Εργαστηριακή Άσκηση 3: Σύστημα Παρακολούθησης Θερμοκρασίας Υγρασίας

Βογιατζής Χαρίσιος
AEM:9192

June 6, 2025

Github Source Code (Lab 3)

1 Εισαγωγή

Ο στόχος της τρίτης εργαστηριακής άσκησης είναι η ανάπτυξη ενός ενσωματωμένου συστήματος IoT για την παρακολούθηση της θερμοκρασίας και της σχετικής υγρασίας περιβάλλοντος. Το σύστημα χρησιμοποιεί έναν αισθητήρα DHT11, έναν αισθητήρα αφής touch, παρέχει μηχανισμό σύνδεσης χρήστη μέσω κωδικού πρόσβασης και AEM, και προσφέρει οπτική ανάδραση μέσω LED καθώς και καταγραφή πληροφοριών μέσω σειριακής επικοινωνίας (UART).

2 Υλοποίηση

2.1 Διακοπές (Interrupts)

Το σύστημα αξιοποιεί πολλαπλές πηγές διακοπών για την αποδοτική διαχείριση των λειτουργιών του:

- **Timer Interrupt**: Μια περιοδική διακοπή (κάθε 1ms, μέσω της `timer_1ms_callback`) χρησιμοποιείται για τη διατήρηση ενός μετρητή χρόνου συστήματος (`system_ms_counter`). Αυτός ο μετρητής είναι κρίσιμος για τον χρονισμό των αναγνώσεων του αισθητήρα, τη διαχείριση των χρονικών ορίων για τις ειδοποιήσεις και τον έλεγχο του ρυθμού του LED.
- **UART RX Interrupt (`uart_rx_isr`)**: Ενεργοποιείται κατά τη λήψη δεδομένων μέσω της σειριακής θύρας UART. Οι εισερχόμενοι χαρακτήρες τοποθετούνται σε μια ουρά (`rx_queue`) για ασύγχρονη επεξεργασία από την κύρια λογική του προγράμματος, επιτρέποντας την εισαγωγή εντολών από τον χρήστη χωρίς να μπλοκάρει την εκτέλεση.
- **Touch Sensor Interrupt (`touch_sensor_isr`)**: Μια εξωτερική διακοπή (EXTI) συνδεδεμένη με τον αισθητήρα αφής. Όταν ο αισθητήρας πιεστεί, η αντίστοιχη ISR θέτει μια σημαία (`g_touch_sensor_pressed_flag`) για να ενημερώσει την κύρια λογική, επιτρέποντας την αλληλεπίδραση του χρήστη (π.χ. για την επαναφορά από κατάσταση ειδοποίησης).

2.2 Βασικές Δομές Δεδομένων

Η κατάσταση και η λειτουργία του συστήματος οργανώνονται γύρω από τις ακόλουθες δομές δεδομένων:

- `SystemState_t`: Μια κεντρική δομή (struct) που περιέχει όλες τις σημαντικές μεταβλητές κατάστασης του συστήματος. Αυτές περιλαμβάνουν:
  - Στοιχεία σύνδεσης (AEM, κατάσταση σύνδεσης).
  - Τρέχουσα κατάσταση λειτουργίας (`current_mode`: `MODE_A_NORMAL` ή `MODE_B_ALERT`).
  - Προτίμηση εμφάνισης δεδομένων (`display_preference`).
  - Διάστημα ανάγνωσης αισθητήρα (`read_interval_ms`).

  - Τελευταίες μετρήσεις από τον DHT11 και την κατάσταση της ανάγνωσης (`last_dht_status`).
  - Μετρητές για συνεχόμενες μετρήσεις εκτός ορίων (για ειδοποιήσεις και πανικό).

- `SystemMode` (enum): Καθορίζει τις κύριες λειτουργικές καταστάσεις: `MODE_A_NORMAL` (κανονική λειτουργία) και `MODE_B_ALERT` (λειτουργία ειδοποίησης).
- `DisplayPreference` (enum): Επιτρέπει στον χρήστη να επιλέξει ποια δεδομένα θα εμφανίζονται (θερμοκρασία, υγρασία, ή και τα δύο).
- `DHT11_StatusTypeDef` (enum): Αναπαριστά την κατάσταση της τελευταίας προσπάθειας ανάγνωσης από τον αισθητήρα DHT11 (π.χ. `DHT11_OK`, `DHT11_ERR_CHECKSUM`).
- `Queue rx_queue`: Μια ουρά FIFO για την προσωρινή αποθήκευση των χαρακτήρων που λαμβάνονται από το UART.

2.3 Ροή Προγράμματος (Program Flow)

Η κύρια λογική του συστήματος εκτελείται εντός ενός ατέρμονα βρόχου στην συνάρτηση `main()`:

1. **Αρχικοποίηση** (`init_system_state`) : Κατά την εκκίνηση, αρχικοποιούνται τα περιφερειακά (GPIOs για LED και αισθητήρα αφής, UART, Timer), οι διακοπές, και η καθολική δομή κατάστασης `g_system_state`.
2. **Σύνδεση Χρήστη** (`handle_login_sequence`) : Πριν την έναρξη της κύριας λειτουργίας, το σύστημα απαιτεί από τον χρήστη να εισάγει έναν έγκυρο κωδικό πρόσβασης. Η πρόσβαση επιτρέπεται μόνο μετά από επιτυχή ταυτοποίηση.
3. **Κύριος Βρόχος (Main Loop)**: Μετά τη σύνδεση, ο κύριος βρόχος διαχειρίζεται τις ακόλουθες λειτουργίες:
   - **Επεξεργασία Εντολών UART**: Ελέγχει την ουρά UART για νέες εντολές από τον χρήστη. Ένα μενού επιτρέπει την αλλαγή του διαστήματος ανάγνωσης, την προτίμηση εμφάνισης, και την αποσύνδεση.
   - **Ανάγνωση Αισθητήρα** (`DHT11_Read`) : Περιοδικά, με βάση το `read_interval_ms`, διαβάζει δεδομένα θερμοκρασίας και υγρασίας από τον DHT11. Υπάρχει η δυνατότητα προσομοίωσης των αναγνώσεων (`NO_HW_TESTING`) για ανάπτυξη χωρίς το φυσικό αισθητήρα.
   - **Έλεγχος Ορίων και Ειδοποιήσεων**: Οι μετρήσεις συγκρίνονται με προκαθορισμένα όρια (`TEMP_ALERT_THRESHOLD`, `HUMIDITY_ALERT_THRESHOLD`, `TEMP_PANIC_THRESHOLD`, `HUMIDITY_PANIC_THRESHOL`
   - **Διαχείριση Καταστάσεων Λειτουργίας**:
     - `MODE_A_NORMAL`: Εάν οι μετρήσεις είναι εντός φυσιολογικών ορίων, το σύστημα παραμένει σε αυτή την κατάσταση, εμφανίζοντας τα δεδομένα.
     - `MODE_B_ALERT`: Εάν οι μετρήσεις υπερβούν τα όρια ειδοποίησης ή πανικού για συγκεκριμένο αριθμό συνεχόμενων αναγνώσεων, το σύστημα μεταβαίνει σε κατάσταση ειδοποίησης. Σε αυτή την κατάσταση, το LED αναβοσβήνει. Για την επαναφορά στην κανονική λειτουργία, απαιτούνται είτε συγκεκριμένος αριθμός συνεχόμενων φυσιολογικών μετρήσεων είτε αλληλεπίδραση μέσω του αισθητήρα αφής.
   - **Διαχείριση Αισθητήρα Αφής**: Εάν ανιχνευθεί πάτημα του αισθητήρα αφής (μέσω της σημαίας `g_touch_sensor_pressed_flag`), το σύστημα μπορεί να εκτελέσει συγκεκριμένες ενέργειες, όπως η προσπάθεια επαναφοράς από την κατάσταση `MODE_B_ALERT`.
   - **Καταγραφή UART**: Οι τρέχουσες μετρήσεις, η κατάσταση του συστήματος και τυχόν ειδοποιήσεις καταγράφονται στη σειριακή θύρα.

2.4 Επικοινωνία με Αισθητήρα DHT11

Ο αισθητήρας θερμοκρασίας και υγρασίας DHT11 χρησιμοποιεί ένα πρωτόκολλο επικοινωνίας ενός σύρματος (1-wire) για την ανταλλαγή δεδομένων με τον μικροελεγκτή. Για τη διαδικασία επικοινωνίας δημιουργήθηκε ένας driver και υλοποιήθηκε στη συνάρτηση `DHT11_Read` στο αρχείο `main.c`. Ακολουθεί συγκεκριμένα βήματα και χρονισμούς:

1. **Σήμα Έναρξης από τον Μικροελεγκτή (MCU):** Ο μικροελεγκτής ξεκινά την επικοινωνία τραβώντας τη γραμμή δεδομένων σε χαμηλή λογική στάθμη (LOW) για τουλάχιστον 18ms. Στη συνέχεια, η γραμμή τίθεται σε υψηλή λογική στάθμη (HIGH) για 20-40μs, σηματοδοτώντας στον DHT11 ότι ο MCU είναι έτοιμος να λάβει δεδομένα. Κατά τη διάρκεια αυτών των βημάτων, η ακίδα δεδομένων του DHT11 (`DHT11_PIN`) ρυθμίζεται αρχικά ως έξοδος (`GPIO_MODE_OUTPUT_PP`) και μετά ως είσοδος με pull-up (`GPIO_MODE_INPUT_PULLUP`) για να μπορέσει ο αισθητήρας να οδηγήσει τη γραμμή.

2. **Απόκριση Αισθητήρα DHT11:** Μετά το σήμα έναρξης από τον MCU, ο DHT11 αποκρίνεται τραβώντας τη γραμμή δεδομένων LOW για περίπου 80μs, ακολουθούμενη από μια περίοδο HIGH για άλλα 80μs. Αυτή η ακολουθία επιβεβαιώνει ότι ο αισθητήρας είναι ενεργός και έτοιμος να στείλει δεδομένα. Η συνάρτηση `DHT11_Read` περιμένει αυτές τις μεταβάσεις στάθμης, με χρονικά όρια (`DHT11_TIMEOUT_US`) για την ανίχνευση σφαλμάτων επικοινωνίας.

3. **Μετάδοση Δεδομένων:** Ο DHT11 στέλνει συνολικά 40 bits (5 bytes) δεδομένων. Αυτά τα bytes αντιστοιχούν σε: ακέραιο μέρος σχετικής υγρασίας, δεκαδικό μέρος σχετικής υγρασίας, ακέραιο μέρος θερμοκρασίας, δεκαδικό μέρος θερμοκρασίας, και ένα άθροισμα ελέγχου (checksum). Κάθε bit δεδομένων ξεκινά με έναν παλμό LOW διάρκειας 50μs. Η τιμή του bit (0 ή 1) καθορίζεται από τη διάρκεια του επόμενου παλμού HIGH:

   - Ένας παλμός HIGH διάρκειας 26-28μs αντιπροσωπεύει ένα λογικό '0'.
   - Ένας παλμός HIGH διάρκειας 70μs αντιπροσωπεύει ένα λογικό '1'.

   Η συνάρτηση `dht11_read_byte_raw`, που καλείται από την `DHT11_Read`, είναι υπεύθυνη για την ανάγνωση ενός byte κάθε φορά, ανιχνεύοντας αυτές τις διάρκειες παλμών για κάθε bit.

4. **Άθροισμα Ελέγχου (Checksum):** Το πέμπτο byte που αποστέλλεται από τον DHT11 είναι το checksum, το οποίο ισούται με το άθροισμα των τεσσάρων προηγούμενων bytes (ακέραιο RH + δεκαδικό RH + ακέραιο Temp + δεκαδικό Temp), λαμβάνοντας υπόψη μόνο τα 8 λιγότερο σημαντικά bits του αθροίσματος. Η συνάρτηση `DHT11_Read` επαληθεύει αυτό το checksum. Εάν το υπολογιζόμενο checksum δεν ταιριάζει με το λαμβανόμενο, επιστρέφεται σφάλμα (`DHT11_ERR_CHECKSUM`).

5. **Απενεργοποίηση Διακοπών:** Για να διασφαλιστεί η ακρίβεια των χρονισμών που είναι κρίσιμοι για το πρωτόκολλο του DHT11, οι διακοπές απενεργοποιούνται καθολικά (`irq_disable()`) πριν από την έναρξη της επικοινωνίας και ενεργοποιούνται ξανά (`irq_enable()`) μετά την ολοκλήρωση της ανάγνωσης των δεδομένων. Αυτό αποτρέπει την παρεμβολή των ISRs στη διαδικασία μέτρησης των παλμών.

Η επιτυχής ανάγνωση επιστρέφει `DHT11_SUCCESS`, ενώ διάφορα σφάλματα (π.χ., timeout, checksum error) υποδεικνύονται με αρνητικές τιμές.

2.5 Διαχείριση Αισθητήρα Αφής

Ο αισθητήρας αφής στο σύστημα χρησιμοποιείται για την επαναφορά από την κατάσταση ειδοποίησης (Alert State) στην κανονική λειτουργία (Normal State). Η διαχείρισή του βασίζεται σε εξωτερικές διακοπές:

1. **Ρύθμιση Υλικού:** Η ακίδα GPIO στην οποία είναι συνδεδεμένος ο αισθητήρας αφής (`TOUCH_SENSOR_PIN`) ρυθμίζεται ως είσοδος (`GPIO_MODE_INPUT`). Επιπλέον, ενεργοποιείται μια εξωτερική διακοπή (`EXTI`) σε αυτή την ακίδα, συνήθως ρυθμισμένη να ενεργοποιείται στην ανιούσα ακμή (rising edge) ή στην κατιούσα ακμή (falling edge), ανάλογα με τη λογική του αισθητήρα. Στην υλοποίηση, η `touch_sensor_init` ρυθμίζει την ακίδα και ενεργοποιεί την `EXTI` καλώντας την `exti_setup`.

2. **Ρουτίνα Εξυπηρέτησης Διακοπής (ISR):** Όταν ο αισθητήρας αφής ενεργοποιείται (πατιέται), η αντίστοιχη γραμμή EXTI προκαλεί μια διακοπή. Η εκτέλεση μεταφέρεται στην αντίστοιχη ρουτίνα εξυπηρέτησης διακοπής, την `touch_sensor_isr`. Αυτή η ISR είναι σχεδιασμένη να είναι πολύ σύντομη και γρήγορη. Η κύρια της λειτουργία είναι να θέσει μια καθολική, πτητική (volatile) σημαία, την `g_touch_sensor_pressed_flag`, σε `true`. Η χρήση μιας `volatile` μεταβλητής είναι σημαντική για να εξασφαλιστεί ότι ο compiler δεν θα προβεί σε βελτιστοποιήσεις που θα μπορούσαν να αγνοήσουν τις αλλαγές που γίνονται από την ISR.

3. **Επεξεργασία στην Κύρια Λογική (Main Loop):** Ο κύριος βρόχος του προγράμματος (μέσα στη συνάρτηση `main`) ελέγχει περιοδικά την κατάσταση της σημαίας `g_touch_sensor_pressed_flag`. Εάν η σημαία είναι `true`, αυτό υποδεικνύει ότι ο αισθητήρας αφής έχει πατηθεί. Η κύρια λογική τότε:
    - Εκτελεί τις απαραίτητες ενέργειες, όπως η μετάβαση από την κατάσταση `ALERT_STATE` στην `NORMAL_STATE`.
    - Επαναφέρει τη σημαία `g_touch_sensor_pressed_flag` σε `false`, ώστε να είναι έτοιμη για την ανίχνευση του επόμενου πατήματος.

Αυτός ο μηχανισμός επιτρέπει στο σύστημα να ανταποκρίνεται άμεσα σε ένα άγγιγμα χωρίς να επιβαρύνει υπερβολικά την ISR.

3 Προκλήσεις Λύσεις

Κατά την ανάπτυξη του συστήματος, προέκυψαν ορισμένες προκλήσεις:

- **Αξιόπιστη Ανάγνωση Αισθητήρα**: Η επικοινωνία με αισθητήρες όπως ο DHT11 απαιτεί ακριβή χρονισμό. Λόγω της software υλοποίησης του delay driver που είχαμε διαθέσιμο και χρησιμοποιήσαμε, η πραγματική χρονική καθυστέρηση ήταν αρκετά μεγαλύτερη από την αναμενόμενη, με σκοπό να χρειαστεί να αναπροσαρμώσουμε πολλές φορές τις τιμές των delay για την υλοποίηση της επικοινωνίας, κάτι το οποίο ήταν πολύ χρονοβόρο.
- **Διαχείριση Διακοπών**: Ο συντονισμός πολλαπλών πηγών διακοπών (timer, UART, EXTI) απαιτεί προσοχή για την αποφυγή race conditions και την εξασφάλιση της απόκρισης του συστήματος.

4 Testing Παραδοχές

Ο κώδικας έχει δοκιμαστεί επί μέρους επιτυχώς αλλά όχι εξ ολοκλήρου ως ολοκληρωμένο σύστημα. Αυτό σημαίνει ότι οι επί μέρους λειτουργίες (επικοινωνία και ανάγνωση μετρήσεων θερμοκρασίας και υγρασίας, ανάγνωση πίεσης touch, ενεργοποίση LED κλπ) έχουν δοκιμαστεί επιτυχώς, αλλά ενδέχεται να υπάρχουν bugs τα οποία να εμφανίζονται στη συνολική λειτουργία του project.

5 Source Code

Ο πηγαίος κώδικας για την Εργαστηριακή Άσκηση 3 είναι διαθέσιμος στο Github: Github Source Code (Lab 3)