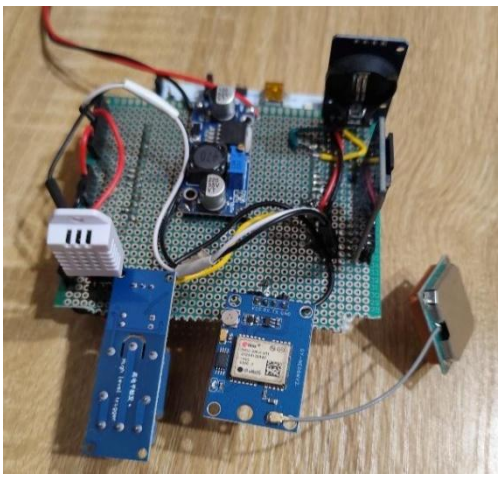
ΔΙΕΘΝΕΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΤΗΣ ΕΛΛΑΔΟΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Σύστημα ελέγχου ρελέ με τον μικροελεγκτή STM32 και δημιουργία SD bootloader



Του φοιτητή Ιορδανίδη Χριστόφορου

Αρ. Μητρώου: 512038

Επιβλέπων

Γιακουμής Άγγελος

Επίκουρος Καθηγητής

Σεπτέμβριος 2025

Τίτλος Π.Ε. Σύστημα ελέγχου ρελέ με τον μικροελεγκτή STM32 και δημιουργία SD bootloader

Κωδικός Π.Ε. 24335

Ιορδανίδης Χριστόφορος

Γιακουμής Άγγελος

Ημερομηνία ανάληψης Π.Ε. 29/12/2024

Ημερομηνία περάτωσης Π.Ε. 11/09/2025

Βεβαιώνω ότι είμαι ο συγγραφέας αυτής της εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, έχω καταγράψει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών, εικόνων και κειμένου, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επιπλέον, βεβαιώνω ότι αυτή η εργασία προετοιμάστηκε από εμένα προσωπικά, ειδικά ως πτυχιακή εργασία, στο Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του ΔΙ.ΠΑ.Ε.

Η έγκριση της πτυχιακής εργασίας από το Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του Διεθνούς Πανεπιστημίου της Ελλάδος, δεν υποδηλώνει απαραιτήτως και αποδοχή των απόψεων του συγγραφέα, εκ μέρους του Τμήματος.

Πρόλογος

Τα ενσωματωμένα συστήματα αποτελούν έναν από τους ταχύτερα αναπτυσσόμενους τομείς της επιστήμης και της τεχνολογίας, με ευρεία εφαρμογή στη βιομηχανία, την υγεία, τις επικοινωνίες και την καθημερινή ζωή. Η συνεχής εξέλιξη των μικροελεγκτών έχει επιτρέψει την ανάπτυξη ολοένα και πιο αποδοτικών, αξιόπιστων και ενεργειακά αυτόνομων συσκευών, οι οποίες μπορούν να ανταποκριθούν σε απαιτήσεις αυτοματισμού, συλλογής δεδομένων και απομακρυσμένου ελέγχου.

Σημαντικό ρόλο σε αυτές τις εφαρμογές διαδραματίζουν οι μηχανισμοί προγραμματισμού και αναβάθμισης, καθώς η δυνατότητα εύκολης ενημέρωσης του λογισμικού αποτελεί κρίσιμο παράγοντα για τη συντήρηση, την ασφάλεια και τη μακροχρόνια λειτουργία ενός συστήματος. Η ενσωμάτωση bootloader σε μικροελεγκτές προσφέρει μία αποδοτική και ευέλικτη λύση, επιτρέποντας τη διαχείριση ενημερώσεων χωρίς την ανάγκη εξειδικευμένων εργαλείων.

Παράλληλα, η χρήση ρελέ σε συνδυασμό με μικροελεγκτές εξακολουθεί να αποτελεί θεμελιώδες στοιχείο σε πληθώρα εφαρμογών, καθώς δίνει τη δυνατότητα ελέγχου ηλεκτρικών φορτίων με απλό και αξιόπιστο τρόπο. Η αξιοποίηση αισθητήρων και συστημάτων πραγματικού χρόνου ενισχύει περαιτέρω τις δυνατότητες των ενσωματωμένων λύσεων, επιτρέποντας την αυτοματοποίηση διαδικασιών και την ακριβή καταγραφή δεδομένων.

Η παρούσα εργασία εντάσσεται σε αυτό το πλαίσιο, με στόχο την ανάδειξη της συμβολής των σύγχρονων μικροελεγκτών στη δημιουργία ολοκληρωμένων και επεκτάσιμων συστημάτων αυτοματισμού.

Περίληψη

Η παρούσα πτυχιακή εργασία έχει ως αντικείμενο την ανάπτυξη ενός συστήματος ελέγχου ρελέ με τον μικροελεγκτή STM32, σε συνδυασμό με τη δημιουργία ενός SD bootloader για την ενημέρωση του firmware. Το σύστημα στοχεύει στη συλλογή και καταγραφή δεδομένων από αισθητήρες θερμοκρασίας και υγρασίας, ενώ οι ρυθμίσεις του ρελέ (ενεργοποίηση και απενεργοποίηση) φορτώνονται δυναμικά από κάρτα SD. Με τον τρόπο αυτό επιτυγχάνεται ευελιξία στη λειτουργία και δυνατότητα επαναπρογραμματισμού χωρίς τη χρήση εξωτερικού προγραμματιστή.

Η υλοποίηση βασίστηκε στον μικροελεγκτή STM32F401RE, ο οποίος υποστηρίζει λειτουργίες χαμηλής κατανάλωσης και πλήθος περιφερειακών, καθιστώντας τον κατάλληλο για εφαρμογές καταγραφής δεδομένων. Το λογισμικό περιλαμβάνει την υποστήριξη του συστήματος αρχείων FatFS, για τη διαχείριση της κάρτας SD, ενώ η συλλογή δεδομένων πραγματοποιείται με τον αισθητήρα DHT22. Επιπλέον, το σύστημα χρησιμοποιεί το RTC DS3231 για την ακριβή χρονοσήμανση των μετρήσεων, το οποίο ενημερώνεται αυτόματα σε εβδομαδιαία βάση μέσω GPS, εξασφαλίζοντας την ακρίβεια χρόνου.

Ο bootloader που αναπτύχθηκε επιτρέπει την αυτόματη φόρτωση νέου κώδικα από την κάρτα SD, βελτιώνοντας σημαντικά τη διαδικασία ενημέρωσης και μειώνοντας την ανάγκη εξειδικευμένου εξοπλισμού. Η αρχιτεκτονική του συστήματος συνδυάζει τόσο το υλικό (αισθητήρες, RTC, GPS, relay) όσο και το λογισμικό (οδηγοί επικοινωνίας, διαχείριση αρχείων, λογική ελέγχου), προσφέροντας μια ολοκληρωμένη λύση.

Τα αποτελέσματα της εργασίας καταδεικνύουν ότι το προτεινόμενο σύστημα μπορεί να λειτουργήσει ως αξιόπιστο και αποδοτικό καταγραφικό δεδομένων, με δυνατότητα ελέγχου ρελέ και εύκολης αναβάθμισης του firmware. Παράλληλα, η υλοποίηση αναδεικνύει τα πλεονεκτήματα της χρήσης STM32 σε εφαρμογές αυτοματισμού, συνδυάζοντας χαμηλή κατανάλωση, επεκτασιμότητα και ακρίβεια.

Relay control system with the STM32 microcontroller and creation of SD bootloader

Iordanidis Christoforos

Abstract

This thesis focuses on the development of a relay control system using the STM32 microcontroller, combined with the creation of an SD bootloader for firmware updates. The system is designed to collect and record data from temperature and humidity sensors, while the relay settings (activation and deactivation) are dynamically loaded from an SD card. This approach provides operational flexibility and enables reprogramming without the need for an external programmer.

The implementation is based on the STM32F401RE microcontroller, which supports low-power modes and a wide range of peripherals, making it suitable for data logging applications. The software integrates the FatFS file system for SD card management, while data collection is performed with the DHT22 sensor. In addition, the system employs the DS3231 RTC for accurate timestamping of measurements, which is automatically updated on a weekly basis via GPS, ensuring precise time synchronization.

The developed bootloader enables the automatic loading of new code from the SD card, significantly improving the update process and reducing the need for specialized equipment. The system architecture combines both hardware (sensors, RTC, GPS, relay) and software (communication drivers, file management, control logic), providing a comprehensive solution. The results demonstrate that the proposed system can operate as a reliable and efficient data logger with relay control and easy firmware upgrade capability. At the same time, the implementation highlights the advantages of using STM32 in automation applications, combining low power consumption, scalability, and accuracy.

Ευχαριστίες

Θα ήθελα να εκφράσω τις θερμές μου ευχαριστίες στον επιβλέποντα καθηγητή μου, Γιακουμή Άγγελο, για την πολύτιμη καθοδήγηση και την υποστήριξή του καθ' όλη τη διάρκεια της εκπόνησης της παρούσας εργασίας. Η εμπειρία και οι γνώσεις του ήταν καθοριστικές για την επιτυχή ολοκλήρωσή της.

Επιπλέον, ευχαριστώ την οικογένειά μου και τους φίλους μου για την ηθική υποστήριξη και την υπομονή τους. Η αμέριστη αγάπη και η ενθάρρυνσή τους ήταν απαραίτητη για να ξεπεράσω τις δυσκολίες και να φτάσω σε αυτό το σημείο.

Τέλος, ευχαριστώ όλους όσους συνέβαλαν με οποιονδήποτε τρόπο στην εκπόνηση της εργασίας μου.

Περιεχόμενα

1 Εισαγωγή

1.1 Σκοπός και στόχοι του συστήματος

Ο σκοπός της παρούσας πτυχιακής0 εργασίας είναι η δημιουργία ενός συστήματος καταγραφής πληροφοριών πλαισίου ενός χώρου (είτε εσωτερικού είτε εξωτερικού), όπως θερμοκρασία και υγρασία. Ο στόχος του συστήματος αυτού είναι ο χρήστης να έχει την δυνατότητα με μία απλή διαδικασία να είναι σε θέση και να το επαναπρογραμματίζει αλλά και να μπορεί να συλλέγει τις μετρήσεις των αισθητήρων. Με βάση τα παραπάνω, το σύστημα προορίζεται για παρακολούθηση χώρων οι οποίοι δεν έχουν απευθείας πρόσβαση στο δίκτυο.

1.2 Σύντομη περιγραφή της λειτουργίας του συστήματος

Η παραπάνω πτυχιακή εργασία περιλαμβάνει την ανάπτυξη ενός συστήματος ελέγχου ρελέ με τον μικροελεγκτή STM32 και τη δημιουργία ενός SD bootloader. Οι ρυθμίσεις του ρελέ (ώρα ενεργοποίησης, ώρα απενεργοποίησης) φορτώνονται από κάρτα SD, ενώ το σύστημα καταγράφει τα δεδομένα του αισθητήρα (θερμοκρασία, υγρασία). Ο προγραμματισμός του STM32 πραγματοποιείται μέσω του SD bootloader, διευκολύνοντας την διαδικασία ενημέρωσης του κώδικα. Επιπλέον, το ρολόι πραγματικού χρόνου (RTC) του συστήματος ενημερώνεται εβδομαδιαία μέσω GPS, διασφαλίζοντας την ακρίβεια του συστήματος.

1.2.1 Επισκόπηση του Συστήματος Καταγραφής Δεδομένων

Ένα σύστημα καταγραφής δεδομένων είναι υπεύθυνο για την περιοδική συλλογή και αποθήκευση δεδομένων από αισθητήρες. Ένα τέτοιο σύστημα μπορεί να σχεδιαστεί με την ακόλουθη δομή:

- **Συλλογή Δεδομένων Αισθητήρων:** Ανάγνωση δεδομένων αισθητήρων οι οποίοι είναι συνδεδεμένοι σε έναν μικροελεγκτή χρησιμοποιώντας ένα πρωτόκολλο επικοινωνίας (I2C, SPI, ADC κ.α).
- **Αποθήκευση Δεδομένων:** Αποθήκευση των συλλεχθέντων δεδομένων σε μνήμη ROM (π.χ EEPROM, FlashROM).
- **Περιοδική Καταγραφή Δεδομένων:** Χρήση ενός χρονοδιακόπτη (timer) η ενός ρολογιού (RTC) για την καταγραφή δεδομένων σε τακτά χρονικά διαστήματα.
- **Διαχείριση και Ανάκτηση Δεδομένων και Καταγραφών:** Ανάκτηση και διαχείριση αποθηκευμένων δεδομένων όταν χρειάζεται. Διάθεση νέου χώρου αποθήκευσης όταν η μνήμη φτάσει σε ένα συγκεκριμένο όριο και εκκαθάριση παλαιών δεδομένων.

1.3 Παρόμοια Συστήματα

Τα συστήματα καταγραφής δεδομένων (data logging systems) αποτελούν βασικό στοιχείο πολλών εφαρμογών στον τομέα της βιομηχανίας, της γεωργίας, των έξυπνων κτιρίων και των περιβαλλοντικών μετρήσεων. Κοινός στόχος αυτών των συστημάτων είναι η συνεχής ή περιοδική μέτρηση, αποθήκευση και συχνά μετάδοση δεδομένων, όπως θερμοκρασία, υγρασία, ένταση φωτός, τάση, ρεύμα, πίεση κ.ά.

Υπάρχουν πολλές έτοιμες λύσεις στην αγορά, είτε ως εμπορικά προϊόντα, είτε ως ανοιχτού κώδικα πλατφόρμες, με διάφορους βαθμούς επεκτασιμότητας, ακρίβειας και κόστους.

1.3.1 Arduino-based Data Loggers

Οι πιο διαδεδομένες λύσεις βασίζονται σε πλακέτες **Arduino**, σε συνδυασμό με αισθητήρες και μονάδες αποθήκευσης όπως **SD κάρτες**. Υπάρχουν πλήθος βιβλιοθηκών για ευκολία ανάπτυξης, ενώ χρησιμοποιούνται συχνά σε εκπαιδευτικά ή ημιεπαγγελματικά περιβάλλοντα. Παραδείγματα:

- **Adafruit Data logger Shield**: Επιτρέπει την καταγραφή δεδομένων από αισθητήρες μέσω Arduino και αποθήκευση σε SD κάρτα, με υποστήριξη ρολογιού RTC για χρονοσήμανση.
- **OpenLog**: Συσκευή χαμηλού κόστους που καταγράφει σειριακά δεδομένα απευθείας σε αρχείο κειμένου σε κάρτα SD.

1.3.2 Εμπορικά Βιομηχανικά Συστήματα

Πιο εξελιγμένα συστήματα διατίθενται από εταιρείες όπως η **National Instruments**, η **Dataq Instruments**, και η **HOBO**. Αυτά περιλαμβάνουν:

- **Ακριβείς μετατροπείς A/D**, με υποστήριξη για πολλαπλά κανάλια.
- **Ενσωματωμένη ασύρματη επικοινωνία** (π.χ. Wi-Fi, GSM, LoRa).
- **Ενεργειακή αυτονομία** με μπαταρία και λειτουργία χαμηλής κατανάλωσης.
- **Πιστοποιήσεις ανθεκτικότητας** για βιομηχανικά ή εξωτερικά περιβάλλοντα.

1.3.3 Σύγχρονες IoT Πλατφόρμες

Τα τελευταία χρόνια έχουν αναπτυχθεί συστήματα βασισμένα σε **IoT μικροελεγκτές** όπως οι **ESP32** και **STM32**, τα οποία συνδυάζουν:

- Ενσωματωμένες διεπαφές Wi-Fi/Bluetooth ή και LoRa.
- Συγχρονισμό ώρας μέσω δικτύου ή GPS.
- Καταγραφή δεδομένων σε SD κάρτα ή απευθείας αποστολή σε cloud υπηρεσίες μέσω **MQTT** ή **HTTP APIs**.
- Δυνατότητα απομακρυσμένης αναβάθμισης (OTA) και παρακολούθησης.

1.3.4 Συγκριτική Αξιολόγηση

Σύστημα	Επεξεργαστής	Αποθήκευση	Επικοινωνία	RTC	Κατανάλωση	Επεκτασιμότητα
Arduino + SD Shield	ATmega328P	SD Card	Όχι	Ναι	Χαμηλή	Υψηλή
STM32 CustomLogger	STM32F401RE	SD Card	Όχι	Ναι	Πολύ χαμηλή	Υψηλή

ESP32 IoT Logger	ESP32 Dual-Core	SD / Cloud	Wi-Fi	Ναι	Μέτρια	Υψηλή
HOBO Industrial Logger	Ιδιόκτητο SoC	Εσωτερική	Bluetooth	Ναι	Πολύ χαμηλή	Περιορισμένη

Πίνακας 1 Αξιολόγηση παρόμοιων data logger συστημάτων

Η παρούσα εργασία προσανατολίζεται στη δημιουργία ενός **ευέλικτου και χαμηλού κόστους καταγραφικού συστήματος**, βασισμένο στο STM32F401RE, το οποίο προσφέρει υψηλό έλεγχο του υλικού, ενεργειακή αποδοτικότητα και δυνατότητα επέκτασης με αισθητήρες και διασύνδεση αποθήκευσης μέσω SD κάρτας. Η υλοποίηση συγκρίνεται ευνοϊκά με αντίστοιχες DIY λύσεις, προσφέροντας μεγαλύτερη ακρίβεια και δυνατότητα ελέγχου του λογισμικού σε χαμηλό επίπεδο.

2 Ανάλυση απαιτήσεων συστήματος

Για την επίτευξη και ορθή λειτουργία του συστήματος οι απαιτήσεις είναι οι παρακάτω:

- **Χρήση μικροελεγκτή**
- **Ενημέρωση firmware συσκευής μέσω bootloader**
- **Συλλογή υγρασίας θερμοκρασίας μέσω αισθητήρα**
- **Χρήση κάρτα μνήμης SD για την αποθήκευση του firmware αλλά και των δεδομένων του αισθητήρα**
- **Χρήση συστήματος αρχείων για την επικοινωνία με την κάρτα μνήμης (FatFS)**
- **Χρήση ρελέ για έλεγχο συσκευών**
- **Χρήση ρολογιού πραγματικού χρόνου για την συλλογή των δεδομένων**
- **Χρήση GPS για την περιοδική ενημέρωση του ρολογιού για τυχόν αποκλίσεις**

2.1 Μικροελεγκτής

2.1.1 Ο μικροελεγκτής STM32F401RE

Ο STM32F401RE είναι ένας ισχυρός μικροελεγκτής της σειράς STM32F4 της STMicroelectronics, βασισμένος στον πυρήνα ARM Cortex-M4 με συχνότητα λειτουργίας έως και 84 MHz. Διαθέτει 512 KB μνήμης Flash και 96 KB SRAM, παρέχοντας επαρκή χωρητικότητα για απαιτητικές εφαρμογές.

2.1.2 Περιφερειακά και Υποστηριζόμενες Λειτουργίες

Ο STM32F401RE περιλαμβάνει πληθώρα περιφερειακών όπως:

- ADC 12-bit με 16 κανάλια για αναλογική καταγραφή δεδομένων.
- USART, I2C, SPI, USB OTG για επικοινωνία με άλλα συστήματα και συσκευές.
- TIMERS (Advanced Control Timers, General Purpose Timers) για ακριβή χρονισμό και έλεγχο PWM.
- RTC (Real-Time-Clock) για καταγραφή και διατήρηση πραγματικού χρόνου.
- DMA (Direct Memory Access) για ταχύτατη μεταφορά δεδομένων μεταξύ περιφερειακών και μνήμης χωρίς την παρέμβαση της CPU.
- Watchdog Timers (IWDG, WWDG) για την ανίχνευση και αντιμετώπιση σφαλμάτων.

2.1.3 Λειτουργίες Χαμηλής Κατανάλωσης

Ο STM32F401RE υποστηρίζει πολλαπλές λειτουργίες χαμηλής κατανάλωσης, όπως:

- **SleepMode:** Μειώνει την κατανάλωση όταν ο επεξεργαστής δεν εκτελεί εντολές, αλλά τα περιφερειακά παραμένουν ενεργά.
- **StopMode:** Η λειτουργία του επεξεργαστή σταματά, αλλά η μνήμη RAM και τα περιφερειακά I/O διατηρούν την κατάστασή τους.
- **StandbyMode:** Η χαμηλότερη κατανάλωση ενέργειας, με διατήρηση του RTC και της μνήμης backup.

2.1.4 Αρχιτεκτονική και Απόδοση

Ο πυρήνας **Cortex-M4** του STM32F401RE υποστηρίζει:

- **Floating Point Unit (FPU):** Για γρήγορους υπολογισμούς κινητής υποδιαστολής.
- **DSP Instructions:** Βελτιστοποίηση στην επεξεργασία σήματος.
- **Nested Vectored Interrupt Controller (NVIC):** Υποστήριξη για διαχείριση διακοπών (interrupts) χαμηλής καθυστέρησης.

2.1.5 Εφαρμογές και Χρήσεις

Ο STM32F401RE είναι ιδανικός για:

- Ενσωματωμένα συστήματα ελέγχου.
- IoT εφαρμογές που απαιτούν χαμηλή κατανάλωση.
- Καταγραφείς δεδομένων (Data Loggers).
- Βιομηχανικά αυτοματοποιημένα συστήματα.
- Ασύρματες συσκευές επικοινωνίας.

Το μικρό του μέγεθος σε συνδυασμό με τις πολλαπλές δυνατότητες επικοινωνίας και την επεκτασιμότητά του τον καθιστούν μια εξαιρετική επιλογή για απαιτητικές ενσωματωμένες εφαρμογές.

2.1.6 Η μνήμη του μικροελεγκτή STM32F401RE

Ο μικροελεγκτής STM32F401RE διαθέτει ενσωματωμένη μνήμη Flash η οποία χρησιμοποιείται για την αποθήκευση του προγράμματος (firmware) καθώς και άλλων στατικών δεδομένων. Η μνήμη αυτή αποτελεί βασικό χαρακτηριστικό κάθε συστήματος ενσωματωμένου ελέγχου, καθώς προσφέρει μη πτητική αποθήκευση διατηρώντας τα δεδομένα της ακόμα και μετά από απώλεια τροφοδοσίας.

Η συνολική χωρητικότητα της Flash μνήμης στον συγκεκριμένο μικροελεγκτή είναι 512 KB. Η μνήμη είναι οργανωμένη σε τομείς (sectors) και υποστηρίζει διαγραφή και εγγραφή σε επίπεδο τομέα. Οι βασικές λειτουργίες που μπορούν να πραγματοποιηθούν στη Flash μνήμη είναι:

- **Διαγραφή (Erase):** Πραγματοποιείται σε επίπεδο τομέα. Κάθε τομέας πρέπει να διαγραφεί πριν από νέα εγγραφή.
- **Εγγραφή (Program):** Γίνεται συνήθως σε λέξεις των 32 bit.
- **Ανάγνωση (Read):** Η ανάγνωση γίνεται άμεσα από τον επεξεργαστή, με πρόσβαση παρόμοια με την SRAM.

Block	Name	Block base addresses	Size
Main memory	Sector 0	0x0800 0000 - 0x0800 3FFF	16 Kbytes
	Sector 1	0x0800 4000 - 0x0800 7FFF	16 Kbytes
	Sector 2	0x0800 8000 - 0x0800 BFFF	16 Kbytes
	Sector 3	0x0800 C000 - 0x0800 FFFF	16 Kbytes
	Sector 4	0x0801 0000 - 0x0801 FFFF	64 Kbytes
	Sector 5	0x0802 0000 - 0x0803 FFFF	128 Kbytes
	Sector 6	0x0804 0000 - 0x0805 FFFF	128 Kbytes
	Sector 7	0x0806 0000 - 0x0807 FFFF	128 Kbytes
System memory		0x1FFF 0000 - 0x1FFF 77FF	30 Kbytes
OTP area		0x1FFF 7800 - 0x1FFF 7A0F	528 bytes
Option bytes		0x1FFF C000 - 0x1FFF C00F	16 bytes

Εικόνα 2.1 Μπλοκ διάγραμμα μνήμης μικροελεγκτή

Η μνήμη Flash είναι προσβάσιμη στον επεξεργαστή μέσω του διαύλου Cortex-M4 I-Code και υποστηρίζει εκτέλεση εντολών απευθείας από τη Flash (XIP – Execute-In-Place), γεγονός που επιτρέπει την εκτέλεση του κώδικα χωρίς ανάγκη αντιγραφής του στην RAM.

Η ταχύτητα προσπέλασης της μνήμης Flash επηρεάζεται από τη συχνότητα λειτουργίας του συστήματος. Για τον λόγο αυτό χρησιμοποιείται ένας μηχανισμός cache και prefetch buffer, καθώς και η δυνατότητα καθορισμού latency cycles μέσω των ρυθμίσεων του Flash interface.

Τέλος, η STM παρέχει τη δυνατότητα προστασίας εγγραφής και ανάγνωσης (read/write protection) σε επιλεγμένους τομείς της Flash, ώστε να διασφαλιστεί η ακεραιότητα και η ασφάλεια του λογισμικού από μη εξουσιοδοτημένες τροποποιήσεις ή αντιγραφή.

Η διαχείριση της Flash μνήμης μπορεί να γίνει είτε με τη χρήση του HAL/LLAPI, είτε απευθείας μέσω προσπέλασης στους καταχωρητές του υποσυστήματος Flash, εφόσον απαιτείται πιο αποδοτικός ή προσαρμοσμένος χειρισμός.

2.2 Ενημέρωση firmware συσκευής μέσω bootloader

Στα ενσωματωμένα συστήματα, οι περιοδικές ενημερώσεις firmware, DFU (Device Firmware Update), είναι απαραίτητες. Παρακάτω καλύπτεται η έννοια του DFU και πώς ρυθμίζεται σε έναν μικροελεγκτή.

2.2.1 DFU (Device Firmware Update)

DFU (Device Firmware Update) είναι η διαδικασία ασφαλούς ενημέρωσης firmware. Το firmware είναι το βασικό λογισμικό που ελέγχει τη λειτουργία ενός συστήματος και οι συνεχείς ενημερώσεις είναι απαραίτητες για την εισαγωγή νέων λειτουργιών ή την επιδιόρθωση τρωτών σημείων ασφαλείας. Συνήθως, το DFU λειτουργεί παράλληλα με ένα bootloader, το οποίο αφαιρεί την υπάρχουσα εφαρμογή και την αντικαθιστά με ένα νέο firmware.

Για την υλοποίηση της λειτουργικότητας DFU, απαιτούνται τα ακόλουθα στοιχεία:

- **Bootloader:** Η μονάδα firmware που είναι υπεύθυνη για την εκτέλεση της διαδικασίας DFU, τη διαγραφή της παλιάς εφαρμογής και την εγκατάσταση του νέου firmware.
- **Application:** Το κύριο firmware που εκτελεί κανονικές λειτουργίες.
- **Βιβλιοθήκες διασύνδεσης:** Μια μονάδα firmware που περιέχει τις βιβλιοθήκες για την επίτευξη της σύνδεσης της συσκευής, που απαιτείται για το DFU.
- **DFU Package:** Ένα αρχείο πακέτου που περιέχει το νέο firmware, διασφαλίζοντας την ακεραιότητα μέσω κρυπτογράφησης και επαλήθευσης υπογραφής.

2.2.2 Ρύθμιση και Εκτέλεση DFU

Τα βασικά βήματα ρύθμισης είναι τα εξής:

- **Δόμηση και προγραμματισμός του Bootloader:** Ο bootloader είναι ένα απαραίτητο στοιχείο για την εκτέλεση του DFU.
- **Δημιουργία πακέτου DFU:** Για την ενημέρωση του firmware, πρέπει να δημιουργηθεί ένα πακέτο DFU.
- **Μετάδοση και Ενημέρωση Firmware:** Το δημιουργημένο πακέτο DFU μπορεί να μεταδοθεί στη συσκευή για ενημέρωση.

2.2.3 Βασικά στοιχεία του Bootloader

Ο bootloader είναι ένα πολύ σημαντικό στοιχείο σε οποιοδήποτε λειτουργικό σύστημα. Ένας bootloader είναι επίσης γνωστός και ως πρόγραμμα εκκίνηση. Είναι ειδικό λογισμικό λειτουργικού συστήματος που φορτώνεται στην ενεργή μνήμη ενός υπολογιστή μετά την εκκίνηση. Επομένως, πρέπει να είναι σαφές ότι αυτό είναι επίσης λογισμικό όπως μια εφαρμογή. Ο όρος bootloader είναι μια συντομευμένη μορφή των λέξεων "bootstrap loader".

2.2.4 Bootloader σε ενσωματωμένα συστήματα

Όπως ένα κανονικό λειτουργικό σύστημα, ο bootloader σε έναν μικροελεγκτή εξυπηρετεί επίσης τον ίδιο σκοπό. Αυτό είναι ένα μικρό κομμάτι λογισμικού που εκτελείται πριν από τον κύριο κώδικα της εφαρμογής σε μια ενσωματωμένη συσκευή. Ο κύριος σκοπός του είναι να αρχικοποιήσει το υλικό, να ρυθμίσει τη μνήμη και να φορτώσει τον κώδικα της εφαρμογής από μια συσκευή αποθήκευσης, όπως μια μνήμη flash, στη RAM. Εάν δεν υπάρχει bootloader, τότε μια εφαρμογή θα αρχίσει να εκτελείται απευθείας.

Ο bootloader επαληθεύει την ακεραιότητα του νέου firmware. Είναι χρήσιμος για την ενημέρωση του firmware και δεν υπάρχει η ανάγκη για εξωτερικό προγραμματιστή. Επίσης, είναι υπεύθυνος για την ασφάλεια του firmware και του συστήματος.

2.3 Αισθητήρες

2.3.1 Επισκόπηση

Σήμερα, όλο και περισσότερα ψηφιακά συστήματα λαμβάνουν πληροφορίες από το εξωτερικό τους περιβάλλον χρησιμοποιώντας αισθητήρες. Οι πιο συχνές εφαρμογές όπου συναντώνται αισθητήρες

είναι οι οικιακές ηλεκτρονικές συσκευές καθώς και διάφοροι αυτοματισμοί που διευκολύνουν την καθημερινότητα των ανθρώπων. Ταυτόχρονα, όμως, εμφανίζονται και σε πιο σύνθετα προβλήματα συμβάλλοντας στην ανάπτυξη των τομέων της μηχανικής, της επιστήμης και της βιομηχανίας. Το γεγονός αυτό καθιστά εξαιρετικά σημαντικό τον σχεδιασμό απλών, γρήγορων και οικονομικών μεθόδων που μετατρέπουν τις αναλογικές πληροφορίες που παρέχονται από τους αισθητήρες σε ψηφιακές πληροφορίες που χειρίζεται το κάθε σύστημα.

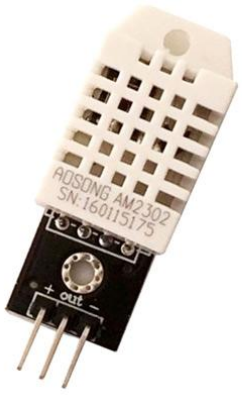
Πιο γενικά, στην σύγχρονη εποχή, οι μετρήσεις πολλών διαφορετικών φυσικών μεγεθών γίνονται με τη βοήθεια αισθητήρων. Οι αισθητήρες αυτοί είναι διατάξεις που διαθέτουν κάποια κατάλληλη ιδιότητα, που μεταβάλλεται ως συνάρτηση του φυσικού μεγέθους που πρόκειται να μετρηθεί. Επομένως, μετρώντας την ιδιότητα του αισθητήρα είναι εφικτός ο ποσοτικός υπολογισμός της τιμής του φυσικού μεγέθους.

Τα ενσωματωμένα συστήματα στηρίζουν τη λειτουργία τους σε αισθητήρες, με τους οποίους συλλέγουν πληροφορία από τον περιβάλλον. Τέτοιοι αισθητήρες μετρούν φυσικά μεγέθη όπως θερμοκρασία, πίεση, υγρασία, ταχύτητα, επιτάχυνση. Σε πολλές περιπτώσεις είναι σύνθετοι αισθητήρες, όπως κάμερες ή υπερηχητικοί αισθητήρες απόστασης. Η τεχνολογία επιτρέπει πλέον τη δημιουργία αισθητήρων με τη μορφή σύνθετων ηλεκτρομηχανικών συστημάτων (MEMS-micro-electro-mechanical-systems) που αναπτύσσονται σε τσιπ πυριτίου και ενσωματώνουν και κυκλώματα μετατροπής του αναλογικού σήματος σε ψηφιακό ή και μονάδες επεξεργασίας. Τέτοια συστήματα ονομάζονται «έξυπνοι αισθητήρες».

Ο μικροελεγκτής λειτουργεί ως ψηφιακός ελεγκτής συλλέγοντας δεδομένα από τους αισθητήρες. Το σύστημα συλλογής δεδομένων (data acquisition), αποτελείται από τους αισθητήρες, που μετατρέπουν τη φυσική ποσότητα που μετριέται σε αναλογικό ηλεκτρικό σήμα.

2.3.2 Αισθητήρας Συστήματος

Στο συγκεκριμένο σύστημα καταγραφής χρησιμοποιήθηκε ο DHT-22 αισθητήρας. Ο DHT-22 είναι ένας βασικός, χαμηλού κόστους, αισθητήρας για την μέτρηση υγρασίας και θερμοκρασίας στον χώρο. Στο εσωτερικό του κρύβει έναν αισθητήρα υγρασίας και ένα θερμίστορ (μεταβλητή αντίσταση που η τιμή της αλλάζει σε σχέση με την θερμοκρασία) διαβάζοντας έτσι τον αέρα που το περιβάλει. Η επικοινωνία ανάμεσα στον μικροελεγκτή και τον αισθητήρα πραγματοποιείται με το πρωτόκολλο 1-wire.



Εικόνα 2.2 Ο αισθητήρας DHT22

Model	DHT22
Power supply	3.3-6V DC
Output signal	digital signal via single-bus
Sensing element	Polymer capacitor
Operating range	humidity 0-100%RH; temperature -40~80Celsius
Accuracy	humidity +-2%RH(Max +-5%RH); temperature <+-0.5Celsius
Resolution or sensitivity	humidity 0.1%RH; temperature 0.1Celsius
Repeatability	humidity +-1%RH; temperature +-0.2Celsius
Humidity hysteresis	+-0.3%RH
Long-term Stability	+-0.5%RH/year
Sensing period	Average: 2s
Interchangeability	fully interchangeable
Dimensions	small size 14*18*5.5mm; big size 22*28*5mm

Εικόνα 2.3 Οι προδιαγραφές του αισθητήρα DHT-22

2.4 Κάρτα SD

Η κάρτα SD, Secure Digital, είναι μια συσκευή αποθήκευσης μνήμης flash που έχει σχεδιαστεί για να παρέχει υψηλής χωρητικότητα, μη πτητικό και επανεγγράψιμο μέσο αποθήκευσης σε μικρό μέγεθος. Αυτές οι συσκευές χρησιμοποιούνται συχνά σε πολλά ηλεκτρονικά προϊόντα, όπως κάμερες, υπολογιστές, συστήματα GPS, κινητά τηλέφωνα κ.α.

Κάρτες SD διατίθενται σε διάφορες χωρητικότητες, κατηγοριοποιημένες ανά τύπο:

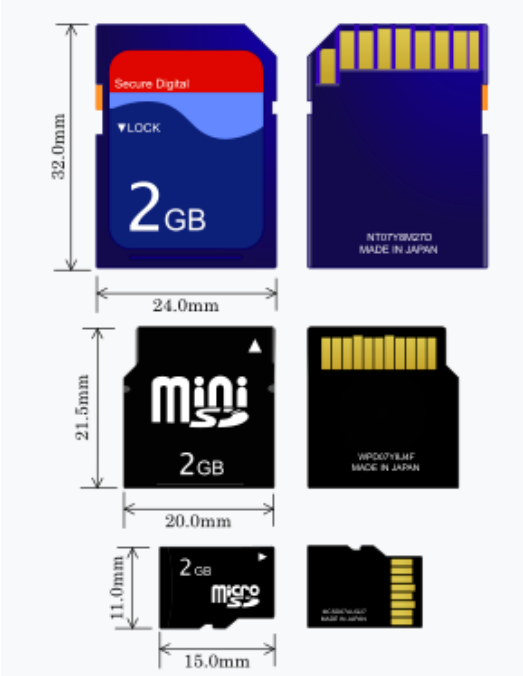
- **SD:** Οι standard κάρτες SD έχουν χωρητικότητα έως 2 GB.
- **SDHC:** Οι κάρτες SDHC (Secure Digital High Capacity) έχουν χωρητικότητα έως 32 GB.
- **SDXC:** Οι κάρτες SDXC (Secure Digital Extended Capacity) μπορούν να αποθηκεύσουν έως 2 TB δεδομένων.
- **SDUC:** Οι κάρτες SDUC (Secure Digital Ultra Capacity) έχουν χωρητικότητα έως 128 TB.

		SD Standard	SDHC Standard	SDXC Standard	SDUC Standard
Capacity		up to 2GB	more than 2GB up to 32GB	more than 32GB up to 2TB	more than 2TB up to 128TB
File System		FAT 12, 16	FAT 32	exFAT	exFAT
SD Logo					
Form Factor Size	SD	32 x 24 x 2.1 mm			
	microSD	11 x 15 x 1.0 mm			

Εικόνα 2.4 Κατηγοριοποίηση καρτών SD

Η οικογένεια SD περιλαμβάνει τρία μεγέθη:

- **Standard SD:** 32 x 24 x 2.1 mm (το πάχος κυμαίνεται από 1.4 έως 2.1mm)
- **MiniSD:** 21.5 x 20 x 1.4 mm.
- **microSD:** 15 x 11 x 1 mm.



Εικόνα 2.5 Μεγέθη καρτών SD

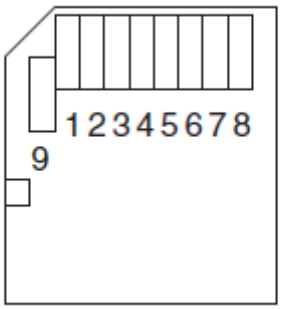
Οι προδιαγραφές της κάρτας SD διατηρούνται από την SD Card Association. Οι κάρτες MiniSD και microSD είναι ηλεκτρικά συμβατές με τις standard κάρτες SD και μπορούν να τοποθετηθούν σε ειδικούς προσαρμογείς και να χρησιμοποιηθούν ως standard κάρτες SD σε τυπικές υποδοχές καρτών.

Εικόνα 2.6 SD Card Adapter

Οι δύο κύριοι μέθοδοι επικοινωνίας ανάμεσα σε μία κάρτα SD και ενός μικροελεγκτή είναι οι εξής:

- **Λειτουργία SPI (Serial Peripheral Interface):** Χρησιμοποιεί τον δίαυλο SPI για επικοινωνία με την κάρτα SD. Έχει χαμηλότερη ταχύτητα, αλλά είναι απλό στην υλοποίηση.
- **Λειτουργία QSPI (Quad Serial Peripheral Interface):** Προσφέρει υψηλότερες ταχύτητες μεταφοράς δεδομένων, καθιστώντας την πιο κατάλληλη για το χειρισμό μεγάλων όγκων δεδομένων.

Η τυπική κάρτα SD έχει 9 pins. Το κάθε pin έχει διαφορετικές λειτουργίες ανάλογα με το πρωτόκολλο διεπαφής.



Εικόνα 2.7 Διάταξη των pin της κάρτας SD

Pin	Name	SD description	SPI description
1	CD/DAT3/CS	Data line 3	Chip select
2	CMD/Datain	Command/response	Host to card command and data
3	VSS	Supply ground	Supply ground
4	VDD	Supply voltage	Supply voltage
5	CLK	Clock	Clock
6	VSS2	Supply voltage ground	Supply voltage ground
7	DAT0	Data line 0	Card to host data and status
8	DAT1	Data line 1	Reserved
9	DAT2	Data line 2	Reserved

Εικόνα 2.8 Λειτουργία κάθε pin SD κάρτας

2.5 Επισκόπηση συστήματος αρχείων και εισαγωγή στο FatFS

2.5.1 Σύστημα αρχείων – Επισκόπηση

Στα ενσωματωμένα συστήματα, η μέθοδος αποθήκευσης και διαχείρισης δεδομένων μπορεί να χαρακτηριστεί ως κρίσιμος παράγοντας. Ειδικά σε περιβάλλοντα μικροελεγκτών, ο σχεδιασμός μιας αποτελεσματικής δομής αποθήκευσης δεδομένων εντός περιορισμένων πόρων είναι απαραίτητος. Επομένως, ένα σύστημα αρχείων έχει ζωτικής σημασίας ρόλο στην οργάνωση, αποθήκευση, ανάγνωση και εγγραφή δεδομένων αποτελεσματικά.

2.5.2 Ρόλος ενός Συστήματος Αρχείων

Ένα σύστημα αρχείων παρέχει μια δομή και μεθοδολογία για την αποτελεσματική διαχείριση δεδομένων σε μια συσκευή αποθήκευσης. Οι κύριες λειτουργίες του περιλαμβάνουν:

- **Αποθήκευση και Διαχείριση Αρχείων:** Οργανώνει και αποθηκεύει δεδομένα σε μονάδες αρχείων.
- **Υποστήριξη Δομής Καταλόγων:** Επιτρέπει τη λογική κατηγοριοποίηση πολλαπλών αρχείων μέσω καταλόγων.
- **Διασφάλιση Ακεραιότητας Δεδομένων:** Προστατεύει τα δεδομένα από αλλοίωση λόγω σφαλμάτων αποθήκευσης ή μη φυσιολογικών τερματισμών.
- **Βελτιστοποιημένη Απόδοση Ανάγνωσης/Εγγραφής:** Εφαρμόζει αποτελεσματικούς αλγορίθμους για γρήγορη πρόσβαση στα δεδομένα σε περιβάλλοντα με περιορισμένους πόρους.

2.5.3 Συστήματα Αρχείων σε Ενσωματωμένα Συστήματα

Ενώ τα συστήματα σταθερών υπολογιστών χρησιμοποιούν συστήματα αρχείων όπως NTFS, ext4 και APFS, τα ενσωματωμένα συστήματα χρησιμοποιούν συνήθως ελαφριά συστήματα αρχείων, συμπεριλαμβανομένων:

- **FatFS:** Το πιο ευρέως χρησιμοποιούμενο σύστημα αρχείων για κάρτες SD και μνήμη flash.
- **LittleFS:** Ανθεκτικό σε διακοπές ρεύματος και υποστηρίζει “wear leveling”, καθιστώντας το κατάλληλο για μνήμη NAND.
- **SPIFFS:** Βελτιστοποιημένο για μικρά μεγέθη μνήμης flash με ελάχιστη επιβάρυνση.

Μεταξύ αυτών, το FatFS είναι εξαιρετικά ευέλικτο και υποστηρίζεται ευρέως σε όλες τις ενσωματωμένες πλατφόρμες, καθιστώντας το κατάλληλη επιλογή για την εφαρμογή του σε συστήματα καταγραφής.

2.5.4 FatFS – Επισκόπηση

FatFS (File Allocation Table File System) είναι μια βιβλιοθήκη συστήματος αρχείων ανοιχτού κώδικα για ενσωματωμένα συστήματα, που αναπτύχθηκε από τον ChaN. Υποστηρίζει συστήματα αρχείων FAT12, FAT16, FAT32 και exFAT και έχει σχεδιαστεί για να λειτουργεί με εξωτερικές συσκευές αποθήκευσης όπως κάρτες SD.

2.5.5 Χαρακτηριστικά του FatFS

Το FatFS προσφέρει τα ακόλουθα βασικά χαρακτηριστικά:

- **Υποστηρίζει FAT12, FAT16, FAT32 και exFAT:** Επιτρέπει τη χρήση σε ένα ευρύ φάσμα συσκευών αποθήκευσης.
- **Ελαφρύ:** Σχεδιασμένο για να χρησιμοποιεί ελάχιστους πόρους, καθιστώντας το ιδανικό για μικροελεγκτές.
- **Ευέλικτη Διασύνδεση:** Η αρθρωτή δομή επιτρέπει την απρόσκοπτη ενσωμάτωση με διάφορα υλικά και λειτουργικά συστήματα.
- **Επίπεδο αφαίρεσης:** Λειτουργεί με κάρτες SD, eMMC, NOR/NAND flash, δίσκους RAM και άλλα μέσα αποθήκευσης.
- **Διαχείριση Αρχείων και Καταλόγων:** Παρέχει λειτουργίες για τη δημιουργία, διαγραφή, ανάγνωση, εγγραφή, μετακίνηση αρχείων και χειρισμό καταλόγων.
- **Χωρίς Υποστήριξη Journaling:** Το FatFS δεν διαθέτει journaling, καθιστώντας το ευάλωτο σε απώλεια δεδομένων σε ξαφνικές διακοπές ρεύματος.

2.5.6 Δομή FatFS

Το FatFS αποτελείται από αρκετές βασικές ενότητες:

- **ff.c:** Η βασική υλοποίηση του συστήματος αρχείων.
- **diskio.c:** Διασύνδεση προγράμματος οδήγησης φυσικού δίσκου που πρέπει να υλοποιηθεί σύμφωνα με τη συσκευή αποθήκευσης.
- **integer.h:** Ορίζει τύπους δεδομένων ακεραίων που χρησιμοποιούνται στο FatFS.
- **ffconf.h:** Αρχείο διαμόρφωσης για την προσαρμογή της συμπεριφοράς του FatFS.

2.5.7 Βασικές Λειτουργίες FatFS

Το FatFS παρέχει βασικές συναρτήσεις/εντολέςAPI για λειτουργίες συστήματος αρχείων:

Συνάρτηση	**Περιγραφή**
f_mount()	Προσάρτηση του συστήματος αρχείων
f_open()	Άνοιγμα ή δημιουργία αρχείου
f_read()	Ανάγνωση δεδομένων από ένα αρχείο
f_write()	Εγγραφή δεδομένων σε ένα αρχείο
f_close()	Κλείσιμο ενός αρχείου

f_unlink()	Διαγραφή ενός αρχείου
f_mkdir()	Δημιουργία καταλόγου
f_opendir()	Άνοιγμα καταλόγου
f_readdir()	Ανάγνωση καταχωρήσεων αρχείων σε έναν κατάλογο

Χρησιμοποιώντας αυτές τις συναρτήσεις, είναι εύκολο να υλοποιηθούν λειτουργίες αποθήκευσης δεδομένων που βασίζονται σε αρχεία στον μικροελεγκτή χρησιμοποιώντας είτε κάρτες SD είτε κάποια άλλη μνήμη flash.

2.5.8 Χρήση του FatFS για Αποθήκευση Δεδομένων

Η χρησιμοποίηση του FatFS επιτρέπει διάφορες εφαρμογές:

- **Καταγραφή Δεδομένων Αισθητήρων:** Αποθήκευση μετρήσεων περιβαλλοντικών αισθητήρων σε κάρτα SD για μεταγενέστερη ανάλυση.
- **Ενημερώσεις Firmware:** Χρήση εξωτερικής αποθήκευσης για ενημερώσεις firmware.
- **Σύστημα Καταγραφής:** Καταγραφή αρχείων εντοπισμού σφαλμάτων και ιστορικού συμβάντων για ανάλυση συστήματος.

2.5.9 Παρατηρήσεις κατά τη χρήση του FatFS

Κατά τη χρήση του FatFS, οι παρακάτω παράγοντες πρέπει να ληφθούν υπόψη:

- **Διάρκεια ζωής συσκευής αποθήκευσης:** Μία μνήμη flash έχει περιορισμένο αριθμό κύκλων εγγραφής, επομένως συνιστάται η εφαρμογή τεχνικών wear leveling.
- **Βελτιστοποίηση χρήσης μνήμης:** Η ελαχιστοποίηση της χρήσης της μνήμης αποφέρει λιγότερη κατανάλωση στο σύστημα καθώς είτε η ανάγνωση είτε η εγγραφή αποφέρει κατανάλωση ενέργειας.
- **Βελτιστοποίηση απόδοσης:** Η χρήση διακοπών (interrupts) κατά την επικοινωνία ανάμεσα σε μικροελεγκτή και μνήμη μειώνει το φόρτο εργασίας του επεξεργαστή (CPU load).

2.5.10 Συμπέρασμα

Το FatFS είναι μια ισχυρή λύση και ένα ελαφρύ σύστημα αρχείων βασισμένο στο FAT που υποστηρίζει διάφορες συσκευές αποθήκευσης και επιτρέπει την αποτελεσματική διαχείριση δεδομένων σε ενσωματωμένα περιβάλλοντα. Η επικοινωνία με την κάρτα SD μπορεί να επιτευχθεί μέσω SPI ή QSPI, επιτρέποντας την απρόσκοπτη αποθήκευση και ανάκτηση δεδομένων που βασίζεται σε αρχεία χρησιμοποιώντας το FatFS, διατηρώντας παράλληλα την αφαιρετικότητα της πλατφόρμας, καθιστώντας την κατάλληλη για ένα ευρύ φάσμα εφαρμογών.

2.6 Προγραμματισμός Ρελέ

2.6.1 Ορισμός

Ο ηλεκτρονόμος, ρελέ (relay), είναι ένας ηλεκτρικός διακόπτης που ανοίγει και κλείνει ένα ηλεκτρικό κύκλωμα υπό τον έλεγχο ενός άλλου ηλεκτρικού κυκλώματος και χρησιμοποιείται κατά κόρον σε αυτοματισμούς.

Κάθε επαφή ενός ηλεκτρονόμου μπορεί να είναι Κανονικά-Ανοικτή (Normally Open, NO), Κανονικά-Κλειστή (Normally Closed, NC) ή μεταγωγικός (change-over), ανάλογα με τον τύπο της.

- **Κανονικά-Ανοικτή:** Συνδέει το κύκλωμα όταν ο ηλεκτρονόμος ενεργοποιείται. Το κύκλωμα αποσυνδέεται όταν ο ηλεκτρονόμος είναι ανενεργός. Μια τέτοια επαφή καλείται επίσης Επαφή Μορφής Α ή επαφή "make". Η επαφή μορφής Α είναι ιδανική για εφαρμογές που απαιτούν την ενεργοποίηση μιας πηγής υψηλής τάσης από απόσταση.
- **Κανονικά-Κλειστή:** Αποσυνδέει το κύκλωμα όταν ο ηλεκτρονόμος ενεργοποιείται. Το κύκλωμα συνδέεται όταν ο ηλεκτρονόμος είναι ανενεργός. Μια τέτοια επαφή καλείται επίσης Επαφή Μορφής Β ή επαφή "break". Η επαφή μορφής Β είναι ιδανική για εφαρμογές που απαιτούν το κύκλωμα να παραμένει κλειστό (ενεργό) μέχρι ο ηλεκτρονόμος να ενεργοποιηθεί.
- **Μεταγωγική:** Μπορεί να ελέγχει δύο κυκλώματα. Ισοδυναμεί με μια επαφή κανονικά-ανοικτή και μια επαφή κανονικά-κλειστή που έχουν ένα κοινό ακροδέκτη. Μια τέτοια επαφή καλείται επίσης Επαφή Μορφής C.

2.7 Ρολόι πραγματικού χρόνου

Τα ρολόγια πραγματικού χρόνου (RTC) είναι κρίσιμα υποσυστήματα στα ενσωματωμένα συστήματα, καθώς επιτρέπουν τη διατήρηση της ακριβούς ώρας και ημερομηνίας, ακόμα και όταν το σύστημα είναι απενεργοποιημένο ή βρίσκεται σε κατάσταση χαμηλής κατανάλωσης. Τα RTC είναι ειδικά σχεδιασμένα για να λειτουργούν αυτόνομα, με εξαιρετικά χαμηλή κατανάλωση ισχύος, χρησιμοποιώντας συνήθως έναν κρύσταλλο 32,768 kHz ως ταλαντωτή.

Η βασική λειτουργία ενός RTC είναι να μετράει το χρόνο, διατηρώντας ακριβείς εγγραφές δευτερολέπτων, λεπτών, ωρών, ημερών, μηνών και ετών. Αυτό το χαρακτηριστικό επιτρέπει σε συστήματα που βασίζονται σε μικροελεγκτές να εκτελούν λειτουργίες συγχρονισμένες με πραγματικό χρόνο, όπως είναι η χρονοσήμανση δεδομένων, η ενεργοποίηση ή απενεργοποίηση συσκευών σε συγκεκριμένες ώρες και η καταγραφή συμβάντων.

Η ενσωμάτωση των RTC σε μικροελεγκτές επιτρέπει τη χρονοπρογραμματισμένη εκτέλεση εργασιών (όπως την συλλογή υγρασίας, θερμοκρασίας) και τη διατήρηση ακριβούς χρονομέτρησης κατά τη διάρκεια λειτουργίας σε χαμηλή κατανάλωση.

Τα RTC χρησιμοποιούνται εκτενώς σε συστήματα χαμηλής κατανάλωσης (Low Power Systems), καθώς επιτρέπουν στον επεξεργαστή να «ξυπνάει» μόνο όταν απαιτείται, μειώνοντας την κατανάλωση ενέργειας. Σε συστήματα καταγραφής, αυτή η λειτουργικότητα είναι εξαιρετικά σημαντική, καθώς συμβάλλει στη μεγάλη διάρκεια ζωής των συσκευών.

Οι τεχνικές συγχρονισμού, όπως ο συγχρονισμός μέσω GPS επιτρέπουν την υψηλή ακρίβεια σε κρίσιμες εφαρμογές, με τις τυχόν αποκλίσεις από τον πραγματικό χρόνο να διορθώνονται ανά τακτά διαστήματα.

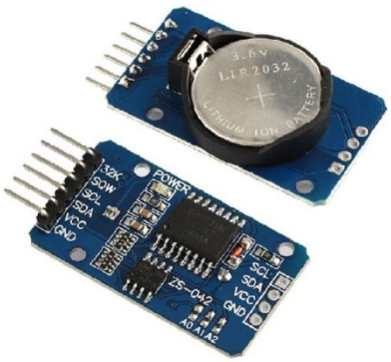
Τα RTC μπορούν να είναι είτε ενσωματωμένα (on-chip) στον μικροελεγκτή είτε εξωτερικά ως ανεξάρτητα ολοκληρωμένα κυκλώματα (IC). Στην παρούσα πτυχιακή, πραγματοποιήθηκε η χρήση ενός εξωτερικού RTC και συγκεκριμένα του DS3231.

2.7.1 Το ρολόι πραγματικού χρόνου DS3231

Το DS3231 είναι ένα ολοκληρωμένο κύκλωμα πραγματικού χρόνου (RTC) υψηλής ακρίβειας που χρησιμοποιεί ταλαντωτή (oscillator) ενσωματωμένο στο ίδιο το ολοκληρωμένο, εξαλείφοντας την ανάγκη για εξωτερικό κρύσταλλο. Η σημαντικότερη διαφορά του από άλλα RTC είναι η δυνατότητά του να διατηρεί εξαιρετική ακρίβεια, με απόκλιση περίπου ±2 ppm (parts per million), κάτι που μεταφράζεται σε μόλις ±1 λεπτό ανά έτος.

Το DS3231 ενσωματώνει επίσης αισθητήρα θερμοκρασίας, ο οποίος χρησιμοποιείται για τη θερμοκρασιακή αντιστάθμιση του ταλαντωτή, βελτιώνοντας έτσι την ακρίβεια του ρολογιού ανεξάρτητα από τις περιβαλλοντικές συνθήκες. Επιπλέον, διαθέτει δύο προγραμματιζόμενα alarms, ανίχνευση απώλειας τροφοδοσίας και υποστήριξη για I2C επικοινωνία, καθιστώντας την ενσωμάτωσή του σε μικροελεγκτές εύκολη και αποδοτική.

Το DS3231 είναι ιδανικό για εφαρμογές που απαιτούν μακροχρόνια ακρίβεια, όπως συστήματα καταγραφής δεδομένων, χρονισμός σε βιομηχανικές εφαρμογές και συγχρονισμός πραγματικού χρόνου σε συσκευές IoT.



Εικόνα 2.9 Το ρολόι πραγματικού χρόνου DS3231

2.8 GPS (Global Positioning System)

2.8.1 Επισκόπηση

Το GPS (Global Positioning System), Παγκόσμιο Σύστημα Στιγματοθέτησης, ή Θεσιθεσίας είναι παγκόσμιο σύστημα εντοπισμού γεωγραφικής θέσης, ακίνητου ή κινούμενου χρήστη, το οποίο βασίζεται σε ένα "πλέγμα" εικοσιτεσσάρων δορυφόρων της Γης, εφοδιασμένων με ειδικές συσκευές

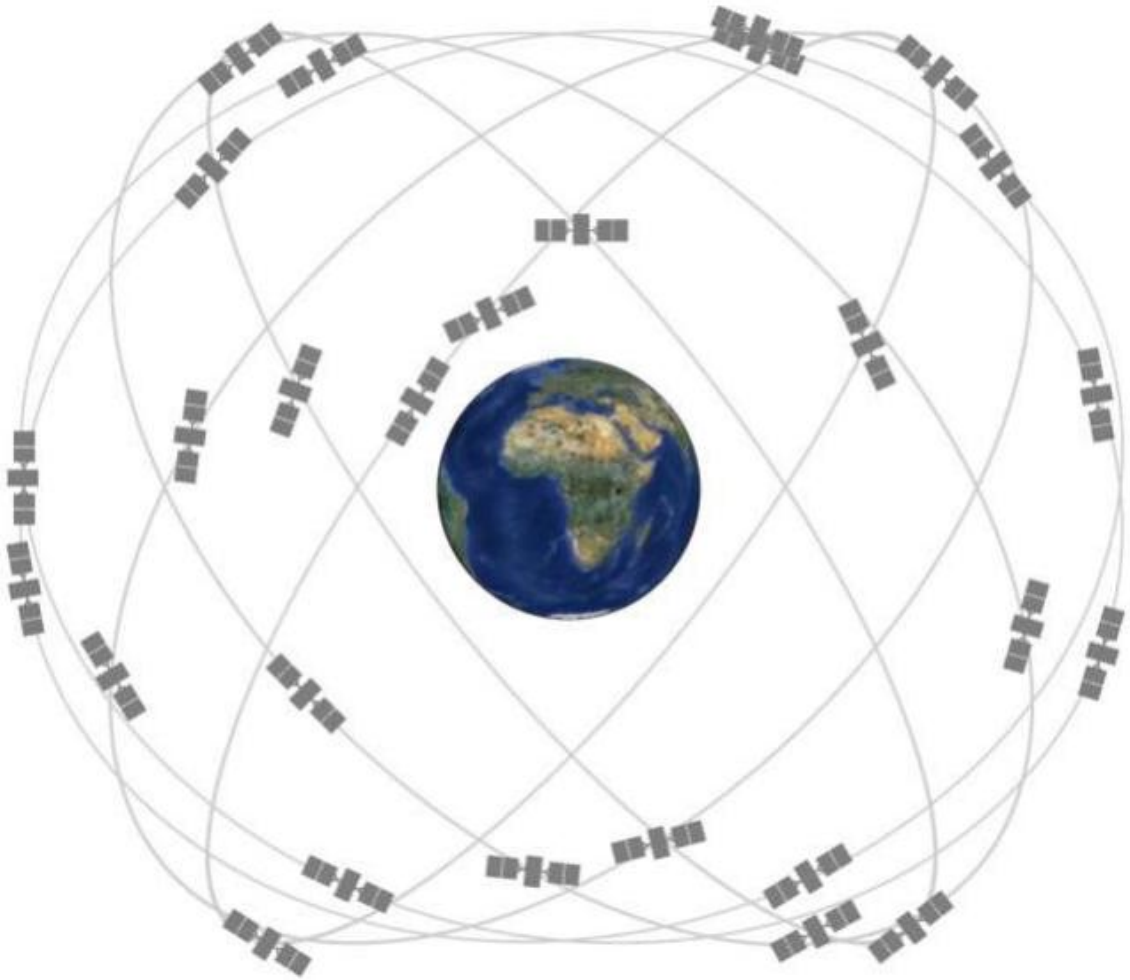
εντοπισμού, οι οποίες ονομάζονται "πομποδέκτες GPS". Οι πομποδέκτες αυτοί παρέχουν ακριβείς πληροφορίες για τη θέση ενός σημείου, το υψόμετρό του, την ταχύτητα και την κατεύθυνση της κίνησης του. Επίσης, σε συνδυασμό με ειδικό λογισμικό χαρτογράφησης μπορούν να απεικονίσουν γραφικά τις πληροφορίες αυτές. Το σύστημα ξεκίνησε από το Υπουργείο Άμυνας των ΗΠΑ και ονομάστηκε NAVSTAR GPS (Navigation Signal Timing and Ranging Global Positioning System).

2.8.2 Λειτουργικά τμήματα

Το σύστημα εντοπισμού θέσης GPS σχηματίζει ένα παγκόσμιο δίκτυο, με εμβέλεια που καλύπτει ξηρά, θάλασσα και αέρα. Εξαιτίας αυτής της έκτασής του, είναι απαραίτητος ο διαχωρισμός του σε επιμέρους τμήματα όπου πραγματοποιούνται όλες οι λειτουργίες του αλλά και ο συντονισμός του. Αναλυτικά, τα τμήματα αυτά είναι:

- **Διαστημικό τμήμα**: Αποτελείται από το δίκτυο των 24 - 32. Οι δορυφόροι αυτοί «σκεπάζουν» ομοιόμορφα με το σήμα τους ολόκληρο τον πλανήτη, γεγονός που αποδεικνύει τη φιλοσοφία που κρύβεται πίσω από τη λειτουργία του συστήματος GPS, δηλαδή τη διαθεσιμότητά του σε κάθε σημείο της Γης, ώστε να μην υπάρχει κίνδυνος να αποπροσανατολιστεί κανείς ποτέ και πουθενά.

Όλοι οι δορυφόροι βρίσκονται σε ύψος 20.200 χιλιομέτρων πάνω από την επιφάνεια της θάλασσας και εκτελούν δύο περιστροφές γύρω από τη Γη κάθε 24ωρο. Η κατασκευάστρια εταιρεία είναι η Rockwell International, η εκτόξευσή τους πραγματοποιήθηκε από το ακρωτήριο Canaveral, ενώ η τροφοδοσία τους με ηλεκτρική ενέργεια πραγματοποιείται μέσω των φωτοβολταϊκών συστημάτων που διαθέτουν.

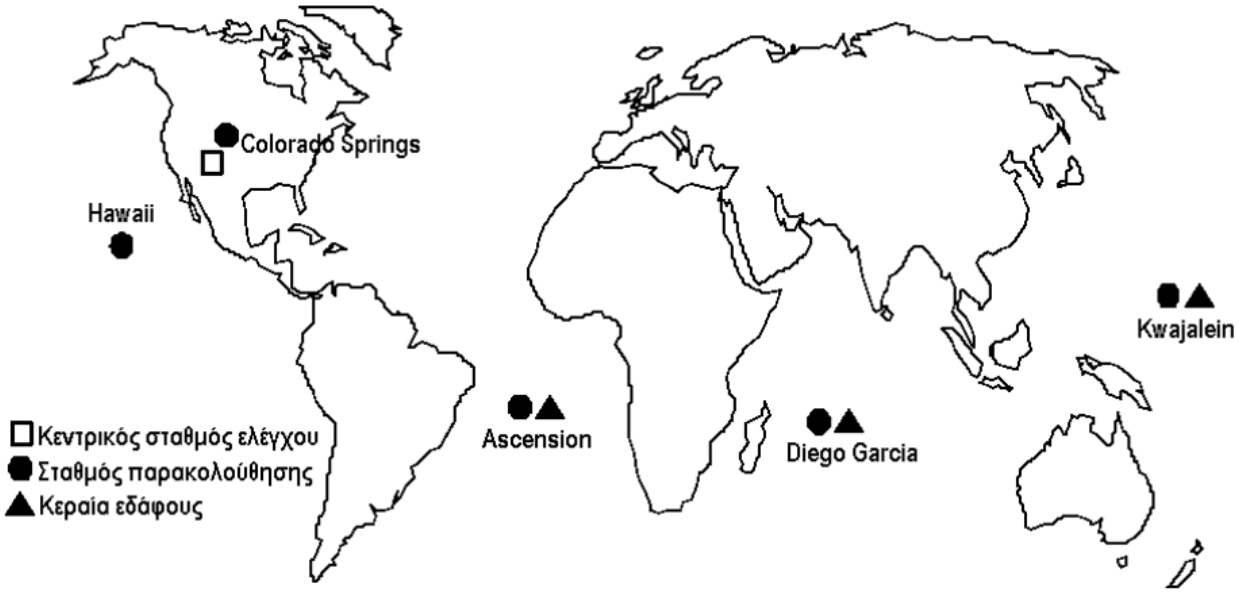


Εικόνα 2.10 Διαστημικό τμήμα GPS, Αστερισμός δορυφόρων

- **Επίγειο τμήμα ελέγχου**: Οι δορυφόροι, όπως είναι αναμενόμενο, είναι πολύ πιθανό να αντιμετωπίσουν ανά πάσα στιγμή προβλήματα στη σωστή λειτουργία τους. Οι έλεγχοι που πραγματοποιούνται σε αυτούς αφορούν στη σωστή τους ταχύτητα και υψόμετρο και στην κατάσταση της επάρκειάς τους σε ηλεκτρική ενέργεια. Παράλληλα, εφαρμόζονται όλες οι διορθωτικές ενέργειες που αφορούν στο σύστημα χρονομέτρησης των δορυφόρων, ώστε να αποτρέπεται η παροχή λανθασμένων πληροφοριών στους χρήστες του συστήματος. Το τμήμα επίγειου ελέγχου αποτελείται από ένα επανδρωμένο και τέσσερα μη επανδρωμένα κέντρα, εγκατεστημένα σε ισάριθμες περιοχές του πλανήτη.

Οι περιοχές αυτές είναι οι εξής: α) Κολοράντο (ΗΠΑ) β) Χαβάη (Ανατολικός Ειρηνικός Ωκεανός) γ) Ascension Island (Ατλαντικός Ωκεανός) δ) Diego Garcia (Ινδικός Ωκεανός) ε) Kwajalein (Δυτικός Ειρηνικός Ωκεανός)

Ο κυριότερος σταθμός βάσης είναι αυτός του Κολοράντο, ο οποίος είναι μάλιστα και ο μοναδικός που βρίσκεται στην ξηρά. Αναλαμβάνει τον έλεγχο της σωστής λειτουργίας των εναπομεινάντων τεσσάρων σταθμών, καθώς και τον συντονισμό τους. Σημειώνοντας τη θέση των σταθμών αυτών πάνω σε έναν παγκόσμιο χάρτη, παρατηρεί κανείς ότι η διάταξή τους δεν είναι τυχαία, αλλά ακολουθούν μια γραμμή παράλληλη με τα γεωγραφικά μήκη της Γης.



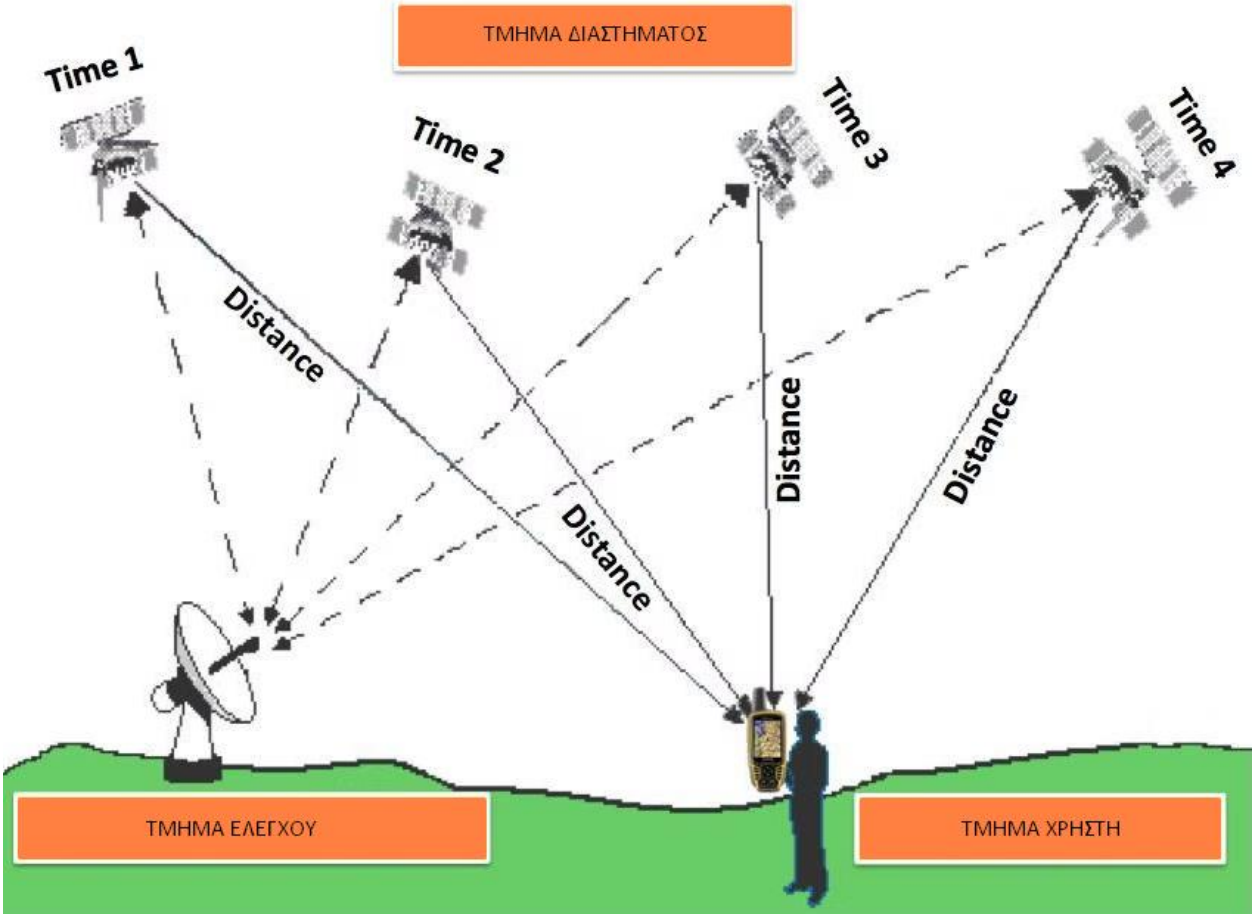
Εικόνα 2.11 Τμήμα Ελέγχου GPS, Βασικά κέντρα ελέγχου

- **Το τμήμα τελικού χρήστη**: Απαρτίζεται από τους χιλιάδες χρήστες δεκτών GPS ανά την υφήλιο. Οι δέκτες αυτοί μπορούν να χρησιμοποιηθούν τόσο κατά τη διάρκεια μιας απλής πεζοπορίας, όσο και σε οχήματα ή θαλάσσια σκάφη και κατά κανόνα διαθέτουν αρκετά μικρές διαστάσεις. Για να προσφέρουν όσο το δυνατόν περισσότερες πληροφορίες, οι δέκτες συνδυάζονται με ειδικό λογισμικό, που προβάλλει ένα χάρτη στην οθόνη της συσκευής GPS. Πρόκειται, δηλαδή, για λογισμικό που λαμβάνει από τους δορυφόρους τις πληροφορίες για το στίγμα του σημείου στο οποίο βρίσκεται ο δέκτης και τις μετατρέπει σε κατανοητή «ανθρώπινη» μορφή, πληροφορώντας τον χρήστη για την ακριβή γεωγραφική του θέση.

2.8.3 Λειτουργία GPS

Η βασική λειτουργία του GPS αναλύεται σε απλά βήματα παρακάτω:

1. Τα ραδιοσήματα που εκπέμπουν οι δορυφόροι GPS παρέχουν την ακριβή τους θέση, την κατάστασή τους και τον ακριβή χρόνο από ατομικά ρολόγια επί του σκάφους.
2. Τα ραδιοφωνικά σήματα GPS ταξιδεύουν δια μέσου του χώρου με την ταχύτητα του φωτός C πάνω από 299,792 km / sec .
3. Μια συσκευή GPS λαμβάνει τα ραδιοσήματα καταγράφοντας την ακριβή ώρα άφιξής τους και τα χρησιμοποιεί για να υπολογίσει την απόστασή της από κάθε δορυφόρο. Για να υπολογίσει την απόστασή του από ένα δορυφόρο μια συσκευή GPS αρκεί να γνωρίζει το χρόνο ταξιδιού του σήματος. Ο χρόνος ταξιδιού του σήματος είναι η διαφορά μεταξύ του χρόνου που μεταδίδει ο δορυφόρος και του χρόνου λήψης του σήματος.
4. Μόλις μια συσκευή GPS γνωρίσει την απόστασή της από τουλάχιστον τέσσερις δορυφόρους, μπορεί να χρησιμοποιήσει τη Γεωμετρία για να καθορίσει την θέση της στη γη σε τρείς διαστάσεις.



Εικόνα 2.12 GPS Αναπαράσταση

3 Τεχνολογίες και εργαλεία

3.1 Ρύθμιση Περιβάλλοντος Ανάπτυξης

Απαιτήσεις Υλικού

Τα ακόλουθα εξαρτήματα υλικού που απαιτούνται για αυτό το σύστημα είναι:

- STM32F401RE Nucleo Board
- Κάρτα SD και μονάδα ανάγνωσης καρτών SD (με υποστήριξη διασύνδεσης SPI)
- Αισθητήρας θερμοκρασίας και υγρασίας DHT22 (ή παρόμοιος αισθητήρας)
- RTC DS3231
- Relay module
- GPS module

Λογισμικό και Βιβλιοθήκες

Τα λογισμικά και οι βιβλιοθήκες που χρησιμοποιήθηκαν είναι:

- STM32CubeIDE (λογισμικό υπεύθυνο για τον προγραμματισμό του μικροελεγκτή)
- FatFS: Βιβλιοθήκη συστήματος αρχείων.
- HAL/LL drivers για την ρύθμιση των καταχωρητών του μικροελεγκτή
- Γλώσσα: C

3.2 Πρωτόκολλα επικοινωνίας υλικού

Στην ενότητα αυτή θα παρουσιαστούν κάποια από τα πρωτόκολλα επικοινωνίας που χρησιμοποιούνται για την επικοινωνία των συσκευών του συστήματος που υλοποιήθηκε. Γενικά, οι τρόποι επικοινωνίας μεταξύ ολοκληρωμένων κυκλωμάτων είναι η σειριακή και η παράλληλη επικοινωνία. Στην παράλληλη επικοινωνία όλα τα bits μιας λέξης δεδομένων μεταδίδονται ταυτόχρονα προς το κύκλωμα αποδέκτη. Βασικό πλεονέκτημα της παράλληλης μετάδοσης είναι ο μικρός χρόνος μεταφοράς δεδομένων, με ταυτόχρονη όμως αύξηση των αγωγών που απαιτούνται. Η παράλληλη επικοινωνία χρησιμοποιείται κυρίως στο εσωτερικό των ολοκληρωμένων κυκλωμάτων, όπου απαιτούνται αρκετά υψηλές ταχύτητες μεταφοράς δεδομένων. Για την επικοινωνία με άλλα ολοκληρωμένα κυκλώματα ή περιφερειακά πιο διαδομένη είναι η σειριακή μετάδοση δεδομένων. Τα πρωτόκολλα επικοινωνίας που χρησιμοποιούνται για την διασύνδεση των συσκευών στο σύστημα που υλοποιήθηκε είναι:

- **SPI (Serial Peripheral Interface Bus)**
- **I2C (Inter-Integrated Circuit)**
- **1-WIRE**
- **UART (Universal Asynchronous Receiver/Transmitter)**

Γίνεται επεξήγηση για τις θεμελιώδεις έννοιες κάθε interface και πώς διαμορφώνονται και χρησιμοποιούνται αυτές οι λειτουργίες με τον μικροελεγκτή της συγκεκριμένης πτυχιακής. Τα

συγκεκριμένα interfaces είναι κατάλληλα για την ανάπτυξη ενσωματωμένων εφαρμογών όπως η συλλογή δεδομένων αισθητήρων, η αποθήκευση εσωτερικής και εξωτερικής μνήμης.

3.2.1 Πρωτόκολλο SPI (Serial Peripheral Interface Bus)

Το SPI αναπτύχθηκε με σκοπό την εύκολη επικοινωνία μεταξύ ολοκληρωμένων και τον καλύτερο τρόπο διασύνδεσης των περιφερειακών μονάδων και των μικροελεγκτών μεταξύ τους. Το πρωτόκολλο SPI ή Serial Peripheral Interface Bus επιτρέπει την σειριακή σύγχρονη επικοινωνία μεταξύ ολοκληρωμένων. Η επικοινωνία αυτή είναι αμφίδρομη. Ο δίαυλος υλοποιήθηκε για πρώτη φορά από την εταιρία Motorola. Οι συσκευές επικοινωνούν μεταξύ τους σε mode Master/Slave. Ο Master του διαύλου είναι το ολοκληρωμένο που παράγει το frame των δεδομένων και το μεταδίδει προς τα ολοκληρωμένα slaves. Σε έναν SPI δίαυλο μπορούν να διασυνδεθούν περισσότερες από μία συσκευές slave χρησιμοποιώντας τις γραμμές Slave Select. Το SPI αποκαλείται και "σειριακός δίαυλος 4 καλωδίων".

Το SPI επιτρέπει σε δεδομένα των 8-bits να αποστέλλονται σύγχρονα και ταυτόχρονα να λαμβάνονται σύγχρονα δεδομένα με ταχύτητα που φτάνει τα 10 Mbps.

Μια τυπική θύρα σειριακής επικοινωνίας SPI διαθέτει τα ακόλουθα σήματα:

- **MISO (Master In Slave Out):** Σήμα εξόδου σειριακών δεδομένων. Μετάδοση δεδομένων από τον slave στον master
- **MOSI (Master Out Slave In):** Σήμα εισόδου σειριακών δεδομένων. Μετάδοση δεδομένων από τον master στον slave
- **SCK (Serial Clock):** Σήμα Χρονισμού που παρέχεται από τον master για συγχρονισμό της μεταφοράς δεδομένων
- **SS (Slave Select):** Σήμα επιλογής μιας συγκεκριμένης συσκευής slave

Σε όλες τις μεταφορές στο SPI το ψηφίο υψηλότερης αξίας (big-endian) στέλνεται πρώτο.

Εικόνα 3.1 Απλό παράδειγμα ενός SPI διαύλου

Τα κύρια χαρακτηριστικά του SPI είναι τα εξής:

- Επιτρέπει τη σύγχρονη επικοινωνία
- Είναι σειριακό
- Είναι πλήρως αμφίδρομο (full-duplex)
- Υπάρχει μόνο ένας Master στο δίαυλο
- Υπάρχουν ένας ή περισσότεροι Slaves στο δίαυλο

Λειτουργία του SPI

Το Ρολόι

Το σήμα του ρολογιού συγχρονίζει την έξοδο των bits δεδομένων από τον master με τη δειγματοληψία των bits από τον slave. Ένα bit δεδομένων μεταφέρεται σε κάθε κύκλο ρολογιού, οπότε η ταχύτητα μεταφοράς δεδομένων καθορίζεται από τη συχνότητα του σήματος του ρολογιού. Η επικοινωνία SPI ξεκινά πάντα από τον master, καθώς ο master διαμορφώνει και δημιουργεί το σήμα του ρολογιού.

Οποιοδήποτε πρωτόκολλο επικοινωνίας όπου οι συσκευές μοιράζονται ένα σήμα ρολογιού είναι γνωστό ως σύγχρονο. Το SPI είναι ένα σύγχρονο πρωτόκολλο επικοινωνίας. Υπάρχουν επίσης ασύγχρονες μέθοδοι που δεν χρησιμοποιούν σήμα ρολογιού. Για παράδειγμα, στην επικοινωνία UART, και οι δύο πλευρές ρυθμίζονται σε ένα προκαθορισμένο baud rate που υπαγορεύει την ταχύτητα και το χρονισμό της μετάδοσης δεδομένων.

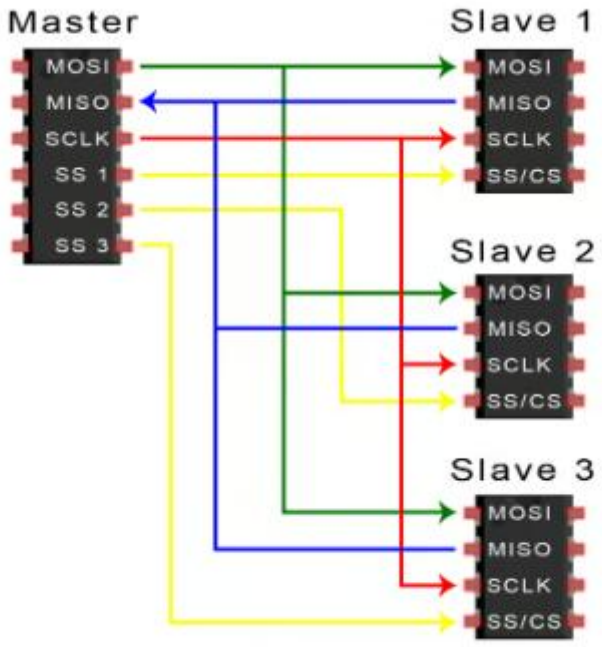
Το σήμα ρολογιού στο SPI μπορεί να τροποποιηθεί χρησιμοποιώντας τις ιδιότητες της πολικότητας του ρολογιού και της φάσης του ρολογιού. Αυτές οι δύο ιδιότητες συνεργάζονται για να καθορίσουν πότε τα bits εξάγονται και πότε γίνεται η δειγματοληψία τους. Η πολικότητα του ρολογιού μπορεί να ρυθμιστεί από τον master για να επιτρέψει την εξαγωγή και τη δειγματοληψία των bits είτε στην ανερχόμενη είτε στην κατερχόμενη ακμή του κύκλου του ρολογιού. Η φάση του ρολογιού μπορεί να ρυθμιστεί για να συμβεί η έξοδος και η δειγματοληψία είτε στην πρώτη είτε στη δεύτερη ακμή του κύκλου του ρολογιού, ανεξάρτητα από το αν ανεβαίνει ή κατεβαίνει.

Επιλογή Slave

Ο master μπορεί να επιλέξει με ποιον slave θέλει να επικοινωνήσει ρυθμίζοντας τη γραμμή CS/SS του slave σε χαμηλό επίπεδο τάσης. Στην κατάσταση αδράνειας, μη μετάδοσης, η γραμμή επιλογής slave διατηρείται σε υψηλό επίπεδο τάσης. Πολλαπλά CS/SS pins ενδέχεται να είναι διαθέσιμα στον master, γεγονός που επιτρέπει την παράλληλη καλωδίωση πολλαπλών slaves. Εάν υπάρχει μόνο ένα pin CS/SS, πολλαπλοί slaves μπορούν να συνδεθούν στον master με αλυσιδωτή σύνδεση.

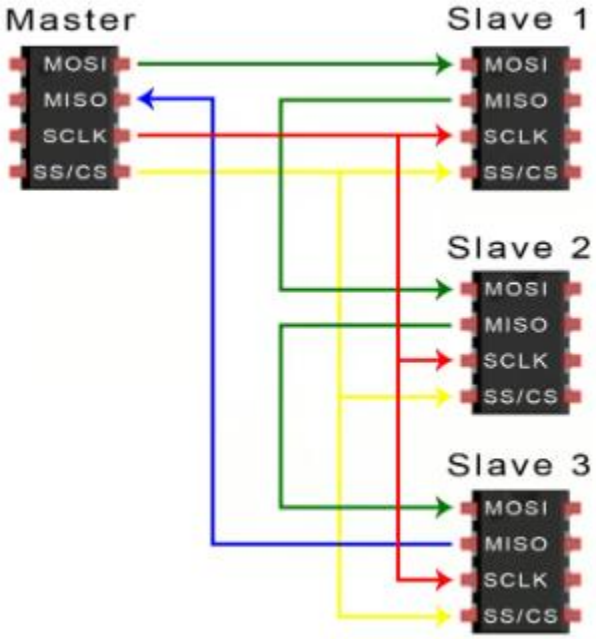
Πολλαπλοί Slaves

Το SPI μπορεί να ρυθμιστεί ώστε να λειτουργεί με έναν μόνο master και έναν μόνο slave, και μπορεί να ρυθμιστεί με πολλαπλούς slaves που ελέγχονται από έναν μόνο master. Υπάρχουν δύο τρόποι σύνδεσης πολλαπλών slaves στον master. Εάν ο master έχει πολλαπλά pins επιλογής slave, οι slaves μπορούν να συνδεθούν παράλληλα ως εξής:



3.2 Συνδεσμολογία Master με πολλαπλά pins επιλογής Slave

Αν είναι διαθέσιμο μόνο ένα pin επιλογής slave, οι slaves μπορούν να συνδεθούν σε αλυσίδα όπως παρακάτω:



3.3Συνδεσμολογία Master με ένα pin επιλογής Slave

MOSI και MISO

Ο master στέλνει δεδομένα στον slave bit προς bit, σειριακά μέσω της γραμμής MOSI. Ο slave λαμβάνει τα δεδομένα που αποστέλλονται από τον master στο pin αυτό. Τα δεδομένα που αποστέλλονται από τον master στον slave αποστέλλονται συνήθως με το πιο σημαντικό bit πρώτο.

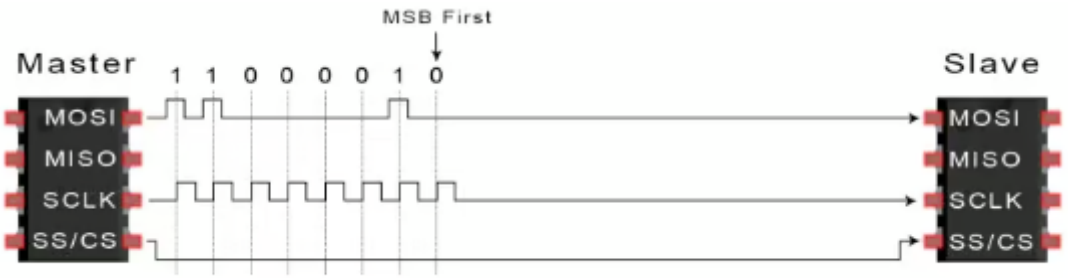
Ο slave μπορεί επίσης να στείλει δεδομένα πίσω στον master μέσω της γραμμής MISO σειριακά. Τα δεδομένα που αποστέλλονται από τον slave πίσω στον master αποστέλλονται συνήθως με το λιγότερο σημαντικό bit πρώτο.

Βήματα της SPI Μετάδοσης Δεδομένων

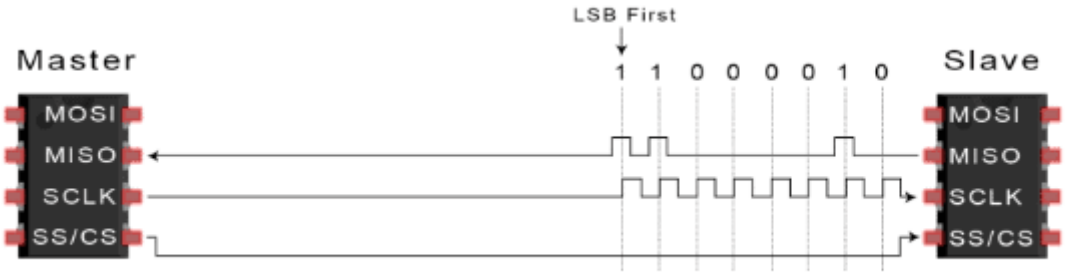
1. Ο master εξάγει το σήμα του ρολογιού:

2. Ο master βάζει το pin SS/CS σε κατάσταση χαμηλής τάσης, η οποία ενεργοποιεί τον slave:

3. Ο master στέλνει τα δεδομένα με ένα bit τη φορά στον slave κατά μήκος της γραμμής MOSI. Ο slave διαβάζει τα bits καθώς λαμβάνονται:



4. Εάν απαιτείται απάντηση, ο slave επιστρέφει δεδομένα με ένα bit τη φορά στον master κατά μήκος της γραμμής MISO. Ο master διαβάζει τα bits καθώς λαμβάνονται:



Πλεονεκτήματα και Μειονεκτήματα του SPI

Πλεονεκτήματα

- Δεν υπάρχουν bit έναρξης και διακοπής, επομένως τα δεδομένα μπορούν να μεταδοθούν συνεχώς χωρίς διακοπή
- Δεν υπάρχει περίπλοκο σύστημα διευθυνσιοδότησης των slaves όπως το I2C
- Υψηλότερος ρυθμός μεταφοράς δεδομένων από το I2C (σχεδόν διπλάσιος)
- Ξεχωριστές γραμμές MISO και MOSI, ώστε τα δεδομένα να μπορούν να αποστέλλονται και να λαμβάνονται ταυτόχρονα (full duplex)

Μειονεκτήματα

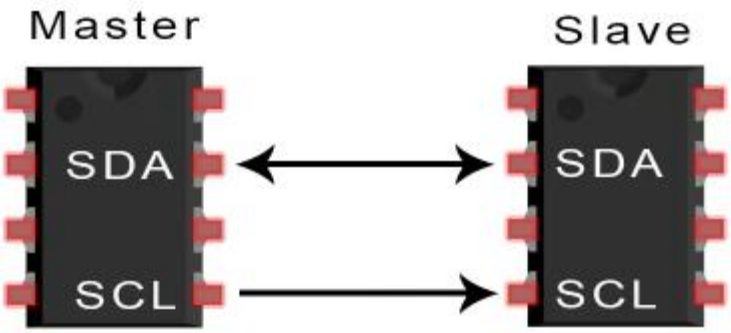
- Χρησιμοποιεί τέσσερα καλώδια σε αντίθεση με τα I2C και UART που χρησιμοποιούν δύο
- Δεν υπάρχει επιβεβαίωση ότι τα δεδομένα έχουν ληφθεί με επιτυχία (Acknowledge bit)
- Δεν υπάρχει μορφή ελέγχου σφαλμάτων όπως το ψηφίο ισοτιμίας στο UART (parity bit)
- Επιτρέπει μόνο έναν master

3.2.2 Πρωτόκολλο I2C (Inter-Integrated Circuit)

Το I2C είναι ένα πρωτόκολλο σειριακής διασύνδεσης διαύλου που χρησιμοποιείται για τη σύνδεση μικροελεγκτών και άλλων εξαρτημάτων σε ενσωματωμένα συστήματα. Με τη χρήση του I2C δίνεται η δυνατότητα σύνδεσης είτε πολλών slaves σε έναν μόνο master, όπως το SPI, είτε η σύνδεση πολλών masters με έναν ή πολλούς slaves.

Η επικοινωνία I2C, γνωστή και ως TWI (Two-Wire Interface) επιτρέπει την επικοινωνία πολλαπλών κόμβων/συσκευών χρησιμοποιώντας δύο καλώδια, γραμμές σήματος και απαιτεί δύο pins:

- **Pin SCL (Ρολόι):** Μεταδίδει το σήμα ρολογιού.
- **Pin SDA (Δεδομένα):** Μεταδίδει και λαμβάνει δεδομένα.



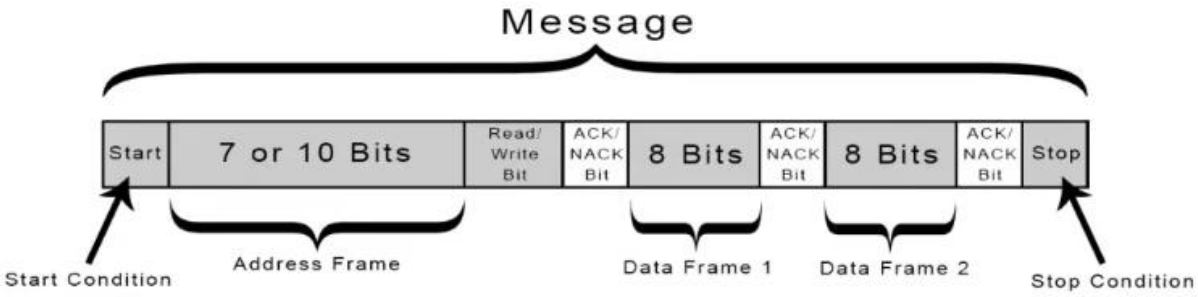
3.4Απεικόνιση πρωτοκόλλου I2C

Το I2C είναι ένα πρωτόκολλο σειριακής επικοινωνίας, οπότε τα δεδομένα μεταφέρονται bit προς bit κατά μήκος ενός μόνο καλωδίου (η γραμμή SDA).

Όπως και το SPI, το I2C είναι σύγχρονο, οπότε η έξοδος των bit συγχρονίζεται με τη δειγματοληψία των bit από ένα σήμα ρολογιού που μοιράζονται ο master και ο slave. Το σήμα ρολογιού ελέγχεται πάντα από τον master.

Λειτουργία I2C

Με το I2C, τα δεδομένα μεταφέρονται σε μηνύματα. Τα μηνύματα χωρίζονται σε πλαίσια δεδομένων. Κάθε μήνυμα έχει ένα πλαίσιο διεύθυνσης που περιέχει τη δυαδική διεύθυνση του slave και ένα ή περισσότερα πλαίσια δεδομένων που περιέχουν τα δεδομένα που μεταδίδονται. Η μορφή ενός I2C μηνύματος περιλαμβάνει επίσης συνθήκες έναρξης και διακοπής, bit ανάγνωσης/εγγραφής και bit ACK/NACK μεταξύ κάθε πλαισίου δεδομένων.



Εικόνα 3.5 Μήνυμα I2C πρωτοκόλλου

Συνθήκη Έναρξης: Η γραμμή SDA μεταβαίνει από υψηλή τάση σε χαμηλή τάση πριν η γραμμή SCL μεταβεί από υψηλή σε χαμηλή.

Συνθήκη Διακοπής: Η γραμμή SDA μεταβαίνει από χαμηλή τάση σε υψηλή τάση μετά την αλλαγή της γραμμής SCL από χαμηλή σε υψηλή.

Πλαίσιο Διεύθυνσης: Μια ακολουθία 7 ή 10 bits μοναδική για κάθε slave που προσδιορίζει την διεύθυνση του slave κατά την επικοινωνία του με τον master.

Bit Ανάγνωσης/Εγγραφής: Ένα μόνο bit που καθορίζει εάν ο master στέλνει δεδομένα στον slave (χαμηλή τάση) ή ζητά δεδομένα από αυτόν (υψηλή τάση).

Bit ACK/NACK: Κάθε πλαίσιο σε ένα μήνυμα ακολουθείται από ένα bit επιβεβαίωσης/μη-επιβεβαίωσης. Εάν ένα πλαίσιο διεύθυνσης ή ένα πλαίσιο δεδομένων ελήφθη με επιτυχία, ένα bit ACK επιστρέφεται στον αποστολέα από τη συσκευή λήψης.

Διευθυνσιοδότηση

Το I2C δεν διαθέτει γραμμές επιλογής slave όπως το SPI, επομένως χρειάζεται έναν άλλο τρόπο για να ενημερώσει τον slave ότι τα δεδομένα αποστέλλονται σε αυτόν και όχι σε έναν άλλο slave. Η διαδικασία αυτή επιτυγχάνεται με την διευθυνσιοδότηση. Το πλαίσιο διεύθυνσης είναι πάντα το πρώτο πλαίσιο μετά το bit έναρξης σε ένα νέο μήνυμα.

Ο master στέλνει τη διεύθυνση του slave με τον οποίο θέλει να επικοινωνήσει σε κάθε slave που είναι συνδεδεμένος σε αυτόν. Κάθε slave στη συνέχεια συγκρίνει τη διεύθυνση που στάλθηκε από τον master με τη δική του διεύθυνση. Εάν η διεύθυνση ταιριάζει, στέλνει ένα bit ACK χαμηλής τάσης πίσω στον master. Εάν η διεύθυνση δεν ταιριάζει, ο slave δεν κάνει τίποτα και η γραμμή SDA παραμένει υψηλή.

Bit Ανάγνωσης/Εγγραφής

Το πλαίσιο διεύθυνσης περιλαμβάνει ένα μόνο bit στο τέλος του που ενημερώνει τον slave εάν ο master θέλει να γράψει δεδομένα σε αυτόν ή να λάβει δεδομένα από αυτόν. Εάν ο master θέλει να στείλει δεδομένα στον slave, το bit ανάγνωσης/εγγραφής έχει χαμηλό επίπεδο τάσης. Εάν ο master ζητά δεδομένα από τον slave, το bit είναι ένα υψηλό επίπεδο τάσης.

Το Πλαίσιο Δεδομένων

Αφού ο master ανιχνεύσει το bit ACK από τον slave, το πρώτο πλαίσιο δεδομένων είναι έτοιμο να σταλεί.

Το πλαίσιο δεδομένων έχει πάντα μήκος 8 bits και αποστέλλεται με το πιο σημαντικό bit πρώτο. Κάθε πλαίσιο δεδομένων ακολουθείται αμέσως από ένα bit ACK/NACK για να επαληθευτεί ότι το πλαίσιο έχει ληφθεί με επιτυχία ή όχι. Το bit ACK πρέπει να ληφθεί είτε από τον master είτε από τον slave, ανάλογα με τον αποστολέα των δεδομένων, πριν από την αποστολή του επόμενου πλαισίου δεδομένων.

Αφού σταλούν όλα τα πλαίσια δεδομένων, ο master μπορεί να στείλει μια συνθήκη διακοπής στον slave για να σταματήσει τη μετάδοση. Η συνθήκη διακοπής είναι μια μετάβαση τάσης από χαμηλή σε υψηλή στη γραμμή SDA μετά από μια μετάβαση από χαμηλή σε υψηλή στη γραμμή SCL, με τη γραμμή SCL να παραμένει υψηλή.

Βήματα I2C Μετάδοσης Δεδομένων

1. Ο master στέλνει την συνθήκη έναρξης σε κάθε συνδεδεμένο slave αλλάζοντας τη γραμμή SDA από υψηλή τάση σε χαμηλή τάση όσο η γραμμή SCL παραμένει σε υψηλή.

2. Ο master στέλνει σε κάθε slave τη διεύθυνση των 7 ή 10 bits του slave με τον οποίο θέλει να επικοινωνήσει, μαζί με το bit ανάγνωσης/εγγραφής.
3. Κάθε slave συγκρίνει τη διεύθυνση που στάλθηκε από τον master με τη δική του διεύθυνση. Εάν η διεύθυνση ταιριάζει, ο slave επιστρέφει ένα bit ACK τραβώντας τη γραμμή SDA σε χαμηλό επίπεδο τάσης. Εάν η διεύθυνση από τον master δεν ταιριάζει με τη δική του διεύθυνση του slave, ο slave αφήνει τη γραμμή SDA σε υψηλό επίπεδο τάσης.
4. Ο master στέλνει ή λαμβάνει το πλαίσιο δεδομένων.
5. Μετά την μεταφορά κάθε πλαισίου δεδομένων, η συσκευή λήψης επιστρέφει ένα άλλο bit ACK στον αποστολέα για να επιβεβαιώσει την επιτυχή λήψη του πλαισίου.
6. Για να σταματήσει η μετάδοση δεδομένων, ο master στέλνει μια συνθήκη διακοπής στον slave μεταβαίνοντας το SDA σε χαμηλή κατάσταση όσο το SCL παραμένει σε υψηλή.

Πλεονεκτήματα και Μειονεκτήματα του I2C

Υπάρχουν πολλοί παράγοντες στο πρωτόκολλο I2C που μπορεί να το καθιστούν πιο σύνθετο σε σύγκριση με άλλα πρωτόκολλα. Παρόλα αυτά υπάρχουν σαφείς λόγοι για τους οποίους μπορεί κάποιος να επιλέξει ή να αποφύγει την χρήση του I2C για τη σύνδεση με μία συγκεκριμένη συσκευή:

Πλεονεκτήματα

- Χρησιμοποιεί μόνο δύο καλώδια
- Υποστηρίζει πολλαπλούς masters και πολλαπλούς slaves
- Το bit ACK/NACK δίνει επιβεβαίωση ότι κάθε frame μεταφέρεται με επιτυχία
- Γνωστό και ευρέως χρησιμοποιούμενο πρωτόκολλο

Μειονεκτήματα

- Χαμηλότερος ρυθμός μεταφοράς δεδομένων από το SPI
- Το μέγεθος του data frame περιορίζεται σε 8 bits
- Πιο περίπλοκο hardware απαιτείται για την υλοποίηση από το SPI

3.2.3 Πρωτόκολλο 1-Wire

Το 1-Wire είναι ένα πρωτόκολλο σειριακής επικοινωνίας που αναπτύχθηκε από την Dallas Semiconductor και επιτρέπει στις συσκευές να επικοινωνούν μέσω μιας μόνο γραμμής δεδομένων και μια σύνδεση γείωσης (GND).

Το πρωτόκολλο 1-Wire εισήχθη για πρώτη φορά τη δεκαετία του 1980, και δημιουργήθηκε για να μειώσει το κόστος καλωδίωσης των περιφερειακών. Είναι ιδανικό για εφαρμογές όπου η ελαχιστοποίηση της καλωδίωσης και της πολυπλοκότητας του συστήματος είναι απαραίτητη.

Ένα βασικό χαρακτηριστικό του πρωτοκόλλου 1-Wire είναι ότι ένα μόνο καλώδιο μεταφέρει τόσο δεδομένα όσο και ρεύμα στις συνδεδεμένες συσκευές, καθιστώντας το ιδανικό για ενεργειακά αποδοτικές εφαρμογές και συσκευές με περιορισμένες διαθέσιμες γραμμές I/O.

Κεφάλαιο 3

Λόγω της απλότητας της σύνδεσης, της χαμηλής κατανάλωσης ενέργειας και της αξιοπιστίας του, το 1-Wire χρησιμοποιείται ευρέως σε εφαρμογές όπως βιομηχανικά συστήματα, ηλεκτρονικά είδη ευρείας κατανάλωσης και συστήματα ελέγχου πρόσβασης. Το πρωτόκολλο χρησιμοποιείται συχνά για τη σύνδεση συσκευών όπως:

- **Αισθητήρες θερμοκρασίας και υγρασίας**
- **EEPROMs**
- **Τσιπ αναγνώρισης και αυθεντικοποίησης (iButton)**
- **Ρολόι πραγματικού χρόνου (RTC)**

Επισκόπηση

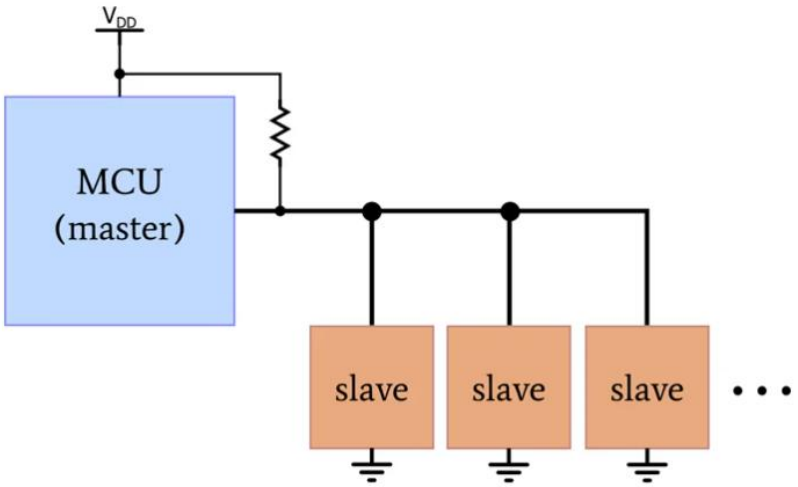
Το πρωτόκολλο 1-Wire δεν διαθέτει ειδική γραμμή χρονισμού, αλλά επίσης δεν είναι ασύγχρονο με τη συνηθισμένη έννοια. Αντίθετα, ο master επιβάλλει αυστηρό χρονισμό μέσω καθορισμένων χρονικών διαστημάτων και κάθε slave απαντά εντός αυτών των ακριβών διαστημάτων. Αυτή η προσέγγιση επιτρέπει την επικοινωνία μέσω μιας μόνο γραμμής δεδομένων χωρίς να απαιτούνται ανεξάρτητα ρολόγια υψηλής ακρίβειας σε κάθε συσκευή, καθώς ο master παρέχει τις απαραίτητες χρονικές αναφορές.

Το 1-Wire υποστηρίζει λειτουργία ημιαμφίδρομης επικοινωνίας. Οι συσκευές στο δίαυλο μπορούν να μεταδίδουν και να λαμβάνουν δεδομένα, αλλά η ταυτόχρονη μετάδοση και λήψη δεν είναι δυνατή. Σε οποιαδήποτε δεδομένη στιγμή, μια συσκευή λειτουργεί ως πομπός, ενώ μια άλλη λειτουργεί ως δέκτης. Παρόλο που κάθε συσκευή στο δίαυλο μπορεί να μεταδώσει δεδομένα, μόνο η κύρια συσκευή μπορεί πάντα να ξεκινήσει την επικοινωνία.

Ο ρυθμός μεταφοράς δεδομένων του 1-Wire είναι σχετικά αργός. Το πρωτόκολλο παρέχει δύο τρόπους λειτουργίας: τυπική λειτουργία (~15 kbps) και λειτουργία υπερ-οδήγησης, υποστηρίζοντας ταχύτητες έως και 125 kbps. Συσκευές που δεν υποστηρίζουν τη λειτουργία υπερ-οδήγησης μπορούν να λειτουργήσουν στον ίδιο δίαυλο με αυτές που την υποστηρίζουν.

Το πρωτόκολλο βασίζεται σε μια αρχιτεκτονική master-slave με σαφή διαίρεση σε συσκευές master και slave. Η δυναμική εναλλαγή ρόλων κατά τη διάρκεια της λειτουργίας του πρωτοκόλλου δεν υποστηρίζεται.

Παρά την απλότητά του, το 1-Wire είναι ένα εξαιρετικά ευέλικτο πρωτόκολλο. Ενσωματώνει πολλές έννοιες που έχουν εφαρμογές σε πιο σύνθετα πρωτόκολλα, καθιστώντας το πολύτιμο τόσο για μελέτη όσο και για πρακτικές εφαρμογές.

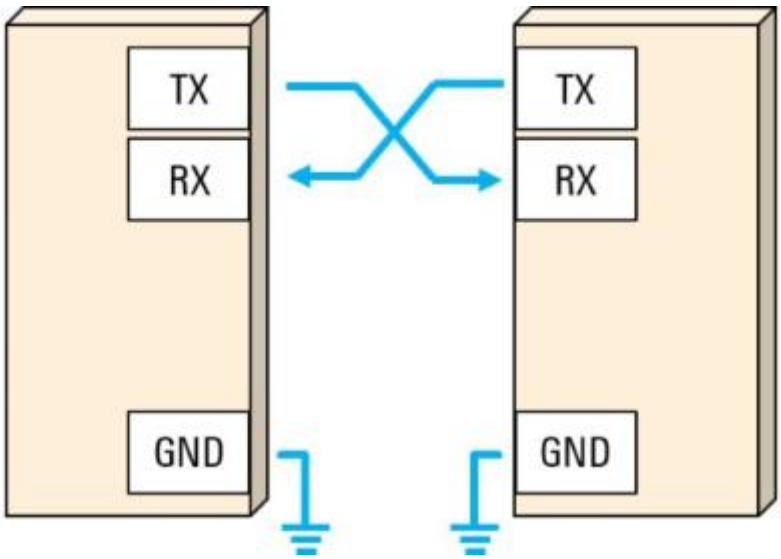


Εικόνα 3.6 Συνδεσμολογία ενός 1-Wire διαύλου

3.2.4 Πρωτόκολλο UART (Universal Asynchronous Receiver/Transmitter)

Οι μικροελεγκτές και οι υπολογιστές, όπως, επίσης, και τα ολοκληρωμένα συστήματα χρησιμοποιούν κυρίως το UART ως μορφή πρωτοκόλλου επικοινωνίας υλικού μεταξύ συσκευών. Μεταξύ των διαθέσιμων πρωτοκόλλων επικοινωνίας, το UART χρησιμοποιεί μόνο δύο καλώδια για τα άκρα μετάδοσης και λήψης.

Εξ ορισμού, το UART είναι ένα πρωτόκολλο επικοινωνίας υλικού που χρησιμοποιεί ασύγχρονη σειριακή επικοινωνία με ρυθμιζόμενη ταχύτητα. Ασύγχρονο σημαίνει ότι δεν υπάρχει σήμα ρολογιού για τον συγχρονισμό των bits εξόδου από τη συσκευή μετάδοσης που πηγαίνουν στο άκρο λήψης. Και τα δύο άκρα έχουν επίσης γείωση.



3.7 Απεικόνιση επικοινωνίας UART μεταξύ δύο συσκευών

Στην UART επικοινωνία τα δεδομένα μεταδίδονται σειριακά, με μία από τις παρακάτω τρεις λειτουργίες:

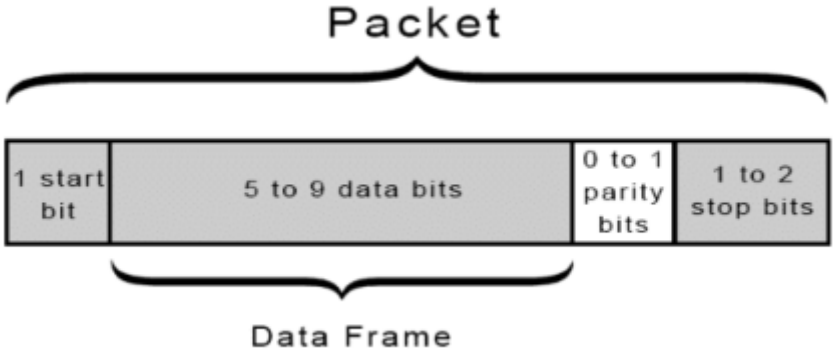
- **Simplex (Μονόδρομη επικοινωνία):** τα δεδομένα αποστέλλονται μόνο προς μία κατεύθυνση.
- **Half duplex (Ημιαμφίδρομη επικοινωνία):** Τα δεδομένα ρέουν προς μία κατεύθυνση κάθε φορά. Κάθε πλευρά μιλάει, αλλά μόνο μία κάθε φορά.
- **Full duplex (Πλήρως αμφίδρομη επικοινωνία):** Ταυτόχρονη επικοινωνία από και προς κάθε κύρια και δευτερεύουσα συσκευή. Και οι δύο πλευρές μπορούν να μεταδίδουν ταυτόχρονα.

Ως ασύγχρονο πρωτόκολλο, κανένα ρολόι δεν ρυθμίζει την ταχύτητα μετάδοσης δεδομένων. Οι χρήστες πρέπει να ρυθμίσουν και τις δύο συσκευές να επικοινωνούν με την ίδια ταχύτητα. Αυτή η ταχύτητα είναι γνωστή ως baud rate, που εκφράζεται σε bits ανά δευτερόλεπτο ή bps. Οι ταχύτητες μετάδοσης ποικίλλουν, από την τυπική ρύθμιση των 9600 baud έως τα 115200 και άνω.

Λειτουργία UART

Το UART που πρόκειται να μεταδώσει δεδομένα λαμβάνει αυτά από έναν δίαυλο δεδομένων. Ο δίαυλος δεδομένων χρησιμοποιείται για την αποστολή αυτών στο UART μέσω μιας άλλης συσκευής, όπως μια CPU, μνήμη ή μικροελεγκτή. Τα δεδομένα μεταφέρονται από τον δίαυλο δεδομένων στο UART μετάδοσης σε παράλληλη μορφή. Αφού το UART μετάδοσης λάβει τα παράλληλα δεδομένα από τον δίαυλο δεδομένων, προσθέτει ένα bit έναρξης, ένα bit ισοτιμίας και ένα bit διακοπής, δημιουργώντας το πακέτο δεδομένων. Στη συνέχεια, το πακέτο δεδομένων εξάγεται σειριακά, bit προς bit στον ακροδέκτη Tx. Το UART λήψης διαβάζει το πακέτο δεδομένων bit προς bit στον ακροδέκτη Rx. Το UART λήψης στη συνέχεια μετατρέπει τα δεδομένα πίσω σε παράλληλη μορφή και αφαιρεί το bit έναρξης, το bit ισοτιμίας και τα bit διακοπής. Τέλος, το UART λήψης μεταφέρει το πακέτο δεδομένων παράλληλα στον δίαυλο δεδομένων στο άκρο λήψης.

Τα δεδομένα που μεταδίδονται μέσω UART οργανώνονται σε πακέτα. Κάθε πακέτο περιέχει 1 bit έναρξης, 5 έως 9 bit δεδομένων (ανάλογα με το UART), ένα προαιρετικό bit ισοτιμίας και 1 ή 2 bit τερματισμού.



3.8 Πακέτο UART επικοινωνίας

Bit έναρξης: Η γραμμή μετάδοσης δεδομένων UART διατηρείται κανονικά σε υψηλή τάση όταν δεν μεταδίδει δεδομένα. Για να ξεκινήσει η μεταφορά δεδομένων, το UART εκπομπής τραβάει τη γραμμή μετάδοσης από υψηλή σε χαμηλή για έναν κύκλο ρολογιού. Όταν το UART λήψης

ανιχνεύσει τη μετάβαση τάσης από υψηλή σε χαμηλή, αρχίζει να διαβάζει τα bits στο πλαίσιο δεδομένων στη συχνότητα του ρυθμού baud.

Πλαίσιο δεδομένων: Το πλαίσιο δεδομένων περιέχει τα πραγματικά δεδομένα που μεταφέρονται. Μπορεί να είναι 5 bit έως 8 bit εάν χρησιμοποιείται ένα bit ισοτιμίας. Εάν δεν χρησιμοποιείται bit ισοτιμίας, το πλαίσιο δεδομένων μπορεί να είναι 9 bit. Στις περισσότερες περιπτώσεις, τα δεδομένα αποστέλλονται με το λιγότερο σημαντικό bit πρώτο.

Bit ισοτιμίας: Η ισοτιμία περιγράφει την αρτιότητα ή την περιττότητα ενός αριθμού. Το bit ισοτιμίας είναι ένας τρόπος για το UART λήψης να γνωρίζει εάν κάποια δεδομένα έχουν αλλάξει κατά τη διάρκεια της μετάδοσης. Τα bits μπορούν να αλλάξουν από ηλεκτρομαγνητική ακτινοβολία, μη αντιστοιχισμένες ταχύτητες baud ή μεταφορές δεδομένων μεγάλων αποστάσεων. Αφού το UART λήψης διαβάσει το πλαίσιο δεδομένων, μετρά τον αριθμό των bits με τιμή 1 και ελέγχει εάν το σύνολο είναι ένας άρτιος ή περιττός αριθμός. Εάν το bit ισοτιμίας είναι 0 (άρτια ισοτιμία), τα bits 1 στο πλαίσιο δεδομένων θα πρέπει να έχουν άθροισμα έναν άρτιο αριθμό. Εάν το bit ισοτιμίας είναι 1 (περιττή ισοτιμία), τα bits 1 στο πλαίσιο δεδομένων θα πρέπει να έχουν άθροισμα έναν περιττό αριθμό. Όταν το bit ισοτιμίας ταιριάζει με τα δεδομένα, το UART γνωρίζει ότι η μετάδοση ήταν χωρίς σφάλματα. Αλλά εάν το bit ισοτιμίας είναι 0 και το σύνολο είναι περιττό, ή το bit ισοτιμίας είναι 1 και το σύνολο είναι άρτιο, το UART γνωρίζει ότι τα bits στο πλαίσιο δεδομένων έχουν αλλάξει.

Bits διακοπής: Για να σηματοδοτηθεί το τέλος του πακέτου δεδομένων, το UART αποστολής οδηγεί τη γραμμή μετάδοσης δεδομένων από χαμηλή τάση σε υψηλή τάση για τουλάχιστον δύο χρονικές διάρκειες bit.

Βήματα μετάδοσης UART

1. Το UART εκπομπής λαμβάνει δεδομένα παράλληλα από τον δίαυλο δεδομένων.
2. Το UART εκπομπής προσθέτει το bit έναρξης, το bit ισοτιμίας και το(τα) bit(s) διακοπής στο πλαίσιο δεδομένων.
3. Ολόκληρο το πακέτο αποστέλλεται σειριακά, ξεκινώντας από το bit έναρξης έως το bit διακοπής, από το UART εκπομπής στο UART λήψης. Το UART λήψης δειγματοληπτεί τη γραμμή δεδομένων στην προκαθορισμένη ταχύτητα μετάδοσης.
4. Το UART λήψης απορρίπτει το bit έναρξης, το bit ισοτιμίας και το bit διακοπής από το πλαίσιο δεδομένων.
5. Το UART λήψης μετατρέπει τα σειριακά δεδομένα ξανά σε παράλληλα και τα μεταφέρει στον δίαυλο δεδομένων στην πλευρά λήψης.

Πλεονεκτήματα:

- Χρησιμοποιεί μόνο δύο καλώδια.
- Δεν απαιτείται σήμα ρολογιού.
- Έχει ένα bit ισοτιμίας για έλεγχο σφαλμάτων.
- Η δομή του πακέτου δεδομένων μπορεί να αλλάξει, εφόσον και οι δύο πλευρές είναι ρυθμισμένες για αυτό.
- Καλά τεκμηριωμένη και ευρέως χρησιμοποιούμενη μέθοδος

Κεφάλαιο 3

Μειονεκτήματα:

- Το μέγεθος του πλαισίου δεδομένων περιορίζεται σε μέγιστο 9 bits.
- Δεν υποστηρίζει πολλαπλά συστήματα slave ή πολλαπλά master.
- Οι ρυθμοί baud κάθε UART πρέπει να είναι εντός του 10% ο ένας του άλλου

Χρήσεις

Μπορεί κανείς να χρησιμοποιήσει UART για πολλές εφαρμογές, όπως:

- Εντοπισμός σφαλμάτων: Η έγκαιρη ανίχνευση σφαλμάτων του συστήματος είναι σημαντική κατά την ανάπτυξη. Η προσθήκη UART μπορεί να βοηθήσει σε αυτό το σενάριο καταγράφοντας μηνύματα από το σύστημα.
- Ιχνηλάτηση λειτουργιών κατασκευής: Τα αρχεία καταγραφής είναι πολύ σημαντικά στην κατασκευή. Καθορίζουν λειτουργίες ειδοποιώντας τους χειριστές για το τι συμβαίνει στη γραμμή παραγωγής.
- Ενημερώσεις πελατών ή χρηστών: Οι ενημερώσεις λογισμικού είναι πολύ σημαντικές. Η ύπαρξη πλήρους, δυναμικού υλικού με λογισμικό με δυνατότητα ενημέρωσης είναι σημαντική για την ύπαρξη ενός πλήρους συστήματος.
- Δοκιμές/επαλήθευση: Η επαλήθευση των προϊόντων πριν φύγουν από τη διαδικασία κατασκευής βοηθά στην παράδοση των καλύτερων δυνατών προϊόντων στους πελάτες.

4 Αρχιτεκτονική & Υλοποίηση Συστήματος

4.1 Λογισμικό

Η ανάπτυξη του λογισμικού πραγματοποιήθηκε με τη χρήση του περιβάλλοντος STM32CubeIDE, το οποίο παρέχει ολοκληρωμένα εργαλεία για την παραμετροποίηση, συγγραφή, μεταγλώττιση και αποσφαλμάτωση (debugging) κώδικα. Το STM32CubeIDE βασίζεται στην πλατφόρμα Eclipse και ενσωματώνει τον compiler arm-none-eabi-gcc, καθώς και εργαλεία προγραμματισμού και αποσφαλμάτωσης μέσω USB (ST-Link debugger), διευκολύνοντας την ανάπτυξη εφαρμογών για μικροελεγκτές STM32.

Η γλώσσα προγραμματισμού που επιλέχθηκε είναι η C, λόγω της υψηλής απόδοσης, του άμεσου ελέγχου σε επίπεδο hardware, αλλά και της ευρείας υιοθέτησής της στα ενσωματωμένα συστήματα (embedded systems). Η υλοποίηση βασίστηκε κυρίως σε άμεση προσπέλαση καταχωρητών (register-level programming), αποφεύγοντας κατά το δυνατόν τη χρήση του STM32 HAL (Hardware Abstraction Layer), εκτός από επιλεγμένες περιπτώσεις όπου οι βιβλιοθήκες HAL χρησιμοποιήθηκαν για τη διευκόλυνση της αρχικοποίησης συγκεκριμένων περιφερειακών.

Ο κώδικας του συστήματος οργανώθηκε σε δύο διακριτά μέρη:

- **Bootloader:** Υπεύθυνος για την ενημέρωση (update) του firmware. Εκτελείται πρώτος κατά την εκκίνηση του συστήματος και διαχειρίζεται τη φόρτωση νέου κώδικα στην Flash μνήμη.
- **Κυρίως Εφαρμογή (Application):** Περιλαμβάνει τις βασικές λειτουργίες του συστήματος, όπως την καταγραφή μετρήσεων από αισθητήρες, τον έλεγχο των ρελέ και την αποθήκευση δεδομένων στην κάρτα SD.

4.1.1 Υποστήριξη Συστήματος Αρχείων και Επικοινωνίες

Για την υποστήριξη του συστήματος αρχείων FAT32 στην κάρτα SD, ενσωματώθηκε και παραμετροποιήθηκε η βιβλιοθήκη FatFS, η οποία διαχειρίζεται την ανάγνωση και εγγραφή αρχείων μέσω του διαύλου SPI του STM32. Η επικοινωνία με τις υπόλοιπες περιφερειακές μονάδες υλοποιήθηκε μέσω των πρωτοκόλλων UART, I2C και 1-Wire, για την αμφίδρομη ανταλλαγή δεδομένων με το GPS, το RTC, και τους αισθητήρες αντίστοιχα.

4.1.2 Λειτουργία Bootloader

Το πρόγραμμα του bootloader είναι υπεύθυνο για τον προγραμματισμό του μικροελεκγτή με το κυρίως πρόγραμμα. Κατά την εκκίνηση του bootloader, γίνεται αρχικοποίηση όλων των συστημάτων που χρησιμοποιούνται για την διαδικασία αυτή. Η αρχικοποίηση αυτή περιλαμβάνει:

- Αρχικοποίηση σειριακής θύρας UART για αποστολή μηνυμάτων με σκοπό των εντοπισμό σφαλμάτων (debugging).
- Ρύθμιση συστήματος χρονισμού (timebase).
- Αρχικοποίηση SPI πρωτοκόλλου επικοινωνίας για τη σύνδεση με την SD Card.
- Αρχικοποίηση FatFS για την πρόσβαση στο σύστημα αρχείων της SD Card.

Εφόσον η αρχικοποίηση είναι επιτυχής, ο bootloader διαβάζει με την χρήση FatFS το κυρίως πρόγραμμα.

Ο bootloader επιχειρεί να εντοπίσει και να ανοίξει το αρχείο main_firmware.bin στην κάρτα SD. Εάν το αρχείο υπάρχει, εκτελείται η διαδικασία ενημέρωσης του firmware:

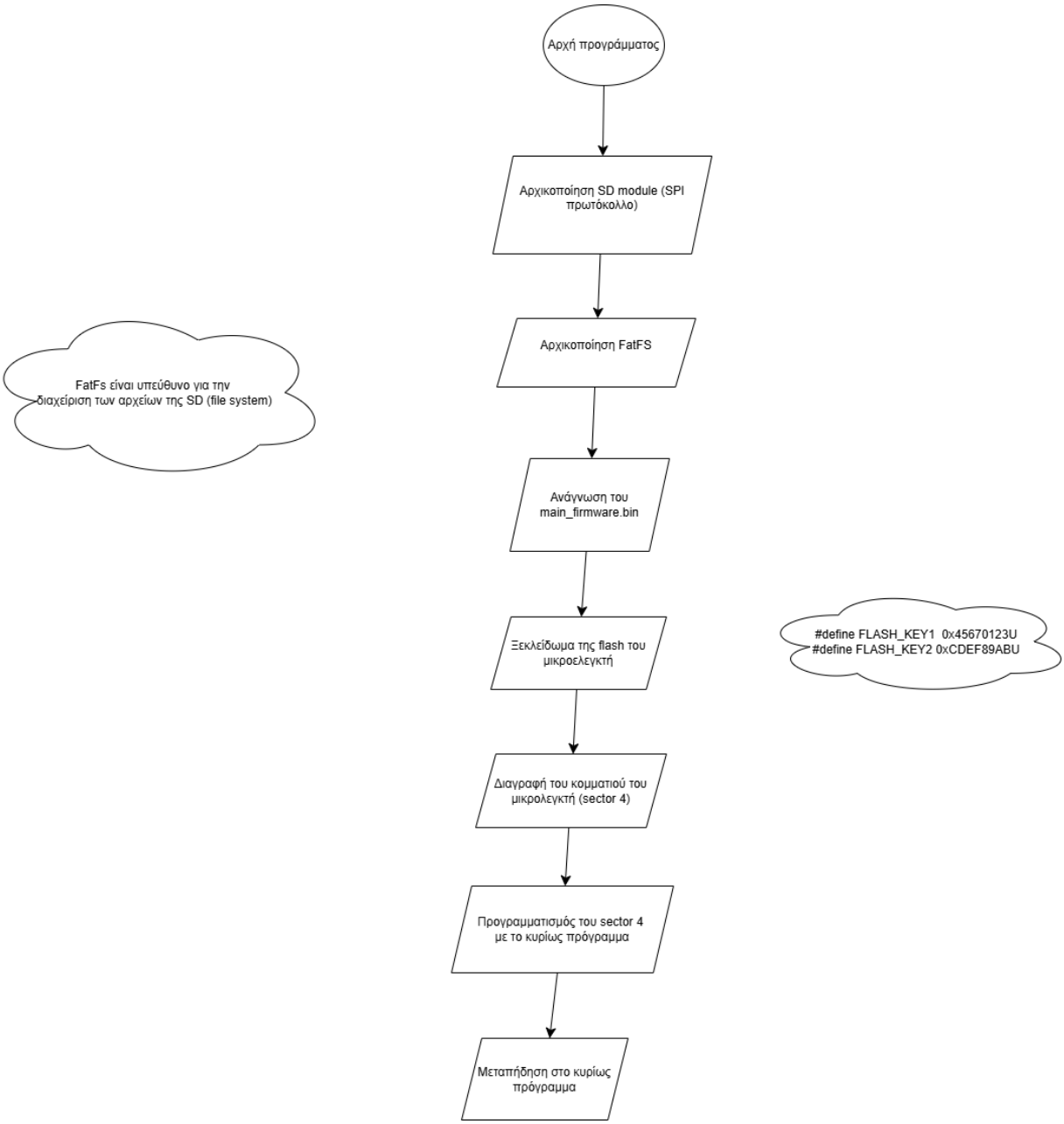
1. Ξεκλειδώνει τη μνήμη Flash του μικροελεγκτή.
2. Διαγράφει τον τομέα 4 (sector 4) της Flash, που προορίζεται για την αποθήκευση του νέου προγράμματος.
3. Διαβάζει διαδοχικά τα δεδομένα του αρχείου σε κομμάτια των 4 bytes (1 word) και τα προγραμματίζει στη Flash, ξεκινώντας από τη διεύθυνση 0x08010000.
4. Μετά την ολοκλήρωση της εγγραφής, κλείνει το αρχείο και απελευθερώνει τη δεσμευμένη μνήμη.
5. Πραγματοποιεί έλεγχο εγκυρότητας στην αρχή της περιοχής εφαρμογής, ελέγχοντας ότι δεν περιέχει την τιμή 0xFFFFFFFF (που δηλώνει κενή ή μη εγγράψιμη μνήμη).
6. Εφόσον υπάρχει έγκυρο πρόγραμμα, ανακτά τη διεύθυνση εκκίνησης από το δεύτερο word της περιοχής (0x08010004) και πραγματοποιεί jump σε αυτήν μέσω δείκτη συνάρτησης, ξεκινώντας έτσι την εκτέλεση της κύριας εφαρμογής.

After reset, write is not allowed in the Flash control register (FLASH_CR) to protect the flash memory against possible unwanted operations due, for example, to electric disturbances. The following sequence is used to unlock this register:

1. Write KEY1 = 0x45670123 in the Flash key register (FLASH_KEYR)
2. Write KEY2 = 0xCDEF89AB in the Flash key register (FLASH_KEYR)

Εικόνα 4.1 Η διαδικασία ξεκλειδώματος της μνήμης flash σύμφωνα με το reference manual του μικροελεκγτή

Για να μπορέσει το κυρίως πρόγραμμα να περαστεί σε ένα τομέα της μνήμης flash (sector) , είναι απαραίτητη η διαγραφή του περιεχομένου του. Αν όλα τα παραπάνω έχουν πραγματοποιηθεί επιτυχώς, το κυρίως πρόγραμμα αποθηκεύεται στον τομέα 4 (sector 4) και ξεκινάει να εκτελείται. Το διάγραμμα ροής της διαδικασίας αυτής φαίνεται στην εικόνα 4.2.



Εικόνα 4.2 Το διάγραμμα ροής της λειτουργίας του bootloader.

4.1.3 Λειτουργία Κυρίως Προγράμματος

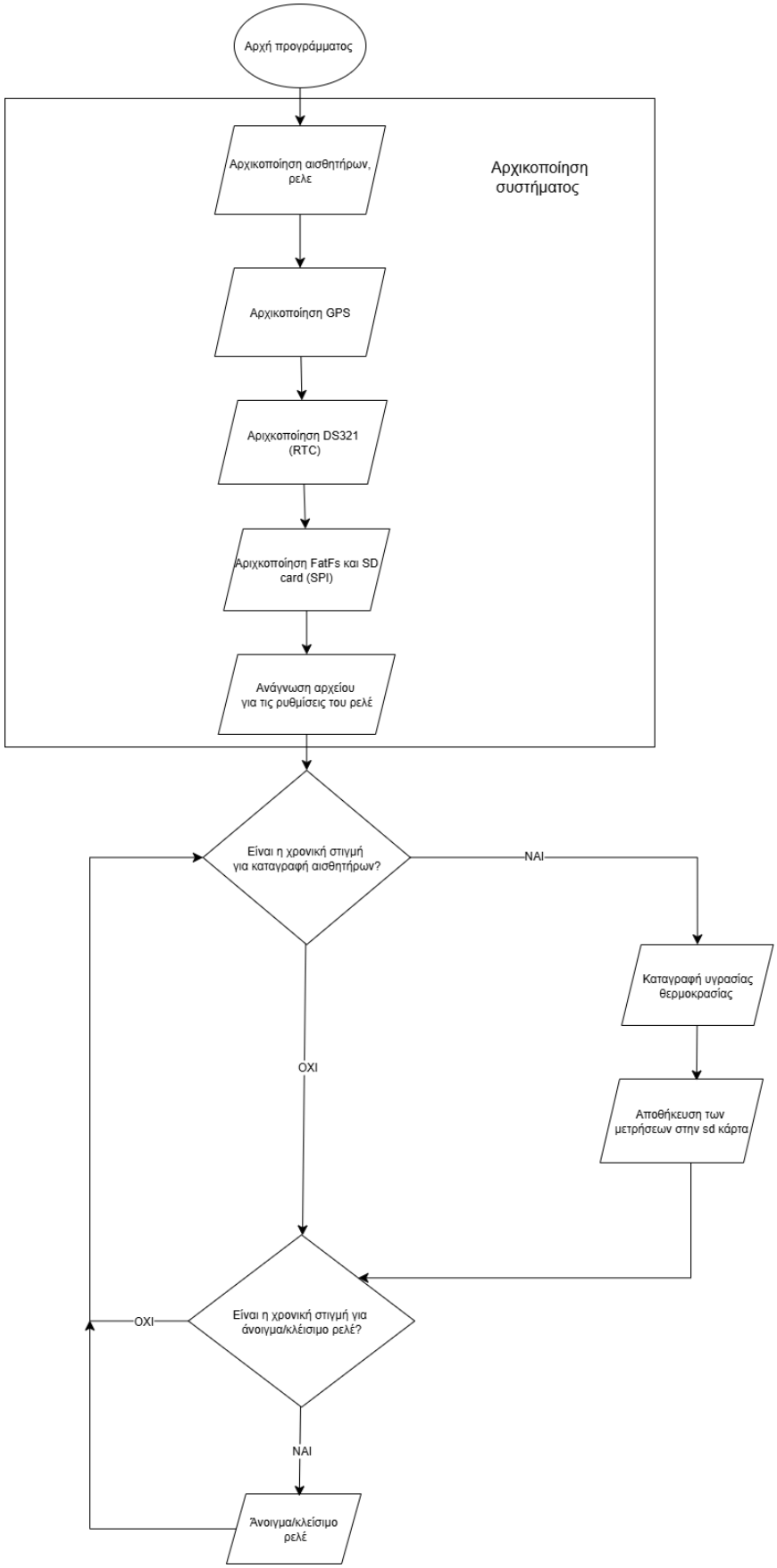
Η κύρια εφαρμογή τρέχει στην περιοχή μνήμης που ορίζει ο bootloader (0x08010000). Ασχολείται με την παρακολούθηση περιβαλλοντικών παραμέτρων (θερμοκρασία, υγρασία), τον έλεγχο ρελέ με βάση προγραμματισμένους χρονισμούς, και την καταγραφή δεδομένων σε κάρτα SD. Κατά την εκκίνηση του προγράμματος, γίνεται αρχικοποίηση του συστήματος, το οποίο περιλαμβάνει:

- **Αρχικοποίηση αισθητήρων και ρελέ**: Ο μικροελεγκτής αρχικοποιεί τους αισθητήρες υγρασίας και θερμοκρασίας, καθώς και το ρελέ, για να διασφαλιστεί η σωστή λειτουργία τους πριν από την καταγραφή δεδομένων.
- **Αρχικοποίηση GPS**: Ο μικροελεγκτής αρχικοποιεί το GPS μέσω πρωτοκόλλου UART λαμβάνοντας την διεθνή ημερομηνία και ώρα (GMT).
- **Αρχικοποίηση DS3231 (RTC):** Το RTC ρυθμίζεται ώστε να παρέχει την ακριβή ώρα και ημερομηνία που έχει ληφθεί από το GPS. Η επικοινωνία πραγματοποιείται μέσω του διαύλου I2C.
- **Αρχικοποίηση FatFs και SD Card (SPI):** Ο μικροελεγκτής αρχικοποιεί το σύστημα αρχείων FAT32 στην SD κάρτα χρησιμοποιώντας το πρωτόκολλο SPI, επιτρέποντας την πρόσβαση στα αρχεία ρυθμίσεων και καταγραφής δεδομένων.
- **Ανάγνωση Αρχείου Ρυθμίσεων:** Το σύστημα φορτώνει τις προκαθορισμένες ρυθμίσεις του ρελέ από ένα αρχείο κειμένου στην SD κάρτα (alarms.txt), ώστε να καθοριστεί πότε πρέπει να ενεργοποιείται ή να απενεργοποιείται.

Μετά την αρχικοποίηση, το πρόγραμμα ελέγχει αν η τρέχουσα χρονική στιγμή, όπως παρέχεται από το RTC, είναι κατάλληλη για την καταγραφή των μετρήσεων. Σε περίπτωση που είναι η χρονική στιγμή για μετρήσεις, το σύστημα:

- Επικοινωνεί με τον αισθητήρα για την συλλογή των τιμών της θερμοκρασίας και της υγρασίας
- Αποθηκεύει τις τιμές αυτές στην SD κάρτα (αρχείο log.txt).

Εκτός από τον έλεγχο για την χρονική στιγμή των μετρήσεων, γίνεται έλεγχος για το αν πρέπει να ανοίξει ή να κλείσει το ρελέ εκείνη την χρονική στιγμή. Με αυτούς τους δύο ελέγχους ολοκληρώνεται η λειτουργία του κυρίως προγράμματος.



4.3 Το διάγραμμα ροής του κυρίως προγράμματος

4.1.3.1 Αρχικοποίηση

Στη συνάρτηση initAll() πραγματοποιούνται όλες οι απαραίτητες αρχικοποιήσεις:

- Σειριακή θύρα UART για debugging.
- Ρύθμιση βάσης χρόνου (timebase).
- Αρχικοποίηση αισθητήρα θερμοκρασίας/υγρασίας (DHT22).
- Επικοινωνία μέσω SPI με την κάρτα SD και αρχικοποίηση FatFS.
- Αρχικοποίηση RTC (DS3231).
- Ρύθμιση UART για επικοινωνία με GPS και προετοιμασία λήψης δεδομένων.

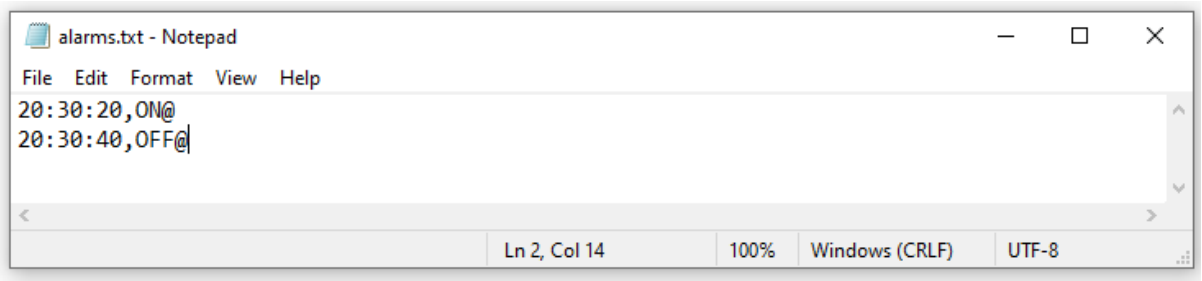
4.1.3.2 Συγχρονισμός Ώρας με GPS

Κατά την εκκίνηση, η εφαρμογή επιχειρεί να συγχρονίσει την ώρα του RTC με αυτή που παρέχεται από το GPS, μέσω της συνάρτησης GpsParse(). Εάν η πρόταση ληφθεί και επεξεργαστεί επιτυχώς:

- Ενημερώνεται η ώρα και ημερομηνία του RTC.
- Υπολογίζεται η ημέρα της εβδομάδας βάσει της μεθόδου του Zeller μέσω της CalculateDayOfWeek().

4.1.3.3 Ανάγνωση Χρονισμών Ρελέ

Κατά την εκκίνηση, το σύστημα διαβάζει από το αρχείο alarms.txt τις ρυθμίσεις για την ενεργοποίηση και απενεργοποίηση του ρελέ. Τα δεδομένα αναλύονται με τη χρήση της sscanf() και αποθηκεύονται σε μεταβλητές τύπου relayConfig. Το αρχείο περιέχει γραμμές της μορφής:



4.4Το αρχείο ρυθμίσεων του ρελέ

Αν το αρχείο δεν είναι διαθέσιμο ή έχει μη έγκυρο περιεχόμενο, το σύστημα παρακάμπτει τον έλεγχο, διασφαλίζοντας την αξιοπιστία.

4.1.3.4 Κύριος Βρόχος Εκτέλεσης (Main Loop)

Μέσα στον ατέρμονα βρόχο του προγράμματος εκτελούνται οι εξής βασικές λειτουργίες:

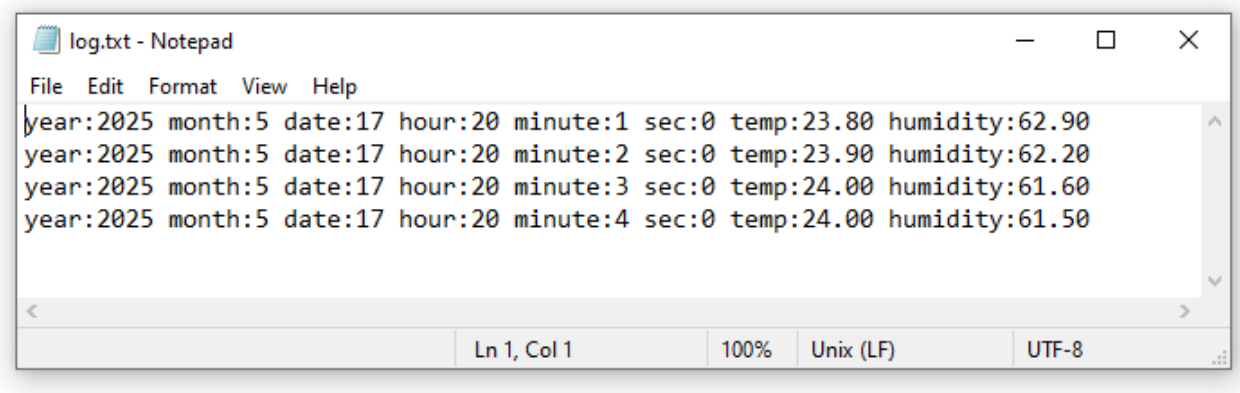
- **Ανάγνωση και Αποθήκευση Μετρήσεων:** Κάθε 1 λεπτό, εφόσον ο τρέχων χρόνος το επιτρέπει, λαμβάνονται μετρήσεις θερμοκρασίας και υγρασίας από τον αισθητήρα DHT22.

Τα δεδομένα αποθηκεύονται στο αρχείο log.txt της κάρτας SD μέσω της συνάρτησης f_write().

- **Έλεγχος και Χειρισμός Ρελέ:** Συγκρίνει την τρέχουσα ώρα με τις προγραμματισμένες τιμές ενεργοποίησης/απενεργοποίησης από το alarms.txt. Αν υπάρχει ταύτιση, ενεργοποιεί ή απενεργοποιεί το ρελέ μέσω pin, με χρήση των συναρτήσεων relay_on() και relay_off(). Χρησιμοποιείται flag ώστε να αποφεύγεται η επανεκτέλεση της ίδιας εντολής μέσα στο ίδιο λεπτό.

4.1.4 Καταγραφή Δεδομένων Αισθητήρων

Οι μετρήσεις θερμοκρασίας και υγρασίας λαμβάνονται από τον DHT22 μέσω του δίαυλου 1-Wire. Οι μετρήσεις αποθηκεύονται σε αρχείο log.txt όπως φαίνεται παρακάτω.



4.5Το αρχείο καταγραφής θερμοκρασίας και υγρασίας

Η καταγραφή γίνεται περιοδικά, ανά 1 λεπτό, πάντα εφόσον το σύστημα είναι ενεργό.

4.2 Υλοποίηση Hardware

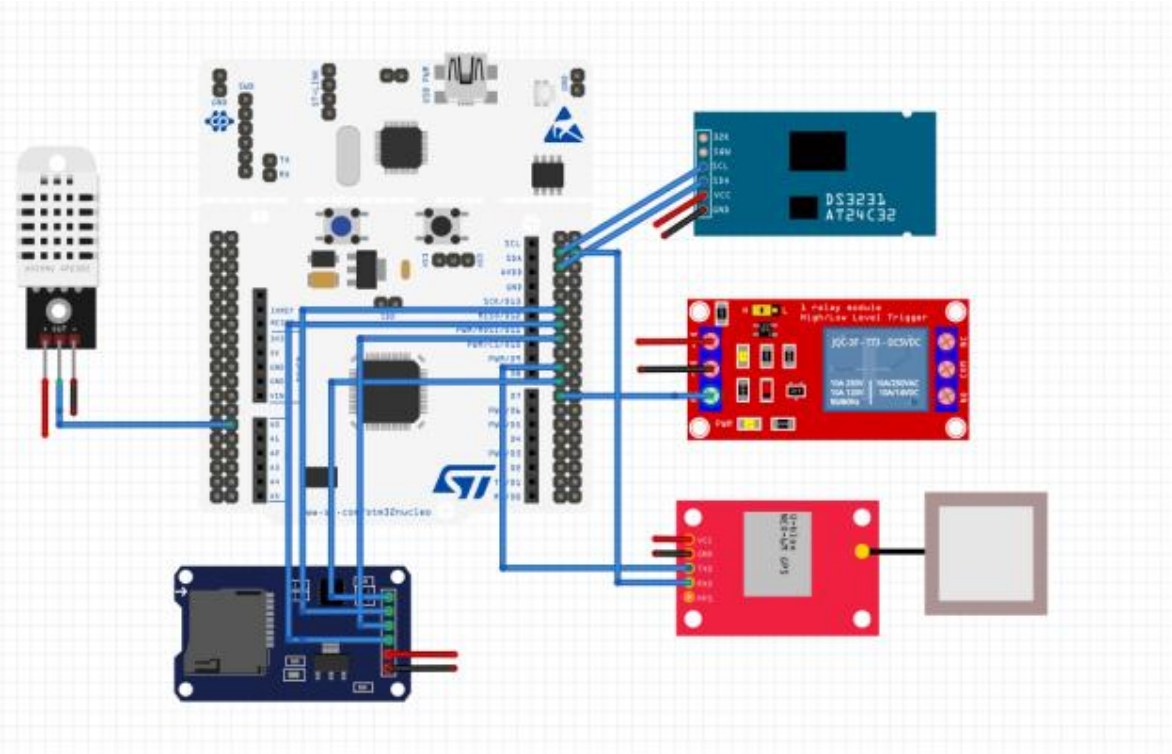
Για την απεικόνιση της υλοποίησης του κυκλώματος, χρησιμοποιήθηκε το λογισμικό Fritzing, το οποίο επιτρέπει την εύκολη σχεδίαση σχηματικών διαγραμμάτων και διατάξεων σε breadboard ή PCB. Στο σχηματικό διάγραμμα αποτυπώνονται οι συνδέσεις του STM32 μικροελεγκτή με τις εξωτερικές περιφερειακές μονάδες του συστήματος.

Το κύκλωμα περιλαμβάνει:

- **Μικροελεγκτή STM32F401RE**, που αποτελεί την κεντρική μονάδα ελέγχου.
- **Κάρτα SD**, η οποία συνδέεται μέσω διαύλου SPI για την αποθήκευση ρυθμίσεων και δεδομένων καταγραφής.
- **GPS module**, το οποίο επικοινωνεί με το STM32 μέσω UART και παρέχει σήμα για τον συγχρονισμό ώρας.
- **RTC DS3231**, που διατηρεί την ώρα του συστήματος και συνδέεται μέσω I2C.
- **Αισθητήρας θερμοκρασίας και υγρασίας (DHT22)**, ο οποίος παρέχει τις περιβαλλοντικές μετρήσεις.
- **Ρελέ**, που ενεργοποιείται μέσω pin για τον έλεγχο εξωτερικού φορτίου.

Κεφάλαιο 4

Το παρακάτω σχηματικό διευκολύνει την κατασκευή του πρωτοτύπου σε breadboard και αποτελεί τη βάση για μελλοντική σχεδίαση τυπωμένου κυκλώματος (PCB), αν απαιτείται.



Εικόνα 4.6 Σχηματικό κυκλώματος του συστήματος καταγραφής

5 Συμπεράσματα

5.1 Επίτευξη

Η παρούσα εργασία επικεντρώθηκε στον σχεδιασμό και την υλοποίηση ενός αυτόνομου και επεκτάσιμου συστήματος καταγραφής δεδομένων βασισμένου στον μικροελεγκτή STM32F401RE και του ελέγχου ρελέ βάσει χρονισμού. Το σύστημα επιτυγχάνει την περιοδική μέτρηση δεδομένων από αισθητήρες, την αποθήκευσή τους σε κάρτα SD, καθώς και τη χρονοσήμανσή τους με χρήση του RTC που συγχρονίζεται μέσω GPS. Επιπλέον, υποστηρίζεται διαχείριση εξόδων μέσω ρελέ, σύμφωνα με παραμετρικά αρχεία ρυθμίσεων που αποθηκεύονται στο σύστημα αρχείων FAT32.

Τα βασικά αποτελέσματα που προέκυψαν από την υλοποίηση και τις δοκιμές περιλαμβάνουν:

- **Αξιόπιστη λειτουργία της καταγραφής** δεδομένων ανά καθορισμένα χρονικά διαστήματα.
- **Επιτυχής αποθήκευση δεδομένων** σε αρχείο .txt σε κάρτα SD.
- **Ακριβής χρονοσήμανση** μέσω GPS και ενημέρωση του RTC.
- **Ευελιξία παραμετροποίησης** μέσω αρχείου ρυθμίσεων, το οποίο διαβάζεται κατά την εκκίνηση του συστήματος.
- **Απλή και σταθερή λειτουργία των ρελέ**, με προγραμματιζόμενα χρονικά διαστήματα ενεργοποίησης/απενεργοποίησης.

Η λειτουργικότητα του συστήματος δοκιμάστηκε σε πραγματικές συνθήκες, και το σύστημα ανταποκρίθηκε με επιτυχία στους καθορισμένους στόχους.

5.2 Βελτιστοποίηση και Επέκταση

Η τρέχουσα υλοποίηση παρέχει τη βάση για ένα σύστημα καταγραφής δεδομένων, ενώ εξακολουθούν να υπάρχουν δυνατότητες για περαιτέρω βελτιστοποίηση και επεκτάσεις. Ορισμένες προτάσεις για μελλοντική εργασία περιλαμβάνουν:

- **Υποστήριξη ασύρματης επικοινωνίας** (π.χ. LoRa ή Wi-Fi) για αποστολή δεδομένων σε απομακρυσμένο server ή cloud υπηρεσία.
- **Υποστήριξη Ενημέρωσης OTA:** Ενεργοποίηση ενημερώσεων firmware για βελτίωση της συντηρησιμότητας (π.χ. Wi-Fi). Αν και η ενημέρωση του firmware μέσω SD κάρτας και bootloader αποτελεί λειτουργική λύση, μια μελλοντική προσθήκη θα μπορούσε να περιλαμβάνει ασύρματη επικοινωνία (π.χ. Wi-Fi ή LoRa) για απομακρυσμένη ενημέρωση του προγράμματος.
- **Ανάπτυξη διεπαφής χρήστη (UI)**, είτε μέσω οθόνης TFT είτε μέσω web interface για τον έλεγχο του συστήματος.
- **Ανίχνευση και αντιμετώπιση σφαλμάτων** στη λειτουργία αισθητήρων ή του GPS, με χρήση watchdog ή fallback μηχανισμών. Επί του παρόντος, αν κάποιο αρχείο (όπως το alarms.txt) δεν είναι σωστά μορφοποιημένο, το σύστημα παραλείπει την εκτέλεση χωρίς ειδοποίηση. Μία λύση θα ήταν να ενσωματωθεί ένας μηχανισμός εντοπισμού σφαλμάτων, ώστε να ενημερώνεται ο χρήστης (π.χ. μέσω σειριακής θύρας ή λυχνίας LED) όταν υπάρχουν προβλήματα στα δεδομένα εισόδου.

- **Προσθήκη επιπλέον αισθητήρων** (π.χ. αισθητήρα φωτός, αισθητήρα βροχής, αισθητήρα ανίχνευσης πυρκαγιάς) για πιο σύνθετη καταγραφή.
- **Επέκταση αριθμού ρελέ και προγραμματισμών:** Η παρούσα υλοποίηση υποστηρίζει έναν μόνο ρελέ και ένα σύνολο χρονισμών. Η δομή του κώδικα μπορεί να επεκταθεί ώστε να υποστηρίζει πολλαπλά ρελέ με ξεχωριστό πρόγραμμα για το καθένα, επιτρέποντας πιο σύνθετες εφαρμογές, όπως πολύζωνη άρδευση.
- **Ενεργειακή βελτιστοποίηση (Low Power Mode)** μέσω χρήσης εξωτερικού RTC με wake-up interrupt και περαιτέρω μείωση κατανάλωσης με deep sleep modes. Αν και το σύστημα προορίζεται για αυτόνομη χρήση, δεν αξιοποιεί επί του παρόντος τις δυνατότητες εξοικονόμησης ενέργειας του μικροελεγκτή STM32. Ο μικροελεγκτής παραμένει ενεργός σε κανονική λειτουργία ακόμα και κατά την αναμονή, γεγονός που οδηγεί σε αυξημένη κατανάλωση. Μια σημαντική βελτίωση θα ήταν η ένταξη του STM32 σε κατάσταση ύπνου (Sleep ή Stop mode) μεταξύ των περιόδων δειγματοληψίας. Για παράδειγμα, μετά την ολοκλήρωση μιας μέτρησης και της αποθήκευσης της, ο μικροελεγκτής θα μπορούσε να εισέλθει σε κατάσταση χαμηλής κατανάλωσης έως το επόμενο ξύπνημα. Η αφύπνιση μπορεί να επιτευχθεί είτε με χρήση του RTC (μέσω interrupt), είτε με εξωτερικό γεγονός.
- **Χρήση EEPROM ή Flash για Αποθήκευση Κατάστασης:** Η τρέχουσα κατάσταση του ρελέ δεν αποθηκεύεται σε μη πτητική μνήμη, επομένως μετά από διακοπή ρεύματος χάνεται. Η αποθήκευση της τελευταίας εντολής σε ενσωματωμένη EEPROM ή ελεύθερη περιοχή της Flash θα επέτρεπε την ανάκτηση της κατάστασης μετά από επανεκκίνηση.
- **Λειτουργία Διαγραφής Δεδομένων:** Εφαρμογή αυτόματης διαγραφής παλαιών δεδομένων.
- **Χρήση αλγορίθμων εξυπνότερης απόφασης:** Στο μέλλον, αντί για απλό χρονισμό, το σύστημα θα μπορούσε να λαμβάνει αποφάσεις βάσει συνθηκών περιβάλλοντος. Για παράδειγμα, αν η υγρασία είναι πάνω από ένα όριο, να μην ενεργοποιείται το ρελέ (π.χ. για άρδευση), ακόμα κι αν έχει προγραμματιστεί. Έτσι, η συμπεριφορά του συστήματος θα προσεγγίζει ένα έξυπνο IoT σύστημα.

Η συνέχιση και εξέλιξη του συστήματος μπορεί να το καταστήσει ιδιαίτερα χρήσιμο σε πραγματικές εφαρμογές, όπως η γεωργία ακριβείας, η περιβαλλοντική παρακολούθηση, ή ακόμα και η βιομηχανική καταγραφή παραγωγής, παρέχοντας αξιόπιστα και χρονομετρημένα δεδομένα από το πεδίο.

6 Παραρτήματα

6.1 main.c bootloader

```
#include <stdio.h>
#include "stm32f4xx.h"
#include "fpu.h"
#include "uart.h"
#include "timebase.h"
#include "adc.h"
#include "string.h"
#include "ff.h"

#include "sd_card_driver.h"
#include "fatfs_app.h"
#include "sd_card_spi.h"
#include "flash_driver.h"

#define MAIN_APP_START_ADDRESS 0x08010000
#define EMPTY_MEM 0xFFFFFFFF

typedef void (*func_ptr) (void);

void jump_to_app(uint32_t addr_value);

int main()
{
unsigned int bytes_read;
uint8_t buffer[4]; // Buffer to hold 4 bytes at a time
```

```
uint32_t word;
uint32_t flash_address = MAIN_APP_START_ADDRESS;

/*Enable FPU*/
fpu_enable();

/*Initialize debug UART*/
debug_uart_init();

/*Initialize timebase*/
timebase_init();

/*Initialize ADC*/
pa1_adc_init();

/*Start ADC*/
start_conversion();

/*Initialize SD*/
sd_cs_pin_init();
sd_card_spi_init();

/*Initialize fatfs*/
fatfs_init();

delay(500);

/* Allocate memory for FATFS and FIL structures */
```

```
FATFS *fat_fs = (FATFS *)malloc(sizeof(FATFS));
FIL *file = (FIL *)malloc(sizeof(FIL));

if (!fat_fs || !file)
{
printf("ERR: Memory Allocation Failed!1\n\r");
free(fat_fs);
free(file);
return 1;
}

/* Mount SD card */
if (f_mount(fat_fs, "", 1) != FR_OK)
{
printf("ERR: SD Card Cannot be Mounted!\n\r");
free(fat_fs);
free(file);
return 1;
}

/* Open the firmware file */
if (f_open(file, "main_firmware.bin", FA_READ) != FR_OK)
{
printf("ERR: Firmware file not found!\n\r");
free(fat_fs);
free(file);
return 1;
```

```
}

/* Unlock flash and erase */
flash_unlock();
flash_sector_erase(4, FLASH_VOLTAGE_RANGE_3);  // Optional: Erase more if firmware is large

/* Read 4 bytes at a time, convert to word, write to flash */
while (f_read(file, buffer, 4, &bytes_read) == FR_OK && bytes_read == 4)
{
word = (buffer[0]) |
  (buffer[1] << 8) |
  (buffer[2] << 16) |
  (buffer[3] << 24);

flash_write_to_addr(flash_address, &word, 1); // Write 1 word (4 bytes)
flash_address += 4;
}

/* Close file and clean up */
f_close(file);
free(fat_fs);
free(file);

/* Jump to new firmware */
jump_to_app(MAIN_APP_START_ADDRESS);

return 0;
}
```

```
void jump_to_app(uint32_t addr_value)
{
uint32_t app_start_address;
func_ptr jump_to_app;

/* Reset Systick */
SysTick->CTRL = 0;
SysTick->LOAD = 0;
SysTick->VAL = 0;

if (*(uint32_t *)addr_value != EMPTY_MEM)
{
printf("DBG: Starting Application \n\r");

app_start_address = *(uint32_t *)(addr_value + 4);
jump_to_app = (func_ptr)app_start_address;
jump_to_app();
}
else
{
printf("DBG: No application found at location \n\r");
}
}
```

6.2 main.c κυρίως κώδικα

```
#include <stdio.h>

#include "stm32f4xx.h"

#include "fpu.h"

#include "debug_uart.h"

#include "timebase.h"

#include "string.h"

#include "ff.h"

#include "bsp.h"

#include "sd_card_driver.h"

#include "fatfs_app.h"

#include "sd_card_spi.h"

#include "flash_driver.h"

#include "dht22.h"

#include "gps_uart.h"

#include "gps.h"

#include "ds3231.h"

#include <stdbool.h>


/* GPS module */

extern uint8_t rxGPS[RX_GPS_BUFFER_MAX_LENGTH];        //!< buffer needed for parsing the receiving data of queue

extern GPS_t GPS;


uint8_t rxGPS_2[RX_GPS_BUFFER_MAX_LENGTH];

uint8_t minute = 0;


/* FATFS */

FATFS fat_fs;
```

```
FIL file;

#define APP_START_ADDRESS  0x08010000
#define NUMBER_OF_ALARM_BYTES 48

typedef struct
{
/* data */
uint8_t hour;
uint8_t minute;
uint8_t sec;
} relayConfig;

typedef enum
{
RELAY_OFF,
RELAY_ON
} relayStatus;

relayConfig relayOn = {0};
relayConfig relayOff = {0};

void SystemInit(void)
{
SCB->VTOR = APP_START_ADDRESS;
}

uint8_t CalculateDayOfWeek(uint8_t date, uint8_t month, uint16_t year)
```

```
{
   if (month < 3)
{
      month += 12;
      year -= 1;
   }

   uint8_t k = year % 100;    // Year of the century
   uint8_t j = year / 100;    // Zero-based century

   // Zeller's formula to calculate the day of the week
   uint8_t dow = (date + 13*(month + 1)/5 + k + k/4 + j/4 + 5*j) % 7;

   // Adjust result to be 1 = Sunday, 2 = Monday, ..., 7 = Saturday
   return ((dow + 5) % 7) + 1;
}

bool getTimeFromGps(void)
{
// true for now
bool found = false;

while(found == false)
{
if (gps_has_pending_data())
{
NVIC_DisableIRQ(USART1_IRQn); // Disable USART1 interrupt
gps_set_data_received();
```

```
memcpy(rxGPS_2, rxGPS, RX_GPS_BUFFER_MAX_LENGTH);
memset(rxGPS, 0, 128*sizeof(rxGPS[0]));
NVIC_EnableIRQ(USART1_IRQn); // Re-enable USART1 interrupt

found = GpsParse((char*)&rxGPS_2, NMEA_GPRMC);
memset(rxGPS_2, 0, 128*sizeof(rxGPS[0]));
}
}
return found;
}

void fatfs_mount(void)
{
   /* Mount */
   FRESULT result =  f_mount(&fat_fs, "", 1);
   if(result != FR_OK )
   {
   printf("ERR: SD Card Cannot be Mounted!\n\r");
   }
}

bool findAlarmTime(uint8_t* alarmInfo, uint8_t length)
{
bool findAlarmStatus = false;

findAlarmStatus = sscanf(alarmInfo, "%d:%d:%d,ON@\r\n%d:%d:%d,OFF@", &relayOn.hour, &relayOn.minute, &relayOn.sec, &relayOff.hour, &relayOff.minute, &relayOff.sec);
```

```
return findAlarmStatus;
}

uint8_t alarm_data[NUMBER_OF_ALARM_BYTES];

bool getTimeOfRelay(void)
{
UINT bytes_read = 0;
bool getTimeStatus = false;

   /* Open the firmware file from SD Card */
FRESULT result =  f_open(&file, "alarms.txt", FA_READ);

   if(result != FR_OK)
   {
     printf("ERR: Alarm File Not Found!\n\r");
   }
   else if (f_read(&file, alarm_data, NUMBER_OF_ALARM_BYTES, &bytes_read) == FR_OK)
{
getTimeStatus = findAlarmTime(alarm_data, NUMBER_OF_ALARM_BYTES);
}
else
{
/* do nothing */
}

return getTimeStatus;
}
```

```
void initAll(void)
{
/* Enable fpu */
fpu_enable();

/* Initialize debug UART */
debug_uart_init();

/* Initialize relay */
relay_init();

/* Initialize timebase */
timebase_init();

/* DHT22 init */
DHT22_Init();

/* Initialize SD */
sd_cs_pin_init();
sd_card_spi_init();

/* Initialize fatfs */
fatfs_init();
fatfs_mount();

/* Initialize RTC */
   DS3231_Init();
```

```
/* Initialize gps uart */
gps_uart_init();

/* Gps init */
GpsInit();

/* Empty the gps buffer */
memset(rxGPS_2, 0, RX_GPS_BUFFER_MAX_LENGTH * sizeof(rxGPS[0]));
}

bool isTimeForSensorsData(void)
{
bool isTime = false;
if (minute == DS3231_GetMinute())
{
isTime = true;

minute = (minute >= 60) ? 0 : (minute + 1);
}

return isTime;
}

bool StoreDataToFile(char* file, char* data, uint8_t length)
{
bool storeRes = false;
unsigned int bytesToWrite = length;
```

```
unsigned int bytesWritten;

FRESULT res;
FIL logFile;
res = f_open(&logFile, file, FA_OPEN_APPEND | FA_WRITE);

if(res != FR_OK)
{
printf("f_open() failed, res = %d\r\n", res);
}
else if (f_write(&logFile, data, bytesToWrite, &bytesWritten) != FR_OK)
{
printf("f_write() failed, res = %d\r\n", res);
}
else if (f_close(&logFile) != FR_OK)
{
printf("f_close() failed, res = %d\r\n", res);
}
else
{
storeRes = true;
}

return storeRes;
}

bool isTimeForRelay(relayStatus* relayStatus)
{
```

```
uint8_t hour, minute, sec;


bool isTimeStatus = true;


hour = DS3231_GetHour();
minute = DS3231_GetMinute();
sec = DS3231_GetSecond();


if ((hour == relayOn.hour) && (minute == relayOn.minute) && (sec == relayOn.sec))
{
*relayStatus = RELAY_ON;
}
else if ((hour == relayOff.hour) && (minute == relayOff.minute) && (sec == relayOff.sec))
{
*relayStatus = RELAY_OFF;
}
else
{
isTimeStatus = false;
}


return isTimeStatus;
}


int main()
{
initAll();
```

```
if (getTimeFromGps())
{
DS3231_SetFullTime(GPS.hour, GPS.minute, GPS.sec);
minute = GPS.minute;
uint8_t dow = CalculateDayOfWeek(GPS.day, GPS.month, GPS.year);
DS3231_SetFullDate(GPS.day, GPS.month, dow, GPS.year);
}
else
{
printf("time from gps failure");
}

if(getTimeOfRelay())
{

}
else
{
printf("time of relay failure");
}

while(1)
{
float temp = 0;
float humidity = 0;
relayStatus relayState = RELAY_OFF;

if (isTimeForSensorsData())
```

```
{

DHT22_GetTemp_Humidity(&temp, &humidity);

char writeBuff[128];

   memset(writeBuff, 0, 128 * sizeof(writeBuff[0]));

sprintf(writeBuff, "year:%d month:%d date:%d dow:%d hour:%d minute:%d sec:%d temp:%.2f humidity:%.2f\n", DS3231_GetYear(), DS3231_GetMonth(), DS3231_GetDate(), DS3231_GetDayOfWeek(), DS3231_GetHour(), DS3231_GetMinute(), DS3231_GetSecond(), temp, humidity);

printf("%s", writeBuff);

StoreDataToFile("log.txt", writeBuff, strlen(writeBuff));

}



if (isTimeForRelay(&relayState))

{

if (relayState == RELAY_ON)

{

relay_on();

}

else

{

relay_off();

}

}

}

}
```

7 Βιβλιογραφία

1. https://elm-chan.org/docs/fat_e.html
2. https://embetronicx.com/tutorials/microcontrollers/stm32/bootloader/bootloader-basics
3. https://www.labcenter.com/blog/sim-bootloaders
4. https://en.wikipedia.org/wiki/Relay
5. https://www.circuitbasics.com/basics-of-the-spi-communication-protocol
6. https://medium.com/@aliaksandr.kavalchuk/chapter-3-the-1-wire-protocol-is-now-published-4d44a35cbfc9
7. https://www.allaboutcircuits.com/technical-articles/low-pin-count-serial-communication-introduction-to-the-1-wire-bus
8. https://en.wikipedia.org/wiki/SD_card
9. https://el.wikipedia.org/wiki/Global_Positioning_System
10. https://www.analog.com/en/resources/analog-dialogue/articles/uart-a-hardware-communication-protocol.html