Ceas de birou

Introducere

Ceas-ul de birou este exact ceea ce spune că este. Un ceas de birou, mobil, care îți arată ora, temperatura și va avea funcții de setare de alarme și temporizator. Am vrut să dezvolt acest proiect deoarece voiam să am un astfel de ceas și proiectul la PM este ocazia de a-l crea și customiza așa cum doresc.

Descriere generală

În modul default, ceasul va afișa pe ecran ora și temperatura. Ulterior, prin interacțiunea cu butoanele (și vizualizarea acțiunilor pe LCD), vor putea fi setate fie un temporizator, fie se poate intra în modul de management al alarmelor (aici se pot vizualiza, edita, șterge sau adăuga alarme). Totodată user-ul va putea edita ora (incrementând / decrementând numărul orei, pentru a seta ora când “se dă ceasul”, timpul exact fiind reținut de un modul specializat).

Hardware Design

Se vor folosi următoarele piese:

- Placă dezvoltare Arduino Nano
- Modul LCD 16×2 cu modul de interfațare I2C
- Modul DS1302
- Buzzer
- Senzor de temperatură și umiditate DHT11
- Power bank pentru alimentare
- 4 butoane

Schema Hardware:

Legături între componente:

- LCD-ul este conectat la pinii de I2C ai Arduino Nano
- Buzzer-ul se conencteaza la pinul D9
- Senzorul de temperatură are linia de date conectată la pinul D10

- Modulul RTC foloseste 3 legături: RST - D2, CLK -D4 și DAT - D3
- Butoanele sunt legate la A[0..3] și folosesc un condensator pentru debouncing și rezistențe de pull-down

Software Design

Pentru dezvoltare s-a folosit PlatformIO. Au fost implementate toate functionalitatile descrise anterior, si anume un manager de alarme (cu 5 alarme disponibile), temporizator, posibilitatea de a configura timpul.

Managerul de alarme foloseste un spatiu de 5 alarme disponibile, salvand orice modificare a acestor alarme pe memoria EEPROM (pentru a asigura persistenta si in timpul deconectarii de la alimentare). Managerul de alarme va interoga, la orice modificare sau la boot-up, cele 5 alarme si va retine timpul ramas pana la trigger (in secunde). Se foloseste un timer pentru generarea de intreruperi la fiecare secunda (detalii mai jos). Cand timpul trece, este activat buzzerul pentru 6 secunde, urmand a se determina cea mai apropiata alarma.

Temporizatorul primeste ca input timpul dorit, urmand ca dupa ce a fost setat sa contorizeze secundele ramase pana la trigger (foloseste acelasi timer pentru contorizarea secundelor). Si acest “modul” foloseste buzzerul cand timpul a expirat.

Configurarea timpului asteapta inputul de la utilizator, urmand sa comunice cu modulul RTC pentru setarea timpului.

Biblioteci folosite: biblioteca pentru LCD I2C si pentru modululele RTC si de temperatura. Motivatie: nu a fost necesar sa reinventez roata.

Optimizari: Am dorit ca acest ceas de birou sa fie cat mai mobil si sa nu depinda de o sursa de alimentare. Astfel am incercat sa diminuez consumul pe cat posibil. O prima decizie pe care am luat-o a fost sa reduc frecventa Nano-ului de la 16 MHz la 8 MHz (prin setarea divizorului de clock al atmega328p la 2). O valoare mai mica de 8MHz rezulta in functionarea anormala a comunicarii cu perifericele. O a doua decizie luata a fost folosirea modului de POWER_DOWN pentru consum redus. Am luat aceasta decizie deoarece ceasul meu nu are nevoie sa faca lucruri in continuu, ci la anumite intervale de timp (spre exemplu, am setat ca ora sa fie actualizata o data la 10 secunde, iar temperatura o data la 1 minut). Astfel, folosesc timerul watchdog pentru a trezi microcontrolerul o data la o secunda. Totodata, trezirea din POWER_DOWN mode se poate face si prin Pin Change Interrupts, ceea ce se preteaza foarte bine pe proiect deoarece pot trezi microcontrolerul si la apasarea butoanelor. Totusi, in urma masuratorilor facute de mine, marele consumator de energie este LCD-ul, si mai exact back-light-ul acestuia, motiv pentru care am folosit si un potentiometru pentru reglarea backlight-ului. Consumul mediu inregistrat cu backlight-ul maxim a fost de 23 mA, fara a lua in considerare timpul cand buzzerul este activat (cand buzzerul este activ, consumul creste pana pe la 70 - 80 mA, dar asta se intampla foarte rar, cateva secunde pe zi).

Din cadrul laboratorului, am folosit urmatoarele functionalitati:

- GPIO (buzzer)
- Timere (WDT si pentru buzzer am implementat de mana functionalitatie de tone si noTone din biblioteca Arduino)
- Intreruperi (WDT si butoane)

- UART (debug)
- I2C (nu am folosit eu de mana, ci biblioteca de interfatare a modulului I2C)

Demo video

Rezultate Obţinute

Care au fost rezultatele obţinute în urma realizării proiectului vostru.

Concluzii

Download

O arhivă (sau mai multe dacă este cazul) cu fişierele obţinute în urma realizării proiectului: surse, scheme, etc. Un fişier README, un ChangeLog, un script de compilare şi copiere automată pe uC crează întotdeauna o impresie bună 😉.

Fişierele se încarcă pe wiki folosind facilitatea **Add Images or other files**. Namespace-ul în care se încarcă fişierele este de tipul **:pm:prj20??:c?** sau **:pm:prj20??:c?:nume_student** (dacă este cazul). **Exemplu:** Dumitru Alin, 331CC → **:pm:prj2009:cc:dumitru_alin**.

Jurnal

Puteți avea și o secțiune de jurnal în care să poată urmări asistentul de proiect progresul proiectului.

Bibliografie/Resurse

Listă cu documente, datasheet-uri, resurse Internet folosite, eventual grupate pe **Resurse Software** şi **Resurse Hardware**.

Export to PDF