή:

```
KalmanX.angle += dt * KalmanX.rate;
```

Για τους παραπάνω λόγους, η μέτρηση του χρόνο γίνεται με έναν μετρητή που αυξάνει μέσω της callback συνάρτησης του System Timer. Όπως είπαμε και παραπάνω, η περίοδος του System Timer είναι 50uSec, οπότε για την μετατροπή σε δευτερόλεπτα:

```
time_now = Kalman_Timer;
Kalman_Timer = 0;
time_now = (double) (time_now / 20000.0f);
```

4.8.3 Μετατροπή εξισώσεων Kalman σε κώδικα

Στη συνέχεια υλοποιείται το φίλτρο το οποίο χρησιμοποιήθηκε για την αναγνώριση. Γίνεται η μετατροπή των μαθηματικών εννοιών που έχουν περιγράφει ήδη, σε κώδικα. Οι εξισώσεις που είναι απαραίτητες αναπτύσσονται με μαθηματικό τρόπο.

Βήμα 1

$$\hat{\boldsymbol{x}}_{k|k-1} = \boldsymbol{F}\hat{x}_{k-1|k-1} + \boldsymbol{B}\dot{\theta}_k$$

$$\begin{bmatrix} \theta \\ \dot{\theta}_b \end{bmatrix}_{k|k-1} = \begin{bmatrix} 1 & -\Delta t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \theta \\ \dot{\theta}_b \end{bmatrix}_{k-1|k-1} + \begin{bmatrix} \Delta t \\ 0 \end{bmatrix} \dot{\theta}_k$$

$$= \begin{bmatrix} \theta - \dot{\theta}_b \Delta t + \dot{\theta}\Delta t \\ \dot{\theta}_b \end{bmatrix}_{k-1|k-1} + \begin{bmatrix} \Delta t \\ 0 \end{bmatrix} \dot{\theta}_k$$

$$= \begin{bmatrix} \theta - \dot{\theta}_b \Delta t \\ \dot{\theta}_b \end{bmatrix}_{k-1|k-1} + \begin{bmatrix} \Delta t \\ 0 \end{bmatrix} \dot{\theta}_k$$

$$= \begin{bmatrix} \theta + \Delta t(\dot{\theta} - \dot{\theta}_b) \\ \dot{\theta}_b \end{bmatrix}$$

Η a priori εκτίμηση της γωνίας $\hat{\theta}_{k|k-1}$ είναι ίση με την εκτίμηση της προηγούμενης κατάστασης συν την μέτρηση χωρίς το σφάλμα επί το delta time Δt . Επειδή δεν μπορούμε απευθείας να υπολογίσουμε το σφάλμα, η εκτίμηση του σφάλματος της a priori κατάστασης είναι ίση με την προηγούμενη. Αυτό γράφεται ως εξής:

```
KalmanX.rate = newRate - KalmanX.bias;
KalmanX.angle += dt * KalmanX.rate;
```

Βήμα 2

$$\boldsymbol{P}_{k|k-1} = \boldsymbol{F}\boldsymbol{P}_{k-1|k-1}\boldsymbol{F}^T + \boldsymbol{Q}_k$$

$$\begin{bmatrix} P_{00} & P_{01} \\ P_{10} & P_{11} \end{bmatrix}_{k|k-1} = \begin{bmatrix} 1 & -\Delta t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} P_{00} & P_{01} \\ P_{10} & P_{11} \end{bmatrix}_{k-1|k-1} \begin{bmatrix} 1 & 0 \\ -\Delta t & 1 \end{bmatrix} + \begin{bmatrix} Q_\theta & 0 \\ 0 & Q_{\dot{\theta}_b} \end{bmatrix} \Delta t$$

$$= \begin{bmatrix} P_{00} - \Delta t P_{10} & P_{01} - \Delta t P_{11} \\ P_{10} & P_{11} \end{bmatrix}_{k-1|k-1} \begin{bmatrix} 1 & 0 \\ -\Delta t & 1 \end{bmatrix} + \begin{bmatrix} Q_\theta & 0 \\ 0 & Q_{\dot{\theta}_b} \end{bmatrix} \Delta t$$

$$= \begin{bmatrix} P_{00} - \Delta t P_{10} - \Delta t(P_{01} - \Delta t P_{11}) & P_{01} - \Delta t P_{11} \\ P_{10} - \Delta t P_{11} & P_{11} \end{bmatrix}_{k-1|k-1} + \begin{bmatrix} Q_\theta & 0 \\ 0 & Q_{\dot{\theta}_b} \end{bmatrix} \Delta t$$

$$= \begin{bmatrix} P_{00} - \Delta t P_{10} - \Delta t(P_{01} - \Delta t P_{11}) + Q_\theta \Delta t & P_{01} - \Delta t P_{11} \\ P_{10} - \Delta t P_{11} & P_{11} + Q_{\dot{\theta}_b} \Delta t \end{bmatrix}$$

Οι παραπάνω εξισώσεις υλοποιούνται ως εξής:

```
KalmanY.P[0][0] += dt * (dt*KalmanY.P[1][1] - KalmanY.P[0][1] -
KalmanY.P[1][0] + KalmanY.Q_angle);
KalmanY.P[0][1] -= dt * KalmanY.P[1][1];
KalmanY.P[1][0] -= dt * KalmanY.P[1][1];
KalmanY.P[1][1] += KalmanY.Q_bias * dt;
```

Βήμα 3

$$\tilde{\boldsymbol{y}}_k = \boldsymbol{z}_k - \boldsymbol{H}\hat{x}_{k|k-1}$$

$$= \boldsymbol{z}_k - \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} \theta \\ \dot{\theta}_b \end{bmatrix}_{k|k-1}$$

$$= \boldsymbol{z}_k - \theta_{k|k-1}$$

```
float y = newAngle - KalmanY.angle; // Angle difference
```

Βήμα 4

$$\boldsymbol{S}_k = \boldsymbol{H}\boldsymbol{P}_{k|k-1}\boldsymbol{H}^T + \boldsymbol{R}$$

$$= \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} P_{00} & P_{01} \\ P_{10} & P_{11} \end{bmatrix}_{k|k-1} \begin{bmatrix} 1 \\ 0 \end{bmatrix} + \boldsymbol{R}$$

$$= P_{00_{k|k-1}} + \boldsymbol{R}$$

$$= P_{00_{k|k-1}} + var(v)$$

```
float S = KalmanY.P[0][0] + KalmanY.R_measure;
```

Βήμα 5

$$\boldsymbol{K}_k = \boldsymbol{P}_{k|k-1}\boldsymbol{H}^T\boldsymbol{S}_k^{-1}$$

$$\begin{bmatrix} K_0 \\ K_1 \end{bmatrix}_k = \begin{bmatrix} P_{00} & P_{01} \\ P_{10} & P_{11} \end{bmatrix}_{k|k-1} \begin{bmatrix} 1 \\ 0 \end{bmatrix} \boldsymbol{S}_k^{-1}$$

$$= \begin{bmatrix} P_{00} \\ P_{10} \end{bmatrix}_{k|k-1} \boldsymbol{S}_k^{-1}$$

$$= \frac{\begin{bmatrix} P_{00} \\ P_{10} \end{bmatrix}_{k|k-1}}{\boldsymbol{S}_k}$$

```
K[0] = KalmanX.P[0][0] / S;
K[1] = KalmanX.P[1][0] / S;
```

Βήμα 6

$$\hat{\boldsymbol{x}}_{k|k} = \hat{\boldsymbol{x}}_{k|k-1} + \boldsymbol{K}_k\, \tilde{\boldsymbol{y}}_k$$

$$\begin{bmatrix} \theta \\ \dot{\theta}_b \end{bmatrix}_{k|k} = \begin{bmatrix} \theta \\ \dot{\theta}_b \end{bmatrix}_{k|k-1} + \begin{bmatrix} K_0 \\ K_1 \end{bmatrix}_k \tilde{\boldsymbol{y}}_k$$

$$= \begin{bmatrix} \theta \\ \dot{\theta}_b \end{bmatrix}_{k|k-1} + \begin{bmatrix} K_0\, \tilde{\boldsymbol{y}} \\ K_1\, \tilde{\boldsymbol{y}} \end{bmatrix}_k$$

```
KalmanY.angle += K[0] * y;
KalmanY.bias += K[1] * y;
```

Βήμα 7

$$\boldsymbol{P}_{k|k} = (\boldsymbol{I} - \boldsymbol{K}_k \boldsymbol{H}) \boldsymbol{P}_{k|k-1}$$

$$\begin{bmatrix} P_{00} & P_{01} \\ P_{10} & P_{11} \end{bmatrix}_{k|k} = \left(\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} - \begin{bmatrix} K_0 \\ K_1 \end{bmatrix}_k \begin{bmatrix} 1 & 0 \end{bmatrix} \right) \begin{bmatrix} P_{00} & P_{01} \\ P_{10} & P_{11} \end{bmatrix}_{k|k-1}$$

$$= \left(\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} - \begin{bmatrix} K_0 & 0 \\ K_1 & 0 \end{bmatrix}_k \right) \begin{bmatrix} P_{00} & P_{01} \\ P_{10} & P_{11} \end{bmatrix}_{k|k-1}$$

$$= \begin{bmatrix} P_{00} & P_{01} \\ P_{10} & P_{11} \end{bmatrix}_{k|k-1} - \begin{bmatrix} K_0\, P_{00} & K_0\, P_{01} \\ K_1\, P_{00} & K_1\, P_{01} \end{bmatrix}$$

```
KalmanY.P[0][0] -= K[0] * P00_temp;
KalmanY.P[0][1] -= K[0] * P01_temp;
KalmanY.P[1][0] -= K[1] * P00_temp;
KalmanY.P[1][1] -= K[1] * P01_temp;
```

4.9 **Bluetooth module**

Όπως αναφέραμε, το κύριο πλεονέκτημα του SPP πρωτοκόλλου είναι ότι μπορούμε να χειριστούμε το Bluetooth module σαν μία απλή σειριακή θύρα. Ότι πληροφορία θέλουμε να αποστείλουμε μέσω Bluetooth, απλά την αποστέλλουμε μέσω UART στο Bluetooth module, το ίδιο και όταν δεχόμαστε πληροφορία. Το Baud Rate είναι ρυθμισμένο από τον κατασκευαστή στα **115200bps** αλλά για ταχύτερη μεταφορά των δεδομένων το αυξάνουμε στα **460800bps**. Ουσιαστικά αυτό το Baud Rate σημαίνει πως για την **πλήρη** αποστολή ενός πακέτου μας που είναι 1436bytes από τον μικροελεγκτή **μέχρι** το Bluetooth module θα χρειαστεί:

1436 / 460800 * (10Bits ανά χαρακτήρα) = **~31.1msec**

Τα Bits ανά χαρακτήρα είναι 10 υπολογίζοντας ότι για κάθε χαρακτήρα έχουμε ορίσει 8bits Data + 1 Stop bit + 1 Start bit.

Για την αλλαγή του Baud Rate αλλά και των παραμέτρων του module, θα πρέπει να στείλουμε την κατάλληλη εντολή στο module ώστε να εισέλθει σε κατάσταση προγραμματισμού (CMD mode). Για την επικοινωνία με το module, έχει υλοποιηθεί η συνάρτηση **uint8_t Send_CMD_Bluetooth(uint8_t *buffer,uint8_t *answer,uint8_t cmp_str,uint16_t timeout)**. Η συνάρτηση αυτή, αποστέλλει την εντολή που παίρνει σαν όρισμα και τη συγκρίνει με την αντίστοιχη απάντηση που και πάλι ορίζουμε εμείς. Φυσικά η συνάρτηση δέχεται και έναν χρόνο Timeout σε msec. Παρακάτω φαίνεται η κλήση της συνάρτηση για να εισέλθει το module σε κατάσταση προγραμματισμού:

```
#define ENTER_COMMAND_MODE "$$$"

if(Send_CMD_Bluetooth(ENTER_COMMAND_MODE,"CMD\r\n",STRCMP,500) == OK)
{
...
}
```

Εντολή	**Επιθυμητή απάντηση**	**Χρόνος Timeout (msec)**	**Επεξήγηση**
"SU,46r"	"AOK\r\n"	100	Ορισμός Baud Rate στα 460800bps
"SA,4\r"	"AOK\r\n"	100	Ενεργοποίηση κωδικού κατά το Pairing
"SN,BluetoothName\r"	"AOK\r\n"	100	Ορισμός ονόματος Bluetooth
"SP,2019\r"	"AOK\r\n"	100	Ορισμός κωδικού Pairing

Πίνακας 5: Οι εντολές/αναμενόμενες απαντήσεις μεταξύ μικροελεγκτή-RN41

4.10 **Πρωτόκολλο επικοινωνίας και δημιουργία πακέτου αποστολής**

Κάθε 250msec δημιουργούμε το πακέτο που πρόκειται να αποσταλεί στο Bluetooth module μέσω της σειριακής θύρας. Για τη δημιουργία του πακέτου διαβάζουμε τις τιμές που έχουμε συλλέξει σε μία δομή της RAM μνήμης. Η συνάρτηση που καλείται για να δημιουργήσει το πακέτο είναι η **uint16_t Build_Packet(void)** και επιστρέφει τον αριθμό των bytes του πακέτου. Ο αριθμός των bytes του πακέτου είναι σταθερός αλλά για μελλοντικές αλλαγές καλό είναι να επιστρέφεται από τη συνάρτηση. Η ανάθεση της αποστολής γίνεται στον DMA και το πρόγραμμα συνεχίζει να εκτελείται.

Όπως σε όλες τις επικοινωνίες, έτσι και στη δικής μας περίπτωση θα πρέπει να ορίσουμε/δημιουργήσουμε ένα δομημένο πρωτόκολλο επικοινωνίας μεταξύ συσκευής και της εφαρμογής Android που θα μας διασφαλίζει τη μεταφορά πακέτων/εντολών αλλά και των απαντήσεων τους, χωρίς σφάλματα και άσκοπες επαναλήψεις. Για αυτό το λόγο, θα περιγράψουμε αναλυτικά τη δομή των πακέτων/εντολών/απαντήσεων καθώς και το Checksum σε συνδυασμό με το CRC που διασφαλίζουν ότι δεν απεστάλησαν δεδομένα που είναι ανακριβή λόγω εσφαλμένης μετάδοσης.

4.10.1 Πρωτόκολλο επικοινωνίας

Το πρωτόκολλο που επιλέξαμε να υλοποιήσουμε για την επικοινωνία μεταξύ συσκευής και Android εφαρμογής αποτελείται **αποκλειστικά** από ASCII χαρακτήρες, ο κύριος λόγος είναι ότι οι τιμές των μεταβλητών που περιέχουν δεδομένα, μπορούν να αναπαρασταθούν με μοναδικό τρόπου σε ASCII. Παρακάτω θα γίνει ανάλυση των χαρακτήρων που απαιτεί το πρωτόκολλο, ώστε θεωρήσει ότι το πακέτο που έφθασε είναι αποδεκτό πριν κάνει περαιτέρω ελέγχους:

- **1ο βήμα - χαρακτήρας '$':** Έναρξη πακέτου (και μηδενισμός pointer)
- **2ο βήμα - χαρακτήρας 'A' ή 'B':** Χαρακτήρας αναγνώρισης αποστολέα
  - 'A' σημαίνει ότι ο αποστολές είναι η Android εφαρμογή
  - 'B' σημαίνει ότι ο αποστολές είναι η συσκευή
- **3ο βήμα - χαρακτήρας 'A' ή 'B':** Χαρακτήρας αναγνώρισης δέκτη
  - 'A' σημαίνει ότι ο δέκτης είναι η Android εφαρμογή
  - 'B' σημαίνει ότι ο δέκτης είναι η συσκευή
- **4ο βήμα - χαρακτήρας 'E' ή 'A' ή 'N':** Αίτηση για πληροφορία ή απάντηση
  - 'E' από το ENQ, ο αποστολέας 'ζητάει' πληροφορία ή στέλνει κάποια εντολή
  - 'A' από το ACK, ο αποστολέας απαντά σε κάποια εντολή ή πακέτο
  - 'N' από το NAK, ο αποστολές απαντά με 'άρνηση' εκτέλεσης
- **5ο βήμα - χαρακτήρας 'D' ή 'C' ή 'R' ή 'W' ή 'A':** Είδος εντολής/απάντησης
  - 'D' από το Data, αφορά πακέτα με τα δεδομένα των μετρήσεων
  - 'C' από το Command, αφορά πακέτα εντολών
  - 'R' από το Read Memory, αφορά πακέτα για ανάγνωση μνήμης (RAM,Flash)
    - Δεν υποστηρίζεται αυτή η λειτουργία αλλά έχει προβλεφθεί για υλοποίηση στο μέλλον
  - 'W' από το Write Memory, αφορά πακέτα για έγγραφη μνήμης (RAM,Flash)

        - Δεν υποστηρίζεται αυτή η λειτουργία αλλά έχει προβλεφθεί για υλοποίηση στο μέλλον
    - ‘Α από το Alive, αφορά πακέτα Alive (όπου απλά ενημερώνεις ότι ‘υπάρχεις’)
        - Δεν υποστηρίζεται αυτή η λειτουργία αλλά έχει προβλεφθεί για υλοποίηση στο μέλλον. Συνήθως χρησιμοποιείται από εφαρμογές που ψάχνουν της σειριακές θύρες για επιθυμητές συσκευές
- **6ο βήμα – χαρακτήρας ‘<’:** Χαρακτήρας έναρξης δεδομένων πακέτου
- **7ο βήμα: - Data:** Δεδομένα πακέτου (εάν υπάρχουν) και CRC (εάν χρειάζεται)
    - Τα πακέτα εντολών έχουν δεδομένα αλλά δεν έχουν CRC μιας και η πληροφορία δεν είναι κρίσιμη όπως τα δεδομένα μετρήσεων
    - Το μέγεθος των Data δεν είναι ανάγκη να είναι σταθερό, υπολογίζεται με βάση τους χαρακτήρες έναρξης και λήξης δεδομένων πακέτου
    - Τα πακέτα Data περιέχουν δεδομένα μετρήσεων, όπως επίσης **περιέχουν και CRC μεγέθους** 16bit για να εξασφαλίσουμε ότι η μεταφορά έγινε χωρίς σφάλματα

- **8ο βήμα – χαρακτήρας ‘>’:** Χαρακτήρας λήξης δεδομένων πακέτου

- **9ο βήμα – Checksum:** Checksum πακέτου
    - Το Checksum υπολογίζεται ανάμεσα στους χαρακτήρες [‘$’:’>’]
    - Το Checksum είναι ένας αριθμός 8bit που υπολογίζεται προσθέτοντας κάθε byte του πακέτου στο εύρος που αναφέρουμε παραπάνω
- **10ο βήμα – χαρακτήρας ‘*’:** Χαρακτήρας λήξης πακέτου

4.10.2 Υπολογισμός Checksum και CRC

Το Checksum είναι από τους απλούς τρόπους ελέγχου αποστολής/αποθήκευσης δεδομένων. Ουσιαστικά αθροίζουμε τις τιμές των θέσεων του πακέτου και παίρνουμε ένα μη προσημασμένο αριθμό 8bit. Μπορεί υπολογιστική ισχύ που απαιτεί να είναι ελάχιστη αλλά δεν μας βοηθά να καταλάβουμε σφάλμα στη ‘σειρά’ των δεδομένων. Για παράδειγμα, το Checksum του πίνακα [‘1’,’2’,’3’,’4’] είναι ίδιο με το Checksum του πίνακα [‘2’,’1’,’3’,’4’], δηλαδή ίσο με 0x32h+0x31+0x33+0x34 = **0xCA**. Η συνάρτηση που καλούμε είναι η παρακάτω:

```
uint8_t Calculate_Checksum(uint8_t *buffer,uint16_t length)
{
        uint8_t temp_checksum=0;
        for(uint16_t i=0;i < length; i++)
        {
                temp_checksum += buffer[i];
        }
        return temp_checksum;
}
```

Το CRC (Cyclic Redundancy Check) είναι μια τεχνική ανίχνευσης σφαλμάτων κατά τη διάρκεια μετάδοσης ή αποθήκευσης δεδομένων. Βασίζεται συνήθως σε κάποιο πολυώνυμο και είναι πιο αξιόπιστο από το απλό Checksum μιας και αντιλαμβάνεται σφάλμα ακόμα και

στη ‘σειρά’ των δεδομένων. Η υπολογιστική ισχύς που απαιτεί είναι μεγαλύτερη από του Checksum αλλά δεν είναι πρόβλημα για την εφαρμογή μας. Η συνάρτηση που καλούμε για τον υπολογισμό ενός CRC των 16bit είναι η παρακάτω:

```
uint16_t gen_crc16(uint8_t *buffer, uint16_t size)
{
        uint16_t crc = 0xFFFF;

        if (buffer && size)
        while (size--)
          {
              crc = (crc >> 8) | (crc << 8);
              crc ^= *buffer++;
              crc ^= ((unsigned char) crc) >> 4;
               crc ^= crc << 12;
               crc ^= (crc & 0xFF) << 5;
          }
           return crc;
}
```

4.10.3 Μετατροπή HEX σε ASCII

Όπως αναφέραμε και παραπάνω, δεν γίνεται να αποσταλούν οι τιμές των μεταβλητών σε HEX μορφή. Ο λόγος είναι ότι αν αποστείλουμε τις τιμές χωρίς να της μετατρέψουμε σε ASCII, μπορούν να πάρουν οποιαδήποτε τιμή και να κάνουν την ανάλυση του πακέτου αδύνατη. Για παράδειγμα:

Χωρίς μετατροπή σε ASCII

- Έστω μία μεταβλητή μεγέθους 16bit (ας υποθέσουμε μη προσημασμένος αριθμούς) με δεκαδική τιμή 9258.
- Τα bytes της μεταβλητής περιέχουν της δεκαεξαδικές τιμές {0x24,0x2A}
- Δηλαδή η μεταβλητή περιέχει τους ASCII χαρακτήρες {‘$’,’*‘}

Είναι προφανές πως με αυτή τη λογική το πακέτο μπορεί να περιέχει οποιοδήποτε χαρακτήρα και κάνει την ανάλυση του πακέτου να περιέχει μεγάλο ρίσκο επειδή δεν μπορούμε να διαχωρίσουμε τη πληροφορία σωστά.

Μετατροπή σε ASCII

- Έστω μία μεταβλητή μεγέθους 16bit (ας υποθέσουμε μη προσημασμένος αριθμούς) με δεκαδική τιμή 9258.
- Τα bytes της μεταβλητής περιέχουν της δεκαεξαδικές τιμές {0x24,0x2A}
- Για να μετατρέψουμε τις δεκαεξαδικές τιμές σε ASCII σημαίνει ότι θα διπλασιάσουμε το μέγεθος. Δηλαδή θα πρέπει να αποστείλουμε τα 4bytes {0x32,0x34,0x32,0x41} = {‘2’,’4’,’2’,’A’}

Με το παραπάνω τρόπο ουσιαστικά μεγαλώνουμε το μέγεθος του πακέτου αποστολής αλλά κερδίζουμε ένα σωστά δομημένο πακέτο. Οποιαδήποτε τιμή και αν πάρουν οι μεταβλητές

μας, πάντα θα μπορούμε να τις αποστείλουμε σε ASCII μορφή με μοναδικό τρόπο. Η απλή συνάρτηση που υλοποιήσαμε για τη μετατροπή, είναι η παρακάτω:

```
void Convert_Hex_To_Ascii_Measures(uint8_t value,uint16_t index)
{
        uint8_t temp_high = value >> 4;
        uint8_t temp_low  = value & 0x0F;

        if(temp_high < 0x0A) { Packet_To_Transmit[index] = temp_high + 48;}
        else { Packet_To_Transmit[index] = temp_high + 55; }


        if(temp_low < 0x0A) { Packet_To_Transmit[index+1] = temp_low + 48; }
        else { Packet_To_Transmit[index+1] = temp_low + 55; }
}
```

4.10.4 Δομή δεδομένων

Τα δεδομένα που αποστέλλουμε μέσα στο πακέτο θα πρέπει να είναι με συγκεκριμένο τρόπο και σειρά ώστε να μπορεί να τα διαχειριστεί και η Android εφαρμογή. Για το λόγο αυτό, κάθε φορά που δημιουργούμε ένα πακέτο για αποστολή οι μεταβλητές τοποθετούνται στη περιοχή των δεδομένων πακέτου με την παρακάτω σειρά:

Περιγραφή	**Μέγεθος μεταβλητής**	**Όνομα μεταβλητής**
Αριθμός πακέτου	16bit	Measure.Number
Τιμή καταχωρητή ACC_X (επιταχυνσιόμετρου)	16bit	Measure.AccInfo[0:49].X
Τιμή καταχωρητή ACC_Y (επιταχυνσιόμετρου)	16bit	Measure.AccInfo[0:49].Y
Τιμή καταχωρητή ACC_Z (επιταχυνσιόμετρου)	16bit	Measure.AccInfo[0:49].Z
Τιμή γωνίας Pitch (degrees)	16bit	Measure.GeneralInfo[0:49].Pitch
Τιμή γωνίας Roll (degrees)	16bit	Measure.GeneralInfo[0:49].Roll
Τιμή ταχύτητας (kh/h)	16bit	Measure.GeneralInfo[0:49].Velocity
Τιμή αναλογικής τάσης γέφυρας (Volts)	8bit	Measure.AnalogRead[0:49]
Τιμή συνολικής απόστασης (meters)	32bit	Measure.Distance
Τιμή καταχωρητή θερμοκρασίας (επιταχυνσιόμετρου)	16bit	Measure.Temperature
Τιμή ADC για τάση μπαταρίας	16bit	Device.Power.Battery_Voltage

Πίνακας 6: Η δομή των δεδομένων ενός πακέτου

Τα στοιχεία του πίνακα 6 με κόκκινο χρώμα, αποστέλλονται αλλά δεν περιέχουν πραγματικές τιμές. Είναι ουσιαστικά οι δύο λειτουργίες που θέλουμε να υλοποιήσουμε μελλοντικά (μέτρηση ταχύτητας και απόστασης)

4.10.5 Παραδείγματα πακέτων επικοινωνίας και απαντήσεις

Τα είδη των πακέτων και των απαντήσεων που αποστέλλονται μεταξύ συσκευής και Android εφαρμογής περιγράφονται παρακάτω. Τα πακέτο με τα δεδομένα από τις μετρήσεις, αποστέλλεται από τη συσκευή προς την εφαρμογή κάθε 250msec. Τα υπόλοιπα πακέτα σηματοδοτούν την έναρξη ή την παύση των μετρήσεων και αποστέλλονται από την εφαρμογή όταν πατηθεί το αντίστοιχο κουμπί της εφαρμογής.

4.10.5.1 Data

Πακέτο με δεδομένα μετρήσεων από το τη συσκευή προς την Android εφαρμογή:

$BAED<0001FFCC0036F80B00C0431A00000000FFC10021F80E00C1435800000000FFB
D0029F80300C2439300000000FFC70021F81C00C143CA00000000FFCF0023F81D00C04
3FE00000000FFAD0037F81F00BF443100000000FFCC002EF71100BE446300000000FFB
B0025F80F00BF449200000000FFBE002EF80E00BF44BF00000000FFC3002DF81600BF
44EA00000000FFB5002DF80200C0451300000000FFB8003EF81B00C1453B00000000FF
BD002BF80800C1456100000000FFBE002CF80200C1458600000000FFAA003CF80E00C
245A900000000FFC20028F80800C245C900000000FFCB003BF80F00C245E700000000F
FC10029F81100C3460500000000FFBF0033F80100C2462200000000FFB40024F81B00C3
463E00000000FFBA002AF70500C3465900000000FFC4001EF81500C3467200000000FF
AF002FF80F00C3468A00000000FFC90029F80D00C346A100000000FFAB003DF80D00
C446B600000000FFCD0031F80400C446CA00000000FFB20017F80100C546DE00000000
FFB30019F80100C646F100000000FFC60034F81500C6470200000000FFB10025F81900C
5471100000000FFC20022F81400C4472100000000FFAD0025F82400C4473200000000FF
C20034F7FF00C3473F00000000FFA9001AF81100C4474C00000000FFBC0024F80C00C
4475900000000FFAD0033F80E00C5476400000000FFAA002AF80C00C7476E00000000F
FC10031F81900C6477800000000FFAC001CF80E00C7478200000000FFCB0039F80600C
6478B00000000FFD0001FF81500C6479400000000FFB4001EF81900C5479B00000000FF
C7001DF81300C447A300000000FFB0002AF82300C547A900000000FFBB002BF82D00
C647B000000000FFC30031F81200C547B500000000FFC20024F80F00C647BA00000000
FFB00038F80300C747BE00000000FFBC002AF80B00C747C200000000FFC80018F80B0
0C747C60000000066445566FFEF0E527F8D>49*

Αξίζει να σημειωθούν τα παρακάτω:

- Με κόκκινο χρώμα είναι ο αύξων αριθμός πακέτου που αυξάνει κατά ένα με κάθε αποστολή πακέτου. Στο συγκεκριμένο παράδειγμα έχει τη τιμή 0x0001
- Με πράσινο χρώμα είναι το CRC των δεδομένων. Στο συγκεκριμένο παράδειγμα έχει την τιμή 0x7F8D
- Με μπλε χρώμα είναι το Checksum του πακέτου. . Στο συγκεκριμένο παράδειγμα έχει την τιμή 0x49

Τα υπόλοιπα bytes περιέχουν τα δεδομένα που σχετίζονται με τις μετρήσεις μας. Το μέγεθος του πακέτου είναι 1434bytes.

Απάντηση σε πακέτο με δεδομένα μετρήσεων από την Android εφαρμογή προς τη συσκευή:

$ABAD<0001>62*

Με κόκκινο χρώμα μέσα στην απάντηση, είναι ο αριθμός του πακέτου που αναφέρεται η απάντηση.

4.10.5.2 Start/End

Εντολή έναρξης διαδικασίας μετρήσεων από την Android εφαρμογή προς τη συσκευή:

$ABEC<S1>2D*

Με κόκκινο χρώμα μέσα στην εντολή, είναι τα 2bytes που δηλώνουν ότι είναι εντολή έναρξης.

Απάντηση σε εντολή έναρξης διαδικασίας μετρήσεων από τη συσκευή προς την Android εφαρμογή:

$BAAC<S1>29*

Με κόκκινο χρώμα μέσα στην εντολή, είναι τα 2bytes που δηλώνουν ότι είναι απάντηση σε εντολή έναρξης.

Εντολή παύσης διαδικασίας μετρήσεων από την Android εφαρμογή προς τη συσκευή:

$ABEC<S0>2C*

Με κόκκινο χρώμα μέσα στην εντολή, είναι τα 2bytes που δηλώνουν ότι είναι εντολή παύσης.

Απάντηση σε εντολή παύσης διαδικασίας μετρήσεων από τη συσκευή προς την Android εφαρμογή:

$BAAC<S06754>29*

Με κόκκινο χρώμα μέσα στην εντολή, είναι τα 2bytes που δηλώνουν ότι είναι απάντηση σε εντολή παύσης. Με πράσινο χρώμα είναι ο αριθμός των πακέτων που απεστάλησαν με δεδομένα μετρήσεων, στα οποία η συσκευή δεν πήρα πίσω απάντηση. Η συσκευή έχει αποθηκευμένα τα πακέτα στα οποία δεν πήρε απάντηση, έτσι **μελλοντικά** μπορεί να υλοποιηθεί η λειτουργία αποστολής αυτών των πακέτων όταν το ζητήσει η Android εφαρμογή.

4.11 USB και πρωτόκολλο επικοινωνίας

Η USB θύρα, χρησιμοποιείται για τη φόρτιση της συσκευής ή για την παραμετροποίηση της συσκευής. Χρησιμοποιήσαμε το πρωτόκολλο USB CDC για να δημιουργήσουμε μια εικονική σειριακή επικοινωνία (Virtual Serial Communication) μεταξύ της συσκευής και του υπολογιστή. Η βιβλιοθήκη για το USB CDC περιέρχεται στα αρχεία που παράγει το STM32Cube, εμείς απλά τροποποιούμε τις συναρτήσεις για λήψη και αποστολή δεδομένων. Τα κύρια πλεονεκτήματα από μια τέτοια υλοποίηση είναι η απλότητα που παρέχει καθώς δεν απαιτεί ανάπτυξη οδηγών (Drivers) από την πλευρά του υπολογιστή. Για την επικοινωνία μεταξύ συσκευής και υπολογιστή, αναπτύξαμε ένα πρόγραμμα σε C#. Μέσω του προγράμματος που υλοποιήσαμε, ο χρήστης μπορεί να διαβάσει ή να αλλάξει παραμέτρους όπως το όνομα της συσκευής ή τον κωδικό Pairing.

4.11.1 Επικοινωνία μεταξύ υπολογιστή-συσκευής

Όπως αναφέραμε, ο χρήστης μπορεί να διαβάσει ή να αλλάξει τις παραμέτρους της συσκευής. Έτσι ουσιαστικά μιλάμε για δύο εντολές που υλοποιούν την ανάγνωση και την εγγραφή. Ο μικροελεγκτής ενημερώνεται για εισερχόμενα bytes από τον υπολογιστή μέσω της callback συνάρτησης **CDC_Receive_FS()**. Παρακάτω φαίνεται ένα μέρος της συνάρτησης αυτής, όπως και οι πιθανές απαντήσεις:

```
static int8_t CDC_Receive_FS(uint8_t* Buf, uint32_t *Len)
{
          uint8_t answer = 0;
          answer = Check_And_Save_Parameters(Buf,Len);

          if(answer == 1) // Reply that parameters saved
        correctly {  CDC_Transmit_FS("$ACK$",5);  }
          else if(answer == 6) // Send back the stored parameters
          {
            uint8_t response_buffer[100] = {0};
            uint8_t checksum = 0;
            Load_Parameters();
            snprintf(response_buffer,100,"$GETPARAM[%s,%s]",Parameters.Name,Parameters.Pass
        );
            checksum = Calculate_Checksum(response_buffer,strlen(response_buffer));
            snprintf(response_buffer,100,"%s%02X%",response_buffer,checksum);
            CDC_Transmit_FS(response_buffer,strlen(response_buffer));
          }
          else if(answer == 5) // Flash error { CDC_Transmit_FS("$FLASH_ERROR$",13); }
          else if(answer == 2) // Wrong checksum { CDC_Transmit_FS("$CHECKSUM$",10); }
          else if (answer == 4) // Wrong parameters {
        CDC_Transmit_FS("$WRONG_PARAM$",13); }

          USBD_CDC_SetRxBuffer(&hUsbDeviceFS, &Buf[0]);
          USBD_CDC_ReceivePacket(&hUsbDeviceFS);
          return (USBD_OK);
}
```

Σε περίπτωση που το πακέτο που δέχεται η συσκευή είναι λάθος, αποστέλλουμε την απάντηση με το λόγο που δεν έγινε δεκτό. Αντίστοιχα και στην desktop εφαρμογή, θα εμφανιστεί κάποιο μήνυμα που θα ενημερώσει το χρήση. Ο χρήστης μπορεί επίσης να δει την επικοινωνία ένα επιθυμεί σε ένα παράθυρο που λειτουργεί σαν Log.

4.11.2 Εντολή ανάγνωσης παραμέτρων και απάντηση

Η παρακάτω εντολή αποστέλλεται από την desktop εφαρμογή προς τη συσκευή για ανάγνωση των παραμέτρων:

$GETPARAM[]%

Η εντολής ανάγνωσης δεν έχει κάτι ιδιαίτερο αλλά η απάντηση της συσκευής θα πρέπει να περιέχει και Checksum για να εξασφαλίσουμε ότι το πακέτο θα μεταφερθεί χωρίς κάποιο λάθος, να σημειώσουμε πως το όνομα και ο κωδικός είναι σε ASCII μορφή

$**GET**PARAM[BluetoothSK,3558]82%

Με κόκκινο χρώμα είναι το όνομα της συσκευής και με πράσινο χρώμα είναι ο κωδικός Pairing. Με μωβ χρώμα είναι το Checksum, το οποίο έχει μετατραπεί σε ASCII. Σε περίπτωση που το πακέτο περιέχει λάθος Checksum, εμφανίζεται αντίστοιχο μήνυμα στην εφαρμογή.

4.11.3 Εντολή αλλαγής παραμέτρων και απάντηση

Η εντολή αλλαγής παραμέτρων είναι παρόμοια με την εντολή ανάγνωσης αλλά περιέχει και Checksum για να εξασφαλίσουμε ότι το πακέτο θα μεταφερθεί χωρίς κάποιο λάθος.

$**SET**PARAM[BluetoothSK,1991]8D%

Η απάντηση σε εντολή αλλαγής παραμέτρων μπορεί να είναι μία από τις παρακάτω:

- **“ACK”:** Η διαδικασία ολοκληρώθηκε επιτυχώς
- **“$WRONG_PARAM$”:** Λάθος τιμές παραμέτρων. Το όνομα πρέπει να είναι αλφαριθμητικός χαρακτήρας και ο κωδικός να είναι ένας τετραψήφιος δεκαδικός αριθμός
- **“$FLASH_ERROR$”:** Αδυναμία εγγραφής παραμέτρων στη Flash μνήμη
- **“$CHECKSUM$”:** Λάθος Checksum

4.12 Μέτρηση τάσης μπαταρίας και γέφυρας

Για τη μέτρηση της τάσης της μπαταρίας και της γέφυρας, χρησιμοποιούμε τον εσωτερικό ADC του μικροελεγκτή. Η ανάλυση του ADC είναι 12bit, η συχνότητα του περιφερειακού είναι ρυθμισμένη στα 12MHz και κάθε μετατροπή χρειάζεται περίπου 20μsec για να ολοκληρωθεί. Την διαδικασία της έναρξης κάθε μετατροπής και της μεταφοράς της τιμής του ADC σε μία θέση-πίνακα της RAM, αναλαμβάνει ο DMA. Το μόνο που χρειάζεται είναι να δηλωθεί μία global μεταβλητή στην οποία ο DMA θα ανανεώνει τις καινούργιες τιμές

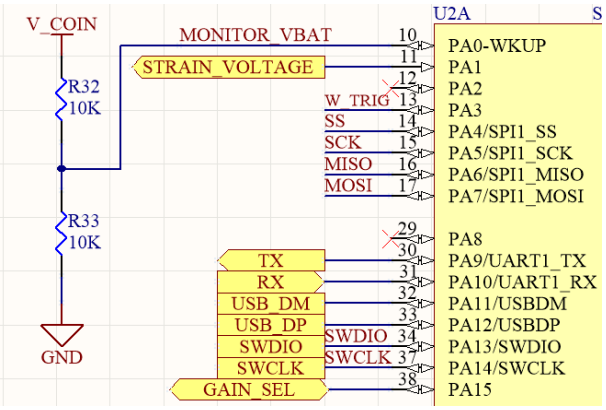
όποτε τον σκανδαλίζουμε και έχει ολοκληρωθεί η μετατροπή. Η συνάρτηση με την οποία σκανδαλίζουμε τον DMA, είναι η παρακάτω:

```
HAL_ADC_Start_DMA(&hadc1, (uint32_t*) DMA_ADCReadings, 2);
```

Με την κλήση της παραπάνω συνάρτησης, ο DMA ξεκινάει δύο μετατροπές (την μία μετά την άλλη) και καταχωρεί το αποτέλεσμα στον πίνακα **uint16_t DMA_ADCReadings[2].** Κάθε φορά που θέλουμε να διαβάσουμε την τιμή του ADC για οποιοδήποτε κανάλι, απλά διαβάζουμε την τιμή της αντίστοιχης θέσης του πίνακα. Για παράδειγμα, για τον υπολογισμό της τάσης της μπαταρίας, παίρνουμε 50 διαδοχικές μετρήσεις και υπολογίζουμε τον μέσο όρο:

```
BatterySUM += (uint16_t) (DMA_ADCReadings[0]*1.6117f);
if(BatterySamples >= 49)
{
        Device.Power.Battery_Voltage = (uint16_t) (BatterySUM/50);
        BatterySamples=0;
        BatterySUM=0;
}
else
{
        BatterySamples++;
}
```

Ο πολλαπλασιασμός που γίνεται με τη τιμή **1.6117f**, είναι για να μετατραπεί η τιμή που επιστρέφει ο ADC σε mVolts. Να σημειωθεί ότι η τάσης της μπαταρίας υποδιπλασιάζεται με τη χρήση ενός διαιρέτη τάσης για να είναι σε αποδεκτά όρια.



Εικόνα 29: Διαιρέτης τάσης για τον υποδιπλασιασμό της τάσης της μπαταρίας προς τον ADC

Αντίστοιχα, όταν θέλουμε να αποθηκεύσουμε την τιμή του καναλιού του ADC που αφορά την γέφυρα (έξοδο του AD8555), απλά παίρνουμε δεύτερη τιμή που περιέχει η δεύτερη θέση του πίνακα μας:

```
Measure.AnalogRead[Measure.Index] = DMA_ADCReadings[1];
```

4.13 **Προγραμματισμός του AD8555**

Για να προγραμματίσουμε το AD8555, χρειάζεται να αποστείλουμε 3 ‘λέξεις’ των 38-bit. Για τον προγραμματισμό του AD8555, υλοποιήσαμε μία βιβλιοθήκη που δημιουργεί τις λέξεις των 32-bit όπως αναφέρει το φύλλο δεδομένων του κατασκευαστή. Τα bits της κάθε λέξης ορίζονται από τις τιμές και τους καταχωρητές που θέλουμε να προγραμματίσουμε. Αξίζει να σημειωθεί πως η τιμή του κάθε bit κατά την αποστολή, ορίζεται από το χρόνο που ο παλμός παραμένει σε υψηλή στάθμη. Επειδή το χρονικό διάστημα που πρέπει ο παλμός να παραμείνει σε υψηλή στάθμη, είναι της τάξεως των μsec. Αυτό σημαίνει πως είτε θα έπρεπε να χρησιμοποιήσουμε επιπλέον Timer είτε να χρησιμοποιήσουμε χρονικές καθυστερήσεις με χρήση εντολών **__NOP()**. Οι επαναλήψεις για να πετύχουμε τους χρόνος που θέλουμε, υπολογίστηκαν πειραματικά (με χρήση παλμογράφου). Η συνάρτηση που αποστέλλει ένα bits προς το AD8555 φαίνεται παρακάτω:

```
void AD8555_BitSend(uint8_t Bit)
{
        NOP_Delay(150); // ~20μs (Width between Pulses, >=10μs)
        HAL_GPIO_WritePin(GPIOA, GAIN_SEL_Pin, GPIO_PIN_SET);
       // ~80μs (Pulse Width for Loading 1 into Shift Register, >=50μs)
       if(Bit)  {   NOP_Delay(600);  }
       // ~6μs (Pulse Width for loading Register, Range between 50 ns and 10μs )
       else  {    NOP_Delay(40);  }
      HAL_GPIO_WritePin(GPIOA, GAIN_SEL_Pin, GPIO_PIN_RESET);
}
```

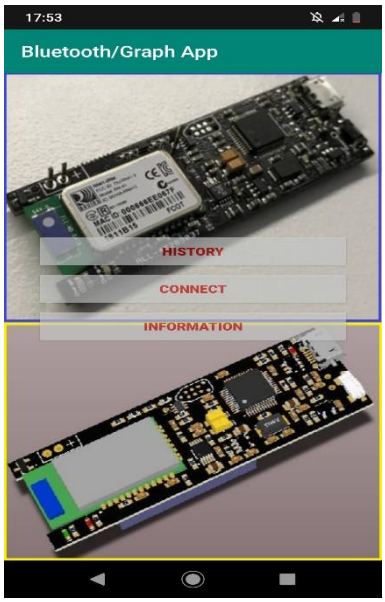
Ιδιαίτερη προσοχή χρειάζεται στο ότι δεν μπορούμε να προγραμματίσουμε τις εσωτερικές ασφάλειες του AD8555 και να παραμείνουν οι ρυθμίσεις μας ακόμα και μετά από διακοπή τροφοδοσίας. Ο λόγος είναι ότι ο κατασκευαστής απαιτεί τάση τροφοδοσίας στο ολοκληρωμένο 5V κατά τον προγραμματισμό των ασφαλειών, πράγμα που απαιτούσε επιπλέον υλικά στη πλακέτα μας. Με βάση όσα αναφέρθηκαν παραπάνω, στη συνάρτηση που στέλνει πακέτα/λέξεις προς το AD8555, γίνεται έλεγχος των ορισμάτων που εισάγει ο χρήστης και αποφεύγει το προγραμματισμό των ασφαλειών. Προγραμματισμός ασφαλειών με λάθος τάση τροφοδοσίας ή με λάθος τιμές, ουσιαστικά αχρηστεύει το ολοκληρωμένο μιας και δε μπορούμε να αναιρέσουμε το λάθος μας! Η συνάρτηση για την αποστολή των λέξεων/πακέτων και ο έλεγχος που αναφέραμε, φαίνεται παρακάτω:

```
uint8_t AD8555_SendPacket(uint8_t Mode,uint8_t Function,uint8_t Value)
{
            // Do not accept Program Mode. It requires 5V supply and
             fuses cannot be changed if we make any mistake!!!
            if(Mode != AD8555_READ &&
               Mode != AD8555_CHANGE_SENSE_CURRENT &&
               Mode != AD8555_SIMULATE)
              {
                    return 0;
              }
               else if(Function != AD8555_SSG_CODE
                  && Function != AD8555_FSG_CODE
                  && Function != AD8555_OFS_CODE
```

```
        && Function != AD8555_OTHER)
    {
        return 0;
    }
    AD8555_StartFrame();
    AD8555_SendParameter(Mode);
    AD8555_SendParameter(Function);
    AD8555_BitSend(1); // Dummy
    AD8555_BitSend(0); // Dummy
    AD8555_ParameterValue(Value);
    AD8555_StopFrame();
    return 1;
}
```

5. Android εφαρμογή

Στο κεφάλαιο αυτό, θα γίνει περιγραφή της Android εφαρμογής όπως και της λογικής που ακολουθήθηκε. Επίσης, θα γίνει αναφορά στα εργαλεία που χρησιμοποιήθηκαν για τη συγγραφή του κώδικα αλλά και των βιβλιοθηκών ανοιχτού κώδικα που χρησιμοποιήσαμε.



Εικόνα 30: Η αρχική οθόνη της Android εφαρμογής

5.1 Λειτουργία εφαρμογής

Η κύρια λειτουργία που εκτελεί η Android εφαρμογή είναι να απεικονίζει σε πραγματικό χρόνο τα δεδομένα μετρήσεων που της αποστέλλει η συσκευή μέσω του Bluetooth SPP πρωτοκόλλου. Για να ξεκινήσει η διαδικασία θα πρέπει να γίνει επιτυχής σύνδεση μεταξύ εφαρμογής και συσκευής. Για να γίνει επιτυχής σύνδεση για πρώτη φορά, θα πρέπει ο χρήστης να κάνει σύζευξη (Pairing) με τη συσκευή και να πληκτρολογήσει το σωστό κωδικό (Pairing key). Από τη στιγμή που η παραπάνω διαδικασία γίνει για πρώτη φορά, η Android συσκευή 'θυμάται' τα στοιχεία και δεν χρειάζεται να τα πληκτρολογούμε κάθε φορά. Τα πακέτα που δέχεται η εφαρμογή θα πρέπει να αναλυθούν με βάση το πρωτόκολλο αποστολής και να γίνει έλεγχος Checksum και CRC για να εξασφαλίσουμε ότι η μετάδοση των δεδομένων έγινε χωρίς σφάλματα. Σε περίπτωση που είτε το Checksum είτε το CRC είναι

λάθος, το πακέτο απορρίπτεται. Λόγω της απεικόνισης σε πραγματικό χρόνο, δεν υπάρχει δυνατότητα να αποσταλεί ξανά κάποιο πακέτο που δεν μεταδόθηκε σωστά ή ακόμα και καθόλου (για παράδειγμα λόγω αποσύνδεσης). Ο μόνος τρόπος να γίνει αποστολή των χαμένων πακέτων είναι όταν ο χρήστης αποφασίσει να σταματήσει τη διαδικασία μετρήσεων. Η συσκευή εάν δε λάβει acknowledge για κάποιο πακέτο εντός κάποιου χρονικού διαστήματος (200msec), αποθηκεύσει το αντίστοιχο πακέτο στη Flash μνήμη. Το πρόβλημα με τη παραπάνω λειτουργία, είναι ότι κατά την δημιουργία των ιστορικών γραφημάτων η εφαρμογή θα πρέπει να διαβάζει **όλο** το αρχείο (και όχι γραμμή-γραμμή) και να τοποθετεί τα πακέτα με τη σωστή σειρά πριν τα αποστείλει στη βιβλιοθήκη γραφικών. Αυτή η διαδικασία είναι χρονοβόρα και δεν έχει νόημα, θα πρέπει **μελλοντικά** να βελτιστοποιηθεί ο τρόπος αποθήκευσης των πακέτων για να μπορεί η εφαρμογή να ταξινομεί τα πακέτα με τη σωστή χρονικά σειρά.

Για την ομαλή λειτουργία και επικοινωνία, η εφαρμογή προσπαθεί να επανασυνδεθεί με την συσκευή σε περίπτωση αποσύνδεσης. Μόλις συνδεθούμε επιτυχώς, ο χρήστης μπορεί με τα κουμπιά Start και End να ξεκινήσει ή να σταματήσει την αποστολή δεδομένων. Στην παρακάτω εικόνα φαίνονται τα πεδία και τα γραφήματα της εφαρμογής πριν ακόμα ο χρήστης ξεκινήσει τη διαδικασία μετρήσεων:

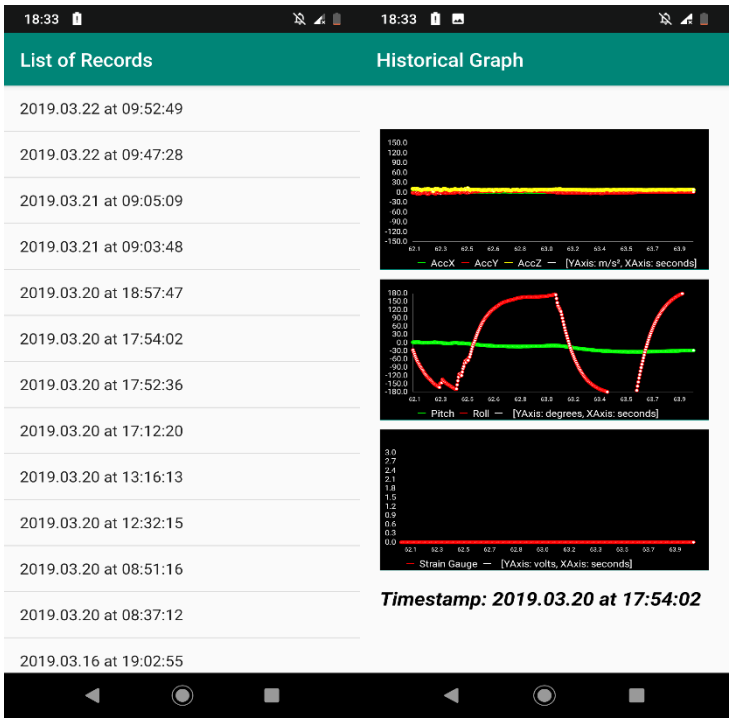
Εικόνα 32: Η οθόνη της εφαρμογής πριν ξεκινήσει η διαδικασία

Στην εικόνα 32 είναι αριθμημένα τα πεδία που απεικονίζουν:

- (1): Το ποσοστό της μπαταρίας της συσκευής (μπάρα)
- (2): Η κατάσταση της σύνδεσης (Connecting ή Connected)
- (3): Το γράφημα που απεικονίζει την επιτάχυνση ανά άξονα σε m/s^2
- (4): Το γράφημα που απεικονίζει τις γωνίες προσανατολισμού σε μοίρες
- (5): Το γράφημα που απεικονίζει την έξοδο του ενισχυτή σε Volts
- (6): Η θερμοκρασία που επιστρέφει το IMU*

***Η θερμοκρασία που επιστρέφει το IMU δεν έχει ακρίβεια και εξαρτάται επίσης από τη θερμοκρασία του PCB**

Μία ακόμα λειτουργία που κρίναμε ότι είναι σημαντική, είναι να μπορεί η εφαρμογή να αποθηκεύει τα γραφήματα και να κρατάει ένα ιστορικό ώστε να μπορεί ο χρήστης εάν το επιθυμήσει να δει παλαιότερες μετρήσεις. Η παραπάνω λειτουργία υλοποιήθηκε και τα δεδομένα αποθηκεύονται στην εσωτερική μνήμη του κινητού τηλεφώνου ή tablet. Έτσι η εφαρμογή διαβάζει τα δεδομένα από το αντίστοιχο αρχείο και δημιουργεί ξανά τα γραφήματα. Ο διαχωρισμός των αρχείων γίνεται με την ημερομηνία δημιουργίας ώστε ο χρήστης να μπορεί να επιλέξει ευκολότερα την επιθυμητή χρονική στιγμή. Στην εικόνα 33, φαίνεται η αριστερά η λίστα με το ιστορικό μετρήσεων όπου ο χρήστης μπορεί να επιλέξει την ημερομηνία/ώρα που επιθυμεί και δεξιά τα γραφήματα που δημιουργήθηκαν από το αντίστοιχο αρχείο:



Εικόνα 33: Η λίστα και τα γραφήματα που δημιουργούνται

5.2 **Android studio**

Το Android Studio είναι το επίσημο περιβάλλον ανάπτυξης για εφαρμογές Android. Πρωτοεμφανίστηκε 16 Μαΐου 2003 σε συνέδριο της Google και είναι βασισμένο στο λογισμικό JetBrains-Intelli J IDEA ενώ είναι κατασκευασμένο σε Java. Επίσης εμπεριέχει το Android SDK το οποίο περιλαμβάνει τις βιβλιοθήκες που χρειάζονται για την ανάπτυξη των εφαρμογών. Το Android Studio προσφέρει διάφορες υπηρεσίες στου χρήστες, κάποιες από αυτές είναι οι εξής:

- Ένα ευέλικτο Cradle-based σύστημα κατασκευής
- Ένα γρήγορο και με πολλές λειτουργίες εξομοιωτή (emulator)
- Δυνατότητα ανάπτυξης εφαρμογών για όλες τις συσκευές Android (Smartphones, Smart watches , Smart TV's)

- Όταν γίνονται αλλαγές στον κώδικα το Android Studio επιτρέπει την άμεση εκτέλεση του χωρίς να δημιουργεί καινούριο εκτελέσιμο APK (Τα αρχεία .apk είναι τα εκτελέσιμα αρχεία εγκατάστασης εφαρμογών Android)
- Περιέχει πρότυπα block κώδικα και επιτρέπει την σύνδεση με το Git Hub προκειμένου να χρησιμοποιηθεί κώδικας που βρίσκεται ανεβασμένος on-line.
- Περιέχει εργαλεία ελέγχου της απόδοσης της χρηστικότητας και έλεγχο συμβατότητας των εφαρμογών
- Υποστήριξη C++
- Υποστήριξή Google Cloud Platform

5.3 Βιβλιοθήκη BluetoothSPPLibrary

Η βιβλιοθήκη "BluetoothSPPLibrary" είναι μία βιβλιοθήκη ανοιχτού κώδικα για Android εφαρμογές που υλοποιούνται στο περιβάλλον του Android Studio. Η βιβλιοθήκη περιέχει όλες τις συναρτήσεις και κλήσεις που χρειαζόμαστε για να υλοποιήσουμε την επικοινωνία μας με βάση το Bluetooth SPP πρωτόκολλο. Το κύριο κριτήριο επιλογής της συγκεκριμένης βιβλιοθήκης ανάμεσα σε αρκετές ανοιχτού κώδικα, είναι η εμπειρία που είχα ήδη μαζί της σε Project προηγούμενων ετών. Για παράδειγμα η βιβλιοθήκη περιέχει όλες τις callback συναρτήσεις που χρειαζόμαστε για να ενημερωνόμαστε για την κατάσταση της σύνδεσης μας, δύο εκ των κυριότερων είναι:

```
@Override
public void onDeviceConnected(BluetoothDevice device)
{
   view.setStatus(R.string.bluetooth_connected);
   view.enableHWButton(true);

   connectionState = true;
}

@Override
public void onDeviceDisconnected(BluetoothDevice device, String
message)
{
   view.setStatus(R.string.bluetooth_connecting);
   view.enableHWButton(false);
   interactor.connectToDevice(device, communicationCallback);
   connectionState = false;
}
```

Αντίστοιχες συναρτήσεις υπάρχουν για να μας ειδοποιούν όταν προκύψει κάποιο σφάλμα στη σύνδεση ή κατά την λήψη πακέτων. Η βιβλιοθήκη μπορεί να βρεθεί στον παρακάτω σύνδεσμο:

https://github.com/akexorcist/Android-BluetoothSPPLibrary

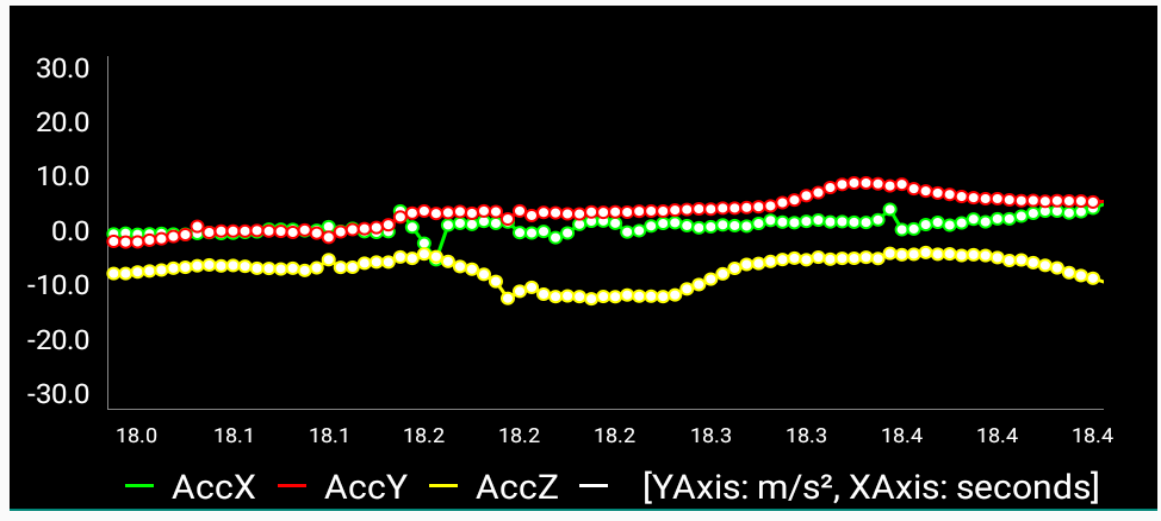
5.4 **Βιβλιοθήκη MPAndroidChart**

Η βιβλιοθήκη "MPAndroidChart" είναι μία βιβλιοθήκη ανοιχτού κώδικα για Android εφαρμογές που υλοποιούνται στο περιβάλλον του Android Studio. Η βιβλιοθήκη χρησιμοποιείται για τη δημιουργία απλών ή σύνθετων γραφημάτων. Οι δυνατότητες/ρυθμίσεις ανά γράφημα είναι αμέτρητες και συνεχώς προστίθενται νέες. Η χρήση της βιβλιοθήκης είναι αρκετά εύκολη για βασικά γραφήματα αλλά περιπλέκεται όταν θέλουμε να απεικονίζουμε δεδομένα σε πραγματικό χρόνο. Το πρώτο πρόβλημα είναι το μέγεθος της μνήμης που απαιτείται καθώς για να μπορεί ο χρήστης να ανατρέψει σε όλο το γράφημα ή να μεγεθύνει συγκεκριμένες περιοχές του γραφήματος, ότι δεδομένο εισάγεται στο γράφημα πρέπει να παραμείνει και στη μνήμη RAM. Αυτό σημαίνει πως εάν έχουμε πολλά δείγματα ανά δευτερόλεπτο, τότε η εφαρμογή μας μετά από λίγες ώρες γίνεται πολύ αργή σε σημείο να μην έχει ουσιαστικό νόημα. Το δεύτερο πρόβλημα είναι ότι οι Android εφαρμογές δεν ενδείκνυται για υλοποιήσεις περιέχουν αυστηρά χρονικά συμβάντα. Αυτό συμβαίνει διότι παράλληλα με την εφαρμογή μας, εκτελούνται διάφορες άλλες λειτουργίες και η μέτρηση χρόνου δεν έχει μεγάλη ακρίβεια. Το πρόβλημα μπορεί να μετριαστεί εάν το χρονισμό των γραφημάτων τον αναλάβει η συσκευή, δηλαδή αυτό που επιλέξαμε εμείς. Φυσικά και πάλι δε μπορούμε να μιλάμε για μεγάλη χρονική ακρίβεια, αφού ανάμεσα στη συσκευή και την εφαρμογή υπάρχουν παράγοντες που δε μπορούμε να ελέγξουμε όπως το Latency του Bluetooth.

Για την απεικόνιση σε πραγματικό χρόνο, κάθε πακέτο που έρχεται από τη συσκευή περιέχει συνολικά 50 μετρήσεις, μία μέτρηση δηλαδή ανά 5msec. Αυτό ουσιαστικά σημαίνει πως εισάγουμε στο γράφημα 50 τιμές κάθε 250msec ανά γραμμή. Ορίζοντας το γράφημα να εμφανίζει πάντα τα **τελευταία 400 δείγματα**, δημιουργούμε μία βάση χρόνου των 2 δευτερολέπτων. Η συνάρτηση που αναφέρουμε είναι η παρακάτω για το γράφημα των επιταχύνσεων:

```
AccChart.setVisibleXRangeMaximum(400);
```

Φυσικά ο χρήστης μόλις σταματήσει τη διαδικασία των μετρήσεων, μπορεί να μικρύνει τη βάση χρόνου ή να ανατρέψει σε παλαιότερα δείγματα. Στη παρακάτω εικόνα φαίνεται το γράφημα μετά από μεγέθυνση που επέβαλε ο χρήστης.



Εικόνα 34: Το γράφημα των επιταχύνσεων μετά από Zoom

Βλέπουμε ότι κάποιες τιμές στον άξονα του χρόνου επαναλαμβάνονται. Το ελάχιστο βήμα που θα μπορούσαμε να απεικονίσουμε είναι 5msec αλλά το πρόβλημα προκύπτει από την αδυναμία της βιβλιοθήκης να διαχωρίσει σωστά τη βάση χρόνου, **έτσι επιβάλλουμε να εμφανίζεται πάντα μόνο το πρώτο δεκαδικό**. Σύμφωνα με τους δημιουργούς της βιβλιοθήκης, είναι ένα από τα προβλήματα που θα διορθωθούν σε μελλοντική αναβάθμιση. Η βιβλιοθήκη μπορεί να βρεθεί στον παρακάτω σύνδεσμο:

https://github.com/PhilJay/MPAndroidChart

6. Desktop εφαρμογή

Στο κεφάλαιο αυτό, θα γίνει περιγραφή της Desktop εφαρμογής όπως και της λογικής που ακολουθήθηκε. Επίσης, θα γίνει αναφορά στα εργαλεία που χρησιμοποιήθηκαν για τη συγγραφή του κώδικα. Η γλώσσα προγραμματισμού που χρησιμοποιήθηκε είναι η C#.

6.1 Λειτουργία εφαρμογής

Οι κύριες λειτουργίες που εκτελεί η Desktop εφαρμογή είναι η χρήση του USB πρωτοκόλλου που αναλύσαμε σε προηγούμενο κεφάλαιο για την ανάγνωση και την αλλαγή των παραμέτρων της συσκευής. Η χρήστης απλά επιλέγει τη θύρα που είναι συνδεδεμένη η συσκευή και η εφαρμογή είναι σε θέση να διαβάσει ή να αλλάξει τις παραμέτρους εάν το επιθυμήσει ο χρήσης.

Εικόνα 35: Το περιβάλλον της Desktop εφαρμογής

6.2 MS Visual studio 2017

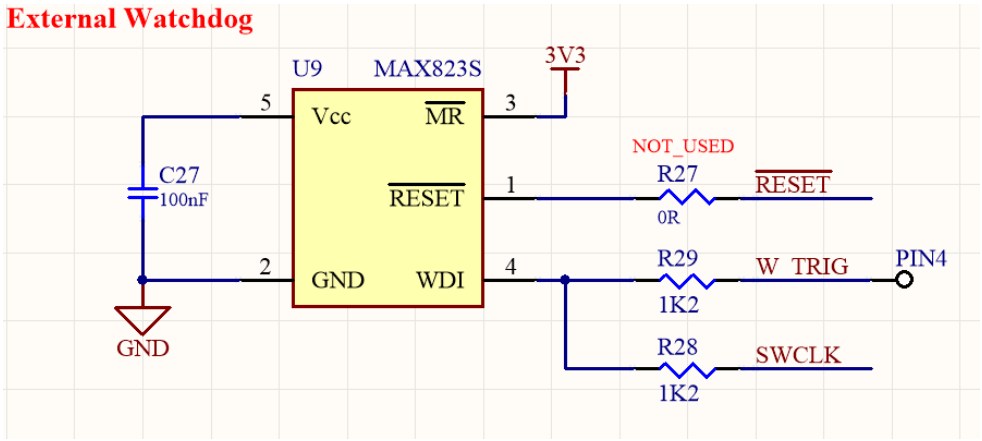
Το Visual Studio 2017 της Microsoft είναι ένα περιβάλλον ανάπτυξης και χρησιμοποιείται για την ανάπτυξη προγραμμάτων ηλεκτρονικών υπολογιστών για τα Microsoft Windows. Επίσης χρησιμοποιεί πλατφόρμες ανάπτυξης λογισμικού της Microsoft, όπως τα Windows API, Windows Forms, Windows Presentation Foundation, το Windows Store και το Microsoft Silverlight. Το Visual Studio περιλαμβάνει έναν code editor και ακόμη το ολοκληρωμένο πρόγραμμα εντοπισμού σφαλμάτων. Το Visual studio υποστηρίζει διάφορες γλώσσες προγραμματισμού και επιτρέπει να επεξεργαστεί το κώδικα ώστε να εντοπίζει τα λάθη του. Κάποιες από τις γλώσσες προγραμματισμού είναι οι Visual Basic, C# ,F#,XML, XAML, Python, Html, Css και Javascript.

7. Σχηματικά και πλακέτα-PCB

Στο κεφάλαιο αυτό, θα γίνει περιγραφή της διαδικασίας σχεδιασμού των σχηματικών και των χαρακτηριστικών της πλακέτας-PCB, καθώς και του εργαλείου που επιλέχθηκε για το σχεδιασμό αυτό. Το πρόγραμμα που χρησιμοποιήσαμε είναι το Altium Designer 2018 και ο κύριος λόγος που χρησιμοποιήθηκε είναι η εμπειρία που προϋπήρχε με το συγκεκριμένο πρόγραμμα, καθώς και η δυνατότητα απεικόνισης σε τρισδιάστατο επίπεδο.

7.1 Σχηματικά

Πριν ξεκινήσει ο σχεδιασμός των σχηματικών, είχαμε ήδη αποφασίσει για τα υλικά που θα χρησιμοποιούσαμε και τα κυκλώματα που θα υλοποιούσαμε. Τα περισσότερα υλικά που χρησιμοποιήθηκαν υπήρχαν ήδη στη προσωπική μου βιβλιοθήκη, το οποίο σημαίνει πως η πιθανότητα για λάθος σχεδίαση υλικού μειώθηκε αφού έχουν ξανά χρησιμοποιηθεί και διορθωθεί κατά το παρελθόν. Η σημείωση "**NOT USED**" δίπλα από υλικό σημαίνει πως το υλικό υπάρχει στη πλακέτα αλλά δεν έχει τοποθετηθεί. Αυτό μας επιτρέπει να προβλέπουμε προβλήματα που θα μπορούσαν να προκύψουν στη πλακέτα. Για παράδειγμα ο εξωτερικός Watchdog και η αντίσταση R27, εάν δεν υπήρχε η αντίσταση R27 ώστε να μπορούμε να αποτρέψουμε τον Watchdog να κάνει reset τον μικροελεγκτή μας, τότε δε θα μπορούσαμε να κάνομε Debug βήμα προς βήμα ή θα έπρεπε να κολλάμε-ξεκολλάμε το ολοκληρωμένο. Εφόσον τελειώσουμε την ανάπτυξη του Firmware, τότε μπορούμε να τοποθετήσουμε την αντίσταση R27.



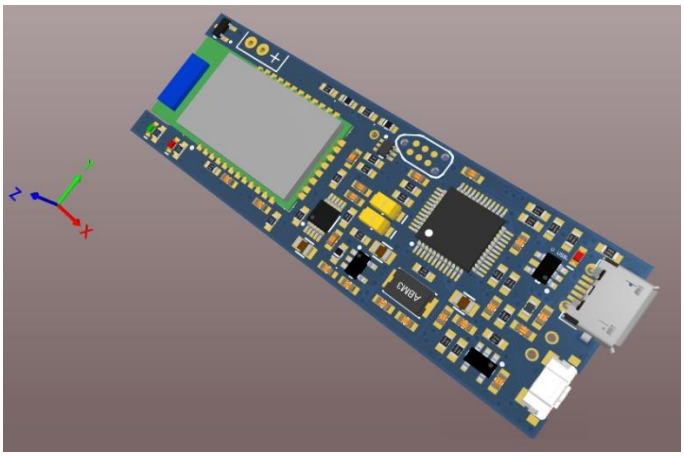
Εικόνα 36: Ο εξωτερικός Watchdog και η αντίσταση R27

7.2 Χαρακτηριστικά πλακέτας

Αξίζει να σημειωθούν τα βασικά χαρακτηριστικά της πλακέτας και ο αριθμός των συνολικών υλικών που τοποθετήθηκαν:

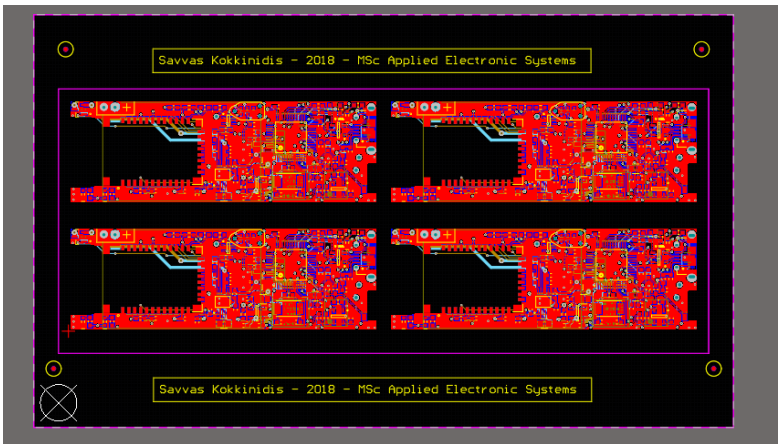
- **Διαστάσεις πλακέτας:** 62.5mm X 21.1mm
- **Αριθμός στρωμάτων (Layers):** 4
- **Ελάχιστη διάμετρος οπής Via:** 0.3mm
- **Αριθμός υλικών:** 96
- **Πλευρά τοποθέτησης υλικών:** Top/Bottom

Όπως αναφέραμε και παραπάνω, η απεικόνιση σε τρισδιάστατο επίπεδο είναι ένα χαρακτηριστικό του Altium Designer που μας βοηθάει κατά τη τοποθέτηση των υλικών και αποτρέπει λάθη. Στην εικόνα 37 φαίνεται το πως εμφανίζεται η πλακέτας σε τρισδιάστατο επίπεδο.



Εικόνα 37: Η πλακέτα σε 3D επίπεδο

Τέλος, πρέπει να σημειωθεί πως ο αριθμός των υλικών και το είδος του κελύφους ορισμένων υλικών, απαιτεί τη χρήση μηχανής Pick and Place. Για την σωστή και γρήγορη ρύθμιση της μηχανής, έπρεπε να δημιουργήσουμε ένα Panel που να περιέχει τη πλακέτα μας συνολικά 4 φορές (2x2). Στην εικόνα 38 φαίνεται αυτό που περιγράφουμε, όπως επίσης και τα Fiducial περιμετρικά του Panel για την ευθυγράμμιση της πλακέτας.



Εικόνα 38: Το Panel για τη τοποθέτηση υλικών με μηχανή Pick and Place

7.3 Altium Designer

Το Altium Designer είναι ένα πρόγραμμα για σχεδιασμό ηλεκτρονικών συστημάτων (EDA). Ξεκίνησε το 2004 (παλαιότερα είχε την ονομασία Protel) και η τελευταία έκδοση είναι η 2019. Καλύπτει όλα τα επίπεδα μιας ηλεκτρονικής κατασκευής όπως:

- Σχεδιασμός και καταγραφή μπροστινού και πίσω μέρους πλακέτας.
- Κατασκευή και φυσικός σχεδιασμός PCB
- Σχεδιασμός Hardware FPGA
- Εφαρμογή συστήματος FPGA και λογισμικό εντοπισμού σφαλμάτων.
- Ενσωματωμένο λογισμικό ανάπτυξης.
- Προσομοίωση κυκλώματος μικτού σήματος.
- Ανάλυση κατασκευαστικής ακεραιότητας σήματος.
- Αναπτυγμένο σύστημα απεικόνισης 3D

8. Συμπεράσματα, σχόλια και προτάσεις για μελλοντική εξέλιξη

Το βασικό συμπέρασμα της εργασίας είναι ότι η απεικόνιση δεδομένων αθλητών σε πραγματικό χρόνο μπορεί να επιτευχθεί με σχεδιασμό μίας απλής συσκευής που θα μπορεί να αποστέλλει τα δεδομένα αυτά σε μία Android εφαρμογή μέσω Bluetooth. Βέβαια υπάρχουν προτάσεις για μελλοντική εξέλιξη που θα μπορούσαν να αυξήσουν τις λειτουργίες της συσκευής όπως:

- Υπολογισμός ταχύτητας
- Υπολογισμός απόστασης
- Αλλαγή του τρόπου αποθήκευσης των δεδομένων στην εφαρμογή
- Αλγόριθμος αναγνώρισης κινήσεων αθλητή σε πραγματικό χρόνο

9. Βιβλιογραφία και Αναφορές

[1] Microcontroller, https://en.wikipedia.org/wiki/Microcontroller
[2] The Insider's Guide To The STM32 ARM Based Microcontroller An Engineer's Introduction To The STM32 Series, Version 1.8, Hitex(UK) Ltd., 2010
[3] STM32F103C8, https://www.stmicroelectronics.com.cn/en/microcontrollers-microprocessors/stm32f103c8.html
[4] Bluetooth, https://www.bluetooth.com/
[5] Marian K. Kazimierczuk, Pulse-width Modulated DC-DC Power Converters, John Wiley & Sons, Ltd (2008)
[6] Lithium battery, https://en.wikipedia.org/wiki/Lithium_battery
[7] Ning Jia, "Detecting Human Falls with a 3-Axis Digital Accelerometer", Analog Devices, Rev. 0, 2009
[8] Inertial measurement unit, https://en.wikipedia.org/wiki/Inertial_measurement_unit
[9] S.Ben Salem, M. Fakhfakh, D. Sellami Masmoudi, M. Loulou, P. Loumeu, and N. Masmoudi, " A high performances CMOS CCII and high frequency applications," Analog Integrated Circuits and Signal Processing, vol. 49, 2006.
[10] Two Flash Technologies Compared: NOR vs NAND, https://www.csd.uoc.gr/~hy428/reading/M-Systems_NANDvsNOR.pdf
[11] N. Assimakis, M. Adam, Steady State Kalman Filter for Periodic Models: A New Approach, Int. J. Contemp. Math. Sciences, Vol. 4
[12] AN5023 Sensor Fusion Kalman Filters, https://www.nxp.com/docs/en/application-note/AN5023.pdf

10.Παράρτημα Α – Εργαλεία και Μετρήσεις

10.1 Εργαλεία που χρησιμοποιήθηκαν

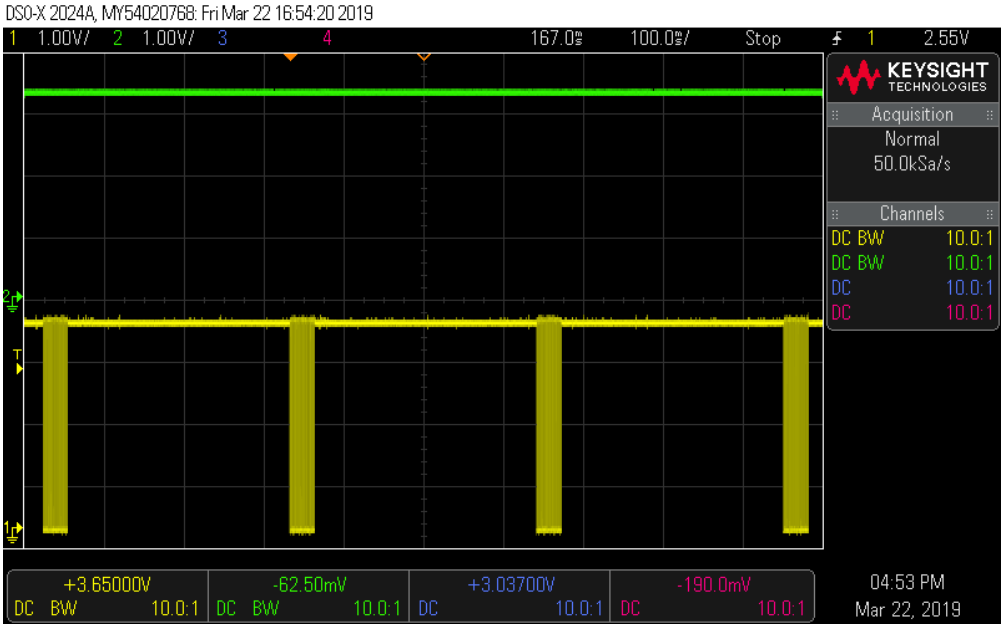
Όλα τα εργαλεία που χρησιμοποιήθηκαν για την ανάπτυξη, υλοποίηση, κατασκευή και διάφορες μετρήσεις, αναγράφονται στον πίνακα 7:

Software
IAR Embedded Workbench ARM 8.32v STM32CubeMC 4.27v Android studio 3.2.1v Microsoft Visual Studio 2017 15.9.6v
Hardware
Παλμογράφος: Keysight DSO-X 2024A **Πολύμετρο:** UNI-T UT61D **Πολύμετρο:** Agilent U1272A **Λογικός αναλυτής:** Saleae Logic 8 **Προγραμματιστής:** ST-LINK/V2

Πίνακας 7: Πίνακας εργαλείων που χρησιμοποιήθηκαν

10.2 Τάση τροφοδοσίας κατά την αποστολή πακέτων

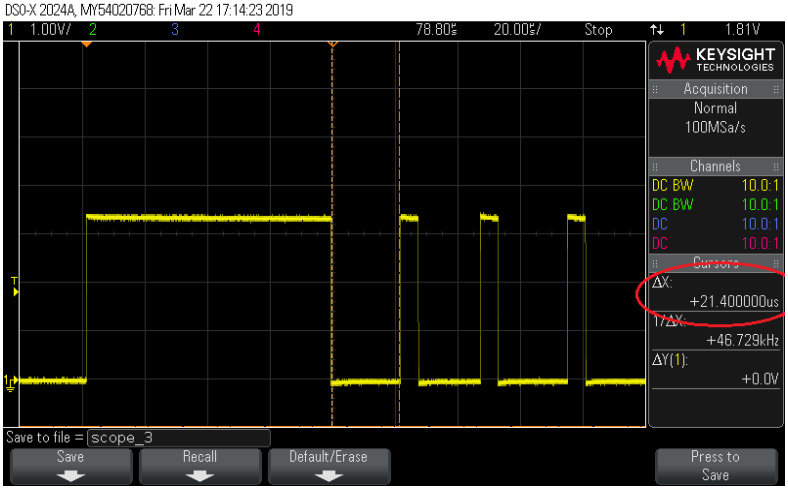
Όπως φαίνεται στην εικόνα 39, η τάση της εξόδου του DC converter είναι σταθερή και χωρίς θόρυβο/βυθίσεις κατά την αποστολή πακέτων μέσω Bluetooth όπου το ρεύμα είναι και το μέγιστο.



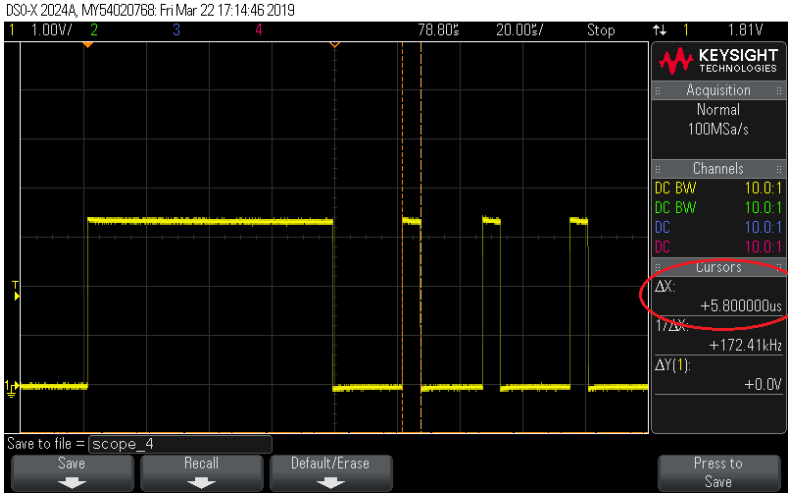
Εικόνα 39: Η τάση εξόδου του DC converter (πράσινο χρώμα) κατά τη διάρκεια της αποστολής πακέτων (με κίτρινο χρόνο είναι τα σήματα στο TX pin)

10.3 Σήματα προγραμματισμού AD8555

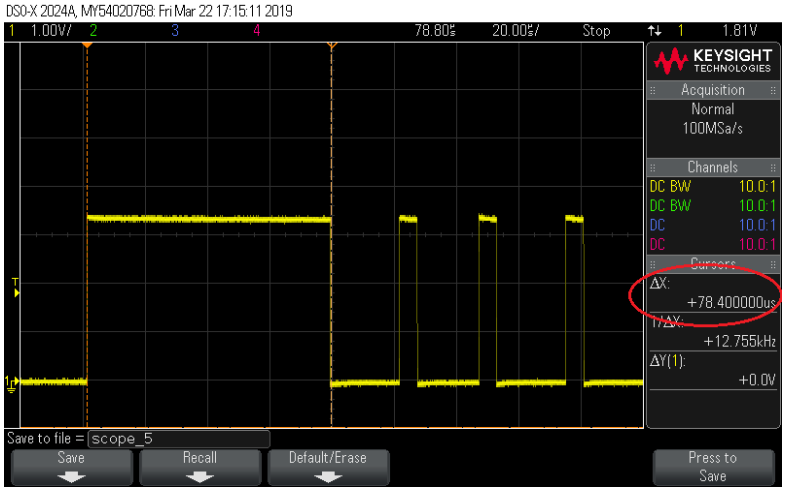
Στις εικόνες 40,41 και 42, φαίνονται τα σήματα που αποστέλλονται από τον μικροελεγκτή προς το ολοκληρωμένο AD8555.



Εικόνα 40: Διάρκεια παλμού χαμηλής στάθμης ανάμεσα στα bits



Εικόνα 41: Διάρκεια παλμού για αποστολή '0' bit



Εικόνα 42: Διάρκεια παλμού για αποστολή '1' bit

10.4 Μέση και μέγιστη τιμή ρεύματος κατά την αποστολή

Στις εικόνες 43 και 44, φαίνεται η μέγιστη και η μέση τιμή του ρεύματος που καταναλώνει η συσκευή από την μπαταρία κατά την αποστολή πακέτων. Οι μετρήσεις έγιναν με τάση μπαταρίας 4.2V.

Εικόνα 43: Μέγιστο ρεύμα 101mA (Peak current) κατά την αποστολή πακέτων

Εικόνα 44: Μέση τιμή ρεύματος 73mA (Peak current) κατά την αποστολή πακέτων

10.5 Κυματομορφή τάσης μπαταρίας κατά την εκφόρτιση

Εικόνα 45: Η τάση της μπαταρίας (άξονας Y) σε Volts κατά την εκφόρτιση. Στον άξονα X, εμφανίζεται ο αύξων αριθμός των δειγμάτων (μέτρηση ανά 5 δευτερόλεπτα).

10.6 Κυματομορφή ρεύματος φόρτισης μπαταρίας

Εικόνα 46: Το ρεύμα φόρτισης της μπαταρίας (άξονας Y) σε mA. Στον άξονα X, εμφανίζεται ο αύξων αριθμός των δειγμάτων (μέτρηση ανά 5 δευτερόλεπτα).

11. Παράρτημα Β – Λίστα υλικών/BOM

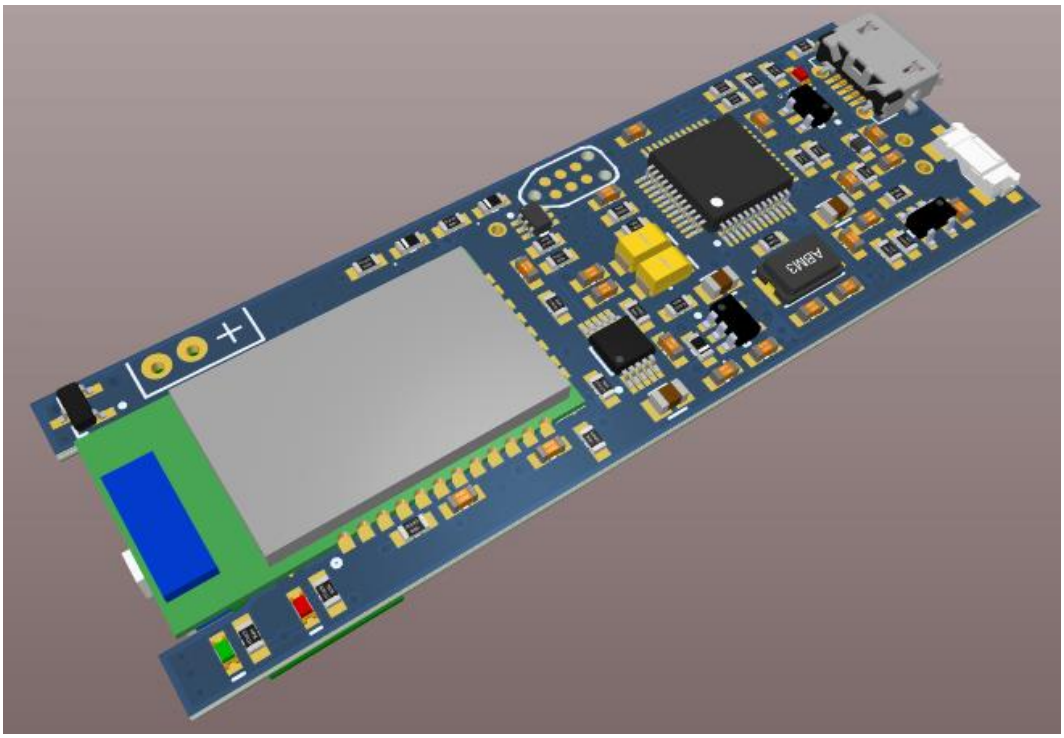
Το κόστος για την κατασκευή της πρώτης πλακέτας, ανέρχεται στα **169 Euro**. Για τις επόμενες 50 πλακέτες (σε μαζική παραγγελία), το κόστος ανά πλακέτα ανέρχεται σε **57,4 Euro**. Να σημειωθεί πώς το Stencil αγοράζεται μόνο τη πρώτη φορά και μπορεί ξαναχρησιμοποιηθεί.

Designator	Comment	Part Number	Quantity	Cost per 1 PCB (Euro)	Cost per 50 PCB (Euro)
PTC1	400mA	0ZCF0075AF2C	1	0,49	0,41
BAT1	BATTERY LITHIUM 3.7V 150mAh	LP701522	1	6,1	4,5
BT1		EVQ-P42B3M	1	0,9	0,75
C1, C2, C6, C12, C14, C19, C21, C22, C23, C25, C26, C27, C28, C29, C30, C31	100nF	CL10B104KB8NNNC	16	0,07	0,02
C3	47uF	TLJR476M010R3200	1	1,7	1
C4, C7, C11	10nF	CL10B103KB8NNNC	3	0,2	0,04
C5	1uF	CL10B104KB8NNNC	1	0,1	0,04
C8, C9	18pF		2	0,1	0,04
C10	4.7nF		1	0,1	0,04
C13	10uF		1	0,1	0,04
C15, C20	100uF/6.3V	TCJA107M006R0150	2	0,75	0,38
C16, C17	4.7uF	JMK107BJ475KA-T	2	0,12	0,04
C18	1.5nF		1	0,2	0,05
CN1	Micro USB	629105150921	1	1,8	1,5
D1, D2, D3	CDBU0530	CDBU0530	3	0,45	0,12
L2, L3	4.7uH	LQM2MPN4R7MG0L	2	0,35	0,22
LED2	RED LED	SML-D12U8WT86C	1	0,3	0,08
LED3	GREEN LED	SML-D13FWT86C	1	0,3	0,08
Q1	SI2333CDS	SI2333CDS-T1-E3	1	0,5	0,35
R1, R3	1M		2	0,5	0,1
R2	100K		1	0,5	0,1
R4, R11	1K		2	0,5	0,1
R5, R6	22R		2	0,5	0,1
R7, R28, R29	1K2		3	0,5	0,1
R8	2K2		1	0,5	0,1
R9, R17, R20, R21, R23, R24, R26, R30, R32, R33, R35	10K		11	0,5	0,05
R12, R15	15K		2	0,5	0,1
R13	30K1		1	0,4	0,1
R14	340K		1	0,5	0,1

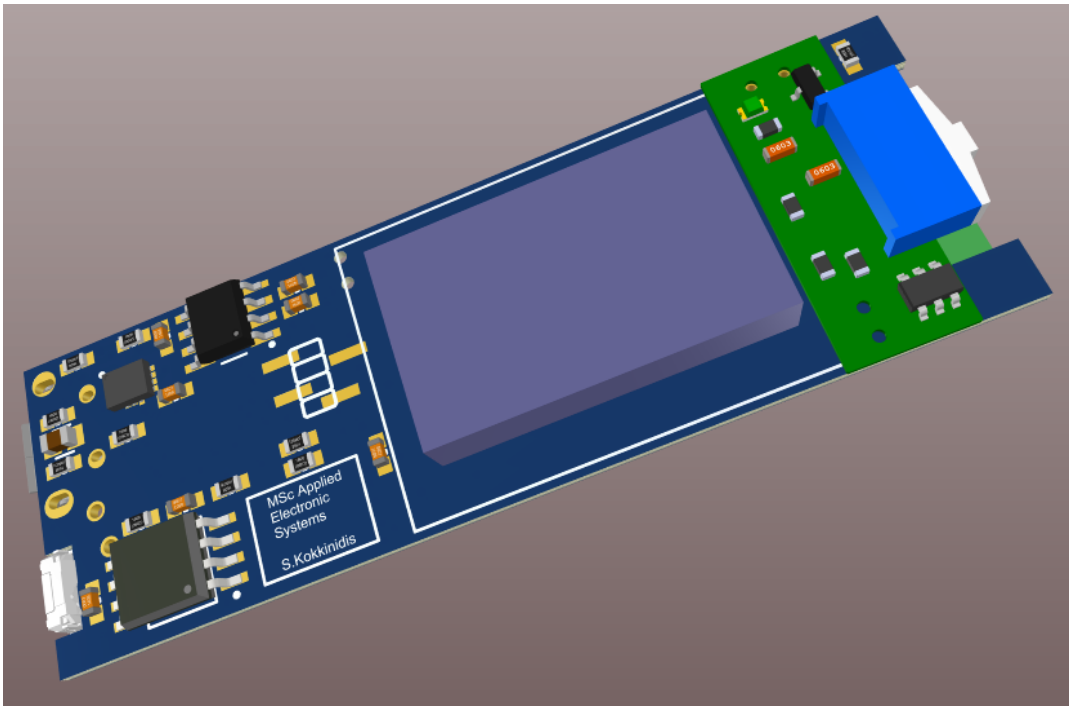
R16	200K		1	0,5	0,1
R27, R34, R36	0R		3	0,5	0,1
R31	4K7		1	0,5	0,1
U1	RN41	RN41-I	1	20	17,5
U2	STM32F103C8T6	STM32F103C8T6	1	5,28	4,1
U3	USBLC6	USBLC6-2SC6	1	0,4	0,2
U4	MCP73831	MCP73831T-2ACI/OT	1	0,5	0,4
U5	LTC3440	LTC3440EMS#TRPBF	1	5,9	5,2
U6	LSM6DS3USTR	LSM6DS3USTR	1	2,5	1,8
U7	AT45DB641E	AT45DB641E	1	3,8	3
U8	AD8555	AD8555ARZ	1	6,8	5,4
U9	MAX823S	MAX823SEXK+T	1	3,8	2,75
U10	SMF05CT	SMF05CT1G	1	0,42	0,23
XT1	8MHz	ABM3-8.000MHZ-D2Y-T	1	0,96	0,76
PCB + Μεταφορικά			1	60	2,5
Stencil + Μεταφορικά			1	25	0
				169,06	**57,39**

Πίνακας 8: BOM

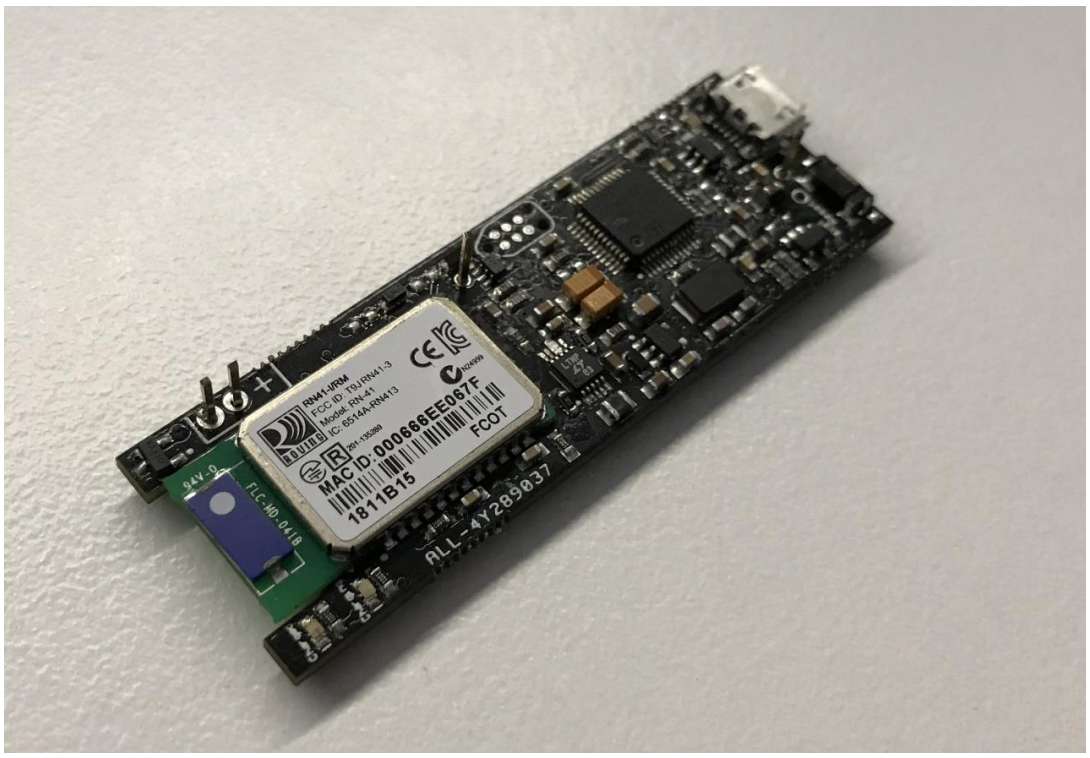
12. Παράρτημα Γ – Φωτογραφίες συσκευής και πλακέτας



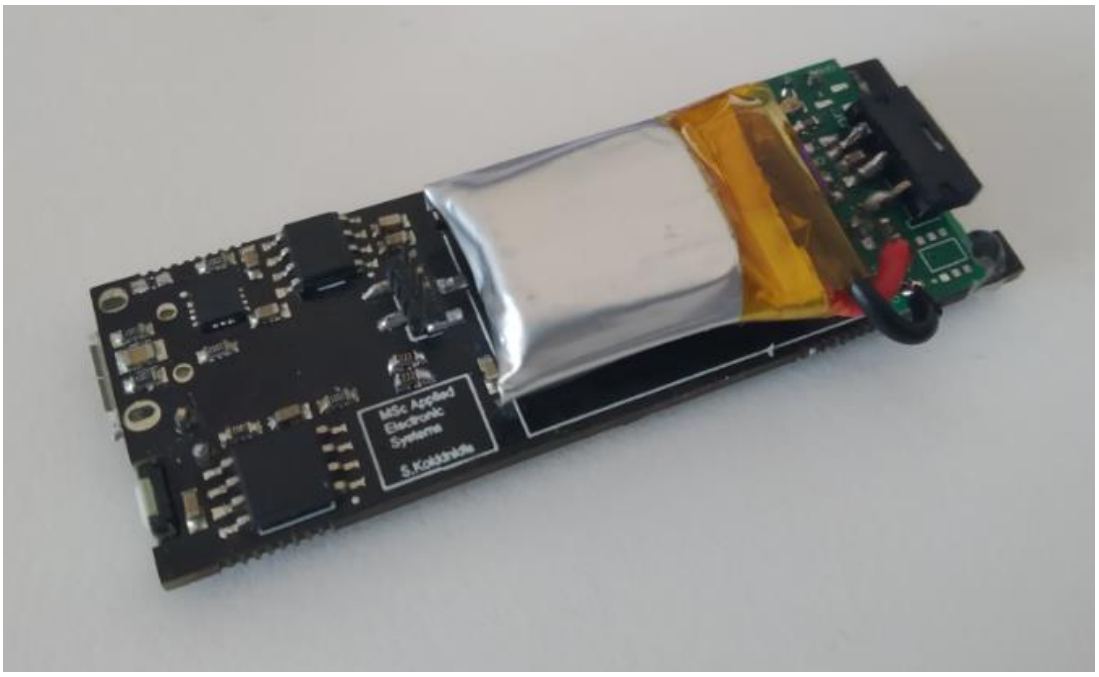
Εικόνα 45: 3D απεικόνιση Top Layer της συσκευής



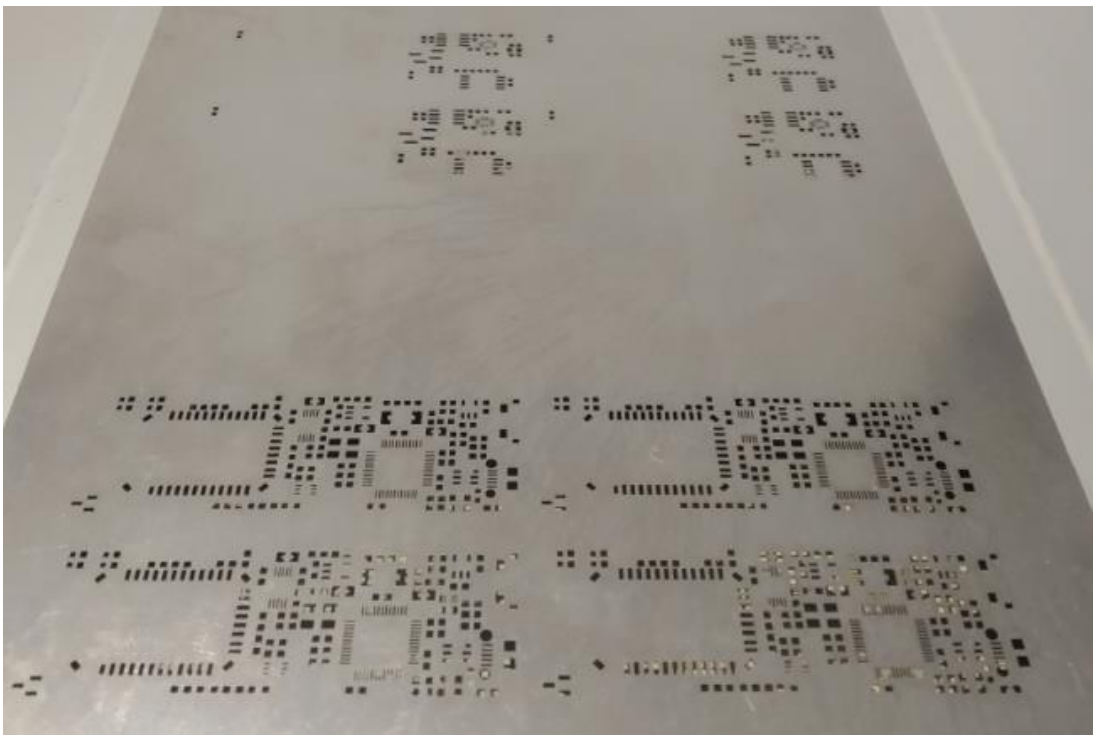
Εικόνα 46: 3D απεικόνιση Bottom Layer της συσκευής



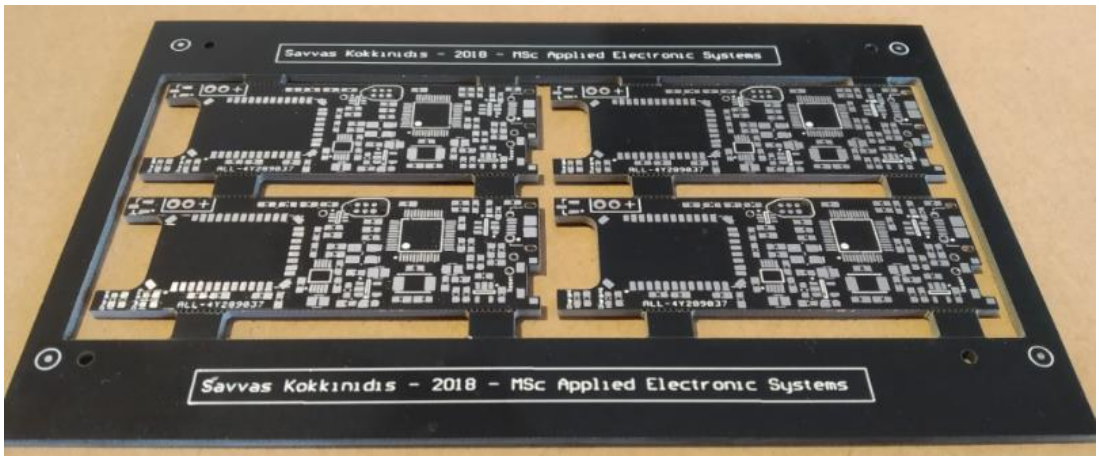
Εικόνα 47: Η πραγματική εικόνα του Top Layer της συσκευής



Εικόνα 48: Η πραγματική εικόνα του Bottom Layer της συσκευής

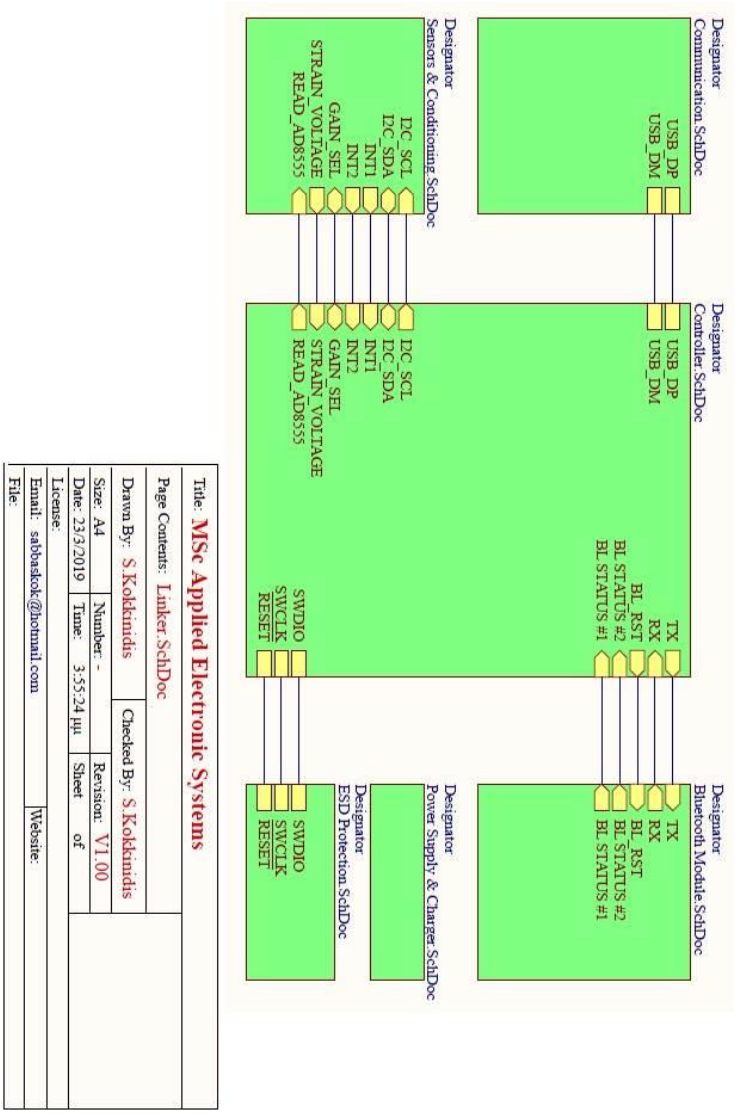


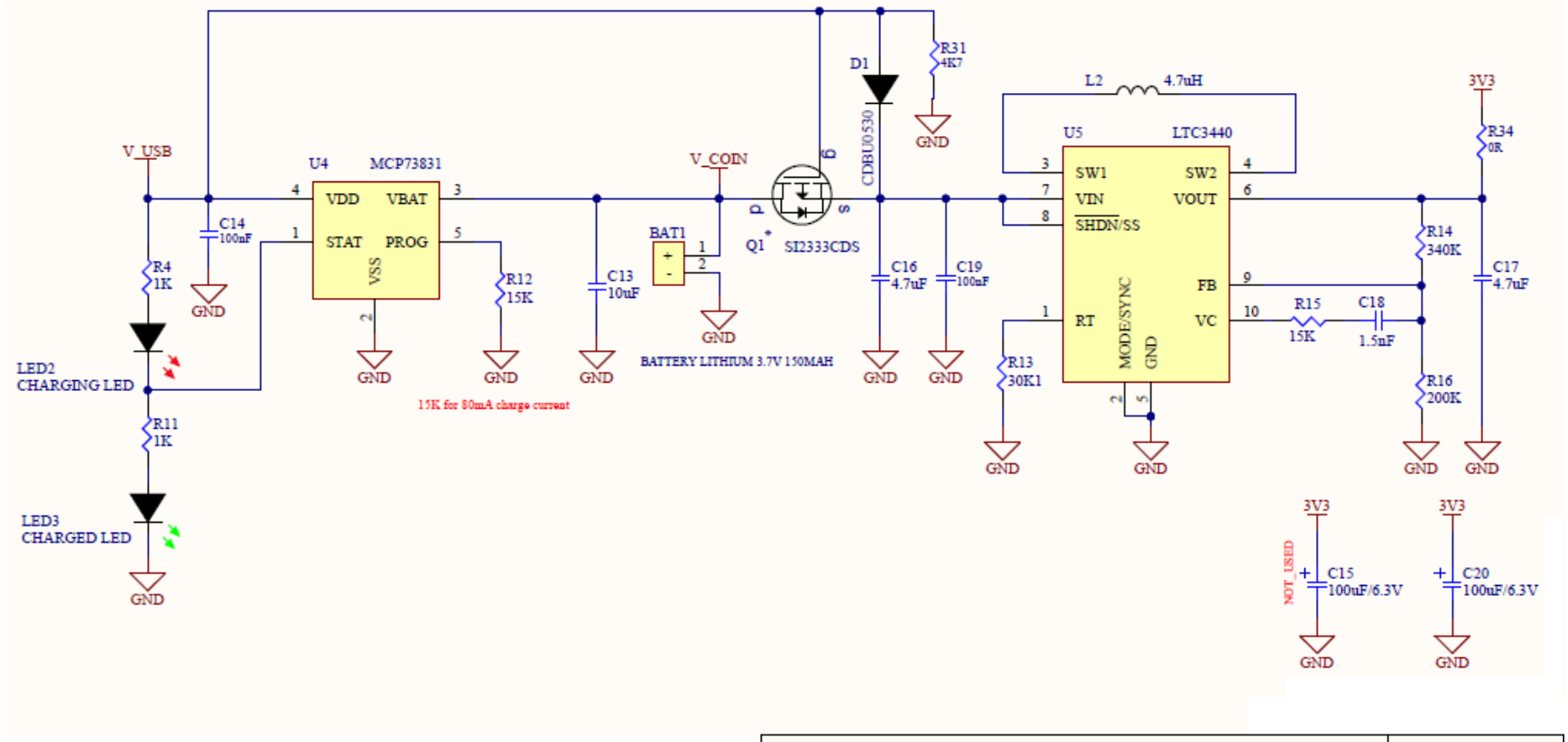
Εικόνα 49: Το stencil που χρησιμοποιήθηκε για την τοποθέτηση της πάστας στο Top και Bottom Layer του πάνελ



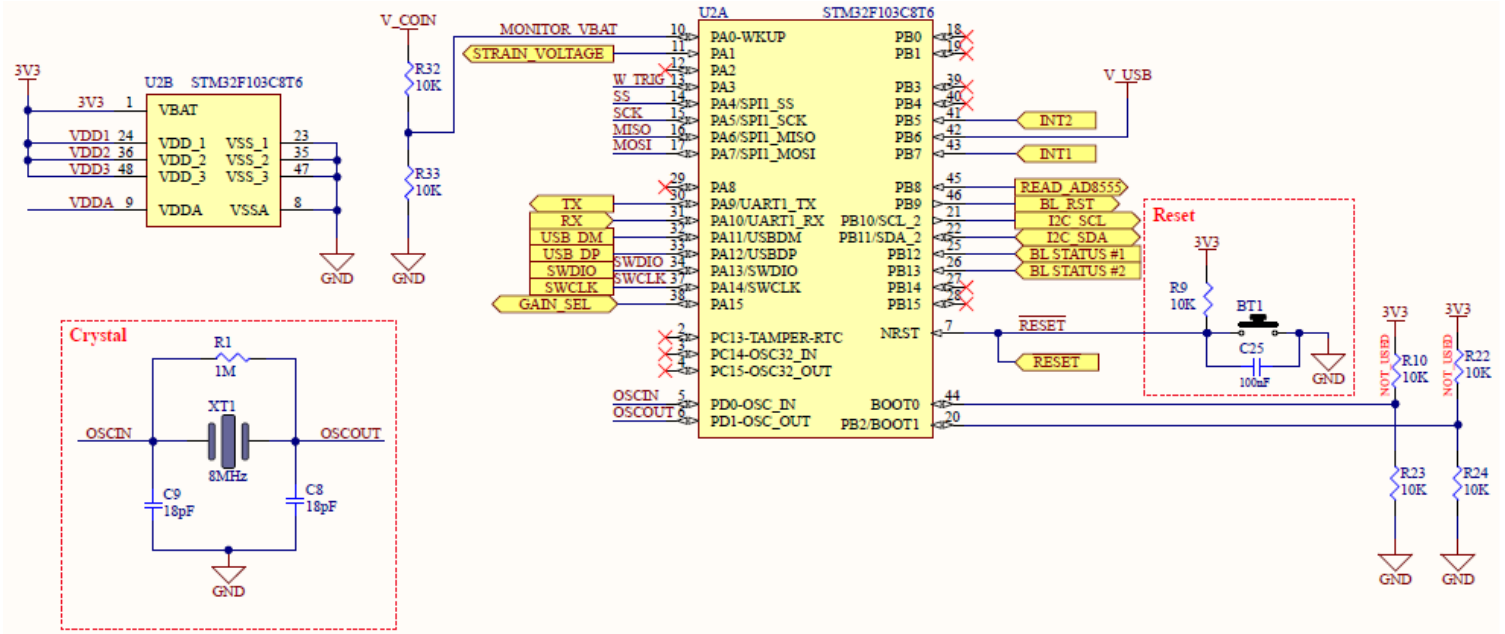
Εικόνα 50: Το Top Layer του πάνελ

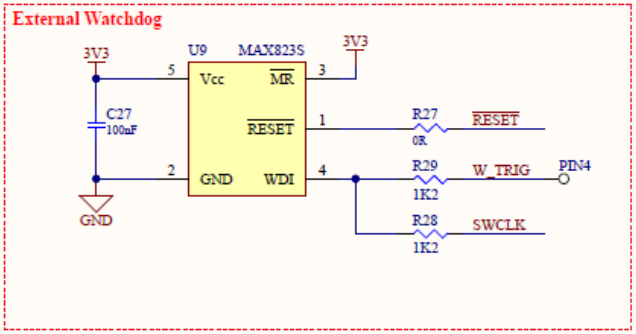
13. Παράρτημα Δ – Σχηματικά και Layout κύριας πλακέτας

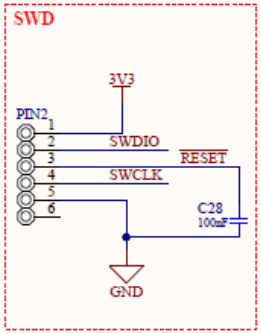


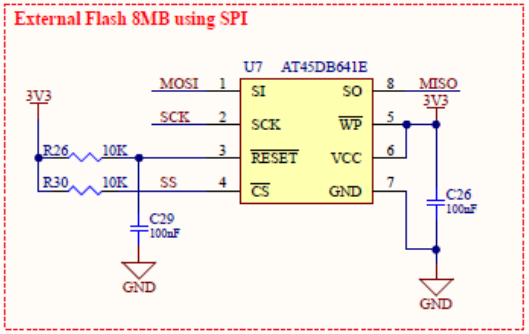


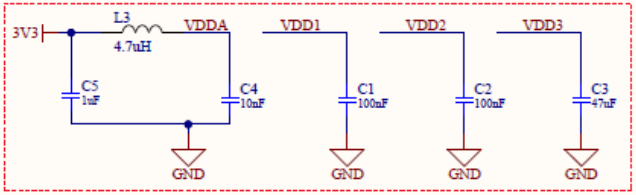
Title: **MSc Applied Electronic Systems**			
Page Contents: Power Supply & Charger.SchDoc			
Drawn By: S.Kokkinidis		Checked By: S.Kokkinidis	
Size: A4	Number: -	Revision: V1.00	
Date: 23/3/2019	Time: 3:55:23 μμ	Sheet of	
License:			
Email: sabbaskok@hotmail.com		Website:	
File:			





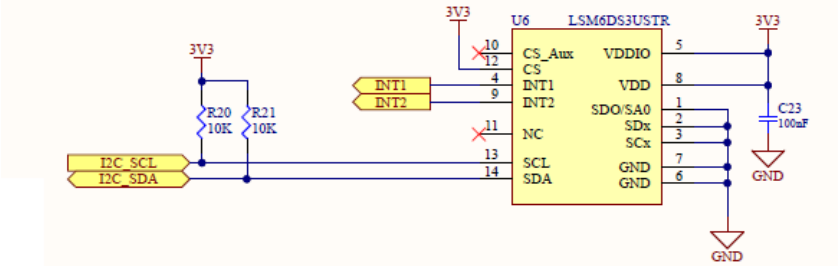




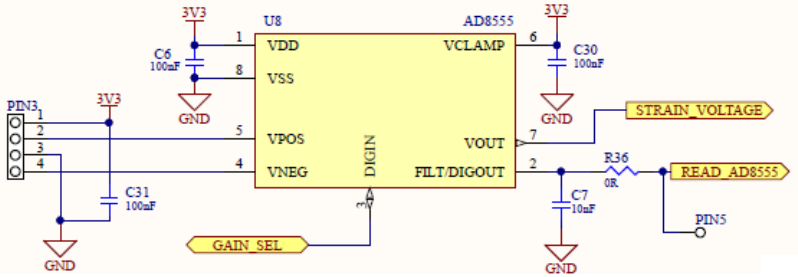


Title: **Smart Tennis Racquet**		
Page Contents: Controller.SchDoc		
Drawn By: S.Kokkinidis	Checked By: S.Kokkinidis	
Size: A4	Number: -	Revision: V1.00
Date: 30/3/2019	Time: 8:17:04 μμ	Sheet 2 of
License:		
Email: sabbaskok@hotmail.com	Website:	
File:		

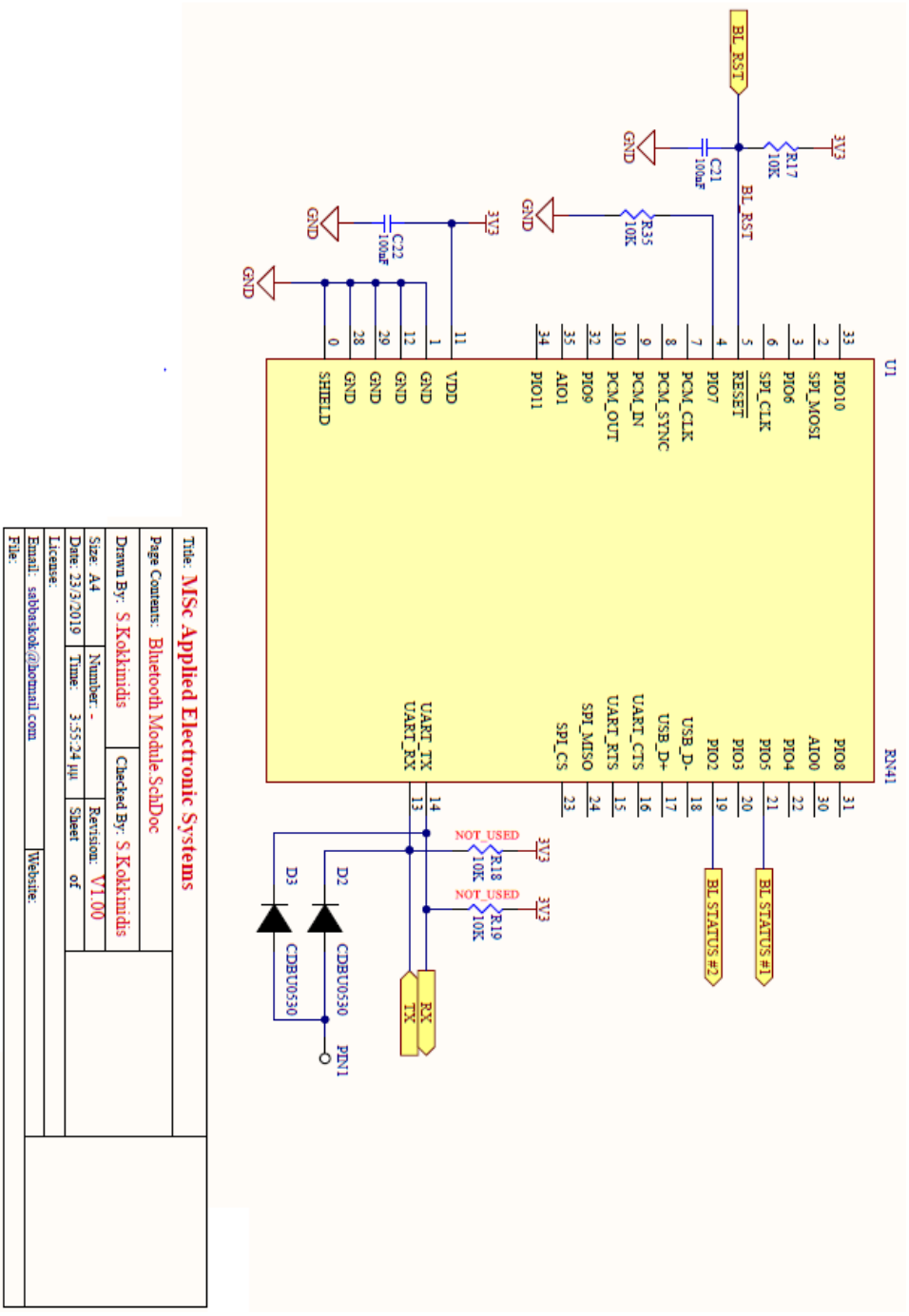
Accelerometer



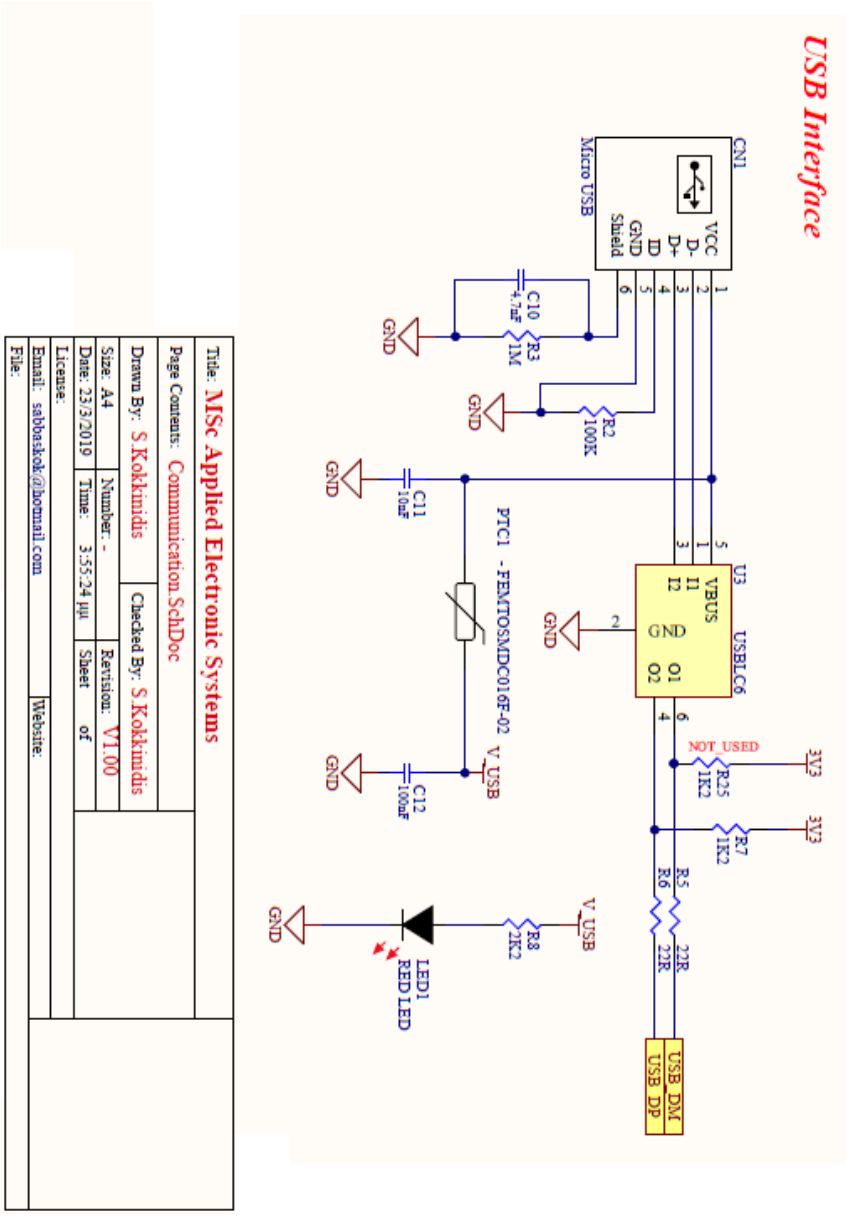
Strain Gauge Sensor & Conditioning



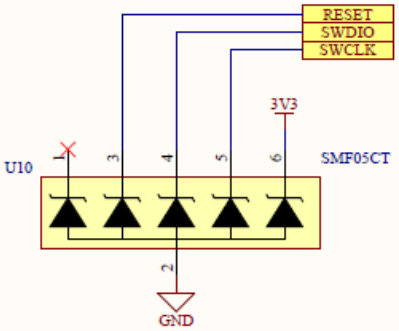
Title: **MSc Applied Electronic Systems**			
Page Contents: Sensors & Conditioning.SchDoc			
Drawn By: S.Kokkinidis		Checked By: S.Kokkinidis	
Size: A4	Number: -	Revision: V1.00	
Date: 23/3/2019	Time: 3:55:23 μμ	Sheet of	
License:			
Email: sabbaskok@hotmail.com		Website:	
File:			



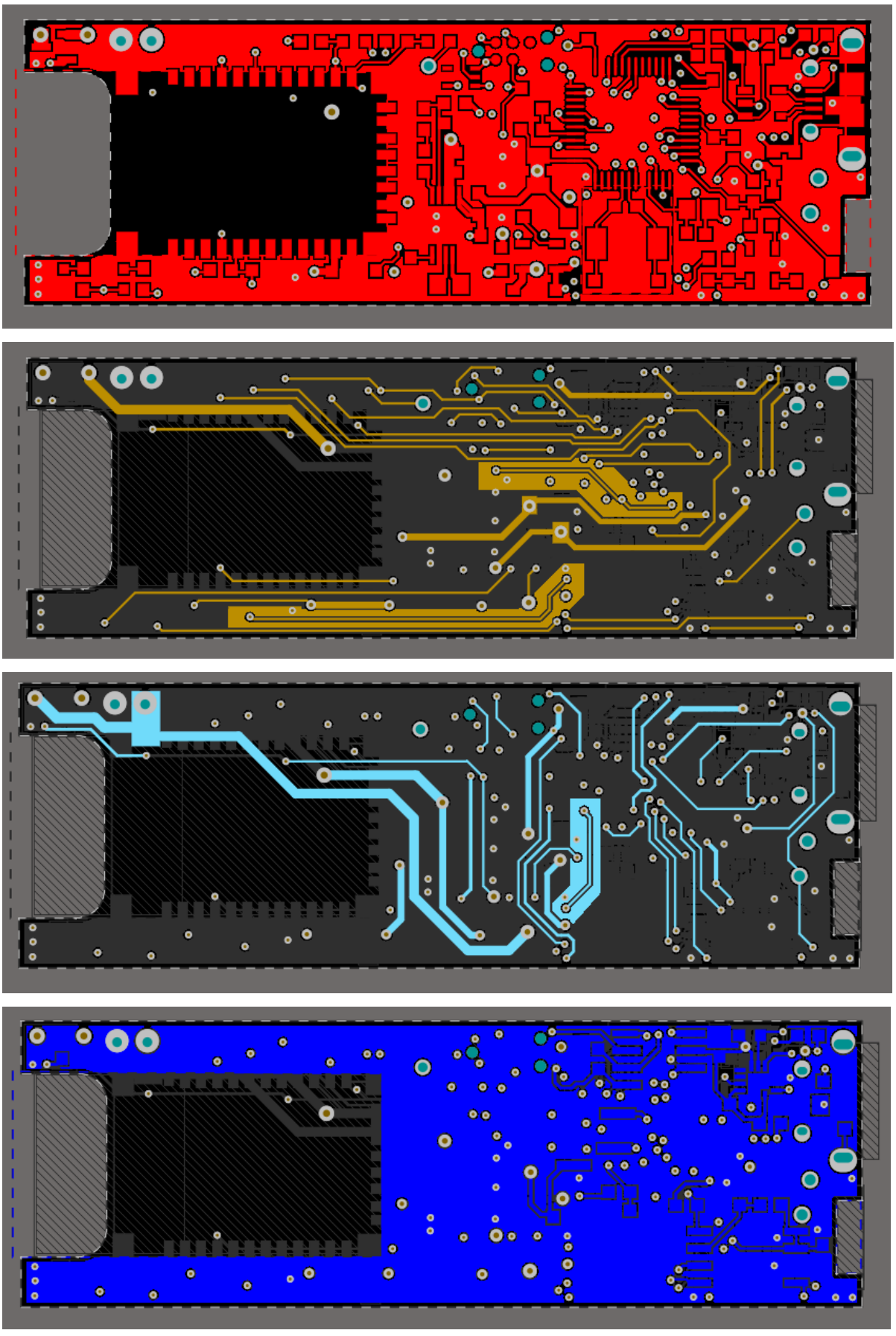
Title: MSc Applied Electronic Systems	
Page Contents: Bluetooth Module.SchDoc	
Drawn By: S.Kokkinidis	Checked By: S.Kokkinidis
Size: A4	Number: -
Date: 23/3/2019	Time: 3:55:24 μμ
License:	
Email: sabbaskok@hotmail.com	Website:
File:	


USB Interface
CN1
Micro USB
VCC
D-
D+
ID
GND
Shield
C10
4.7nF
R3
1M
R2
100K
C11
10nF
C12
100nF
U3
USBLC6
VBUS
I1
I2
GND
O1
O2
PTC1 - FEMTOSMDC016F-02
V_USB
NOT_USED
R25
1K2
R7
1K2
3V3
R5
22R
R6
22R
USB_DM
USB_DP
R8
2K2
LED1
RED LED
GND
Title: MSc Applied Electronic Systems
Page Contents: Communication.SchDoc
Drawn By: S.Kokkinidis
Checked By: S.Kokkinidis
Size: A4
Number: -
Revision: V1.00
Date: 23/3/2019
Time: 3:55:24 μμ
Sheet of
License:
Email: sabbaskok@hotmail.com
Website:
File:

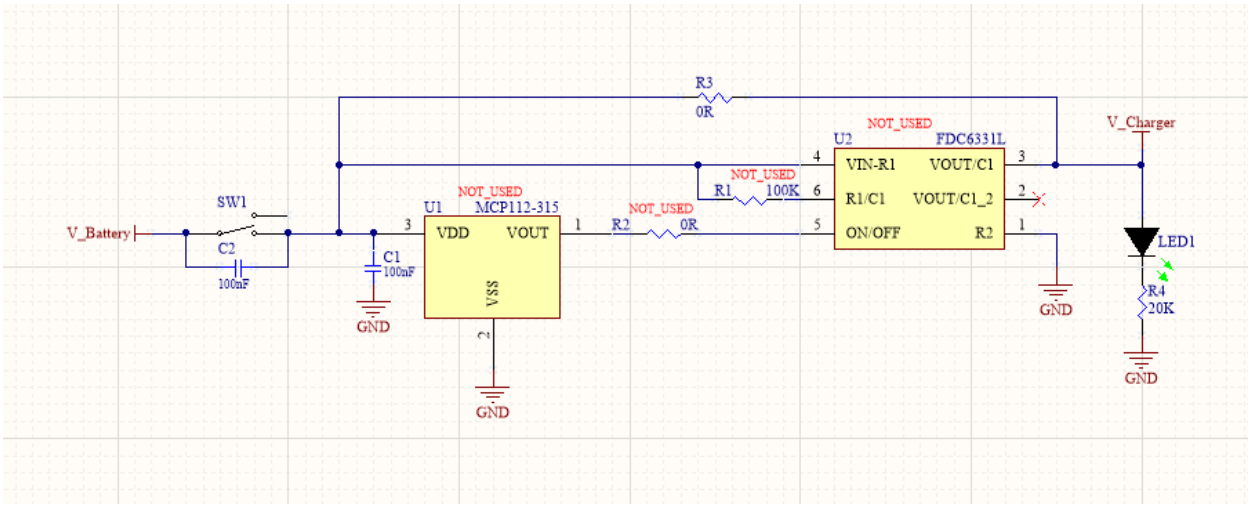
SWD Connector ESD Protection

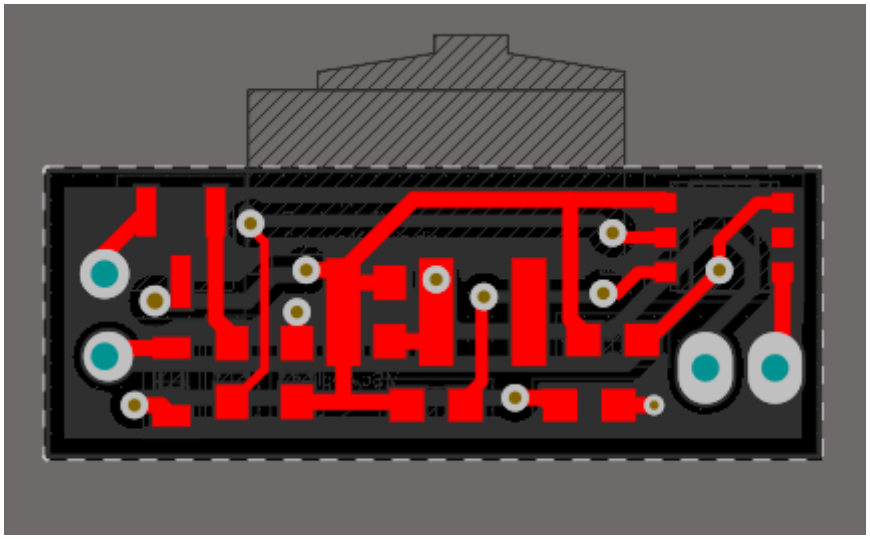


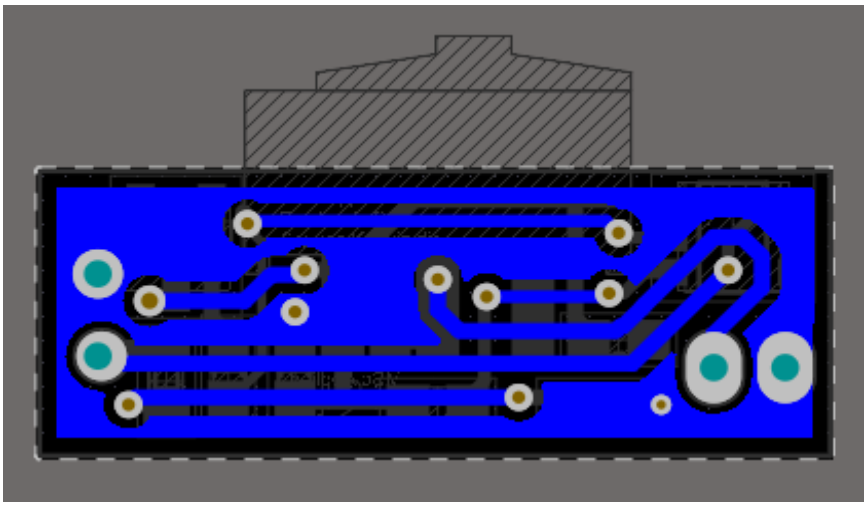
Title: **MSc Applied Electronic Systems**			
Page Contents: ESD Protection.SchDoc			
Drawn By: S.Kokkinidis		Checked By: S.Kokkinidis	
Size: A4	Number: -	Revision: V1.00	
Date: 23/3/2019	Time: 3:55:23 μμ	Sheet of	
License:			
Email: sabbaskok@hotmail.com		Website:	
File:			



14. Παράρτημα Ε – Σχηματικό και Layout πλακέτας μπαταρίας







15. Παράρτημα ΣΤ – Κώδικας μικροελεγκτή

15.1 main.c

```
/**
  ******************************************************************************
  * @file           : main.c
  * @brief          : Savvas Kokkinidis - sabbaskok@hotmail.com
  ******************************************************************************
  * This notice applies to any and all portions of this file
  * that are not between comment pairs USER CODE BEGIN and
  * USER CODE END. Other portions of this file, whether
  * inserted by the user or by software development tools
  * are owned by their respective copyright owners.
  *
  * Copyright (c) 2018 STMicroelectronics International N.V.
  * All rights reserved.
  *
  * Redistribution and use in source and binary forms, with or without
  * modification, are permitted, provided that the following conditions are met:
  *
  * 1. Redistribution of source code must retain the above copyright notice,
  *    this list of conditions and the following disclaimer.
  * 2. Redistributions in binary form must reproduce the above copyright notice,
  *    this list of conditions and the following disclaimer in the documentation
  *    and/or other materials provided with the distribution.
  * 3. Neither the name of STMicroelectronics nor the names of other
  *    contributors to this software may be used to endorse or promote products
  *    derived from this software without specific written permission.
  * 4. This software, including modifications and/or derivative works of this
  *    software, must execute solely and exclusively on microcontroller or
  *    microprocessor devices manufactured by or for STMicroelectronics.
  * 5. Redistribution and use of this software other than as permitted under
  *    this license is void and will automatically terminate your rights under
  *    this license.
  *
  * THIS SOFTWARE IS PROVIDED BY STMICROELECTRONICS AND CONTRIBUTORS "AS IS"
  * AND ANY EXPRESS, IMPLIED OR STATUTORY WARRANTIES, INCLUDING, BUT NOT
  * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
  * PARTICULAR PURPOSE AND NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY
  * RIGHTS ARE DISCLAIMED TO THE FULLEST EXTENT PERMITTED BY LAW. IN NO EVENT
  * SHALL STMICROELECTRONICS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT,
  * INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
  * LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA,
  * OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF
  * LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
  * NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE,
  * EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
  *
  ******************************************************************************
  */
/* Includes ------------------------------------------------------------------*/
#include "main.h"
#include "stm32f1xx_hal.h"

#include "usb_device.h"
#include <stdio.h>
#include <string.h>
#include "accelerometer_gyroscope.h"
#include "bios.h"
#include "ad8555.h"


ADC_HandleTypeDef hadc1;
```

```
61. DMA_HandleTypeDef hdma_adc1;
62. I2C_HandleTypeDef hi2c2;
63. SPI_HandleTypeDef hspi1;
64. UART_HandleTypeDef huart1;
65. DMA_HandleTypeDef hdma_usart1_tx;
66.
67.
68. extern  MainStruct Device;
69. extern  UART       Uart;
70. extern  Measurements   Measure;
71. extern parameters_struct Parameters;
72. extern state_info Flash_State;
73. extern Flash_struct Flash_Parameters;
74. uint8_t Packet_To_Transmit[2000];
75. uint8_t Sending_Started = 0;
76. uint8_t Reset_CMD [4] = {0xF0,0x00,0x00,0x00};
77. uint8_t Check_Kalman = 1;
78. uint16_t DMA_ADCReadings[2];
79.
80. void SystemClock_Config(void);
81. static void MX_GPIO_Init(void);
82. static void MX_DMA_Init(void);
83. static void MX_ADC1_Init(void);
84. static void MX_I2C2_Init(void);
85. static void MX_SPI1_Init(void);
86. static void MX_USART1_UART_Init(unsigned long  BaudRate);
87.
88. int main(void)
89. {
90.   NOP_Delay(50000);
91.   HAL_Init();
92.   Initialize_States();
93.   MX_GPIO_Init();
94.   MX_I2C2_Init();
95.   NOP_Delay(5000);
96.   SystemClock_Config();
97.
98.   MX_DMA_Init();
99.   MX_ADC1_Init();
100.     MX_SPI1_Init();
101.     MX_USART1_UART_Init(460800);
102.     MX_USB_DEVICE_Init();
103.
104.
105.
106.     Load_Parameters();
107.
108.     Device.Bluetooth.BaudRate = 460800;
109.     Device.Bluetooth.Protocol = BLUETOOTH_CMD_MODE;
110.     Device.Bluetooth.State    = BLUETOOTH_START;
111.
112.     Device.Power.Battery_Voltage = 4200;
113.     Device.Power.Charging_Status = NOT_CHARGING;
114.     Device.Power.USB_Detected    = USB_NOT_PLUGGED;
115.
116.     HAL_UART_Receive_IT(&huart1,Uart.IncommingBuffer+Uart.Index,1);
117.     Clear_UART(CLEAR_FLAG,CLEAR_NUMBER);
118.
119.     while(Accelerometer_Gyroscope(ACC_1666Hz,
120.                                   ACC_RANGE_16G,
121.                                   ACC_FILTER_400Hz,
122.                                   GYR_1666Hz,
123.                                   GYR_RANGE_2000dps,
124.                                   MAX_ACC_GYR_INIT_ERRORS) != OK)
125.     {
```

```
126.       HAL_Delay(1000);
127.     }
128.
129.     HAL_GPIO_WritePin(BL_RST_GPIO_Port, BL_RST_Pin, GPIO_PIN_RESET);
130.     HAL_Delay(1000);
131.     HAL_GPIO_WritePin(BL_RST_GPIO_Port, BL_RST_Pin, GPIO_PIN_SET);
132.     HAL_Delay(1000);
133.     // We will try to enter on Command mode using 460800  baud rate
134.     if(Send_CMD_Bluetooth(ENTER_COMMAND_MODE,"CMD\r\n",STRCMP,500) == OK)
135.     {
136.       if(Parameters.Change_Name_Pass == '1')
137.       {
138.            uint8_t buffer[30] = {0};
139.            snprintf(buffer,6,"SA,4\r"); // Set bluetooth on Pin Code authentication
    mode
140.            Send_CMD_Bluetooth(buffer,"AOK\r\n",STRCMP,100);
141.            snprintf(buffer,30,"SN,%s\r",Parameters.Name);    // Set the new Name
142.            Send_CMD_Bluetooth(buffer,"AOK\r\n",STRCMP,100);
143.            snprintf(buffer,10,"SP,%s\r",Parameters.Pass);
144.            if(Send_CMD_Bluetooth(buffer,"AOK\r\n",STRCMP,100) == OK) // Set the new
    Pass
145.            {
146.                    buffer[0] = 0;
147.                    if(Flash_WriteBuffer(PARAMETER__ADDRESS,buffer,1) != OK)
148.                    {
149.                      Flash_WriteBuffer(PARAMETER__ADDRESS,buffer,1);
150.                    }
151.            }
152.       }
153.     }
154.     else // Try to communicate on 115200
155.     {
156.        MX_USART1_UART_Init(115200);
157.        HAL_UART_Receive_IT(&huart1,Uart.IncommingBuffer+Uart.Index,1);
158.        Clear_UART(CLEAR_FLAG,CLEAR_NUMBER);
159.        HAL_GPIO_WritePin(BL_RST_GPIO_Port, BL_RST_Pin, GPIO_PIN_RESET);
160.        HAL_Delay(1000);
161.        HAL_GPIO_WritePin(BL_RST_GPIO_Port, BL_RST_Pin, GPIO_PIN_SET);
162.        HAL_Delay(1000);
163.        if(Send_CMD_Bluetooth(ENTER_COMMAND_MODE,"CMD\r\n",STRCMP,500) == OK)
164.        {
165.           Send_CMD_Bluetooth("SU,46r","AOK\r\n",STRCMP,500);
166.        }
167.     }
168.     Device.Bluetooth.Protocol = BLUETOOTH_SPP_MODE;
169.
170.
171.
172.     HAL_GPIO_WritePin(BL_RST_GPIO_Port, BL_RST_Pin, GPIO_PIN_RESET);
173.     HAL_Delay(1000);
174.     HAL_GPIO_WritePin(BL_RST_GPIO_Port, BL_RST_Pin, GPIO_PIN_SET);
175.     HAL_Delay(1000);
176.
177.
178.
179.     Get_and_Check_Flash_Working_Area();
180.     Init_Kalman_Algorithm();
181.     Clear_Velocity(1);
182.
183.
184.     AD8555_SendPacket(AD8555_SIMULATE,AD8555_SSG_CODE,0);
185.     AD8555_SendPacket(AD8555_SIMULATE,AD8555_FSG_CODE,0);
186.     AD8555_SendPacket(AD8555_SIMULATE,AD8555_OFS_CODE,127);
187.
188.     Device.System.Time_Before_Start_Sending = TIME_BEFORE_SENDING;
```

```
189.
190.      while (1)
191.      {
192.        Flash_Write_Using_States(); // Execute the states for Flash write
193.
194.        if((Device.System.Status == SYSTEM_RUNNING_STATUS) && (Device.System.Time_Befor
    e_Start_Sending == 0))
195.        {
196.            if(Sending_Started == 0)
197.            {
198.              Init_Kalman_Algorithm();
199.              HAL_Delay(500);
200.              Sending_Started = 1;
201.            }
202.            if(Measure.Ticking >= TICKING_SAMPLE)
203.            {
204.              if(Get_IMU_Value())
205.              {
206.                    Measure.Ticking = 0;
207.                    Check_Kalman = 1;
208.                    Execute_Kalman_Filter();
209.                    Read_ADC_Sample();
210.                    Measure.Index++;
211.              }
212.            }
213.            else if((Measure.Ticking >= TICKING_SAMPLE_1_5) && (Check_Kalman == 1))
214.            {
215.              if(Get_IMU_Value())
216.              {
217.                    HAL_ADC_Start_DMA(&hadc1, (uint32_t*) DMA_ADCReadings, 2);//start
    the DMA collecting the data
218.                    Execute_Kalman_Filter();
219.                    Check_Kalman = 2;
220.              }
221.            }
222.            else if((Measure.Ticking >= TICKING_SAMPLE_2_5) && (Check_Kalman == 2))
223.            {
224.              if(Get_IMU_Value())
225.              {
226.                    HAL_ADC_Start_DMA(&hadc1, (uint32_t*) DMA_ADCReadings, 2);//start
    the DMA collecting the data
227.                    Execute_Kalman_Filter();
228.                    Check_Kalman = 3;
229.              }
230.            }
231.            else if((Measure.Ticking >= TICKING_SAMPLE_3_5) && (Check_Kalman == 3))
232.            {
233.              if(Get_IMU_Value())
234.              {
235.                    HAL_ADC_Start_DMA(&hadc1, (uint32_t*) DMA_ADCReadings, 2);//start
    the DMA collecting the data
236.                    Execute_Kalman_Filter();
237.                    Check_Kalman = 4;
238.              }
239.            }
240.            else if((Measure.Ticking >= TICKING_SAMPLE_4_5) && (Check_Kalman == 4))
241.            {
242.              if(Get_IMU_Value())
243.              {
244.                    HAL_ADC_Start_DMA(&hadc1, (uint32_t*) DMA_ADCReadings, 2);//start
    the DMA collecting the data
245.                    Execute_Kalman_Filter();
246.                    Check_Kalman = 1;
247.              }
248.            }
```

```
249.
250.
251.
252.            if(Measure.Index >= MAX_SAMPLES_PER_PACKET)
253.            {
254.              uint16_t Packet_Size;
255.              Packet_Size = Build_Packet();
256.              HAL_UART_Transmit_DMA(&huart1, Packet_To_Transmit,Packet_Size);
257.              Device.System.Timeout_Per_Packet = 4200;
258.              Device.System.Answer_Pending     = 1;
259.              Device.System.Number_Pending     = Measure.Number;
260.              Measure.Number++;
261.              Measure.Index = 0;
262.            }
263.       }
264.
265.       if((Device.System.Answer_Pending == PENDING) && (Device.System.Timeout_Per_Packe
    t == 0)) // Time to write Packet on Flash due to timeout
266.       {
267.          if(Flash_State.locked_by == BY_NOONE)
268.          {
269.                Device.System.Pointers_Refresh_Pending             = PENDING;
270.                Device.System.Answer_Pending                       = NOT_PENDING;
271.                Flash_State.locked_by                             = BY_WRITE_OPERATIO
    N;
272.                Flash_Parameters.Size                             = DATA_SIZE_TO_STOR
    E;
273.                Flash_Parameters.Address                          = FLASH_SIZE_OF_PAG
    E * Flash_Parameters.write_pointer *PAGE_AREAS_PER_PACKET;
274.                memcpy(Flash_Parameters.WriteData,&Measure,Flash_Parameters.Size);
275.                Queue_Push();
276.          }
277.       }
278.
279.       if((Device.System.Pointers_Refresh_Pending == PENDING) &&  (Device.System.Answer
    _Pending == NOT_PENDING))
280.       {
281.          if(Flash_State.locked_by == BY_NOONE)
282.          {
283.              Device.System.Pointers_Refresh_Pending = NOT_PENDING;
284.              Prepare_Pointer_For_Store();
285.          }
286.       }
287.
288.       switch(Uart.Flag) // Check flags related to UART packets
289.       {
290.            case DATA_PACKET_ACK_FLAG:
291.                 if(Device.System.Answer_Pending)
292.                 {
293.                   Device.System.Answer_Pending=0;
294.                   Device.System.Timeout_Per_Packet = 4200;
295.                 }
296.                 Clear_UART(CLEAR_FLAG,CLEAR_NUMBER);
297.               break;
298.            case CMD_START_SENDING_FLAG:
299.                 if(Device.System.Status != SYSTEM_RUNNING_STATUS)
300.                 {
301.                   Clear_Velocity(1);
302.                   Flash_Parameters.read_pointer = Flash_Parameters.write_pointer;
303.                   Device.System.Time_Before_Start_Sending = TIME_BEFORE_SENDING;
304.                 }
305.                 Send_Packet(START_PACKET);
306.                 Device.System.Status = SYSTEM_RUNNING_STATUS;
307.                 Clear_UART(CLEAR_FLAG,CLEAR_NUMBER);
308.               break;
```

```
309.            case CMD_STOP_SENDING_FLAG:
310.                  Send_Packet(STOP_PACKET);
311.                  Measure.Ticking = 0;
312.                  Device.System.Time_Before_Start_Sending = TIME_BEFORE_SENDING;
313.                  Device.System.Status = SYSTEM_STOPPED_STATUS;
314.                  Clear_UART(CLEAR_FLAG,CLEAR_NUMBER);
315.                break;
316.            default: break;
317.        }
318.      }
319.    }
320.
321.    /**
322.      * @brief System Clock Configuration
323.      * @retval None
324.      */
325.    void SystemClock_Config(void)
326.    {
327.
328.      RCC_OscInitTypeDef RCC_OscInitStruct;
329.      RCC_ClkInitTypeDef RCC_ClkInitStruct;
330.      RCC_PeriphCLKInitTypeDef PeriphClkInit;
331.
332.        /**Initializes the CPU, AHB and APB busses clocks
333.        */
334.      RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSE;
335.      RCC_OscInitStruct.HSEState = RCC_HSE_ON;
336.      RCC_OscInitStruct.HSEPredivValue = RCC_HSE_PREDIV_DIV1;
337.      RCC_OscInitStruct.HSIState = RCC_HSI_ON;
338.      RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
339.      RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSE;
340.      RCC_OscInitStruct.PLL.PLLMUL = RCC_PLL_MUL9;
341.      if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
342.      {
343.        _Error_Handler(__FILE__, __LINE__);
344.      }
345.
346.        /**Initializes the CPU, AHB and APB busses clocks
347.        */
348.      RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
349.                                    |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
350.      RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
351.      RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
352.      RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV2;
353.      RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;
354.
355.      if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_2) != HAL_OK)
356.      {
357.        _Error_Handler(__FILE__, __LINE__);
358.      }
359.
360.      PeriphClkInit.PeriphClockSelection = RCC_PERIPHCLK_ADC|RCC_PERIPHCLK_USB;
361.      PeriphClkInit.AdcClockSelection = RCC_ADCPCLK2_DIV6;
362.      PeriphClkInit.UsbClockSelection = RCC_USBCLKSOURCE_PLL_DIV1_5;
363.      if (HAL_RCCEx_PeriphCLKConfig(&PeriphClkInit) != HAL_OK)
364.      {
365.        _Error_Handler(__FILE__, __LINE__);
366.      }
367.
368.        /**Configure the Systick interrupt time
369.        */
370.      HAL_SYSTICK_Config(HAL_RCC_GetHCLKFreq()/20000);
371.
372.        /**Configure the Systick
373.        */
```

```
374.      HAL_SYSTICK_CLKSourceConfig(SYSTICK_CLKSOURCE_HCLK);
375.
376.      /* SysTick_IRQn interrupt configuration */
377.      HAL_NVIC_SetPriority(SysTick_IRQn, 0, 0);
378.    }
379.
380.    /* ADC1 init function */
381.    static void MX_ADC1_Init(void)
382.    {
383.        /* USER CODE BEGIN ADC1_Init 0 */
384.
385.      /* USER CODE END ADC1_Init 0 */
386.
387.      ADC_ChannelConfTypeDef sConfig = {0};
388.
389.      /* USER CODE BEGIN ADC1_Init 1 */
390.
391.      /* USER CODE END ADC1_Init 1 */
392.      /**Common config
393.      */
394.      hadc1.Instance = ADC1;
395.      hadc1.Init.ScanConvMode = ADC_SCAN_ENABLE;
396.      hadc1.Init.ContinuousConvMode = ENABLE;
397.      hadc1.Init.DiscontinuousConvMode = DISABLE;
398.      hadc1.Init.ExternalTrigConv = ADC_SOFTWARE_START;
399.      hadc1.Init.DataAlign = ADC_DATAALIGN_RIGHT;
400.      hadc1.Init.NbrOfConversion = 2;
401.      hadc1.Init.NbrOfDiscConversion = 0;
402.      if (HAL_ADC_Init(&hadc1) != HAL_OK)
403.      {
404.        Error_Handler();
405.      }
406.      /**Configure Regular Channel
407.      */
408.      sConfig.Channel = ADC_CHANNEL_0;
409.      sConfig.Rank = ADC_REGULAR_RANK_1;
410.      sConfig.SamplingTime = ADC_SAMPLETIME_239CYCLES_5;
411.      if (HAL_ADC_ConfigChannel(&hadc1, &sConfig) != HAL_OK)
412.      {
413.        Error_Handler();
414.      }
415.      /**Configure Regular Channel
416.      */
417.      sConfig.Channel = ADC_CHANNEL_1;
418.      sConfig.Rank = ADC_REGULAR_RANK_2;
419.      sConfig.SamplingTime = ADC_SAMPLETIME_239CYCLES_5;
420.      if (HAL_ADC_ConfigChannel(&hadc1, &sConfig) != HAL_OK)
421.      {
422.        Error_Handler();
423.      }
424.      /* USER CODE BEGIN ADC1_Init 2 */
425.
426.      /* USER CODE END ADC1_Init 2 */
427.
428.    }
429.
430.    /* I2C2 init function */
431.    static void MX_I2C2_Init(void)
432.    {
433.
434.      hi2c2.Instance = I2C2;
435.      hi2c2.Init.ClockSpeed = 400000;
436.      hi2c2.Init.DutyCycle = I2C_DUTYCYCLE_2;
437.      hi2c2.Init.OwnAddress1 = 0;
438.      hi2c2.Init.AddressingMode = I2C_ADDRESSINGMODE_7BIT;
```

```
439.      hi2c2.Init.DualAddressMode = I2C_DUALADDRESS_DISABLE;
440.      hi2c2.Init.OwnAddress2 = 0;
441.      hi2c2.Init.GeneralCallMode = I2C_GENERALCALL_DISABLE;
442.      hi2c2.Init.NoStretchMode = I2C_NOSTRETCH_DISABLE;
443.      if (HAL_I2C_Init(&hi2c2) != HAL_OK)
444.      {
445.        _Error_Handler(__FILE__, __LINE__);
446.      }
447.
448.    }
449.
450.    /* SPI1 init function */
451.    void MX_SPI1_Init(void)
452.    {
453.      /* SPI1 parameter configuration*/
454.      hspi1.Instance = SPI1;
455.      hspi1.Init.Mode = SPI_MODE_MASTER;
456.      hspi1.Init.Direction = SPI_DIRECTION_2LINES;
457.      hspi1.Init.DataSize = SPI_DATASIZE_8BIT;
458.      hspi1.Init.CLKPolarity = SPI_POLARITY_LOW;
459.      hspi1.Init.CLKPhase = SPI_PHASE_1EDGE;
460.      hspi1.Init.NSS = SPI_NSS_SOFT;
461.      hspi1.Init.BaudRatePrescaler = SPI_BAUDRATEPRESCALER_32;
462.      hspi1.Init.FirstBit = SPI_FIRSTBIT_MSB;
463.      hspi1.Init.TIMode = SPI_TIMODE_DISABLE;
464.      hspi1.Init.CRCCalculation = SPI_CRCCALCULATION_DISABLE;
465.      hspi1.Init.CRCPolynomial = 10;
466.      if (HAL_SPI_Init(&hspi1) != HAL_OK)
467.      {
468.        _Error_Handler(__FILE__, __LINE__);
469.      }
470.
471.    }
472.
473.    /* USART1 init function */
474.    static void MX_USART1_UART_Init(unsigned long  BaudRate)
475.    {
476.
477.      huart1.Instance = USART1;
478.      huart1.Init.BaudRate = BaudRate;
479.      huart1.Init.WordLength = UART_WORDLENGTH_8B;
480.      huart1.Init.StopBits = UART_STOPBITS_1;
481.      huart1.Init.Parity = UART_PARITY_NONE;
482.      huart1.Init.Mode = UART_MODE_TX_RX;
483.      huart1.Init.HwFlowCtl = UART_HWCONTROL_NONE;
484.      huart1.Init.OverSampling = UART_OVERSAMPLING_16;
485.      if (HAL_UART_Init(&huart1) != HAL_OK)
486.      {
487.        _Error_Handler(__FILE__, __LINE__);
488.      }
489.    }
490.
491.    /**
492.      * Enable DMA controller clock
493.      */
494.    static void MX_DMA_Init(void)
495.    {
496.      /* DMA controller clock enable */
497.      __HAL_RCC_DMA1_CLK_ENABLE();
498.
499.      /* DMA interrupt init */
500.      /* DMA1_Channel1_IRQn interrupt configuration */
501.      HAL_NVIC_SetPriority(DMA1_Channel1_IRQn, 0, 0);
502.      HAL_NVIC_EnableIRQ(DMA1_Channel1_IRQn);
503.      /* DMA1_Channel4_IRQn interrupt configuration */
```

```
504.      HAL_NVIC_SetPriority(DMA1_Channel4_IRQn, 0, 0);
505.      HAL_NVIC_EnableIRQ(DMA1_Channel4_IRQn);
506.    }
507.
508.    /** Configure pins as
509.            * Analog
510.            * Input
511.            * Output
512.            * EVENT_OUT
513.            * EXTI
514.    */
515.    static void MX_GPIO_Init(void)
516.    {
517.
518.      GPIO_InitTypeDef GPIO_InitStruct;
519.
520.      /* GPIO Ports Clock Enable */
521.      __HAL_RCC_GPIOD_CLK_ENABLE();
522.      __HAL_RCC_GPIOA_CLK_ENABLE();
523.      __HAL_RCC_GPIOB_CLK_ENABLE();
524.
525.      /*Configure GPIO pin Output Level */
526.      HAL_GPIO_WritePin(GPIOA, W_TRIG_Pin|GAIN_SEL_Pin, GPIO_PIN_RESET);
527.      HAL_GPIO_WritePin(GPIOA, SS_Pin, GPIO_PIN_SET);
528.
529.      /*Configure GPIO pin Output Level */
530.      HAL_GPIO_WritePin(BL_RST_GPIO_Port, BL_RST_Pin, GPIO_PIN_SET);
531.
532.      /*Configure GPIO pins : W_TRIG_Pin SS_Pin GAIN_SEL_Pin */
533.      GPIO_InitStruct.Pin = W_TRIG_Pin|SS_Pin|GAIN_SEL_Pin;
534.      GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
535.      GPIO_InitStruct.Pull = GPIO_NOPULL;
536.      GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_HIGH;
537.      HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);
538.
539.      /*Configure GPIO pins : RESERVED_Pin BL_STATUS_1_Pin BL_STATUS_2_Pin V_USB_Pin
540.                               READ_AD8555_Pin */
541.      GPIO_InitStruct.Pin = RESERVED_Pin|BL_STATUS_1_Pin|BL_STATUS_2_Pin|V_USB_Pin
542.                              |READ_AD8555_Pin;
543.      GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
544.      GPIO_InitStruct.Pull = GPIO_PULLDOWN;
545.      HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);
546.
547.      /*Configure GPIO pins : INT2_Pin INT1_Pin */
548.      GPIO_InitStruct.Pin = INT2_Pin|INT1_Pin;
549.      GPIO_InitStruct.Mode = GPIO_MODE_IT_RISING;
550.      GPIO_InitStruct.Pull = GPIO_NOPULL;
551.      HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);
552.
553.      /*Configure GPIO pin : BL_RST_Pin */
554.      GPIO_InitStruct.Pin = BL_RST_Pin;
555.      GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
556.      GPIO_InitStruct.Pull = GPIO_NOPULL;
557.      GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
558.      HAL_GPIO_Init(BL_RST_GPIO_Port, &GPIO_InitStruct);
559.
560.    }
561.
562.    /* USER CODE BEGIN 4 */
563.
564.    /* USER CODE END 4 */
565.
566.    /**
567.      * @brief  This function is executed in case of error occurrence.
568.      * @param  file: The file name as string.
```

```
569.      * @param  line: The line in file as a number.
570.      * @retval None
571.      */
572.   void _Error_Handler(char *file, int line)
573.   {
574.     /* USER CODE BEGIN Error_Handler_Debug */
575.     /* User can add his own implementation to report the HAL error return state */
576.     while(1)
577.     {
578.     }
579.     /* USER CODE END Error_Handler_Debug */
580.   }
581.
582.   #ifdef  USE_FULL_ASSERT
583.   /**
584.     * @brief  Reports the name of the source file and the source line number
585.     *         where the assert_param error has occurred.
586.     * @param  file: pointer to the source file name
587.     * @param  line: assert_param error line source number
588.     * @retval None
589.     */
590.   void assert_failed(uint8_t* file, uint32_t line)
591.   {
592.     /* USER CODE BEGIN 6 */
593.     /* User can add his own implementation to report the file name and line number,
594.        tex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) */
595.     /* USER CODE END 6 */
596.   }
597.   #endif /* USE_FULL_ASSERT */
598.
599.   /**
600.     * @}
601.     */
602.
603.   /**
604.     * @}
605.     */
606.
607.   /************************ (C) COPYRIGHT STMicroelectronics *****END OF FILE****/
```

15.2 accelerometer_gyroscope.c

```
/*******************************************************************************
* @file           : accelerometer_gyroscope.c
* @version        : v1.0
* @brief          : Savvas Kokkinidis - sabbaskok@hotmail.com
*******************************************************************************/

#include "accelerometer_gyroscope.h"
#include "stm32f1xx_hal.h"
#include "bios.h"

#define UC (unsigned char)

extern I2C_HandleTypeDef hi2c2;
extern  uint16_t counter_debug;
extern  MainStruct Device;
extern  UART       Uart;
extern  Measurements   Measure;

uint8_t Accelerometer_Gyroscope(uint8_t ACC_RATE,uint8_t ACC_SCALE,uint8_t ACC_FILTER,
uint8_t GYR_RATE,uint8_t GYR_SCALE,uint8_t MAX_ERRORS)
{
    uint8_t Read_Buffer_I2C[2] =  {0,0};
    uint8_t Write_Buffer_I2C[2] = {0,0};
    uint8_t Error_Counts;

    Error_Counts = MAX_ERRORS;
    while(Error_Counts)
    {
      if(HAL_I2C_Mem_Read(&hi2c2, LSM6DS3_I2C_ADDRESS, LSM6DS3_ACC_GYRO_WHO_AM_I_REG,
I2C_MEMADD_SIZE_8BIT, Read_Buffer_I2C, 1, 5000) ==HAL_OK)
      {
        if(Read_Buffer_I2C[0] == WHO_I_AM_RESPONSE) { break;}
      }
      else
      {
        HAL_Delay(100);
      }
      Error_Counts--;
      if(Error_Counts == 0) { return 0; }
    }

    Error_Counts = MAX_ERRORS;
    while(Error_Counts)
    {
      Read_Buffer_I2C[0] = 0x00;
      Read_Buffer_I2C[1] = 0x00;
      // Prepare the CTRL register for Accelerometer
      Write_Buffer_I2C[0]  = ACC_RATE<<4;
      Write_Buffer_I2C[0] |= ((ACC_SCALE<<2) & 0x0F);
      Write_Buffer_I2C[0] |= (ACC_FILTER & 0x01);
      // Prepare the CTRL register for Gyroscope
      Write_Buffer_I2C[1]  = GYR_RATE<<4;
      Write_Buffer_I2C[1] |= ((GYR_SCALE<<1) & 0x0F);
      Write_Buffer_I2C[1] &=  0xFE; // The Bit_0 must be always 0. Secure it.

      if(HAL_I2C_Mem_Write(&hi2c2, LSM6DS3_I2C_ADDRESS, LSM6DS3_ACC_GYRO_CTRL1_XL, I2C
_MEMADD_SIZE_8BIT, Write_Buffer_I2C, 2, 5000) ==HAL_OK)
      {

        if(HAL_I2C_Mem_Read(&hi2c2, LSM6DS3_I2C_ADDRESS, LSM6DS3_ACC_GYRO_CTRL1_XL, I2
C_MEMADD_SIZE_8BIT, Read_Buffer_I2C, 2, 5000) ==HAL_OK)
        {
```

```
59.           if((Read_Buffer_I2C[0] == Write_Buffer_I2C[0]) && (Read_Buffer_I2C[1] == Wri
    te_Buffer_I2C[1]))
60.           {
61.              break;
62.           }
63.           else
64.           {
65.             HAL_Delay(5);
66.           }
67.         }
68.         else
69.         {
70.           HAL_Delay(5);
71.         }
72.       }
73.       else
74.       {
75.         HAL_Delay(5);
76.       }
77.       Error_Counts--;
78.       if(Error_Counts == 0) { return 0; }
79.     }
80.     return 1;
81. }
82. 
83. char Get_IMU_Value(void)
84. {
85.   uint8_t Read_Buffer_I2C[14] =  {0,0,0,0,0,0,0,0,0,0,0,0,0,0};
86. 
87.   if(Measure.Index == 0 || Measure.Index == (MAX_SAMPLES_PER_PACKET/2))
88.   {
89.       Measure.GeneralInfo[Measure.Index].Roll  =  Measure.GeneralInfo[MAX_SAMPLES_PER_
    PACKET-1].Roll;
90.       Measure.GeneralInfo[Measure.Index].Pitch  = Measure.GeneralInfo[MAX_SAMPLES_PER_
    PACKET-1].Pitch;
91.       if(HAL_I2C_Mem_Read(&hi2c2, LSM6DS3_I2C_ADDRESS, LSM6DS3_ACC_GYRO_OUT_TEMP_L, I2
    C_MEMADD_SIZE_8BIT, Read_Buffer_I2C, 14, 5000) ==HAL_OK)
92.       {
93.           Measure.Temperature              =   (int16_t) (Read_Buffer_I2C[1]<<8 | Read
    _Buffer_I2C[0]);
94.           Measure.GyrInfo[Measure.Index].X =   (int16_t) (Read_Buffer_I2C[3]<<8 | Read
    _Buffer_I2C[2]);
95.           Measure.GyrInfo[Measure.Index].Y =   (int16_t) (Read_Buffer_I2C[5]<<8 | Read
    _Buffer_I2C[4]);
96.           Measure.GyrInfo[Measure.Index].Z =   (int16_t) (Read_Buffer_I2C[7]<<8 | Read
    _Buffer_I2C[6]);
97.           Measure.GyrInfo[Measure.Index].Z *=   -1;
98.           Measure.AccInfo[Measure.Index].X =   (int16_t) (Read_Buffer_I2C[9]<<8 | Read
    _Buffer_I2C[8]);
99.           Measure.AccInfo[Measure.Index].Y =   (int16_t) (Read_Buffer_I2C[11]<<8 | Rea
    d_Buffer_I2C[10]);
100.            Measure.AccInfo[Measure.Index].Z =   (int16_t) (Read_Buffer_I2C[13]<<8 |
    Read_Buffer_I2C[12]);
101.            Measure.AccInfo[Measure.Index].Z *=  -1;
102.        }
103.        else
104.        {
105.          memset(Read_Buffer_I2C,0,sizeof(Read_Buffer_I2C));
106.          return 0;
107.        }
108.        return 1;
109.    }
110.    else
111.    {
```

```
112.          Measure.GeneralInfo[Measure.Index].Roll  =  Measure.GeneralInfo[Measure.Index
    -1].Roll;
113.          Measure.GeneralInfo[Measure.Index].Pitch  = Measure.GeneralInfo[Measure.Index
    -1].Pitch;
114.          if(HAL_I2C_Mem_Read(&hi2c2, LSM6DS3_I2C_ADDRESS, LSM6DS3_ACC_GYRO_OUTX_L_G, I
    2C_MEMADD_SIZE_8BIT, Read_Buffer_I2C, 12, 5000) == HAL_OK)
115.          {
116.              Measure.GyrInfo[Measure.Index].X =   (int16_t) (Read_Buffer_I2C[1]<<8 | R
    ead_Buffer_I2C[0]);
117.              Measure.GyrInfo[Measure.Index].Y =   (int16_t) (Read_Buffer_I2C[3]<<8 | R
    ead_Buffer_I2C[2]);
118.              Measure.GyrInfo[Measure.Index].Z =   (int16_t) (Read_Buffer_I2C[5]<<8 | R
    ead_Buffer_I2C[4]);
119.              Measure.GyrInfo[Measure.Index].Z *=    -1;
120.              Measure.AccInfo[Measure.Index].X =   (int16_t) (Read_Buffer_I2C[7]<<8 | R
    ead_Buffer_I2C[6]);
121.              Measure.AccInfo[Measure.Index].Y =   (int16_t) (Read_Buffer_I2C[9]<<8 | R
    ead_Buffer_I2C[8]);
122.              Measure.AccInfo[Measure.Index].Z =   (int16_t) (Read_Buffer_I2C[11]<<8 |
    Read_Buffer_I2C[10]);
123.              Measure.AccInfo[Measure.Index].Z *=    -1;
124.          }
125.          else
126.          {
127.            memset(Read_Buffer_I2C,0,sizeof(Read_Buffer_I2C));
128.            return 0;
129.          }
130.          return 1;
131.      }
132.    }
```

15.3 accelerometer_gyroscope.h

```
/**
*****************************************************************************
* @file           : accelerometer_gyroscope.h
* @version        : V1.00
* @brief          : Savvas Kokkinidis - sabbaskok@hotmail.com
*****************************************************************************
**/

#include "stm32f1xx_hal.h"

/* Define to prevent recursive inclusion -------------------------------------*/
#ifndef __ACCELEROMETER_GYROSCOPE__H__
#define __ACCELEROMETER_GYROSCOPE__H__

#ifdef __cplusplus
 extern "C" {
#endif

/************** I2C Address ******************/
#define  SDO_SA0_Pin '1'
#if SDO_SA0_Pin == '1'
    #define LSM6DS3_I2C_ADDRESS (0x6A<<1)
#else
    #define LSM6DS3_I2C_ADDRESS (0x6B<<1)
#endif

#define UC (unsigned char)

/************** Register description *******************/
#define LSM6DS3_FUNC_CFG_ACCESS             0x01
#define LSM6DS3_ACC_GYRO_SENSOR_SYNC_TIME       0x04
#define LSM6DS3_ACC_GYRO_SENSOR_SYNC_EN         0x05
#define LSM6DS3_ACC_GYRO_FIFO_CTRL1             0x06
#define LSM6DS3_ACC_GYRO_FIFO_CTRL2             0x07
#define LSM6DS3_ACC_GYRO_FIFO_CTRL3             0x08
#define LSM6DS3_ACC_GYRO_FIFO_CTRL4             0x09
#define LSM6DS3_ACC_GYRO_FIFO_CTRL5             0x0A
#define LSM6DS3_ACC_GYRO_ORIENT_CFG_G           0x0B
#define LSM6DS3_ACC_GYRO_REFERENCE_G            0x0C
#define LSM6DS3_ACC_GYRO_INT1_CTRL              0x0D
#define LSM6DS3_ACC_GYRO_INT2_CTRL              0x0E
#define LSM6DS3_ACC_GYRO_WHO_AM_I_REG           0x0F
#define LSM6DS3_ACC_GYRO_CTRL1_XL           0x10
#define LSM6DS3_ACC_GYRO_CTRL2_G            0x11
#define LSM6DS3_ACC_GYRO_CTRL3_C            0x12
#define LSM6DS3_ACC_GYRO_CTRL4_C            0x13
#define LSM6DS3_ACC_GYRO_CTRL5_C            0x14
#define LSM6DS3_ACC_GYRO_CTRL6_G            0x15
#define LSM6DS3_ACC_GYRO_CTRL7_G            0x16
#define LSM6DS3_ACC_GYRO_CTRL8_XL           0x17
#define LSM6DS3_ACC_GYRO_CTRL9_XL           0x18
#define LSM6DS3_ACC_GYRO_CTRL10_C           0x19
#define LSM6DS3_ACC_GYRO_MASTER_CONFIG          0x1A
#define LSM6DS3_ACC_GYRO_WAKE_UP_SRC            0x1B
#define LSM6DS3_ACC_GYRO_TAP_SRC            0x1C
#define LSM6DS3_ACC_GYRO_D6D_SRC            0x1D
#define LSM6DS3_ACC_GYRO_STATUS_REG             0x1E
#define LSM6DS3_ACC_GYRO_OUT_TEMP_L             0x20
#define LSM6DS3_ACC_GYRO_OUT_TEMP_H             0x21
#define LSM6DS3_ACC_GYRO_OUTX_L_G           0x22
#define LSM6DS3_ACC_GYRO_OUTX_H_G           0x23
#define LSM6DS3_ACC_GYRO_OUTY_L_G           0x24
#define LSM6DS3_ACC_GYRO_OUTY_H_G           0x25
```

```
64. #define LSM6DS3_ACC_GYRO_OUTZ_L_G           0x26
65. #define LSM6DS3_ACC_GYRO_OUTZ_H_G           0x27
66. #define LSM6DS3_ACC_GYRO_OUTX_L_XL              0x28
67. #define LSM6DS3_ACC_GYRO_OUTX_H_XL              0x29
68. #define LSM6DS3_ACC_GYRO_OUTY_L_XL              0x2A
69. #define LSM6DS3_ACC_GYRO_OUTY_H_XL              0x2B
70. #define LSM6DS3_ACC_GYRO_OUTZ_L_XL              0x2C
71. #define LSM6DS3_ACC_GYRO_OUTZ_H_XL              0x2D
72. #define LSM6DS3_ACC_GYRO_SENSORHUB1_REG         0x2E
73. #define LSM6DS3_ACC_GYRO_SENSORHUB2_REG         0x2F
74. #define LSM6DS3_ACC_GYRO_SENSORHUB3_REG         0x30
75. #define LSM6DS3_ACC_GYRO_SENSORHUB4_REG         0x31
76. #define LSM6DS3_ACC_GYRO_SENSORHUB5_REG         0x32
77. #define LSM6DS3_ACC_GYRO_SENSORHUB6_REG         0x33
78. #define LSM6DS3_ACC_GYRO_SENSORHUB7_REG         0x34
79. #define LSM6DS3_ACC_GYRO_SENSORHUB8_REG         0x35
80. #define LSM6DS3_ACC_GYRO_SENSORHUB9_REG         0x36
81. #define LSM6DS3_ACC_GYRO_SENSORHUB10_REG        0x37
82. #define LSM6DS3_ACC_GYRO_SENSORHUB11_REG        0x38
83. #define LSM6DS3_ACC_GYRO_SENSORHUB12_REG        0x39
84. #define LSM6DS3_ACC_GYRO_FIFO_STATUS1           0x3A
85. #define LSM6DS3_ACC_GYRO_FIFO_STATUS2           0x3B
86. #define LSM6DS3_ACC_GYRO_FIFO_STATUS3           0x3C
87. #define LSM6DS3_ACC_GYRO_FIFO_STATUS4           0x3D
88. #define LSM6DS3_ACC_GYRO_FIFO_DATA_OUT_L        0x3E
89. #define LSM6DS3_ACC_GYRO_FIFO_DATA_OUT_H        0x3F
90. #define LSM6DS3_ACC_GYRO_TIMESTAMP0_REG         0x40
91. #define LSM6DS3_ACC_GYRO_TIMESTAMP1_REG         0x41
92. #define LSM6DS3_ACC_GYRO_TIMESTAMP2_REG         0x42
93. #define LSM6DS3_ACC_GYRO_STEP_COUNTER_L         0x4B
94. #define LSM6DS3_ACC_GYRO_STEP_COUNTER_H         0x4C
95. #define LSM6DS3_ACC_GYRO_FUNC_SRC           0x53
96. #define LSM6DS3_ACC_GYRO_TAP_CFG1           0x58
97. #define LSM6DS3_ACC_GYRO_TAP_THS_6D             0x59
98. #define LSM6DS3_ACC_GYRO_INT_DUR2           0x5A
99. #define LSM6DS3_ACC_GYRO_WAKE_UP_THS            0x5B
100.    #define LSM6DS3_ACC_GYRO_WAKE_UP_DUR            0x5C
101.    #define LSM6DS3_ACC_GYRO_FREE_FALL              0x5D
102.    #define LSM6DS3_ACC_GYRO_MD1_CFG            0x5E
103.    #define LSM6DS3_ACC_GYRO_MD2_CFG            0x5F
104.
105.
106.    #define WHO_I_AM_RESPONSE                               0x69
107.
108.    #define              POWER_DOWN                         0x00
109.    #define MAX_ACC_GYR_INIT_ERRORS                         0x0A
110.
111.
112.    // Parameters related with Accelerometer's Initialization
113.    #define ACC_POWER_DOWN                                  0x00
114.    #define     ACC_12_5Hz                                  0x01
115.    #define       ACC_26Hz                                  0x02
116.    #define       ACC_52Hz                                  0x03
117.    #define      ACC_104Hz                                  0x04
118.    #define      ACC_208Hz                                  0x05
119.    #define      ACC_416Hz                                  0x06
120.    #define      ACC_833Hz                                  0x07
121.    #define     ACC_1666Hz                                  0x08
122.    #define     ACC_3330Hz                                  0x09
123.    #define     ACC_6660Hz                                  0x0A
124.
125.    #define   ACC_RANGE_2G                                  0x00
126.    #define  ACC_RANGE_16G                                  0x01
127.    #define   ACC_RANGE_4G                                  0x02
128.    #define   ACC_RANGE_8G                                  0x03
```

```
129.
130.    #define ACC_FILTER_400Hz                                0x00
131.    #define ACC_FILTER_200Hz                                0x01
132.    #define ACC_FILTER_100Hz                                0x02
133.    #define  ACC_FILTER_50Hz                                0x03
134.
135.    // Parameters related with Gyroscope's Initialization
136.    #define GYR_POWER_DOWN                                  0x00
137.    #define     GYR_12_5Hz                                  0x01
138.    #define       GYR_26Hz                                  0x02
139.    #define       GYR_52Hz                                  0x03
140.    #define      GYR_104Hz                                  0x04
141.    #define      GYR_208Hz                                  0x05
142.    #define      GYR_416Hz                                  0x06
143.    #define      GYR_833Hz                                  0x07
144.    #define     GYR_1666Hz                                  0x08
145.
146.    #define    GYR_RANGE_125dps                             0x01
147.    #define    GYR_RANGE_250dps                             0x00
148.    #define    GYR_RANGE_500dps                             0x02
149.    #define   GYR_RANGE_1000dps                             0x04
150.    #define   GYR_RANGE_2000dps                             0x06
151.
152.
153.
154.    uint8_t Accelerometer_Gyroscope(uint8_t ACC_RATE,uint8_t ACC_SCALE,uint8_t ACC_FILT
    ER,uint8_t GYR_RATE,uint8_t GYR_SCALE,uint8_t MAX_ERRORS);
155.    char Get_IMU_Value(void);
156.
157.    #ifdef __cplusplus
158.    }
159.    #endif
160.    #endif
```

15.4 bios.c

```
/*******************************************************************************
* @file           : bios.c
* @version        : v1.0
* @brief          : Savvas Kokkinidis - sabbaskok@hotmail.com
*******************************************************************************/

#include "bios.h"
#include "stm32f1xx_hal.h"
#include "math.h"

#define ADD_NEWLINE_FOR_BL_LIBRARY
#define RAD_TO_DEG      57.2957f
#define CRC16 0x8005

extern UART_HandleTypeDef huart1;
extern uint8_t Packet_To_Transmit[2000];
extern state_info Flash_State;
extern SPI_HandleTypeDef hspi1;
extern ADC_HandleTypeDef hadc1;
extern state_info Flash_State;
extern Flash_struct Flash_Parameters;
extern uint16_t DMA_ADCReadings[2]; //ADC Readings

MainStruct       Device;
Measurements     Measure;
UART             Uart;
Velocity_struct  Velocity_Measurements;
parameters_struct Parameters;

uint32_t Velocity_Timer = 0;

uint8_t Read_ADC_Sample(void)
{
   static uint8_t BatterySamples=0;
   static uint32_t BatterySUM=0;
   uint8_t Invalid = 0;


    BatterySUM += (uint16_t) (DMA_ADCReadings[0]*1.6117f);
    if(BatterySamples >= 49)
    {
      Device.Power.Battery_Voltage = (uint16_t) (BatterySUM/50);
      BatterySamples=0;
      BatterySUM=0;
    }
    else
    {
      BatterySamples++;
    }



   if(Invalid)
   {
       Measure.AnalogRead[Measure.Index] = 0xFFFF;
   }
   else
   {
     if(1)
     {
        Measure.AnalogRead[Measure.Index] = DMA_ADCReadings[1];
```

```
77.       }
78.       else
79.       {
80.          Measure.AnalogRead[Measure.Index] = 0xFFFF;
81.       }
82.    }
83.
84.    return 1;
85. }
86.
87. void Send_Packet(uint8_t packet)
88. {
89.   uint8_t index=0;
90.   uint8_t temp_checksum = 0;
91.
92.
93.   Queue_Status(); // Refresh Flash_Parameters.bytes_available variable
94.
95.   Packet_To_Transmit[index++] = SOH;
96.   Packet_To_Transmit[index++] = MY_ID;
97.   Packet_To_Transmit[index++] = AP_ID;
98.   Packet_To_Transmit[index++] = ACK;
99.   Packet_To_Transmit[index++] = CMD_PACKET;
100.  Packet_To_Transmit[index++] = STX;
101.  Packet_To_Transmit[index++] = 'S';
102.
103.  if(packet == START_PACKET)
104.  {
105.    Packet_To_Transmit[index++] = '1';
106.  }
107.  else
108.  {
109.    Packet_To_Transmit[index++] = '0';
110.    Convert_Hex_To_Ascii_Measures((uint8_t) (Flash_Parameters.bytes_available>>8),inde
    x);
111.    index += 2;
112.    Convert_Hex_To_Ascii_Measures((uint8_t) (Flash_Parameters.bytes_available),index);
113.    index += 2;
114.  }
115.  Packet_To_Transmit[index++] = ETX;
116.  temp_checksum = Calculate_Checksum(Packet_To_Transmit,index);
117.  Convert_Hex_To_Ascii_Measures(temp_checksum,index);
118.  index += 2;
119.  Packet_To_Transmit[index++] = EOT;
120.  #ifdef ADD_NEWLINE_FOR_BL_LIBRARY
121.    Packet_To_Transmit[index++] = '\r';
122.    Packet_To_Transmit[index++] = '\n';
123.  #endif
124.  HAL_UART_Transmit_DMA(&huart1, Packet_To_Transmit,index);
125. }
126.
127. uint16_t Build_Packet(void)
128. {
129.   uint16_t index=0;
130.   uint8_t temp_checksum = 0;
131.   uint8_t index_struct = 0;
132.   uint16_t current_CRC = 0;
133.
134.   Packet_To_Transmit[index++] = SOH;
135.   Packet_To_Transmit[index++] = MY_ID;
136.   Packet_To_Transmit[index++] = AP_ID;
137.   Packet_To_Transmit[index++] = ENQ;
138.   Packet_To_Transmit[index++] = DATA_PACKET;
```

```
  Packet_To_Transmit[index++] = STX;
  Convert_Hex_To_Ascii_Measures((uint8_t) (Measure.Number>>8),index);
  index += 2;
  Convert_Hex_To_Ascii_Measures((uint8_t)    (Measure.Number),index);
  index += 2;
  for(index_struct=0; index_struct < MAX_SAMPLES_PER_PACKET; index_struct++)
  {
    // Accelerometer//
    Convert_Hex_To_Ascii_Measures((uint8_t) (Measure.AccInfo[index_struct].X>>8),index
);
    index += 2;
    Convert_Hex_To_Ascii_Measures((uint8_t) (Measure.AccInfo[index_struct].X),index);
    index += 2;
    Convert_Hex_To_Ascii_Measures((uint8_t) (Measure.AccInfo[index_struct].Y>>8),index
);
    index += 2;
    Convert_Hex_To_Ascii_Measures((uint8_t) (Measure.AccInfo[index_struct].Y),index);
    index += 2;
    Convert_Hex_To_Ascii_Measures((uint8_t) (Measure.AccInfo[index_struct].Z>>8),index
);
    index += 2;
    Convert_Hex_To_Ascii_Measures((uint8_t) (Measure.AccInfo[index_struct].Z),index);
    index += 2;
    //General
    Convert_Hex_To_Ascii_Measures((uint8_t) (Measure.GeneralInfo[index_struct].Pitch>>
8),index);
    index += 2;
    Convert_Hex_To_Ascii_Measures((uint8_t) (Measure.GeneralInfo[index_struct].Pitch),
index);
    index += 2;
    Convert_Hex_To_Ascii_Measures((uint8_t) (Measure.GeneralInfo[index_struct].Roll>>8
),index);
    index += 2;
    Convert_Hex_To_Ascii_Measures((uint8_t) (Measure.GeneralInfo[index_struct].Roll),i
ndex);
    index += 2;
    Convert_Hex_To_Ascii_Measures((uint8_t) (Measure.GeneralInfo[index_struct].Velocit
y>>8),index);
    index += 2;
    Convert_Hex_To_Ascii_Measures((uint8_t) (Measure.GeneralInfo[index_struct].Velocit
y),index);
    index += 2;
    Convert_Hex_To_Ascii_Measures((uint8_t) (Measure.AnalogRead[index_struct]>>8),inde
x);
    index += 2;
    Convert_Hex_To_Ascii_Measures((uint8_t) (Measure.AnalogRead[index_struct]),index);
    index += 2;
  }
  Measure.Distance = 0x66445566;
  Convert_Hex_To_Ascii_Measures((uint8_t) (Measure.Distance>>24),index);
  index += 2;
  Convert_Hex_To_Ascii_Measures((uint8_t) (Measure.Distance>>16),index);
  index += 2;
  Convert_Hex_To_Ascii_Measures((uint8_t) (Measure.Distance>>8),index);
  index += 2;
  Convert_Hex_To_Ascii_Measures((uint8_t) (Measure.Distance),index);
  index += 2;
  Convert_Hex_To_Ascii_Measures((uint8_t) (Measure.Temperature>>8),index);
  index += 2;
  Convert_Hex_To_Ascii_Measures((uint8_t) (Measure.Temperature),index);
  index += 2;
  if(Device.Power.USB_Detected == USB_PLUGGED_IN)
  {
```

```
    Convert_Hex_To_Ascii_Measures((uint8_t) (0xFF),index);
    index += 2;
    Convert_Hex_To_Ascii_Measures((uint8_t) (0xFF),index);
    index += 2;
  }
  else
  {
    Convert_Hex_To_Ascii_Measures((uint8_t) (Device.Power.Battery_Voltage>>8),index);
    index += 2;
    Convert_Hex_To_Ascii_Measures((uint8_t) (Device.Power.Battery_Voltage),index);
    index += 2;
  }
  current_CRC = gen_crc16(Packet_To_Transmit+6,index-6);
  Convert_Hex_To_Ascii_Measures((uint8_t) (current_CRC>>8),index);
  index += 2;
  Convert_Hex_To_Ascii_Measures((uint8_t) (current_CRC),index);
  index += 2;
  Packet_To_Transmit[index++] = ETX;
  temp_checksum = Calculate_Checksum(Packet_To_Transmit,index);
  Convert_Hex_To_Ascii_Measures(temp_checksum,index);
  index += 2;
  Packet_To_Transmit[index++] = EOT;
  #ifdef ADD_NEWLINE_FOR_BL_LIBRARY
    Packet_To_Transmit[index++] = '\r';
    Packet_To_Transmit[index++] = '\n';
  #endif
  return(index);
}

uint8_t Convert_Ascii_To_Hex(uint8_t *buffer)
{
  uint8_t temp_high = 0;
  uint8_t temp_low  = 0;
  uint8_t value     = 0;

  if(*buffer  < 'A')
  {
    temp_high = *buffer - 48;
  }
  else
  {
    temp_high = *buffer - 55;
  }

  if(*(buffer+1) < 'A')
  {
    temp_low = *(buffer+1) - 48;
  }
  else
  {
    temp_low = *(buffer+1) - 55;
  }

  temp_high &= 0x0F;
  temp_low &= 0x0F;
  value = (uint8_t) (temp_high*0x10 + temp_low);
  return value;
}

void Convert_Hex_To_Ascii(uint8_t value,uint8_t *buffer)
{
  uint8_t temp_high = value >> 4;
  uint8_t temp_low  = value & 0x0F;
```

```
255.
256.   if(temp_high < 0x0A)
257.   {
258.     *buffer = temp_high + 48;
259.   }
260.   else
261.   {
262.     *buffer = temp_high + 55;
263.   }
264.
265.   if(temp_low < 0x0A)
266.   {
267.     *(buffer+1) = temp_low + 48;
268.   }
269.   else
270.   {
271.     *(buffer+1) = temp_low + 55;
272.   }
273. }
274.
275. void Convert_Hex_To_Ascii_Measures(uint8_t value,uint16_t index)
276. {
277.   uint8_t temp_high = value >> 4;
278.   uint8_t temp_low  = value & 0x0F;
279.
280.   if(temp_high < 0x0A) { Packet_To_Transmit[index] = temp_high + 48;}
281.   else { Packet_To_Transmit[index] = temp_high + 55; }
282.
283.   if(temp_low < 0x0A) { Packet_To_Transmit[index+1] = temp_low + 48; }
284.   else { Packet_To_Transmit[index+1] = temp_low + 55; }
285. }
286.
287. uint8_t Send_CMD_Bluetooth(uint8_t *buffer,uint8_t *answer,uint8_t cmp_str,uint16_t ti
     meout)
288. {
289.    uint16_t counting = 0;
290.    uint8_t  return_value=0;
291.
292.    Clear_UART(CLEAR_FLAG,CLEAR_NUMBER);
293.    memset(Uart.IncommingBuffer,0,UART_MAX_LEN);
294.
295.    HAL_UART_Transmit_IT(&huart1,buffer,strlen(buffer));
296.    while(Uart.Flag != CMD_PACKET_CAME)
297.    {
298.      HAL_Delay(1);
299.      counting++;
300.      if(counting >= timeout) { return_value = TIMEOUT; }
301.    }
302.
303.   if(return_value != TIMEOUT)
304.   {
305.      Uart.IncommingBuffer[UART_MAX_LEN-1] = 0;
306.      if(cmp_str == STRCMP)
307.      {
308.        if(strcmp(answer,Uart.IncommingBuffer) == 0)
309.        {
310.           return_value = OK;
311.        }
312.        else
313.        {
314.           return_value = FAILED;
315.        }
316.      }
```

```
317.      else
318.      {
319.        if(strcmp(answer,Uart.IncommingBuffer) != 0)
320.        {
321.           return_value = OK;
322.        }
323.        else
324.        {
325.           return_value = FAILED;
326.        }
327.      }
328.   }
329.   Clear_UART(CLEAR_FLAG,CLEAR_NUMBER);
330.   memset(Uart.IncommingBuffer,0,UART_MAX_LEN);
331.   return return_value;
332. }
333.
334. uint8_t Calculate_Checksum(uint8_t *buffer,uint16_t length)
335. {
336.   uint8_t temp_checksum=0;
337.   for(uint16_t i=0;i < length; i++)
338.   {
339.     temp_checksum += buffer[i];
340.   }
341.   return temp_checksum;
342. }
343.
344. void Clear_UART(uint8_t Clear_Flag,uint8_t Clear_Number)
345. {
346.   Uart.Status  =  UART_FREE;
347.   Uart.Index   =  0;
348.   Uart.Command =  INVALID_PACKET;
349.   Uart.From_ID =  0;
350.   Uart.To_ID   =  0;
351.   Uart.Response = 0;
352.   Uart.Checksum = 0;
353.   Uart.DataLen  = 0;
354.   Uart.DataStartIndex = 0;
355.   Uart.DataEndIndex = 0;
356.   Uart.CalculatedChecksum = 0xFF;
357.   if(Clear_Flag)
358.   {
359.     Uart.Flag    = CLEAR_PACKET_FLAG;
360.   }
361.   if(Clear_Number)
362.   {
363.     Uart.PacketNumber = 0;
364.   }
365. }
366.
367. uint16_t gen_crc16(uint8_t *buffer, uint16_t size)
368. {
369.     uint16_t crc = 0xFFFF;
370.
371.     if (buffer && size)
372.     while (size--)
373.     {
374.             crc = (crc >> 8) | (crc << 8);
375.             crc ^= *buffer++;
376.             crc ^= ((unsigned char) crc) >> 4;
377.             crc ^= crc << 12;
378.             crc ^= (crc & 0xFF) << 5;
379.      }
```

```
    return crc;
}


void NOP_Delay(uint32_t mdelay)
{
  do
  {
    __NOP();
  }
  while (mdelay --);
}

void Initialize_States(void)
{
   Flash_State.current_state = FLASH_WRITE_START;
   Flash_State.locked_by     = BY_NOONE;
   Flash_State.fails         = 0;
   Flash_State.retry         = 0;
   Device.Bluetooth.Protocol  = BLUETOOTH_CMD_MODE;
}

void SPI0_out(volatile unsigned char *Data, uint16_t DataLength)
{
  /* Send the byte */
   __HAL_SPI_ENABLE(&hspi1);
  SPIx_WriteData((uint8_t *)Data, DataLength);
}

void SPI0_in(volatile unsigned char *Data, uint16_t DataLength)
{
  /* Send the byte */
   __HAL_SPI_ENABLE(&hspi1);
  SPIx_ReadData((uint8_t *)Data, DataLength);
}

void SPIx_WriteData(uint8_t *DataIn, uint16_t DataLength)
{
  HAL_StatusTypeDef status = HAL_OK;

  status = HAL_SPI_Transmit(&hspi1, DataIn, DataLength, 10000);

  /* Check the communication status */
  if(status != HAL_OK)
  {
    /* Execute user timeout callback */
    SPIx_Error();
  }
}



void SPIx_ReadData(uint8_t *DataOut, uint16_t DataLength)
{
  HAL_StatusTypeDef status = HAL_OK;

  status = HAL_SPI_Receive(&hspi1, DataOut, DataLength, 30000);

  /* Check the communication status */
  if(status != HAL_OK)
  {
    /* Execute user timeout callback */
    SPIx_Error();
```

```
443.   }
444. }
445.
446. void SPIx_Error (void)
447. {
448.   /* De-initialize the SPI communication BUS */
449.   HAL_SPI_DeInit(&hspi1);
450.
451.   hspi1.Instance = SPI1;
452.   hspi1.Init.Mode = SPI_MODE_MASTER;
453.   hspi1.Init.Direction = SPI_DIRECTION_2LINES;
454.   hspi1.Init.DataSize = SPI_DATASIZE_8BIT;
455.   hspi1.Init.CLKPolarity = SPI_POLARITY_LOW;
456.   hspi1.Init.CLKPhase = SPI_PHASE_1EDGE;
457.   hspi1.Init.NSS = SPI_NSS_SOFT;
458.   hspi1.Init.BaudRatePrescaler = SPI_BAUDRATEPRESCALER_32;
459.   hspi1.Init.FirstBit = SPI_FIRSTBIT_MSB;
460.   hspi1.Init.TIMode = SPI_TIMODE_DISABLE;
461.   hspi1.Init.CRCCalculation = SPI_CRCCALCULATION_DISABLE;
462.   hspi1.Init.CRCPolynomial = 10;
463.
464.   /* Re-Initiaize the SPI communication BUS */
465.   HAL_SPI_Init(&hspi1);
466. }
467.
468. void Reset_Flash_Queue(void)
469. {
470.     Flash_Parameters.read_pointer=0;
471.     Flash_Parameters.write_pointer=0;
472. }
473.
474. uint8_t Queue_Status(void)
475. {
476.  if(Flash_Parameters.read_pointer == ( Flash_Parameters.write_pointer + 1 ) % (QUEUE_S
    IZE))
477.  {
478.     Flash_Parameters.bytes_available=QUEUE_SIZE;
479.     return QUEUE_FULL;
480.  }
481.  else if(Flash_Parameters.read_pointer == Flash_Parameters.write_pointer)
482.  {
483.     Flash_Parameters.bytes_available=0;
484.     return QUEUE_EMPTY;
485.  }
486.  else
487.  {
488.    if(Flash_Parameters.read_pointer < Flash_Parameters.write_pointer)
489.    {
490.      Flash_Parameters.bytes_available = (Flash_Parameters.write_pointer-
    Flash_Parameters.read_pointer);
491.    }
492.    else
493.    {
494.      Flash_Parameters.bytes_available= (QUEUE_SIZE-
    Flash_Parameters.read_pointer)+ Flash_Parameters.write_pointer;
495.    }
496.    return QUEUE_VALID;
497.  }
498. }
499.
500. void Queue_Push(void)
501. {
502.     if(Queue_Status() == QUEUE_FULL)
```

```
    {
      Queue_Pop();
    }
    Flash_Parameters.write_pointer = (Flash_Parameters.write_pointer+1)%(QUEUE_SIZE);
}

void Queue_Pop(void)
{
  Flash_Parameters.read_pointer = (Flash_Parameters.read_pointer+1)%(QUEUE_SIZE);
}

uint8_t Load_Parameters(void)
{
  if(Flash_Read_Bytes(PARAMETER__ADDRESS,&Parameters,21) != OK)
  {
    HAL_Delay(100);
    Flash_Read_Bytes(PARAMETER__ADDRESS,&Parameters,21);
  }
  return OK;
}

uint8_t Check_And_Save_Parameters(uint8_t* Buf, uint32_t *Len)
{
     uint32_t length = *Len;
     uint8_t index = 0;
     uint8_t index1 = 0;
     uint8_t comma_index = 0;
     uint8_t Name_buffer[20] = {0};
     uint8_t Pass_buffer[10] = {0};
     uint8_t data_length = 0;
     uint8_t calculated_checksum = 0;
     uint8_t incoming_checksum = 0;

     if(length > 5)
     {
        length -=3;
     }

     if(Buf[0]== '$'
     && Buf[1]== 'S'
     && Buf[2]== 'E'
     && Buf[3]== 'T'
     && Buf[4]== 'P'
     && Buf[5]== 'A'
     && Buf[6]== 'R'
     && Buf[7]== 'A'
     && Buf[8]== 'M'
     && Buf[9]== '['
     && Buf[length-4] ==  ']'
     && Buf[length-1] ==  '%' )
     {

        incoming_checksum = Convert_Ascii_To_Hex(Buf+length-3);
        calculated_checksum = Calculate_Checksum(Buf,length-3);
        if(incoming_checksum == calculated_checksum)
        {
           data_length = length-15;
           for(index = 0;index < data_length;index++)
           {
               if(Buf[10+index] != ',')
               {
                  Name_buffer[index] = Buf[10+index];
               }
```

```
                else
                {
                   comma_index = 10+index;
                   Name_buffer[index] = 0;
                   Name_buffer[14] = 0;
                   break;
                }
            }

            for(index1 = 0;index1 < index;index1++)
            {
               if((Name_buffer[index1] >= '0' &&  Name_buffer[index1] <= '9') || (Name_
buffer[index1] >= 'a' &&  Name_buffer[index1] <= 'z')|| (Name_buffer[index1] >= 'A' &&
  Name_buffer[index1] <= 'Z') )
                {
                  ;
                }
                else
                {
                  return 4;
                }
            }

            for(index = comma_index+1;index < (comma_index+5);index++)
            {
                   Pass_buffer[index-comma_index-1] = Buf[index];
            }
            Pass_buffer[4]=0;

           for(index1 = 0;index1 < 4;index1++)
           {
               if(Pass_buffer[index1] >= '0' &&  Pass_buffer[index1] <= '9')
               {
                 ;
               }
               else
               {
                 return 4;
               }
           }
           Parameters.Change_Name_Pass = '1'; // Time to change parameters
           for(index=0;index<15;index++)
           {
             Parameters.Name[index] = Name_buffer[index];
           }
           Parameters.Name[14] = 0;

           for(index=0;index<5;index++)
           {
             Parameters.Pass[index] = Pass_buffer[index];
           }
           Parameters.Pass[4] = 0;

           if(Flash_WriteBuffer(PARAMETER__ADDRESS,&Parameters,sizeof(Parameters)) == O
K)
           {
             return 1;
           }
           else
           {
              if(Flash_WriteBuffer(PARAMETER__ADDRESS,&Parameters,sizeof(Parameters)) =
= OK)
              {
```

```
625.                return 1;
626.             }
627.             else
628.             {
629.                return 5;
630.             }
631.          }
632.        }
633.        else
634.        {
635.          return 2; // Wrong chekcsum
636.        }
637.     }
638.     else if(Buf[0]== '$'
639.     && Buf[1]== 'G'
640.     && Buf[2]== 'E'
641.     && Buf[3]== 'T'
642.     && Buf[4]== 'P'
643.     && Buf[5]== 'A'
644.     && Buf[6]== 'R'
645.     && Buf[7]== 'A'
646.     && Buf[8]== 'M'
647.     && Buf[9]== '['
648.     && Buf[10] ==  ']'
649.     && Buf[11] ==  '%' )
650.     {
651.        return 6; // Wrong chekcsum
652.     }
653.     else
654.     {
655.      return 3; // Wrong format
656.     }
657. }
658.
659. void Prepare_Pointer_For_Store(void)
660. {
661.   uint16_t current_CRC = 0;
662.   uint8_t write_bytes[10] = 0;
663.
664.
665.   Flash_Parameters.Area_Writes++; // Increase by 1 before we prepare to write on flash
666.
667.   write_bytes[4] = (uint8_t) (Flash_Parameters.Area_Writes>>24);
668.   write_bytes[3] = (uint8_t) (Flash_Parameters.Area_Writes>>16);
669.   write_bytes[2] = (uint8_t) (Flash_Parameters.Area_Writes>>8);
670.   write_bytes[1] = (uint8_t) (Flash_Parameters.Area_Writes);
671.   write_bytes[5] = (uint8_t) (Flash_Parameters.read_pointer>>8);
672.   write_bytes[4] = (uint8_t) (Flash_Parameters.read_pointer);
673.   write_bytes[7] = (uint8_t) (Flash_Parameters.write_pointer>>8);
674.   write_bytes[6] = (uint8_t) (Flash_Parameters.write_pointer);
675.   current_CRC = gen_crc16(write_bytes,8);
676.   write_bytes[9] = (uint8_t) (current_CRC>>8);
677.   write_bytes[8] = (uint8_t) (current_CRC);
678.
679.   Flash_State.locked_by             = BY_WRITE_OPERATION;
680.   Flash_Parameters.Size             = 10;
681.   Flash_Parameters.Address          = FIRST_AREA_CHECK__ADDRESS+((Flash_Parameters.Cur
    rrent_Area-1)*FLASH_SIZE_OF_PAGE);
682.
683.   if(Flash_Parameters.Area_Writes > MAXIMUM_WRITES_PER_FLASH_AREA) // It's time to go
    the next page....
684.   {
685.     Flash_Parameters.Currrent_Area++;
```

```
686.     Flash_Parameters.Area_Writes=0;
687.   }
688.   memcpy(Flash_Parameters.WriteData,write_bytes,Flash_Parameters.Size);
689. }
690.
691. void Clear_Velocity(uint8_t Clear_Counter)
692. {
693.     if(Clear_Counter)
694.     {
695.       Velocity_Measurements.counter = 0;
696.     }
697.     Velocity_Measurements.Acceleration_X = 0;
698.     Velocity_Measurements.Acceleration_Y = 0;
699.     Velocity_Measurements.Acceleration_Z = 0;
700.     Velocity_Measurements.Last_Acceleration_X = 0;
701.     Velocity_Measurements.Last_Acceleration_Y = 0;
702.     Velocity_Measurements.Last_Acceleration_Z = 0;
703.     Velocity_Measurements.Velocity_X = 0;
704.     Velocity_Measurements.Velocity_Y = 0;
705.     Velocity_Measurements.Velocity_Z = 0;
706.     Velocity_Timer = 0;
707. }
708.
709.
710. void Calculate_Speed(void)
711. {
712.     float cosx = cosf(Measure.GeneralInfo[Measure.Index].Pitch/100.0f/RAD_TO_DEG);
713.     float sinx = sinf(Measure.GeneralInfo[Measure.Index].Pitch/100.0f/RAD_TO_DEG);
714.     float cosy = cosf(Measure.GeneralInfo[Measure.Index].Roll/100.0f/RAD_TO_DEG);
715.     float siny = sinf(Measure.GeneralInfo[Measure.Index].Roll/100.0f/RAD_TO_DEG);
716.     float cosz = cosf(0);
717.     float sinz = sinf(0);
718.     float Alpha = 0.6;
719.     float coszcosx = cosz * cosx;
720.     float sinzcosx = sinz * cosx;
721.     float coszsinx = sinx * cosz;
722.     float sinzsinx = sinx * sinz;
723.     static unsigned char count_Xacc_same = 0;
724.     static unsigned char count_Yacc_same = 0;
725.     static unsigned char count_Zacc_same = 0;
726.     double time_now;
727.     time_now = Velocity_Timer;
728.     Velocity_Timer = 0;
729.     time_now = (double) ( time_now / 50000.0f);
730.     static char count_time_stopped = 0;
731.
732.     if(Velocity_Measurements.counter >= 20)
733.     {
734.             float mat[3][3];
735.             mat[0][0] = cosz * cosy;
736.             mat[0][1] = -cosy * sinz;
737.             mat[0][2] = siny;
738.             mat[1][0] = sinzcosx + (coszsinx * siny);
739.             mat[1][1] = coszcosx - (sinzsinx * siny);
740.             mat[1][2] = -sinx * cosy;
741.             mat[2][0] = (sinzsinx) - (coszcosx * siny);
742.             mat[2][1] = (coszsinx) + (sinzcosx * siny);
743.             mat[2][2] = cosy * cosx;
744.
745.             //Velocity_Measurements.Acceleration_X =
    (Velocity_Measurements.Acceleration_X *  Alpha) + ((1.0f -
    Alpha)*Velocity_Measurements.Last_Acceleration_X);
```

```
746.            Velocity_Measurements.Acceleration_X = (Measure.AccInfo[Measure.Index].X *
     mat[0][0] + Measure.AccInfo[Measure.Index].Y * mat[1][0] + Measure.AccInfo[Measure.In
    dex].Z * mat[2][0])*0.004392f; // Convert to m/s2;
747.            //Velocity_Measurements.Acceleration_Y =
    (Velocity_Measurements.Acceleration_Y *  Alpha) + ((1.0f -
    Alpha)*Velocity_Measurements.Last_Acceleration_Y);
748.            Velocity_Measurements.Acceleration_Y = (Measure.AccInfo[Measure.Index].X *
     mat[0][1] + Measure.AccInfo[Measure.Index].Y * mat[1][1] + Measure.AccInfo[Measure.In
    dex].Z * mat[2][1])*0.004392f; // Convert to m/s2;
749.            //Velocity_Measurements.Acceleration_Z =
    (Velocity_Measurements.Acceleration_Z *  Alpha) + ((1.0f -
    Alpha)*Velocity_Measurements.Last_Acceleration_Z);
750.            Velocity_Measurements.Acceleration_Z = (Measure.AccInfo[Measure.Index].X *
     mat[0][2] + Measure.AccInfo[Measure.Index].Y * mat[1][2] + Measure.AccInfo[Measure.In
    dex].Z * mat[2][2])*0.004392f; // Convert to m/s2;
751.            Velocity_Measurements.Acceleration_Z -= 9.80665;
752.
753.            Velocity_Measurements.Velocity_X += time_now*Velocity_Measurements.Acceler
    ation_X;
754.            Velocity_Measurements.Velocity_Y += time_now*Velocity_Measurements.Acceler
    ation_Y;
755.            Velocity_Measurements.Velocity_Z += time_now*Velocity_Measurements.Acceler
    ation_Z;
756.
757.
758.            if(abs(Velocity_Measurements.Acceleration_X-
    Velocity_Measurements.Last_Acceleration_X) < 3.0f)
759.            {
760.              count_Xacc_same++;
761.              if(count_Xacc_same >= 100)
762.              {
763.                count_Xacc_same = 0;
764.                count_Yacc_same = 0;
765.                count_Zacc_same = 0;
766.                Clear_Velocity(0);
767.              }
768.            }
769.            else
770.            {
771.                count_Xacc_same = 0;
772.            }
773.
774.
775.            if(abs(Velocity_Measurements.Acceleration_Y-
    Velocity_Measurements.Last_Acceleration_Y) < 3.0f)
776.            {
777.              count_Yacc_same++;
778.              if(count_Yacc_same >= 100)
779.              {
780.                count_Xacc_same = 0;
781.                count_Yacc_same = 0;
782.                count_Zacc_same = 0;
783.                Clear_Velocity(0);
784.              }
785.            }
786.            else
787.            {
788.                count_Yacc_same = 0;
789.            }
790.
791.
792.            if(abs(Velocity_Measurements.Acceleration_Z-
    Velocity_Measurements.Last_Acceleration_Z) < 3.0f)
793.            {
```

```
              count_Zacc_same++;
              if(count_Zacc_same >= 100)
              {
                count_Xacc_same = 0;
                count_Yacc_same = 0;
                count_Zacc_same = 0;
                Clear_Velocity(0);
              }
            }
            else
            {
                count_Zacc_same = 0;
            }


            Velocity_Measurements.Last_Acceleration_X = Velocity_Measurements.Accelera
tion_X;
            Velocity_Measurements.Last_Acceleration_Y = Velocity_Measurements.Accelera
tion_Y;
            Velocity_Measurements.Last_Acceleration_Z = Velocity_Measurements.Accelera
tion_Z;


            Velocity_Measurements.Velocity  =sqrtf(powf(Velocity_Measurements.Velocity
_X,2)+powf(Velocity_Measurements.Velocity_Y,2)+powf(Velocity_Measurements.Velocity_Z,2
));
            Measure.GeneralInfo[Measure.Index].Velocity = Velocity_Measurements.Veloci
ty;
    }
    else
    {

        Velocity_Measurements.counter++;
    }
}
```

12.5 bios.h

```
/**
*****************************************************************************
* @file           : bios.h
* @version        : V1.00
* @brief          : Savvas Kokkinidis - sabbaskok@hotmail.com
*****************************************************************************
**/

#include "stm32f1xx_hal.h"
#pragma pack(1)
/* Define to prevent recursive inclusion -------------------------------------*/
#ifndef __BIOS__H__
#define __BIOS__H__

#define TOTAL_FLASH_MEMORY 0x00800000
#define FLASH_SIZE_OF_PAGE 256
#define TOTAL_FLASH_PAGES  TOTAL_FLASH_MEMORY/FLASH_SIZE_OF_PAGE
#define FLASH_MAX_AREAS_PROINTERS   100
#define PAGE_AREAS_PER_PACKET       4
#define FIRST_AREA_CHECK__ADDRESS   TOTAL_FLASH_MEMORY-(110*FLASH_SIZE_OF_PAGE)
#define PARAMETER__ADDRESS  TOTAL_FLASH_MEMORY-(5*FLASH_SIZE_OF_PAGE)
#define DATA_SIZE_TO_STORE          708

#define UART_MAX_LEN 500

#define START_PACKET '1'
#define STOP_PACKET '2'

#define CLEAR_FLAG   1
#define CLEAR_NUMBER 1
#define KEEP_FLAG    0
#define KEEP_NUMBER  0

#define MAXIMUM_WRITES_PER_FLASH_AREA 90000
#define FIRST_FLASH_AREA_ID  1
#define SECOND_FLASH_AREA_ID 2
#define THIRD_FLASH_AREA_ID  3
#define FOURTH_FLASH_AREA_ID 4

#define QUEUE_SIZE ((TOTAL_FLASH_PAGES/4)-120) // 4 Pages per packet (it is 3 but data
   can be increased in future) - 120 pages (100 for Flash pointers + 1 for parameters  +
   19 for future use)

#define QUEUE_VALID 2
#define QUEUE_FULL  1
#define QUEUE_EMPTY 0

#define BY_NOONE               0
#define BY_WRITE_OPERATION     1

#define UART_FREE                               0x00
#define UART_WAITING_ANDRS_RES_COMMAND          0x01
#define UART_WAITING_STX                        0x02
#define UART_WAITING_ETX                        0x03
#define UART_WAITING_EOT                        0x04

#define TIME_BEFORE_SENDING                     20

#define SYSTEM_RUNNING_STATUS                   '1'
#define SYSTEM_STOPPED_STATUS                   '0'

#define PENDING       1
#define NOT_PENDING   0
```

```
62.
63.
64. #define SOH '$'
65. #define ENQ 'E'
66. #define ACK 'A'
67. #define NAK 'N'
68. #define STX '<'
69. #define ETX '>'
70. #define EOT '*'
71.
72. #define ENTER_COMMAND_MODE "$$$"
73. #define EXIT_COMMAND_MODE "---\r\n"
74.
75. #define INVALID_PACKET   0
76. #define DATA_PACKET     'D'
77. #define CMD_PACKET      'C'
78. #define RDME_PACKET     'R'
79. #define WRME_PACKET     'W'
80. #define ALIVE_PACKET    'A'
81.
82. #define CLEAR_PACKET_FLAG   0
83.
84. #define PERIOD_SAMPLES           5 // mSec
85. #define TICKING_SAMPLE           PERIOD_SAMPLES*20
86. #define TICKING_SAMPLE_1_5       PERIOD_SAMPLES/5
87. #define TICKING_SAMPLE_2_5       2*(PERIOD_SAMPLES/5)
88. #define TICKING_SAMPLE_3_5       3*(PERIOD_SAMPLES/5)
89. #define TICKING_SAMPLE_4_5       4*(PERIOD_SAMPLES/5)
90. #define TICKING_SAMPLE_5_5       PERIOD_SAMPLES
91.
92. #define TICKING_SAMPLE_HALF      PERIOD_SAMPLES/2
93. #define MAX_SAMPLES_PER_PACKET   50
94.
95. #define DATA_PACKET_ENQ_FLAG  0x01
96. #define DATA_PACKET_ACK_FLAG  0x02
97. #define DATA_PACKET_NAK_FLAG  0x03
98.
99. #define CMD_START_SENDING_FLAG   0x04
100.    #define CMD_STOP_SENDING_FLAG    0x05
101.    #define CMD_RESERVED_FLAG        0x06
102.
103.    #define RDME_PACKET_ENQ_FLAG  0x07
104.    #define RDME_PACKET_ACK_FLAG  0x08
105.    #define RDME_PACKET_NAK_FLAG  0x09
106.
107.    #define WRME_PACKET_ENQ_FLAG  0x0A
108.    #define WRME_PACKET_ACK_FLAG  0x0B
109.    #define WRME_PACKET_NAK_FLAG  0x0C
110.
111.    #define ALIVE_PACKET_ENQ_FLAG 0x0D
112.    #define ALIVE_PACKET_ACK_FLAG 0x0E
113.    #define ALIVE_PACKET_NAK_FLAG 0x0F
114.
115.
116.    #define CMD_PACKET_CAME       0x10
117.
118.    #define STRSTR                0x01
119.    #define STRCMP                0x00
120.
121.    //#define PC_ID 'P'
122.    #define AP_ID 'A'
123.    #define MY_ID 'B'
124.
125.
126.    #define FAILED           0
```

```
127.  #define OK               1
128.  #define TIMEOUT          2
129.  #define NOT_ALLOWED      4
130.
131.  #define BLUETOOTH_CHECK_STATUS_PERIOD 100*20
132.  #define BLUETOOTH_CONNECTED_TICKS     (BLUETOOTH_CHECK_STATUS_PERIOD*9)/10
133.
134.  #define BLUETOOTH_START             0
135.  #define BLUETOOTH_DISCOVERABLE      1
136.  #define BLUETOOTH_COMMAND_MODE      2
137.  #define BLUETOOTH_CONNECTED         3
138.  #define BLUETOOTH_NOT_CONNECTED     4
139.  #define BLUETOOTH_SPP_MODE          0
140.  #define BLUETOOTH_HCI_MODE          1
141.  #define BLUETOOTH_CMD_MODE          2
142.  #define BLUETOOTH_BR_115200         0
143.
144.  #define NOT_CHARGING     0
145.  #define CHARGING         1
146.  #define BATTERY_FULL     2
147.
148.  #define USB_NOT_PLUGGED  0
149.  #define USB_PLUGGED_IN   1
150.
151.
152.  #define FLASH_WRITE_START                0x00  //
153.  #define FLASH_WRITE_CHECK_ADDRESS        0x01
154.  #define FLASH_WRITE_CHECK_LENGTH         0x02
155.  #define FLASH_WRITE_BUFFER_FROM_MEMORY   0x03
156.  #define FLASH_COMPARE_BUFFER_TO_MEMORY   0x04
157.  #define FLASH_BUILD_IN_ERASE             0x05
158.  #define FLASH_BEFORE_COMPARE             0x06
159.  #define FLASH_READY_AND_COMPARE          0x07
160.  #define FLASH_BUFFER_TO_MEMORY_COMPARE   0x08
161.  #define FLASH_CHECK_COMPARE              0x09
162.  #define FLASH_WRITE_DONE                 0x0B
163.  #define FLASH_WRITE_FAILED               0x0C
164.  #define FLASH_CHECK_RETRIES              50
165.  #define FLASH_WRITE_MAX_FAILS            0x02
166.
167.  typedef struct
168.  {
169.     uint8_t  Status;
170.     uint8_t  IncommingBuffer[UART_MAX_LEN];
171.     uint16_t  Index;
172.     uint8_t  From_ID;
173.     uint8_t  To_ID;
174.     uint8_t  Response;
175.     uint8_t  Command;
176.     uint32_t PacketNumber;
177.     uint8_t  Flag;
178.     uint8_t  Checksum;
179.     uint8_t  CalculatedChecksum;
180.     uint8_t  DataStartIndex;
181.     uint8_t  DataEndIndex;
182.     uint8_t  DataLen;
183.  }UART;
184.
185.  typedef struct
186.  {
187.     uint8_t  Mode;
188.     uint8_t  Status;
189.     uint8_t  Pointers_Refresh_Pending;
190.     uint8_t  Answer_Pending;
191.     uint16_t Number_Pending;
```

```
192.      uint16_t Timeout_Per_Packet;
193.      uint16_t Time_Before_Start_Sending;
194.   }SysState;
195.
196.   typedef struct
197.   {
198.      uint8_t Protocol;
199.      uint8_t State;
200.      uint32_t BaudRate;
201.   }BL;
202.
203.   typedef struct
204.   {
205.      uint8_t Charging_Status;
206.      uint16_t  Battery_Voltage;
207.      uint8_t USB_Detected;
208.   }PowerManagement;
209.
210.   typedef struct
211.   {
212.      SysState        System;
213.      BL              Bluetooth;
214.      PowerManagement Power;
215.   }MainStruct;
216.
217.   typedef struct
218.   {
219.      int16_t    X;
220.      int16_t    Y;
221.      int16_t    Z;
222.   }AccelerometerMeasurements;
223.
224.   typedef struct
225.   {
226.      int16_t    X;
227.      int16_t    Y;
228.      int16_t    Z;
229.   }GyroscopeMeasurements;
230.
231.   typedef struct
232.   {
233.      int16_t    Pitch;
234.      int16_t    Roll;
235.      uint16_t   Velocity;
236.   }GeneralMeasurements;
237.
238.   typedef __packed struct
239.   {
240.           uint16_t Number;
241.           AccelerometerMeasurements AccInfo[MAX_SAMPLES_PER_PACKET];
242.           GeneralMeasurements       GeneralInfo[MAX_SAMPLES_PER_PACKET];
243.           uint16_t                  AnalogRead[MAX_SAMPLES_PER_PACKET];
244.           uint32_t                  Distance;
245.           int16_t                   Temperature;
246.           uint16_t Ticking;
247.           uint16_t Index;
248.           GyroscopeMeasurements     GyrInfo[MAX_SAMPLES_PER_PACKET];
249.   }Measurements;
250.
251.   typedef  struct
252.   {
253.      unsigned char counter;
254.      float Velocity_X;
255.      float Velocity_Y;
256.      float Velocity_Z;
```

```
257.      float Acceleration_X;
258.      float Acceleration_Y;
259.      float Acceleration_Z;
260.      float Last_Acceleration_X;
261.      float Last_Acceleration_Y;
262.      float Last_Acceleration_Z;
263.      float Velocity;
264.   }Velocity_struct;
265.
266.   typedef  struct
267.   {
268.      uint8_t Change_Name_Pass;
269.      uint8_t Name[15];
270.      uint8_t Pass[5];
271.   }parameters_struct;
272.
273.   typedef  struct
274.   {
275.      uint8_t  current_state;
276.      uint8_t  locked_by;
277.      uint16_t refresh;
278.      uint16_t fails;
279.      uint16_t retry;
280.      uint32_t ticks_now;
281.      uint32_t ticks_last;
282.   }state_info;
283.
284.   typedef  struct
285.   {
286.      uint8_t  WriteData[908];
287.      uint32_t Address;
288.      uint16_t Size;
289.      uint8_t  Currrent_Area;
290.      uint32_t Area_Writes;
291.      uint16_t read_pointer;
292.      uint16_t write_pointer;
293.      uint16_t bytes_available;
294.   }Flash_struct;
295.
296.   typedef  struct
297.   {
298.      uint16_t read_pointer;
299.      uint16_t write_pointer;
300.      uint16_t bytes_available;
301.   }ring_buffer;
302.
303.   uint8_t Read_ADC_Sample(void);
304.   void Send_Packet(uint8_t packet);
305.   uint16_t Build_Packet(void);
306.   void Convert_Hex_To_Ascii(uint8_t value,uint8_t *buffer);
307.   void Convert_Hex_To_Ascii_Measures(uint8_t value,uint16_t index);
308.   uint8_t Convert_Ascii_To_Hex(uint8_t *buffer);
309.   uint8_t Calculate_Checksum(uint8_t *buffer,uint16_t length);
310.   void Clear_UART(uint8_t Clear_Flag,uint8_t Clear_Number);
311.   void NOP_Delay(uint32_t mdelay);
312.   void SPI0_out(volatile unsigned char *Data, uint16_t DataLength);
313.   void SPI0_in(volatile unsigned char *Data, uint16_t DataLength);
314.   uint8_t Send_CMD_Bluetooth(uint8_t *buffer,uint8_t *answer,uint8_t cmp_str,uint16_t
    timeout);
315.   void SPIx_WriteData(uint8_t *DataIn, uint16_t DataLength);
316.   uint8_t Load_Parameters(void);
317.   void SPIx_ReadData(uint8_t *DataOut, uint16_t DataLength);
318.   uint16_t gen_crc16 (uint8_t *buffer, uint16_t size);
319.   void SPIx_Error (void);
320.   void Queue_Pop(void);
```

```
321.    void Queue_Push(void);
322.    uint8_t Queue_Status(void);
323.    void Prepare_Pointer_For_Store(void);
324.    void Reset_Flash_Queue(void);
325.    void Clear_Velocity(uint8_t Clear_Counter);
326.    uint8_t Check_And_Save_Parameters(uint8_t* Buf, uint32_t *Len);
327.    #endi
```

12.6 ad8555.c

```
/*******************************************************************************
* @file           : ad8555.c
* @version        : v1.0
* @brief          : Savvas Kokkinidis - sabbaskok@hotmail.com
*******************************************************************************/

#include "ad8555.h"
#include "stm32f1xx_hal.h"
#include "bios.h"


extern I2C_HandleTypeDef hi2c2;
extern  uint16_t counter_debug;
extern  MainStruct Device;
extern  UART       Uart;
extern  Measurements   Measure;



uint8_t AD8555_SendPacket(uint8_t Mode,uint8_t Function,uint8_t Value)
{
    // Do not accept Program Mode. It requires 5V supply and fuses cannot be changed if we make any mistake!!!
    if(Mode != AD8555_READ && Mode != AD8555_CHANGE_SENSE_CURRENT && Mode != AD8555_SIMULATE)
    {
      return 0;
    }
    else if(Function != AD8555_SSG_CODE
         && Function != AD8555_FSG_CODE
         && Function != AD8555_OFS_CODE
         && Function != AD8555_OTHER)
    {
      return 0;
    }
    AD8555_StartFrame();
    AD8555_SendParameter(Mode);
    AD8555_SendParameter(Function);
    AD8555_BitSend(1); // Dummy
    AD8555_BitSend(0); // Dummy
    AD8555_ParameterValue(Value);
    AD8555_StopFrame();
    return 1;
}


/*
  12bits Start Frame: 1 0 0 0   0 0 0 0   0 0 0 1
*/
void AD8555_StartFrame(void)
{
    uint8_t repeat;
    AD8555_BitSend(1);
    for(repeat = 0; repeat < 10; repeat++)
    {
      AD8555_BitSend(0);
    }
    AD8555_BitSend(1);
}

/*
  12bits Stop Frame: 0 1 1 1   1 1 1 1   1 1 1 0
*/
```

```
62. void AD8555_StopFrame(void)
63. {
64.     uint8_t repeat;
65.     AD8555_BitSend(0);
66.     for(repeat = 0; repeat < 10; repeat++)
67.     {
68.       AD8555_BitSend(1);
69.     }
70.     AD8555_BitSend(0);
71. }
72.
73. void AD8555_BitSend(uint8_t Bit)
74. {
75.     NOP_Delay(150); // ~20µs (Width between Pulses, >=10µs)
76.     HAL_GPIO_WritePin(GPIOA, GAIN_SEL_Pin, GPIO_PIN_SET);
77.     if(Bit)
78.     {
79.       NOP_Delay(600);   // ~85µs (Pulse Width for Loading 1 into Shift Register,
    >=50µs)
80.     }
81.     else
82.     {
83.       NOP_Delay(40);   // ~6µs (Pulse Width for Loading 0 into Shift Register, Range
    between 50 ns and 10µs )
84.     }
85.     HAL_GPIO_WritePin(GPIOA, GAIN_SEL_Pin, GPIO_PIN_RESET);
86. }
87.
88. void AD8555_SendParameter(uint8_t Value)
89. {
90.     AD8555_BitSend((Value >> 1) & 0x01);
91.     AD8555_BitSend(Value & 0x01);
92. }
93.
94. void AD8555_ParameterValue(uint8_t Value)
95. {
96.     uint8_t repeat = 8;
97.     do
98.     {
99.         repeat--;
100.            AD8555_BitSend((Value >> repeat) & 1);
101.        } while(repeat > 0);
102.    }
```

12.7 ad8555.h

```
/**
*****************************************************************************
* @file           : ad8555.h
* @version        : V1.00
* @brief          : Savvas Kokkinidis - sabbaskok@hotmail.com
*****************************************************************************
**/

#include "stm32f1xx_hal.h"

/* Define to prevent recursive inclusion -------------------------------------*/
#ifndef __AD8555__H__
#define __AD8555__H__


#define AD8555_CHANGE_SENSE_CURRENT 0
#define AD8555_SIMULATE             1
#define AD8555_PROGRAM              2
#define AD8555_READ         3

#define AD8555_SSG_CODE             0
#define AD8555_FSG_CODE             1
#define AD8555_OFS_CODE             2
#define AD8555_OTHER                3

#ifdef __cplusplus
 extern "C" {
#endif

uint8_t AD8555_SendPacket(uint8_t Mode,uint8_t Function,uint8_t Value);
void AD8555_StartFrame(void);
void AD8555_StopFrame(void);
void AD8555_BitSend(uint8_t Bit);
void AD8555_SendParameter(uint8_t Value);
void AD8555_ParameterValue(uint8_t Value);

#ifdef __cplusplus
}
#endif

#endif
```

12.8 flash.c

```
/*******************************************************************************
* @file           : flash.c
* @version        : v1.0
* @brief          : Savvas Kokkinidis - sabbaskok@hotmail.com
*******************************************************************************/

#include "stm32f1xx_hal.h"
#include "bios.h"

#define FLASH_CS_Lo()      HAL_GPIO_WritePin(GPIOA, SS_Pin, GPIO_PIN_RESET);
#define FLASH_CS_Hi()      HAL_GPIO_WritePin(GPIOA, SS_Pin, GPIO_PIN_SET);


extern SPI_HandleTypeDef hspi1;
extern  uint16_t counter_debug;
extern  MainStruct Device;
extern  UART       Uart;
extern  Measurements   Measure;

state_info Flash_State;
Flash_struct Flash_Parameters;

void DelayMicrosecond(uint32_t time);
void DelayMilisecond(uint32_t time);

//-----------------------------------------------------------------------------
//-----------------------------------------------------------------------------
//-----------------------------------------------------------------------------
//
//   Flash eeprom commands
//
const struct FlashCommands_Struct
{
    unsigned char FlashReadPage[1];

    unsigned char WriteBuffer1[2];
    unsigned char WriteBuffer2[2];
    unsigned char ReadBuffer1[2];
    unsigned char ReadBuffer2[2];

    unsigned char Status[1];
    unsigned char ManufacturerAndDevice_ID[1];

    unsigned char Erase[1];
    unsigned char BufferToMainMemoryPageProgramWithBultInErase[2];
    unsigned char MainMemoryPageToBufferTransfer[2];
    unsigned char MainMemoryPageToBufferCompare[2];
    unsigned char FlashContinousArrayRead;

    unsigned char WriteBuffer[2];
    unsigned char ReadBuffer[2];
}FlashCommand =
{
   {0xD2},       // FlashReadPage[1];
   {0x84, 0x00}, // WriteBuffer1[2];
   {0x87, 0x00}, // WriteBuffer2[2];
   {0xD4, 0x00}, // ReadBuffer1[2];
   {0xD6, 0x00}, // ReadBuffer2[2];
   {0xD7},       // Status[1];
   {0x9F},       // ManufacturerAndDevice_ID[1];
   {0x81},       // Erase[1];
   {0x83, 0x86}, // BufferToMainMemoryPageProgramWithBultInErase[2];
   {0x53, 0x55}, // MainMemoryPageToBufferTransfer[2];
```

```
64.   {0x60, 0x61}, // MainMemoryPageToBufferCompare[2];
65.   {0xE8},       // FlashContinousArrayRead;
66.   {0x84, 0x87}, // WriteBuffer[2];
67.   {0xD4, 0xD6}  // ReadBuffer[2];
68. };
69.
70.
71. const struct FlashMask_Struct_AT45DB641E
72. {
73.    unsigned char Ready;
74.    unsigned char IsReady;
75.
76.    unsigned char Communication;
77.    unsigned char Communication_OK;
78.    unsigned char Compare;
79.    unsigned char Compare_OK;
80. }FlashMask_AT45DB641E =
81. {
82.    0xBD, // Ready              10111101
83.    0xBC, //
    Is_Ready           10111100

84.    0x3D, // Communication      00111101
85.    0x3C, // Communication_OK   00111100
86.    0xFD, // Compare            11111101
87.    0xBC  // Compare_OK         10111100
88. };
89.
90. //*****************************************************
91. //                  Flash_Status
92. //*****************************************************
93. //
94. // The routine is used to check the flash's status
95. //
96. unsigned char Flash_Status(void)
97. {
98.    volatile unsigned char FlashResponse;
99.    volatile char          Result;
100.
101.         //DelayMicrosecond(1);
102.         FLASH_CS_Lo();
103.         DelayMicrosecond(1);
104.         SPI0_out((unsigned char *)FlashCommand.Status, sizeof(FlashCommand.Status));
105.         SPI0_in(&FlashResponse, sizeof(FlashResponse));
106.         DelayMicrosecond(1);
107.         FLASH_CS_Hi();
108.         DelayMicrosecond(1);
109.
110.      return FlashResponse;
111.   }
112.
113.
114.
115.   //*****************************************************
116.   //                  Flash_Get_Status
117.   //*****************************************************
118.   //
119.   //
120.   uint8_t Flash_Get_Status(unsigned char *buff, unsigned char len)
121.   {
122.        volatile char Result;
123.
124.        if (len > 1)
```

```
125.         {
126.            *buff = 0;
127.            return FAILED;
128.         }
129.
130.          FLASH_CS_Lo();
131.          DelayMicrosecond(5);
132.
133.          SPI0_out((unsigned char *)FlashCommand.Status, sizeof(FlashCommand.Status));
134.          SPI0_in(buff, 1);
135.
136.          DelayMicrosecond(5);
137.          FLASH_CS_Hi();
138.          DelayMicrosecond(5);
139.
140.
141.
142.
143.       return OK;
144.    }
145.
146.
147.
148.    //*****************************************************
149.    //      Flash_Get_Manufacturer_and_Device ID
150.    //*****************************************************
151.    //
152.    //
153.    uint8_t Flash_Get_Manufacturer_and_Device(unsigned char *buff, unsigned char len)
154.    {
155.       volatile char Result;
156.
157.       if (len > 4)
158.       {
159.          *buff = 0;
160.          return FAILED;
161.       }
162.          //DelayMicrosecond(1);
163.          FLASH_CS_Lo();
164.          DelayMicrosecond(1);                    // no delay need for flash to come
    online.
165.
166.          SPI0_out((unsigned char *)FlashCommand.ManufacturerAndDevice_ID, sizeof(Flash
    Command.ManufacturerAndDevice_ID));
167.          //DelayMicrosecond(1);
168.          SPI0_in(buff, 4);
169.
170.          DelayMicrosecond(1);
171.          FLASH_CS_Hi();
172.          DelayMicrosecond(1);
173.
174.       return OK;
175.    }
176.
177.
178.
179.    //*****************************************************
180.    //                  Flash_Ready
181.    //*****************************************************
182.    //
183.    // The routine checks if the flash is ready to accept new data
184.    //
185.    uint8_t Flash_Ready(uint16_t Efforts)
186.    {
187.       volatile unsigned char FlashResponse;
```

```
188.
189.       while(Efforts--)
190.       {
191.
192.          FlashResponse = Flash_Status();
193.
194.          if(FlashResponse>>7)
195.          {
196.             return OK;
197.          }
198.          DelayMicrosecond(10);
199.       }
200.       return FAILED;
201.    }
202.
203.
204.
205.    //*****************************************************
206.    //                  Flash_Fast_Communication
207.    //*****************************************************
208.    //
209.    // The routine checks if there is communication with the flash
210.    //
211.    uint8_t Flash_Fast_Communication(void)
212.    {
213.       volatile unsigned int  Efforts = 20;
214.       volatile unsigned char FlashResponse;
215.
216.       while(Efforts--)
217.       {
218.
219.          FlashResponse = Flash_Status();
220.
221.          if ((FlashResponse & FlashMask_AT45DB641E.Communication) == FlashMask_AT45DB6
    41E.Communication_OK)
222.          {
223.             return OK;
224.          }
225.       }
226.       return FAILED;
227.    }
228.
229.
230.
231.    //*****************************************************
232.    //                  Flash_Compare
233.    //*****************************************************
234.    //
235.    // The routine is used to verify if a previous write condition
236.    //
237.    uint8_t Flash_Compare(unsigned int Efforts)
238.    {
239.       volatile unsigned char FlashResponse;
240.
241.       DelayMicrosecond(20);
242.       while(Efforts--)
243.       {
244.
245.          FlashResponse = Flash_Status();
246.
247.          if (!((FlashResponse >> 6) & 0x01))
248.          {
249.            break;
250.          }
251.       }
```

```
252.
253.       if (Efforts > 0)
254.          return OK;
255.
256.       return FAILED;
257.    }
258.
259.
260.    //*****************************************************
261.    //                    Flash_WriteInternalBuffer
262.    //*****************************************************
263.    //
264.    // The routine is used to write the flash's RAM buffer and places them into '*data'
265.    //
266.    uint8_t Flash_WriteInternalBuffer(unsigned int Addr, unsigned char bo, unsigned cha
    r Be_Verified, unsigned char *data, volatile unsigned intlen)
267.    {
268.       volatile unsigned char address[5];
269.       volatile char Result;
270.
271.       //-----------------------------------------------------------------------------
272.       //-------------------------- check if flash is ready ------------------------
273.       //-----------------------------------------------------------------------------
    -
274.
275.
276.          if (Flash_Ready(1000) == FAILED || bo > 1 || len > FLASH_SIZE_OF_PAGE) //the
    selected buffer is not the buffer0 or the buffer1 or
277.                   return FAILED;                                          // data lenght
    is not less or equal to FLASH_SIZE_OF_PAGE then
278.                                                        // return error
    (FAILED)
279.
280.          //-----------------------------------------------------------------------------
    -
281.          //-------------------------- send data to ram flash buffer 1 ----------------
    -
282.          //-----------------------------------------------------------------------------
    -
283.
284.          address[0] = FlashCommand.WriteBuffer[bo];
285.          address[1] = 0;
286.          address[2] = (unsigned char) (Addr >> 8);
287.          address[3] = (unsigned char) Addr;
288.          //DelayMicrosecond(1);
289.          FLASH_CS_Lo();         //flash chip select LO
290.          DelayMicrosecond(1);
291.
292.          SPI0_out(address, 4);
293.          //DelayMicrosecond(1);
294.          SPI0_out(data, len);
295.
296.          DelayMicrosecond(1);
297.          FLASH_CS_Hi();
298.          DelayMicrosecond(1);
299.
300.
301.       return OK;
302.    }
303.
304.
305.    //*****************************************************
306.    //                  Flash_MainPageToBufferTranfer
307.    //*****************************************************
308.    //
```

```
309.    uint8_t Flash_MainPageToBufferTranfer(unsigned int Page, unsigned char bo)
310.    {
311.       volatile unsigned char address[4];
312.       volatile char Result;
313.
314.       if (bo > 1)
315.          return FAILED;
316.
317.
318.
319.          //-------------------------------------------------------------------------
    -
320.          //------------------------- check if flash is ready -----------------------
    -
321.          //-------------------------------------------------------------------------
    -
322.
323.               if (Flash_Ready(1000) == FAILED)
324.                   return FAILED;
325.
326.            address[0] = FlashCommand.MainMemoryPageToBufferTransfer[bo];
327.            address[1] = (unsigned char) ((Page >> 8) & 0x7F);
328.            address[2] = (unsigned char) (Page & 0xFF);
329.            address[3] = 0;
330.
331.            //-------------------------------------------------------------------------
    ---
332.            //------------------------- send data to ram flash buffer 1 -------------
    ---
333.            //-------------------------------------------------------------------------
    ---
334.            //DelayMicrosecond(1);
335.            FLASH_CS_Lo();         //flash chip select LO
336.                DelayMicrosecond(1);
337.
338.                SPI0_out(address, sizeof(address));
339.
340.                DelayMicrosecond(1);
341.            FLASH_CS_Hi();
342.            DelayMicrosecond(1);
343.
344.
345.
346.          return OK;
347.    }
348.
349.
350.
351.    //*********************************************************
352.    //      Flash_BufferToMainPageProgramWith_BuiltInErase
353.    //*********************************************************
354.    //
355.    uint8_t Flash_BufferToMainPageProgramWith_BuiltInErase(unsigned int Page, unsigned
    char bo)
356.    {
357.       volatile unsigned char address[4];
358.       volatile char Result;
359.
360.       if (bo > 1)
361.          return FAILED;
362.
363.
364.
365.          //-------------------------------------------------------------------------
    -
```

```
366.          //------------------------- check if flash is ready ----------------------
    -
367.          //------------------------------------------------------------------------
    -
368.                if (Flash_Ready(1000) == FAILED)
369.                  return FAILED;
370.
371.            address[0] = FlashCommand.BufferToMainMemoryPageProgramWithBultInErase[bo];
372.            address[1] = (unsigned char) ((Page >> 8) & 0x7F);
373.            address[2] = (unsigned char) ((Page) & 0xFF);
374.            address[3] = 0;
375.
376.            //------------------------------------------------------------------------
    ---
377.            //------------------------- send data to ram flash buffer 1 --------------
    ---
378.            //------------------------------------------------------------------------
    ---
379.            //DelayMicrosecond(1);
380.            FLASH_CS_Lo();          //flash chip select LO
381.                DelayMicrosecond(1);
382.
383.                SPI0_out(address, sizeof(address));
384.
385.                DelayMicrosecond(1);
386.            FLASH_CS_Hi();
387.            DelayMicrosecond(1);
388.
389.                Result = Flash_Ready(1000);
390.
391.
392.
393.       return Result;
394.    }
395.
396.
397.
398.
399.    //*****************************************************
400.    //          Flash_MainMemoryPageToBufferCompare
401.    //*****************************************************
402.    //
403.    uint8_t Flash_MainMemoryPageToBufferCompare(unsigned int Page, unsigned char bo)
404.    {
405.       volatile unsigned char address[4];
406.       volatile char Result;
407.
408.       if (bo > 1)
409.          return FAILED;
410.
411.    if (Flash_Ready(1000) == FAILED)
412.             return FAILED;
413.
414.          address[0] = FlashCommand.MainMemoryPageToBufferCompare[bo];
415.          address[1] = (unsigned char) ((Page >> 8) & 0x7F);
416.          address[2] = (unsigned char) ((Page) & 0xFF);
417.          address[3] = 0;
418.
419.          // Main Memory Page To Buffer1 Compare
420.          //DelayMicrosecond(1);
421.          FLASH_CS_Lo();
422.          DelayMicrosecond(1);
423.
424.          SPI0_out(address, sizeof(address));
425.
```

```
426.          DelayMicrosecond(1);
427.          FLASH_CS_Hi();
428.          DelayMicrosecond(1);
429.          Result = Flash_Ready(1000);
430.          if (Result == OK) Result = Flash_Compare(100);
431.
432.
433.       return Result;
434.    }
435.
436.
437.    //*****************************************************
438.    //                Flash_WriteBuffer
439.    //*****************************************************
440.    //
441.    char Flash_WriteBuffer(unsigned long AbsoluteAddress, void *AnyType, unsigned int l
    en)
442.    {
443.       unsigned char *buffer = AnyType;
444.       unsigned long x;
445.       unsigned int  Page;
446.       unsigned int  addr;
447.       signed char   Result = 1;
448.       char   Efforts;
449.       unsigned int  _Len = len;
450.
451.       //-------------------------------------------------------------------------------
452.       x = AbsoluteAddress;
453.       //-------------------------------------------------------------------------------
454.       while (x < (AbsoluteAddress + len))
455.       {
456.          //=========================================
457.          addr = x % FLASH_SIZE_OF_PAGE;
458.          Page = x / FLASH_SIZE_OF_PAGE;
459.          //=========================================
460.          for (Efforts = 1; Efforts < 6; Efforts++)
461.          {
462.             Result = Flash_MainPageToBufferTranfer(Page, 1);
463.             if (Result != OK)
464.             {
465.                DelayMicrosecond(300);
466.                continue;
467.             }
468.
469.             if ((_Len + addr) >= FLASH_SIZE_OF_PAGE)
470.                Result = Flash_WriteInternalBuffer(addr, 1, 1, buffer, FLASH_SIZE_OF_PA
    GE - addr);
471.             else if ((_Len + addr) < FLASH_SIZE_OF_PAGE)
472.                Result = Flash_WriteInternalBuffer(addr, 1, 1, buffer, _Len);
473.             else
474.                Result = FAILED;
475.
476.             if (Result == 1) Result = Flash_BufferToMainPageProgramWith_BuiltInErase(P
    age, 1);
477.             if (Result == 1) Result = Flash_MainMemoryPageToBufferCompare(Page, 1);
478.
479.             if (Result == OK)
480.                break;
481.
482.             DelayMicrosecond(300);
483.          }
484.          //=========================================
485.          if ((_Len + addr) >= FLASH_SIZE_OF_PAGE)
486.          {
487.             buffer += (FLASH_SIZE_OF_PAGE - addr);
```

```
488.             x      += (FLASH_SIZE_OF_PAGE - addr);
489.             _Len   -= (FLASH_SIZE_OF_PAGE - addr);
490.          }
491.          else if ((_Len + addr) < FLASH_SIZE_OF_PAGE)
492.          {
493.             buffer += _Len;
494.             x      += _Len;
495.             _Len    = 0;
496.          }
497.          //=========================================
498.          if (Efforts >= 6 || Result != OK)
499.             return FAILED;
500.          //=========================================
501.       }
502.       //-----------------------------------------------------------------------------
503.       return Result;
504.    }
505.    void DelayMicrosecond(uint32_t time)
506.    {
507.      time *= 8;
508.      do
509.      {
510.        __NOP();
511.      }
512.      while (time --);
513.    }
514.
515.    void DelayMilisecond(uint32_t time)
516.    {
517.        HAL_Delay(time);
518.    }
519.
520.    char Flash_ReadInternalBuffer(unsigned int Addr, unsigned char bo, unsigned char *d
    ata, unsigned int len)
521.    {
522.       volatile unsigned char address[5];
523.       volatile char Result;
524.
525.
526.          if (Flash_Ready(1000) == FAILED || bo > 1) // check if flash is ready
527.             return FAILED;
528.
529.          address[0] = FlashCommand.ReadBuffer[bo];
530.          address[1] = 0;
531.          address[2] = (unsigned char) (Addr >> 8);
532.          address[3] = (unsigned char) Addr;
533.          address[4] = 0;
534.          //DelayMicrosecond(1);
535.          FLASH_CS_Lo();         //flash chip select LO
536.          DelayMicrosecond(5);
537.
538.          SPI0_out(address, sizeof(address));
539.          //DelayMicrosecond(1);
540.          SPI0_in(data, len);
541.          DelayMicrosecond(5);
542.          FLASH_CS_Hi();
543.          DelayMicrosecond(5);
544.
545.
546.
547.       return OK;
548.    }
549.
550.    char Flash_Read_Bytes(unsigned long AbsoluteAddress, void *buffer, unsigned int len
    )
```

```
551.   {
552.      volatile unsigned char address[8];
553.      volatile unsigned int  AddressInPage;
554.      volatile unsigned int  Page;
555.      volatile unsigned char Effort;
556.      volatile char Result;
557.
558.      if (Flash_Ready(1000) == FAILED)
559.            return FAILED;
560.
561.          AddressInPage = AbsoluteAddress % FLASH_SIZE_OF_PAGE;
562.          Page          = AbsoluteAddress / FLASH_SIZE_OF_PAGE;
563.
564.          address[0] = FlashCommand.FlashContinousArrayRead;
565.          address[1] = (Page >> 8) & 0x7F;
566.          address[2] = (Page) & 0xFF;
567.          if (AddressInPage & 0x0100) address[2] |= 1;
568.          address[3] = (unsigned char) AddressInPage;
569.          address[4] = 0;
570.          address[5] = 0;
571.          address[6] = 0;
572.          address[7] = 0;
573.              FLASH_CS_Lo();
574.              DelayMicrosecond(1);
575.
576.             SPI0_out(address, sizeof(address));
577.             SPI0_in(buffer, len);
578.
579.             DelayMicrosecond(1);
580.             FLASH_CS_Hi();
581.             DelayMicrosecond(1);
582.
583.
584.        return OK;
585.   }
586.
587.
588.   uint8_t Flash_Write_Using_States(void)
589.   {
590.           static uint8_t Flash_Is_Ready = FAILED;
591.           static uint32_t x;
592.           static uint8_t *buffer;
593.           static uint16_t  Page;
594.           static uint16_t  addr;
595.           static uint16_t  _Len;
596.           static uint8_t address[4];
597.           static uint32_t TickLast = 0;
598.           static uint32_t TickNow = 0;
599.
600.
601.           TickNow = HAL_GetTick();
602.           if(TickNow <= (TickLast+5)) { return 0; }
603.           TickLast = TickNow;
604.
605.           switch (Flash_State.current_state)
606.           {
607.               case FLASH_WRITE_START:
608.                 if(Flash_State.locked_by == BY_WRITE_OPERATION)
609.                 {
610.                          Flash_State.current_state = FLASH_WRITE_CHECK_ADDRESS;
611.                          Flash_State.locked_by     = BY_WRITE_OPERATION;
612.                          Flash_State.retry         = FLASH_CHECK_RETRIES;
613.                          buffer = Flash_Parameters.WriteData;
614.                          x = Flash_Parameters.Address;
615.                          _Len = Flash_Parameters.Size;
```

```
616.                 }
617.               break;
618.               case FLASH_WRITE_CHECK_ADDRESS:
619.                 if((Flash_Parameters.Address+Flash_Parameters.Size) > TOTAL_FLASH_MEM
    ORY)
620.                 {
621.                    Flash_State.current_state = FLASH_WRITE_FAILED;
622.                 }
623.                 else
624.                 {
625.                    Flash_Is_Ready = FAILED;
626.                    Flash_State.current_state = FLASH_WRITE_CHECK_LENGTH;

627.                    Flash_State.retry   = FLASH_CHECK_RETRIES;
628.                 }
629.                break;
630.                case FLASH_WRITE_CHECK_LENGTH:
631.                    if(x < (Flash_Parameters.Address + Flash_Parameters.Size))
632.                    {
633.                      Flash_State.current_state = FLASH_WRITE_BUFFER_FROM_MEMORY;
634.                      addr = x % FLASH_SIZE_OF_PAGE;
635.                      Page = x / FLASH_SIZE_OF_PAGE;
636.                      Flash_Is_Ready = FAILED;
637.                    }
638.                    else
639.                    {
640.                      Flash_State.current_state = FLASH_WRITE_DONE;
641.                    }
642.                break;
643.                case FLASH_WRITE_BUFFER_FROM_MEMORY:
644.                  if(Flash_Is_Ready != OK)
645.                  {
646.                     Flash_Is_Ready = Flash_Ready(20);
647.                     if(Flash_Is_Ready != OK)
648.                     {
649.                       if(Flash_State.retry) { Flash_State.retry--; }
650.                       else
651.                       {
652.                         Flash_State.current_state = FLASH_WRITE_FAILED;
653.                       }
654.                     }
655.                     else
656.                     {
657.                        Flash_State.retry = FLASH_CHECK_RETRIES;
658.                     }
659.                  }
660.                  else
661.                  {
662.                           address[0] = FlashCommand.MainMemoryPageToBufferTransfer[1]
    ;
663.                           address[1] = (unsigned char) ((Page >> 8) & 0x7F);
664.                           address[2] = (unsigned char) (Page & 0xFF);
665.                           address[3] = 0;
666.
667.                           FLASH_CS_Lo();
668.                           DelayMicrosecond(1);
669.
670.                           SPI0_out(address, sizeof(address));
671.
672.                           DelayMicrosecond(1);
673.                           FLASH_CS_Hi();
674.                           DelayMicrosecond(1);
675.                           Flash_Is_Ready = FAILED;
676.                           Flash_State.current_state = FLASH_COMPARE_BUFFER_TO_MEMORY;
```

```
                        Flash_State.retry   = FLASH_CHECK_RETRIES;
                    }
                break;
                case FLASH_COMPARE_BUFFER_TO_MEMORY:
                    if(Flash_Is_Ready != OK)
                    {
                           Flash_Is_Ready = Flash_Ready(20);
                           if(Flash_Is_Ready != OK)
                           {
                             if(Flash_State.retry) { Flash_State.retry--; }
                             else
                             {
                               Flash_State.retry = FLASH_CHECK_RETRIES;
                               Flash_State.current_state = FLASH_WRITE_FAILED;
                             }
                           }
                           else
                           {
                              Flash_State.retry = FLASH_CHECK_RETRIES;
                           }
                    }
                    else
                    {
                        if ((_Len + addr) >= FLASH_SIZE_OF_PAGE)
                        {
                                 address[0] = FlashCommand.WriteBuffer[1];
                                 address[1] = 0;
                                 address[2] = (unsigned char) (addr >> 8);
                                 address[3] = (unsigned char) addr;
                                 FLASH_CS_Lo();
                                 DelayMicrosecond(1);
                                 SPI0_out(address, 4);
                                 SPI0_out(buffer, FLASH_SIZE_OF_PAGE - addr);

                                 DelayMicrosecond(1);
                                 FLASH_CS_Hi();
                                 DelayMicrosecond(1);
                        }
                        else if ((_Len + addr) < FLASH_SIZE_OF_PAGE)
                        {
                                 address[0] = FlashCommand.WriteBuffer[1];
                                 address[1] = 0;
                                 address[2] = (unsigned char) (addr >> 8);
                                 address[3] = (unsigned char) addr;
                                 FLASH_CS_Lo();
                                 DelayMicrosecond(1);
                                 SPI0_out(address, 4);
                                 SPI0_out(buffer, _Len);
                                 DelayMicrosecond(1);
                                 FLASH_CS_Hi();
                                 DelayMicrosecond(1);
                        }
                        Flash_Is_Ready = FAILED;
                        Flash_State.current_state = FLASH_BUILD_IN_ERASE;
                        Flash_State.retry   = FLASH_CHECK_RETRIES;
                    }
                break;
                case FLASH_BUILD_IN_ERASE:
                   if(Flash_Is_Ready == FAILED)
                   {
                           Flash_Is_Ready = Flash_Ready(20);
                           if(Flash_Is_Ready != OK)
                           {
                             if(Flash_State.retry) { Flash_State.retry--; }
                             else
```

```
741.                           {
742.                             Flash_State.retry = FLASH_CHECK_RETRIES;
743.                             Flash_State.current_state = FLASH_WRITE_FAILED;
744.                           }
745.                         }
746.                         else
747.                         {
748.                            Flash_State.retry = FLASH_CHECK_RETRIES;
749.                         }
750.                    }
751.                    else if(Flash_Is_Ready == OK)
752.                    {
753.                         address[0] = FlashCommand.BufferToMainMemoryPageProgramWithBu
    ltInErase[1];
754.                         address[1] = (unsigned char) ((Page >> 8) & 0x7F);
755.                         address[2] = (unsigned char) ((Page) & 0xFF);
756.                         address[3] = 0;
757.
758.                         FLASH_CS_Lo();
759.                         DelayMicrosecond(1);
760.                         SPI0_out(address, sizeof(address));
761.                         DelayMicrosecond(1);
762.                         FLASH_CS_Hi();
763.                         DelayMicrosecond(1);
764.                         Flash_Is_Ready = OK + 1;
765.                    }
766.                    else
767.                    {
768.                        if(Flash_Ready(20) != OK)
769.                         {
770.                           if(Flash_State.retry) { Flash_State.retry--; }
771.                           else
772.                           {
773.                             Flash_State.retry = FLASH_CHECK_RETRIES;
774.                             Flash_State.current_state = FLASH_WRITE_FAILED;
775.                           }
776.                         }
777.                         else
778.                         {
779.                           Flash_Is_Ready = FAILED;
780.                           Flash_State.current_state = FLASH_BEFORE_COMPARE;

781.                           Flash_State.retry    = FLASH_CHECK_RETRIES;

782.                         }
783.                    }
784.                break;
785.                case FLASH_BEFORE_COMPARE:
786.                   if(Flash_Is_Ready != OK)
787.                   {
788.                         Flash_Is_Ready = Flash_Ready(20);
789.                         if(Flash_Is_Ready != OK)
790.                         {
791.                           if(Flash_State.retry) { Flash_State.retry--; }
792.                           else
793.                           {
794.                             Flash_State.retry = FLASH_CHECK_RETRIES*5;
795.                             Flash_State.current_state = FLASH_WRITE_FAILED;
796.                           }
797.                         }
798.                         else
799.                         {
800.                            Flash_State.retry = FLASH_CHECK_RETRIES;
801.                         }
802.                    }
```

```
803.                    else
804.                    {
805.                            address[0] = FlashCommand.MainMemoryPageToBufferCompare[1];
806.                            address[1] = (unsigned char) ((Page >> 8) & 0x7F);
807.                            address[2] = (unsigned char) ((Page) & 0xFF);
808.                            address[3] = 0;
809.                            FLASH_CS_Lo();
810.                            DelayMicrosecond(1);
811.                            SPI0_out(address, sizeof(address));
812.                            DelayMicrosecond(1);
813.                            FLASH_CS_Hi();
814.                            DelayMicrosecond(1);
815.                            Flash_Is_Ready = FAILED;
816.                            Flash_State.retry = FLASH_CHECK_RETRIES;
817.                            Flash_State.current_state = FLASH_READY_AND_COMPARE;
818.                    }
819.                 break;
820.                 case FLASH_READY_AND_COMPARE:
821.                    if(Flash_Is_Ready != OK)
822.                    {
823.                          Flash_Is_Ready = Flash_Ready(20);
824.                          if(Flash_Is_Ready != OK)
825.                          {
826.                            if(Flash_State.retry) { Flash_State.retry--; }
827.                            else
828.                            {
829.                              Flash_State.retry = FLASH_CHECK_RETRIES;
830.                              Flash_State.current_state = FLASH_WRITE_FAILED;
831.                            }
832.                          }
833.                          else
834.                          {
835.                             Flash_State.retry = FLASH_CHECK_RETRIES;
836.                          }
837.                    }
838.                    else
839.                    {
840.                          if(Flash_Compare(5) == OK)
841.                          {
842.                            if ((_Len + addr) >= FLASH_SIZE_OF_PAGE)
843.                            {
844.                               buffer += (FLASH_SIZE_OF_PAGE - addr);
845.                               x      += (FLASH_SIZE_OF_PAGE - addr);
846.                               _Len   -= (FLASH_SIZE_OF_PAGE - addr);
847.                            }
848.                            else if ((_Len + addr) < FLASH_SIZE_OF_PAGE)
849.                            {
850.                               buffer += _Len;
851.                               x      += _Len;
852.                               _Len    = 0;
853.                            }
854.                            Flash_State.retry = FLASH_CHECK_RETRIES;
855.                            Flash_State.current_state = FLASH_WRITE_CHECK_LENGTH;
856.                          }
857.                          else
858.                          {
859.                            if(Flash_State.retry) { Flash_State.retry--; }
860.                            else
861.                            {
862.                              Flash_State.retry = FLASH_CHECK_RETRIES;
863.                              Flash_State.current_state = FLASH_WRITE_FAILED;
864.                            }
865.                          }
866.                    }
```

```
867.                 break;
868.                 case FLASH_WRITE_DONE:
869.                   Flash_State.current_state = FLASH_WRITE_START;
870.                   Flash_State.locked_by = BY_NOONE;
871.                 break;
872.                 case FLASH_WRITE_FAILED:
873.                   Flash_State.current_state = FLASH_WRITE_START;
874.                   Flash_State.locked_by = BY_NOONE;
875.                 break;
876.                 default: break;
877.           }
878.           return 1;
879.   }
880.
881.   uint8_t Get_and_Check_Flash_Working_Area(void)
882.   {
883.     uint8_t  change_area;
884.     uint8_t  count_errors;
885.     uint8_t  temp_buffer[10];
886.     uint16_t stored_crc;
887.     uint16_t calculated_crc;
888.     uint32_t temp_writes;
889.
890.     change_area=0;
891.
892.     while(change_area < FLASH_MAX_AREAS_PROINTERS)
893.     {
894.           count_errors=0;
895.           while(Flash_Read_Bytes(FIRST_AREA_CHECK__ADDRESS+(change_area*FLASH_SIZE_OF
    _PAGE),temp_buffer,10) != OK)
896.           {
897.             HAL_Delay(100);
898.             count_errors++;
899.             if(count_errors > 5)
900.             {
901.               return 0;
902.              }
903.           }
904.           temp_writes        = temp_buffer[3] << 8;
905.           temp_writes        |=  0x000000ff & temp_buffer[2];
906.           temp_writes        =temp_writes << 8;
907.           temp_writes        |=  0x000000ff & temp_buffer[1];
908.           temp_writes        =temp_writes << 8;
909.           temp_writes        |=  0x000000ff & temp_buffer[0];
910.
911.           Flash_Parameters.read_pointer        = temp_buffer[5] << 8;
912.           Flash_Parameters.read_pointer        |=  0x00ff & temp_buffer[4];
913.
914.           Flash_Parameters.write_pointer        = temp_buffer[7] << 8;
915.           Flash_Parameters.write_pointer        |=  0x00ff & temp_buffer[6];
916.
917.           stored_crc         = temp_buffer[9]<<8;
918.           stored_crc         |= temp_buffer[8];
919.           calculated_crc = gen_crc16(temp_buffer,8);
920.
921.           if(temp_writes == 0xFFFFFFFF && stored_crc == 0xFFFF) // Seems it is the
    first time
922.           {
923.             Flash_Parameters.Area_Writes = 0;
924.             Flash_Parameters.Currrent_Area = FIRST_FLASH_AREA_ID+change_area;
925.             return 1;
926.           }
927.           else if(stored_crc == calculated_crc)
928.           {
929.             if(temp_writes < MAXIMUM_WRITES_PER_FLASH_AREA)
```

```
930.             {
931.               Flash_Parameters.Area_Writes = temp_writes;
932.               Flash_Parameters.Currrent_Area = FIRST_FLASH_AREA_ID+change_area;
933.               return 1;
934.             }
935.           }
936.           change_area++;
937.     }
938.
939.     Flash_Parameters.Currrent_Area = FIRST_FLASH_AREA_ID;
940.     Flash_Parameters.bytes_available = 0;
941.     Flash_Parameters.read_pointer = 0;
942.     Flash_Parameters.write_pointer=0;
943.     Flash_Parameters.Area_Writes = 0;
944.     return 1;
945.   }
```

12.9 kalman.c

```
/*******************************************************************************
* @file           : kalman.c
* @version        : V1.0
* @brief          : Savvas Kokkinidis - sabbaskok@hotmail.com
*******************************************************************************/

#include "bios.h"
#include "kalman.h"
#include "stm32f1xx_hal.h"
#include "accelerometer_gyroscope.h"
#include <math.h>

extern I2C_HandleTypeDef hi2c2;
extern UART_HandleTypeDef huart1;
extern DMA_HandleTypeDef hdma_usart1_tx;
extern  MainStruct Device;
extern  UART       Uart;
extern  Measurements   Measure;
extern state_info Flash_State;
extern Flash_struct Flash_Parameters;
extern   uint8_t Debug_String[100];
extern uint8_t Debug_String_n;

#define RAD_TO_DEG      57.295779f
#define RESTRICT_PITCH

Kalman_struct KalmanX;
Kalman_struct KalmanY;

uint32_t Kalman_Timer = 0;

void Init_Kalman_Algorithm(void)
{
    double accX, accY, accZ;
    uint8_t Read_Buffer_I2C[6] =  {0,0,0,0,0,0};


    KalmanX.Q_angle      = 0.001f;
    KalmanX.Q_bias       = 0.003f;
    KalmanX.R_measure    = 0.03f;
    KalmanX.angle        = 0.0f;
    KalmanX.bias         = 0.0f;
    KalmanX.P[0][0]      = 0.0f;
    KalmanX.P[0][1]      = 0.0f;
    KalmanX.P[1][0]      = 0.0f;
    KalmanX.P[1][1]      = 0.0f;

    KalmanY.Q_angle      = 0.001f;
    KalmanY.Q_bias       = 0.003f;
    KalmanY.R_measure    = 0.03f;
    KalmanY.angle        = 0.0f;
    KalmanY.bias         = 0.0f;
    KalmanY.P[0][0]      = 0.0f;
    KalmanY.P[0][1]      = 0.0f;
    KalmanY.P[1][0]      = 0.0f;
    KalmanY.P[1][1]      = 0.0f;


    while(HAL_I2C_Mem_Read(&hi2c2, LSM6DS3_I2C_ADDRESS, LSM6DS3_ACC_GYRO_OUTX_L_XL, I2
C_MEMADD_SIZE_8BIT, Read_Buffer_I2C, 6, 5000) !=HAL_OK)
    {
      HAL_Delay(2);
    }
```

```
63.
64.     accX =   (int16_t) (Read_Buffer_I2C[1]<<8 | Read_Buffer_I2C[0]);
65.     accY =   (int16_t) (Read_Buffer_I2C[3]<<8 | Read_Buffer_I2C[2]);
66.     accZ =   (int16_t) (Read_Buffer_I2C[5]<<8 | Read_Buffer_I2C[4]);
67.
68.     #ifdef RESTRICT_PITCH
69.         double roll  = atan2(accY, accZ) * RAD_TO_DEG;
70.         double pitch = atan(-accX / sqrt(accY * accY + accZ * accZ)) * RAD_TO_DEG;
71.     #else
72.         double roll  = atan(accY / sqrt(accX * accX + accZ * accZ)) * RAD_TO_DEG;
73.         double pitch = atan2(-accX, accZ) * RAD_TO_DEG;
74.     #endif
75.
76.    KalmanX_setAngle(roll);
77.    KalmanY_setAngle(pitch);
78.    Kalman_Timer = 0;
79. }
80.
81. void Execute_Kalman_Filter(void)
82. {
83.   double time_now;
84.   double kalAngleX=0;
85.   double kalAngleY=0;
86.   double accX, accY, accZ;
87.   double gyroX,gyroY,gyroZ;
88.
89.   time_now = Kalman_Timer;
90.   Kalman_Timer = 0;
91.   time_now = (double) ( time_now / 20000.0f);
92.
93.   accX = (int16_t) Measure.AccInfo[Measure.Index].X;
94.   accY = (int16_t) Measure.AccInfo[Measure.Index].Y;
95.   accZ = (int16_t) Measure.AccInfo[Measure.Index].Z;
96.   gyroX = (int16_t) Measure.GyrInfo[Measure.Index].X;
97.   gyroY = (int16_t) Measure.GyrInfo[Measure.Index].Y;
98.   gyroZ = (int16_t) Measure.GyrInfo[Measure.Index].Z;
99.   #ifdef RESTRICT_PITCH // Eq. 25 and 26
100.        double roll  = atan2(accY, accZ) * RAD_TO_DEG;
101.        double pitch = atan(-accX / sqrt(accY * accY + accZ * accZ)) * RAD_TO_DEG;
102.    #else // Eq. 28 and 29
103.        double roll  = atan(accY / sqrt(accX * accX + accZ * accZ)) * RAD_TO_DEG;
104.        double pitch = atan2(-accX, accZ) * RAD_TO_DEG;
105.    #endif
106.
107.    double gyroXrate = gyroX * 0.007f; // Convert to deg/s
108.    double gyroYrate = gyroY * 0.007f; // Convert to deg/s
109.
110.    #ifdef RESTRICT_PITCH
111.
112.    if ((roll < -90 && kalAngleX > 90) || (roll > 90 && kalAngleX < -90))
113.    {
114.      KalmanX_setAngle(roll);
115.      kalAngleX = roll;
116.    }
117.    else
118.    {
119.      kalAngleX = KalmanX_getAngle(roll, gyroXrate, time_now); // Calculate the angle
    using a Kalman filter
120.    }
121.
122.    if (abs(kalAngleX) > 90)
123.    {
124.      gyroYrate = - gyroYrate; // Invert rate, so it fits the restriced accelerometer
    reading
125.    }
```

```
126.      kalAngleY = KalmanY_getAngle(pitch, gyroYrate, time_now);
127.      #else
128.        // This fixes the transition problem when the accelerometer angle jumps between
    -180 and 180 degrees
129.        if ((pitch < -90 && kalAngleY > 90) || (pitch > 90 && kalAngleY < -90))
130.        {
131.          KalmanY_setAngle(pitch);
132.          kalAngleY = pitch;
133.        }
134.        else
135.        {
136.          kalAngleY = KalmanY_getAngle(pitch, gyroYrate, time_now); // Calculate the
    angle using a Kalman filter
137.        }
138.
139.        if (abs(kalAngleY) > 90)
140.        {
141.          gyroXrate = -gyroXrate; // Invert rate, so it fits the restriced
    accelerometer reading
142.        }
143.        kalAngleX = KalmanX_getAngle(roll, gyroXrate, time_now); // Calculate the angle
    using a Kalman filter
144.      #endif
145.
146.       Measure.GeneralInfo[Measure.Index].Roll  =  (int16_t) (kalAngleX*100);
147.       Measure.GeneralInfo[Measure.Index].Pitch  = (int16_t) (kalAngleY*100);
148.
149.    }
150.
151.
152.    float KalmanX_getAngle(float newAngle, float newRate, float dt)
153.    {
154.        KalmanX.rate = newRate - KalmanX.bias;
155.        KalmanX.angle += dt * KalmanX.rate;
156.
157.        KalmanX.P[0][0] += dt * (dt*KalmanX.P[1][1] - KalmanX.P[0][1] - KalmanX.P[1][0]
     + KalmanX.Q_angle);
158.        KalmanX.P[0][1] -= dt * KalmanX.P[1][1];
159.        KalmanX.P[1][0] -= dt * KalmanX.P[1][1];
160.        KalmanX.P[1][1] += KalmanX.Q_bias * dt;
161.
162.        float S = KalmanX.P[0][0] + KalmanX.R_measure; // Estimate error
163.
164.        float K[2]; // Kalman gain - This is a 2x1 vector
165.        K[0] = KalmanX.P[0][0] / S;
166.        K[1] = KalmanX.P[1][0] / S;
167.
168.        /* Step 3 */
169.        float y = newAngle - KalmanX.angle; // Angle difference
170.        /* Step 6 */
171.        KalmanX.angle += K[0] * y;
172.        KalmanX.bias += K[1] * y;
173.
174.        /* Step 7 */
175.        float P00_temp = KalmanX.P[0][0];
176.        float P01_temp = KalmanX.P[0][1];
177.
178.        KalmanX.P[0][0] -= K[0] * P00_temp;
179.        KalmanX.P[0][1] -= K[0] * P01_temp;
180.        KalmanX.P[1][0] -= K[1] * P00_temp;
181.        KalmanX.P[1][1] -= K[1] * P01_temp;
182.
183.        return KalmanX.angle;
184.    }
185.
```

```
186.    float KalmanY_getAngle(float newAngle, float newRate, float dt)
187.    {
188.        KalmanY.rate = newRate - KalmanY.bias;
189.        KalmanY.angle += dt * KalmanY.rate;
190.
191.        KalmanY.P[0][0] += dt * (dt*KalmanY.P[1][1] - KalmanY.P[0][1] - KalmanY.P[1][0]
    + KalmanY.Q_angle);
192.        KalmanY.P[0][1] -= dt * KalmanY.P[1][1];
193.        KalmanY.P[1][0] -= dt * KalmanY.P[1][1];
194.        KalmanY.P[1][1] += KalmanY.Q_bias * dt;
195.
196.
197.
198.        float S = KalmanY.P[0][0] + KalmanY.R_measure; // Estimate error
199.
200.        float K[2]; // Kalman gain - This is a 2x1 vector
201.        K[0] = KalmanY.P[0][0] / S;
202.        K[1] = KalmanY.P[1][0] / S;
203.
204.        /* Step 3 */
205.        float y = newAngle - KalmanY.angle; // Angle difference
206.        /* Step 6 */
207.        KalmanY.angle += K[0] * y;
208.        KalmanY.bias += K[1] * y;
209.
210.        /* Step 7 */
211.        float P00_temp = KalmanY.P[0][0];
212.        float P01_temp = KalmanY.P[0][1];
213.
214.        KalmanY.P[0][0] -= K[0] * P00_temp;
215.        KalmanY.P[0][1] -= K[0] * P01_temp;
216.        KalmanY.P[1][0] -= K[1] * P00_temp;
217.        KalmanY.P[1][1] -= K[1] * P01_temp;
218.
219.        return KalmanY.angle;
220.    }
221.
222.
223.    // Y functions
224.    void KalmanY_setAngle(float angle)
225.    {
226.      KalmanY.angle = angle;
227.    }
228.
229.    float KalmanY_getRate(void)
230.    {
231.      return KalmanY.rate;
232.    }
233.
234.
235.    void KalmanY_setQangle(float Q_angle)
236.    {
237.      KalmanY.Q_angle = Q_angle;
238.    }
239.
240.    void KalmanY_setQbias(float Q_bias)
241.    {
242.      KalmanY.Q_bias = Q_bias;
243.    }
244.
245.    void KalmanY_setRmeasure(float R_measure)
246.    {
247.      KalmanY.R_measure = R_measure;
248.    }
249.
```

```
250.  float KalmanY_getQangle(void)
251.  {
252.    return KalmanY.Q_angle;
253.  }
254.
255.  float KalmanY_getQbias(void)
256.  {
257.    return KalmanY.Q_bias;
258.  }
259.
260.  float KalmanY_getRmeasure(void)
261.  {
262.    return KalmanY.R_measure;
263.  }
264.
265.
266.  // X functions
267.  void KalmanX_setAngle(float angle)
268.  {
269.    KalmanX.angle = angle;
270.  }
271.
272.  float KalmanX_getRate(void)
273.  {
274.    return KalmanX.rate;
275.  }
276.
277.
278.  void KalmanX_setQangle(float Q_angle)
279.  {
280.    KalmanX.Q_angle = Q_angle;
281.  }
282.
283.  void KalmanX_setQbias(float Q_bias)
284.  {
285.    KalmanX.Q_bias = Q_bias;
286.  }
287.
288.  void KalmanX_setRmeasure(float R_measure)
289.  {
290.    KalmanX.R_measure = R_measure;
291.  }
292.
293.  float KalmanX_getQangle(void)
294.  {
295.    return KalmanX.Q_angle;
296.  }
297.
298.  float KalmanX_getQbias(void)
299.  {
300.    return KalmanX.Q_bias;
301.  }
302.
303.  float KalmanX_getRmeasure(void)
304.  {
305.    return KalmanX.R_measure;
306.  }
```

12.10 kalman.h

```
/**
********************************************************************************
* @file           : kalman.h
* @version        : V1.00
* @brief          : Savvas Kokkinidis - sabbaskok@hotmail.com
********************************************************************************
**/

#include "stm32f1xx_hal.h"

/* Define to prevent recursive inclusion -------------------------------------*/
#ifndef __KALMAN__H__
#define __KALMAN__H__



typedef  struct
{
    float Q_angle; // Process noise variance for the accelerometer
    float Q_bias; // Process noise variance for the gyro bias
    float R_measure; // Measurement noise variance - this is actually the variance of the measurement noise

    float angle; // The angle calculated by the Kalman filter - part of the 2x1 state vector
    float bias; // The gyro bias calculated by the Kalman filter - part of the 2x1 state vector
    float rate; // Unbiased rate calculated from the rate and the calculated bias - you have to call getAngle to update the rate

    float P[2][2]; // Error covariance matrix - This is a 2x2 matrix
}Kalman_struct;


void Init_Kalman_Algorithm(void);
float KalmanX_getAngle(float newAngle, float newRate, float dt);
void KalmanX_setAngle(float angle);
float KalmanX_getRate(void);
void KalmanX_setQangle(float Q_angle);
void KalmanX_setQbias(float Q_bias);
void KalmanX_setRmeasure(float R_measure);
float KalmanX_getQangle(void);
float KalmanX_getQbias(void);
float KalmanX_getRmeasure(void);
float KalmanY_getAngle(float newAngle, float newRate, float dt);
void KalmanY_setAngle(float angle);
float KalmanY_getRate(void);
void KalmanY_setQangle(float Q_angle);
void KalmanY_setQbias(float Q_bias);
void KalmanY_setRmeasure(float R_measure);
float KalmanY_getQangle(void);
float KalmanY_getQbias(void);
float KalmanY_getRmeasure(void);


#ifdef __cplusplus
}
#endif

#endif
```

16. Παράρτημα Z – Διαστάσεις και Βάρος

- **Μήκος**: 62.5mm
- **Πλάτος:** 21.1mm
- **Ύψος:** 15mm
- **Βάρος:** 13g