Deadlines Alarm Clock - Culea Cosmin - 331CA

Introducere

Deadlines Alarm Clock, dupa cum sugereaza si numele este un sistem de stocare si de atentionare a studentilor asupra deadline-urilor pe care acestia le au la facultate. Cu totii stim cat de incarcate pot fi anumite perioade din facultate (sa nu uitam de anul 2 semestrul 2), iar de cele mai multe ori te gasesti in mijlocul semestrului cu 3-4-5 teme/proiecte suprapuse carora nu le mai poti face fata, nu stii cat timp mai ai pentru fiecare, trebuie sa intrebi mereu prin stanga si prin dreapta: “CAND ERA DEADLINE-UL LA PA???”, astfel iritandu-i si mai mult pe colegii tai. Cateodata esti atat de bulversat incat nici nu mai stii ce zi a saptamanii este (mai ales daca au trecut 4 zile si 4 nopti rezolvand tema la protocoale).

De aceea, propun acest sistem cu alarma care stocheaza pana la 6 deadline-uri (an/luna/zi/ora/minut/secunda) alaturi de un mesaj cu numele materiei, si te atentioneaza inainte cu 3, 2 si o zi inainte de deadline pentru a fi sigurx ca ai timp suficient sa-ti poti rezolva tema. In plus, ai acces la o listare a tuturor deadline-urilor pentru a-ti putea face o imagine de ansamblu asupra lor, poti face update pe un deadline sau il poti chiar sterge in momentul in care ai terminat proiectul/ti-ai dat seama ca ai puncte destule pe parcurs si nu mai are rost sa o faci.

Descriere generală

Sistemul preia date de la utilizator prin intermediul a 6 butoane NEXT, PREV, UP, DOWN, SELECT si MENU. Meniul are 2 optiuni de LIST si ADD prin care se pot lista toate deadline-urile stocate anterior si de adaugare de dealine-uri noi. In comanda LIST vor fi afisate pe rand alarmele cu cu data si ora la care sunt ele setate si se poate naviga printre ele prin intermediul butoanelor NEXT si PREV. In momentul cand este listat un anumit deadline se poate apasa buton SELECT si se vor afisa 2 optiuni de comenzi si anume UPDATE sau DELETE. Atat in cadrul comenzii de update si de add vor fi afisate pe rand anul, luna, ziua, ora, minutul, secunda care pot fi modificate incremental cu ajutorul butoanelor UP and DOWN, iar navigarea prin data se realizeaza tot prin NEXT si PREV.

Pe LCD va fi afisata constant data si ora curenta preluata de la modulul RTC, mai putin in momentele in care utilizatorul realizeaza o anumita comanda. Buzzerul va fi activat cu 3, 2, si o zi inainte de deadline, anuntand cate zile ramase mai sunt pana cand tema/proiectul la materia respectiva trebuie predata. De asemenea, sistemul dispune si de un senzor de miscare care va verifica daca este sau nu cineva in camera. In cazul in care seznorul nu detecteaza miscare in camera, alarma nu va suna la ora fixata, ci doar in momentul in care a simtit prezenta cuiva. Astfel, alarma ta nu va suna sa deranjeze colegii de apartament/vecinii cand tu nu esti acasa, insa te va anunta imediat ce ai ajuns ca ai pierdut o atentionare de deadline.

Schema bloc

Hardware Design

Piesele folosite în cadrul proiectului sunt:

- Arduino Uno R3 ATmega328P
- Breadboard
- Ecran LCD 1602
- Senzor de miscare PIR
- Buzzer
- RTC (Real Time Clock)
- modul sursa de alimentare 5V
- battery holder
- 6 baterii
- 6 butoane
- fire jumper si dupont

Schema Electrica

Software Design

Organizarea codului

Codul este organizat in mai multe fisiere .cpp, fiecare cu headerul aferent:

- **utils.cpp/utils.h** contin define-urile pentru maparea pinilor si multe altele ajutatoare, majoritatea structurilor de date folosite pentru stocarea deadline-urilor, numele materiilor, flaguri de intarziere pentru momentele cand senzorul nu detecteaza miscare, lcd, rtc, dar si array-uri constante de mapare numarul lunii → numele acesteia, numarul zilei din saptamana → numele zilei din saptamana, numarul lunii → numarul de zile a lunii respective

```
/* Face deep copy pe cele doua structuri (din source in destination) */
int copy_dt(Ds1302::DateTime *destination, Ds1302::DateTime *source);

/* Functie care invalideaza data primita, dandu-i valori invalide pentru
```

```
fiecare camp. */
int invalid_dt(Ds1302::DateTime *dt);

/* Functie care verifica daca data este invalida. */
int is_invalid_dt(Ds1302::DateTime *dt);

/* Aduna o zi la data primita. Functie folositoare la verificare alarmei
inainte cu 3, 2, o zi. */
void increment_day(Ds1302::DateTime &dt);

/* Verifica daca doua date sunt egale. */
int are_dates_equal(Ds1302::DateTime &dt1, Ds1302::DateTime &dt2);

/* Functie care primeste 2 date, si verifica dt1 == dt2 || dt1 + o zi == dt2
|| dt1 + 2 zile == dt2
 * || dt1 + 3 zile == dt si intoarce un define corespunzator in functie de
cum ar trebui sa sune
 * alarma sau nu (NOT_NOW, NOW, ONE_DAY, TWO_DAYS, THREE_DAYS).
 */
int is_deadline_due(Ds1302::DateTime &dt1, Ds1302::DateTime &dt2);
```

- **introduce_to_lcd.cpp/introduce_to_lcd.h** contin functii care permit interfatarea cu lcd-ul pentru introducerea fiecarui camp dintr-o structura de tip date si numele materiei. Parametrul next_op este folosit pentru stocarea urmatoarei operatii care trebuie realizata dupa operatia curenta in functie de ce buton a fost apasat, pentru a fi mai departe procesata de functia introduce_date_to_lcd. Valori posibile pentru next_op sunt NEXT_YEAR, NEXT_MONTH, NEXT_DAY, NEXT_HOUR, NEXT_MINUTES, NEXT_SECONDS, NEXT_SELECT.

```
void print_introduce_year(char *year, char *next_op);

void print_introduce_month(char *month, char *next_op);

void print_introduce_day(char month, char *day, char *next_op);

void print_introduce_hour(char *hour, char *next_op);

void print_introduce_minutes(char *minutes, char *next_op);

void print_introduce_seconds(char *seconds, char *next_op);

void introduce_subject_to_lcd(char* subject);

void introduce_date_to_lcd(Ds1302::DateTime &dt);
```

- **commands.cpp/command.h** contin functiile cu implementarea pentru afisare a meniului, dar si de interfatare a utilizatorului cu meniul prin intermediului lcd-ului (se foloseste de asemenea de functiile din introduce_to_lcd.h).

```
/* Afiseaza pe lcd data trimisa ca parametru cu toate campurile. */
void print_date(Ds1302::DateTime &dt);
```

```
/* Printeaza un deadline-ul care are indexul primit ca parametru in array-ul
dts */
void print_deadline(int index);

/* Functia intoarce urmatorul index al unei date valide din cadrul
array-ului dts dupa
 * indexul dat ca parametru
 */
int next_index(int index);

/* Functia intoarce indexul unei date valide din cadrul array-ului dts
anterior intexului
 * dat ca parametru
 */
int prev_index(int index);

/* Functie care face update sau sterge un deadline cu indexul primit ca
parametru, in functie
 * de butonul apasat de utilizator
 */
void update_or_delete(int *index);

/* Functie care listeaza deadline-urile pe rand, cu posibilitatea iterarii
prin acestea prin
 * apasarea butoanelor NEXT sau PREV. Se iese din functie in momentul
apasarii butonului MENU.
 */
void list();

/* Functie care adauga un nou deadline prin introducerea datei si a materiei
interfatate
 * prin butoane si lcd
 */
void add();

/* Functie care afiseaza meniul cu comenzile ADD si LIST si asteapta input
din partea utilizatorului.
 * Prin apasarea butonul NEXT se apeleaza functia list(), iar prin apasarea
butonului PREV se
 * apeleaza functia add(). Se iese din functie la apasarea butonului MENU.
 */
void display_menu();
```

- **song.cpp/song.h** contin functia de activare a alarmei. Contine de asemenea si 2 array-uri care contin frecventele si duratele notelor. Pentru generarea melodiei am downloadat fisierul .mid de aici si am convertit fisierul in cod arduino prin aplicatia asta. Valorile frecventelor si ale duratelor le-am extras cu un script creat de mine.

```
/* Functia activeaza alarma pentru deadline-ul dt_index, cu deadline_state
fiind NOW, ONE_DAY,
 * TWO_DAYS sau THREE_DAYS, pe pinul buzzerPin
```

```
 */
void play_alarm(int buzzerPin, int dt_index, int deadline_state);
```

Logica implementarii

Logica implementarii se afla in fisierul **deadlines_alarm.ino** care contin functiile **setup()**, **loop()** si includ toate fisierele header anterior mentionate.

- Functia de **setup()** initializeaza pinii butoanelor, al buzerului si senzorului. De asemenea, initializeaza structura specifica RTC-ului, si celelalte structuri pentru stocarea deadlinepurilor. De asemenea, codul comentat din cadrul functiei se poate decomenta pentru a reinitializa valorile de data si ora din modulul RTC, prin introducerea interactiva cu ajutorul LCD-ului.

```
void setup() {
  lcd.init();
  lcd.clear();
  lcd.backlight();

  pinMode(BUTTON_DOWN, INPUT_PULLUP);
  pinMode(BUTTON_UP, INPUT_PULLUP);
  pinMode(BUTTON_MENU, INPUT_PULLUP);
  pinMode(BUTTON_NEXT, INPUT_PULLUP);
  pinMode(BUTTON_PREV, INPUT_PULLUP);
  pinMode(BUTTON_SELECT, INPUT_PULLUP);
  pinMode(BUZZER, OUTPUT);
  pinMode(PIR, INPUT);

  rtc.init();
  /* // Code if the RTC fails to print correct date time
    Ds1302::DateTime dt;
    introduce_date_to_lcd(dt);
    rtc.setDateTime(&dt); */

  /* Initializeaza deadline-urilor si array-ul de delayed flag */
  for (int i = 0; i < 6; i++) {
    invalid_dt(&dts[i]);
    delayed[i] = NOT_NOW;
  }
}
```

- In cadrul functiei **loop()**, se va afisa de fiecare data data si ora curenta extrasa din RTC intr-un mod eficient (se face comparatia cu valorile datei si orei extrase anterior si se va modifica pe LCD doar campul care este diferit). Acest lucru este necesar pentru a putea creste vizibilitatea pe LCD, intrucat am observat ca daca afisez de fiecare data toate campurile, LCD-ul va avea un comportament ciudat. Mai apoi se itereaza prin toate deadline-urile stocate, iar daca sensorul detecteaza miscare si starea deadline-ului indica sunarea alarmei sau alarma a fost amanata, atunci se va apela functia corespunzatoare de sunare a alarmei. Daca sensorul insa nu detecteaza miscare si starea deadline-ului indica sunarea alarmei atunci starea sa va fi stocata in array-ul de flaguri. Dupa ce se proceseaza toate deadline-urile, se verifica si starea butonului de MENU si se apeleaza in

cazul in care este activata aceasta functionalitate functia diplay_menu care se ocupa mai departe de toate functionalitatile puse la dispozitie de commands.cpp si commands.h.

```
void loop() {
  Ds1302::DateTime dt;
  int deadline_due = 0;

  /* Printeaza data si ora curenta extrasa din RTC */
  rtc.getDateTime(&dt);
  print_date(dt);

  /* Pastreaza o copie globala a datei afisate anterior pentru
   * afisarea eficienta a datei la lcd
   */
  copy_dt(&prev_dt, &dt);

  /* Itereaza prin toate deadline-urile pentru a verifica daca
   * trebuie sunata alarma
   */
  for (int i = 0; i < 6; i++) {
    /* Extrage starea deadline-ului curent */
    int deadline_state = is_deadline_due(dt, dts[i]);

    /* Reface structura dt cu data curenta, aceasta fiind modificata
anterior in
     * functia is_deadline_due
     */
    copy_dt(&dt, &prev_dt);

    /* Citeste starea senzorului de miscare */
    int state_sensor = digitalRead(PIR);

    /* Daca senzorul detecteaza miscare */
    if (state_sensor == HIGH) {
          /* Daca starea deadline-ului indica faptul ca alarma trebuie
activata */
          if (deadline_state != NOT_NOW){
            play_alarm(BUZZER, i, deadline_state);
            deadline_due = 1;
          /* Daca alarma a fost amanata anterior, acum trebuie activata */
          } else if (delayed[i] != NOT_NOW) {
            play_alarm(BUZZER, i, delayed[i]);
            deadline_due = 1;
          }

          /* Daca este ultima alarma a deadline-ului, atunci deadline-ul
este sters*/
          if (deadline_state == NOW || delayed[i] == NOW) {
            invalid_dt(&dts[i]);
          }
```

```
          delayed[i] = NOT_NOW;
      /* Daca senzorul nu detecteaza miscare si starea deadline-ului este de
a fi activata
       * atunci se pastreaza in array-ul delayed pentru a putea fi sunata
ulterior
       */
    } else if (deadline_state != NOT_NOW){
        delayed[i] = deadline_state;
    }

  }

  /* Curata LCD-ul daca alarma a fost activata */
  if (deadline_due == 1) {
    lcd.clear();
    invalid_dt(&prev_dt);
  }

  /* Daca functionalitatea meniului este activata prin buton, atunci
   * se apeleaza functia display_menu
   */
  if (digitalRead(BUTTON_MENU) == LOW) {
    delay(200);
    display_menu();
    lcd.clear();
    invalid_dt(&prev_dt);
    delay(200);
  }
}
```

Biblioteci folosite

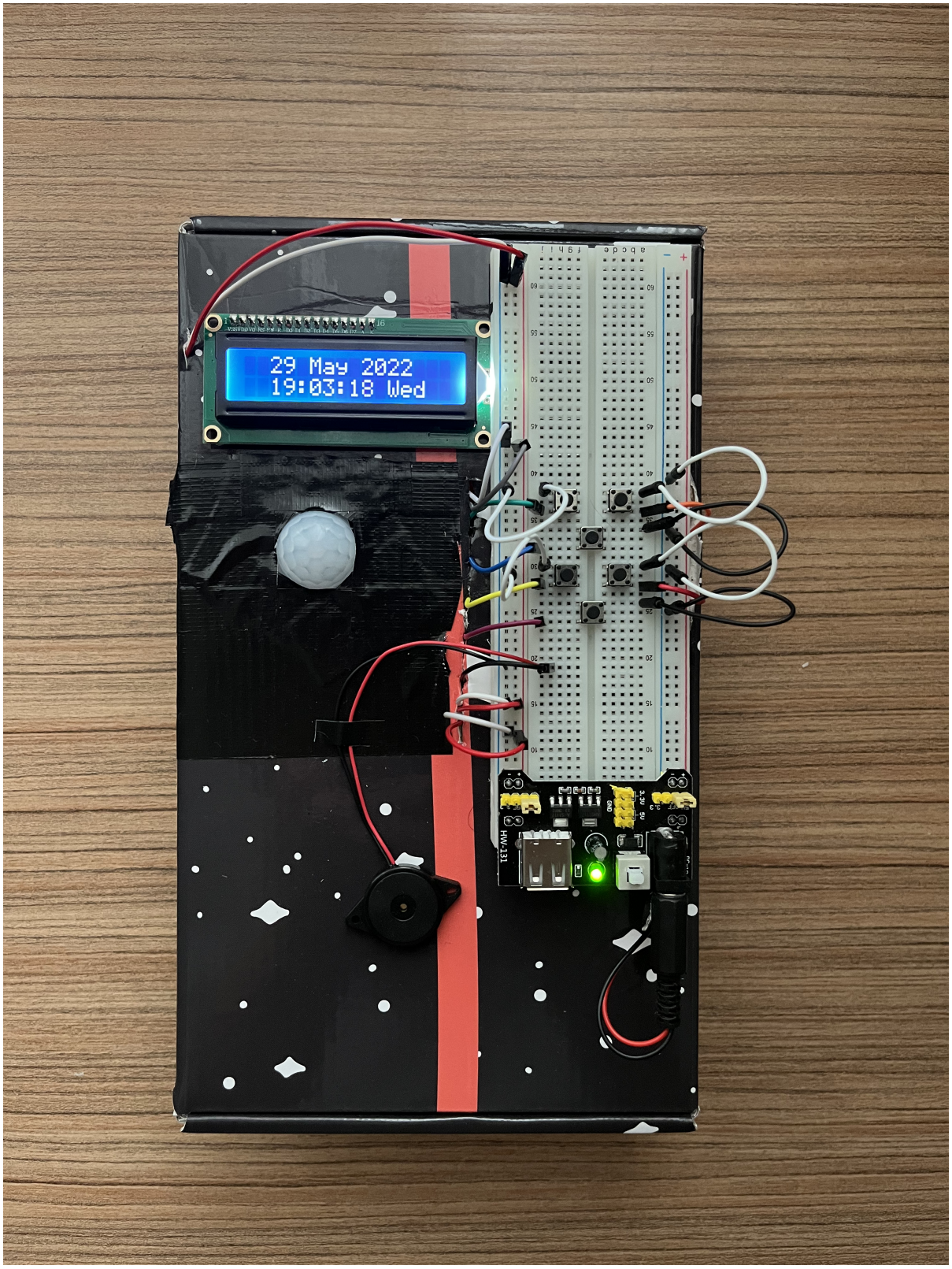
- **LiquidCrystal_I2C.h** pentru lucrul cu LCD-ul
- **Ds1302.h** pentru lucrul cu RTC-ul si pentru stocarea datelor de tip date

Mediul de dezvoltare

- Arduino IDE

Rezultate Obţinute

Proiectul final contine toate functionalitatile mentionate in descriere. Pentru mai multe detalii despre cum se poate folosi puteti accesa sectiunea Videos de aici.


29 May 2022
19:03:18 Wed
HW-131


WARNING! This is a Blackhole. Opening it may change everything about you.

Concluzii

Desi la inceput am fost foarte speriat de acest proiect, intrucat nu aveam experienta mai deloc sa lucrez cu componente hardware (in afara laboratoarelor de pm), nu stiam cat de mult timp ar lua sa il finalizez si nu stiam la ce probleme ar trebui sa ma astept, pot spune ca a fost o experienta foarte placuta. Totul a fost mai greu pana mi-au venit piesele, tocmai pentru ca nu stiam la ce sa ma astept. Apoi am incercat usor usor sa pun toate pisele cap la cap, sa le testez, sa caut tutoriale pe youtube si am realizat cat de multe resurse utile exista.

Implementarea software mi-a placut cel mai mult, dar a fost si destul de challenging, intrucat desi poate nu pare foarte complicata ideea proiectului in sine, a fost destul de dificil si extrem de mult cod de scris ca sa pot implementa toate functionalitatile asa cum mi le doream. Am avut parte de foarte multe buguri evident, mai ales la implementarea logicii butoanelor, pe care am reusit sa le rezolv pe parcurs prin testarea continua (+ printf-uri cat cuprinde pe interfata seriala).

Din punct de vedere hardware, spre bucuria mea, nu am intampinat foarte multe dificultati, asa cum am zis am gasit foarte multe materiale care m-au ajutat sa inteleg principiile de functionare ale componentelor. Singurul meu regret este ca nu mi-am cumparat un buzzer sau un difuzor mai performant, intrucat nu sunt destul de multumit de cum se audea alarma. Am incercat sa convertesc niste melodii, dar nu se aud asa cum m-as fi asteptat.

Download

Arhiva cu fisierele sursa .cpp .h si .ino se pot gasi aici.

Jurnal

- **10.05.2022** Alegere tema proiect
- **15.05.2022** Milestone 1: Introducere, Descriere, Schema Bloc și Componente
- **29.05.2022** Milestone 2: Schema electrică, Software Design, Rezultatele Obținute, Arhivă

Bibliografie/Resurse

- https://github.com/Treboada/Ds1302
- https://www.robofun.ro/componente/mini-difuzor-brick.html
- https://electropeak.com/learn/interfacing-ds1302-real-time-clock-rtc-module-with-arduino/
- https://www.youtube.com/watch?v=FxaTDvs34mM

Export to PDF