Daniel BĂLTEANU (78286) - Ceas cu alarma in patru timpi

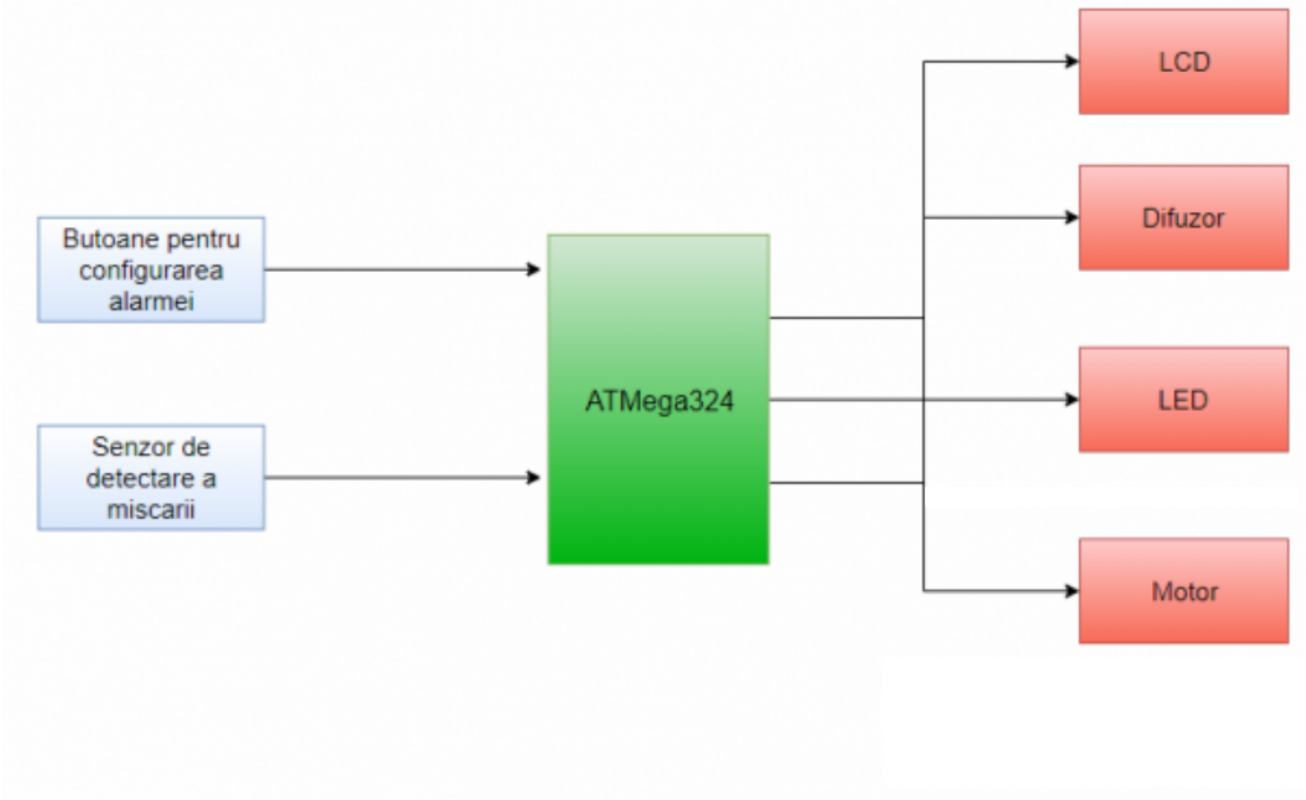
Autorul poate fi contactat la adresa: **Login pentru adresa**

Introducere

Ceasul cu alarma in patru timpi are ca scop principal inlaturarea riscului de a opri alarma si a adormi la loc. Desi somnul este bun, nimeni nu isi doreste sa intarzie la facultate, examene sau la munca. O alarma obisnuita are un simplu buton de oprire a acesteia, fapt ce o face foarte vulnerabila la dorinta puternica a oamenilor de a mai dormi cateva minute. In acest sens, ceasul cu alarma in patru timpi obliga utilizatorul sa se dea jos din pat pentru a opri alarma. Mai mult decat atat, aceasta alarma reduce necesitatea de a auzi un sunet neplacut care sa ne trezeasca. Astfel, cei patru timpi in care functioneaza alarma sunt: 1. Pornirea unui ventilator; 2. Aprinderea luminilor; 3. Pornirea unei melodii cu o intensitatea redusa; 4. Folosirea tuturor celor de mai sus in acelasi timp. In acest sens, in oricare dintre cei 4 timpi alarma poate fi oprita prin simpla coborare din pat. Scopul este de a face trezirea cat mai usoara si treptata. Cu toate acestea, pasul 4 inlatura riscul de a nu trezi utilizatorul.

Asadar, proiectul este util deoarece majoritatea oamenilor folosesc o alarma pentru a se trezi, iar aceasta alarma inlatura dependenta de telefon, care se poate descarca, face trezirea mai placuta, dar in acelasi timp te obliga sa te dai jos din pat.

Descriere generală



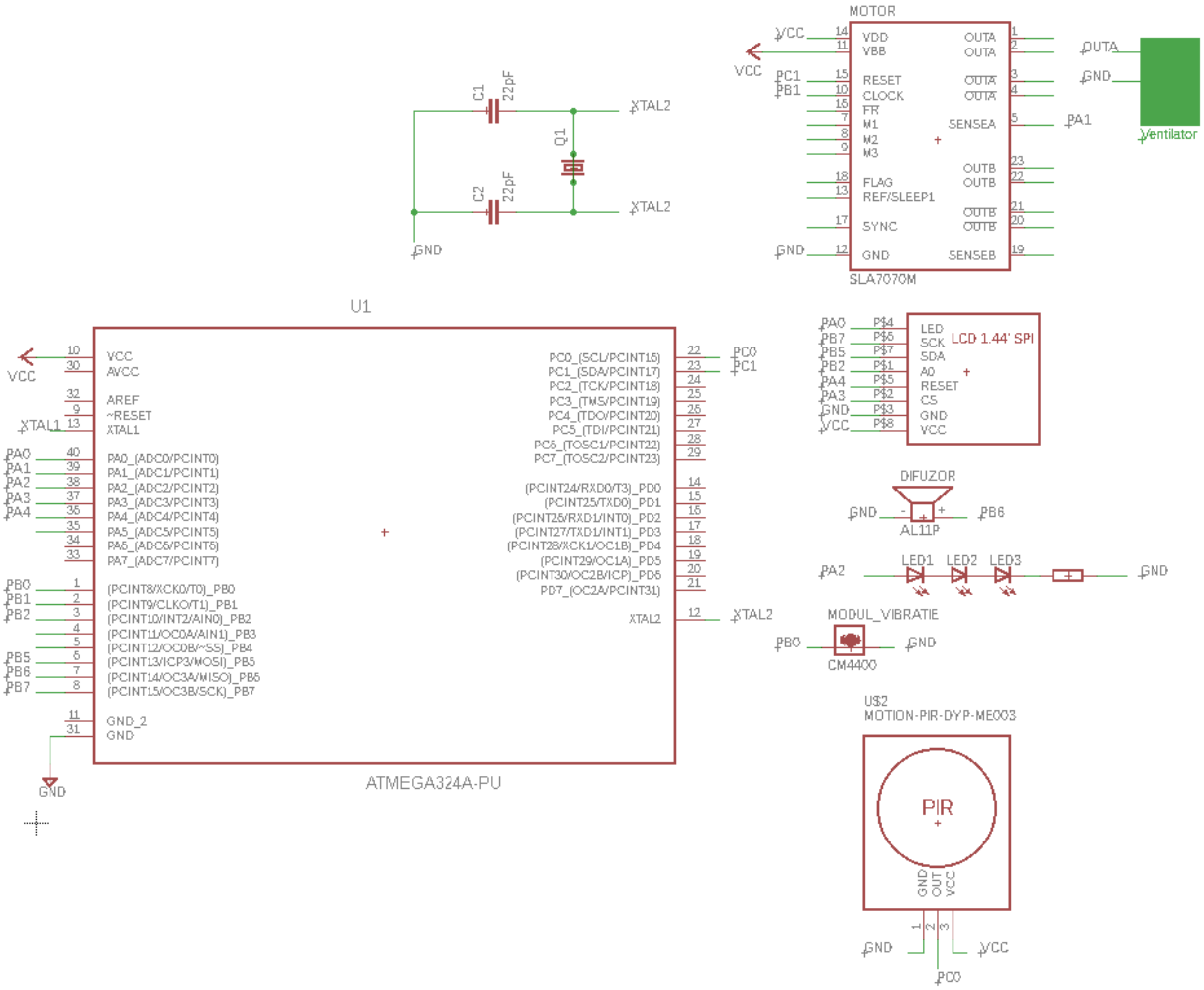
Interactiune modulelor este foarte simpla. Input-ul introdus de la butoane reprezinta alegerea orei la care sa sune alarma. De asemenea, se primeste si un semnal de la senzorul de miscare folosit pentru a opri alarma. In ceea ce priveste output-ul: 1. LCD-ul va afisa ora; 2. Difuzorul va reda sunetul de alarma; 3. Motorul va actiona un ventilator.

Dupa setarea alarmei, cand se ajunge la momentul alarmei, cele patru etape incep pe rand, cate 10 secunde, la final ramanand toate modulele active (ventilatorul, difuzorul si LED-ul) pana la oprire. In orice moment, alarma va putea fi oprita numai prin detectarea coborarii din pat.

Hardware Design

Nume piesa	Cantitate
Placa de baza	1
Senzor de miscare	1
Motor	1
LCD	1
Difuzor	1
Tranzistor	1
Dioda	1
LED	1
Butoane	4
Rezistente	
Fire legatura	

In continuare se afla schema electrica a proiectului (realizata cu ajutorul aplicatiei EAGLE).



Software Design

Pentru realizarea partii software a proiectului am folosit avr-gcc, precum si softul oferit in sectiunea Fisiere de pe site (Bootloader USB).

Descriere Software

In acest sens, realizarea softului poate fi impartite in mai multe sectiuni. Prima si cea mai usoara parte este adaugarea logicii pentru butoane. Astfel, am folosit 3 butoane pentru setarea alarmei: doua pentru alegerea orei si a minutului, iar altul pentru confirmarea selectiei. Urmatorul lucru de realizat este utilizarea unui Real Time Clock pentru a compara timpul la care s-a setat alarma cu timpul actual. In momentul in care s-a ajuns la ora la care trebuie sa sune alarma, se efectueaza urmatoarea secventa: - In primele 10 secunde functioneaza se aprinde un ventilator, cu ajutorul unui tranzistor (adica se aplica tensiune in baza tranzistorului pentru a lasa curentul sa treaca din baterii spre motor); - In urmatoarele 10 secunde se va aprinde lumina; - La urmatorul pas va suna alarma prin intermediul unui buzzer; - Ultimul pas reprezinta combinarea celor 3 tipuri de alarme anterioare. In orice moment, alarma va fi oprita la detectarea miscarii de catre un PIR.

De asemenea, proiectul foloseste un ecran LCD pentru afisarea unor mesaje pentru a sti care este ora in orice moment si la cat a fost setata alarma.

Initializari

```
void IO_init()
{
        //buton PD5 = Confirmare alarma
        //buton PD4 = Setare Ora (in mod circular 0-24)
        //buton PD1 = Setare Minut (0-60 cu pas 5)
        DDRD &= ~(1 << PD1) & ~(1 << PD4) & ~(1 << PD5);
        //Rezistenta de pull-up
        PORTD |= (1 << PD1) | (1 << PD4) | (1 << PD5);

        //Motor
        DDRA |= (1 << PA0);

        //LED
        DDRA |= (1 << PA3);

        //Buzzer
        DDRA |= (1 << PA6);
        PORTA &= ~(1 << PA6);

        //Senzor de miscare
        DDRA &= ~(1 << PA1);

        //Intreruperi pentru butoane si senzor de miscare
        PCICR = 0;
        PCICR |= (1 << PCIE3) | (1 << PCIE0);
        PCMSK3 = 0;
        PCMSK0 = 0;
        PCMSK3 |= (1 << PCINT24) | (1 << PCINT25)
                        | (1 << PCINT28) | (1 << PCINT29);
        PCMSK0 |= (1 << PCINT1);
}
```

Tratare intreruperi

Tratare intrerupere venita de la senzorul de miscare

```
ISR(PCINT0_vect)
{
        if ((PINA & (1 << PA1)) != 0) {
                //detectare miscare
                if (OCR1B == 1 && alarm_started == 1)
```

```
                        OCR1B = 2;
        }
}
```

Tratare intreruperi de la butoane

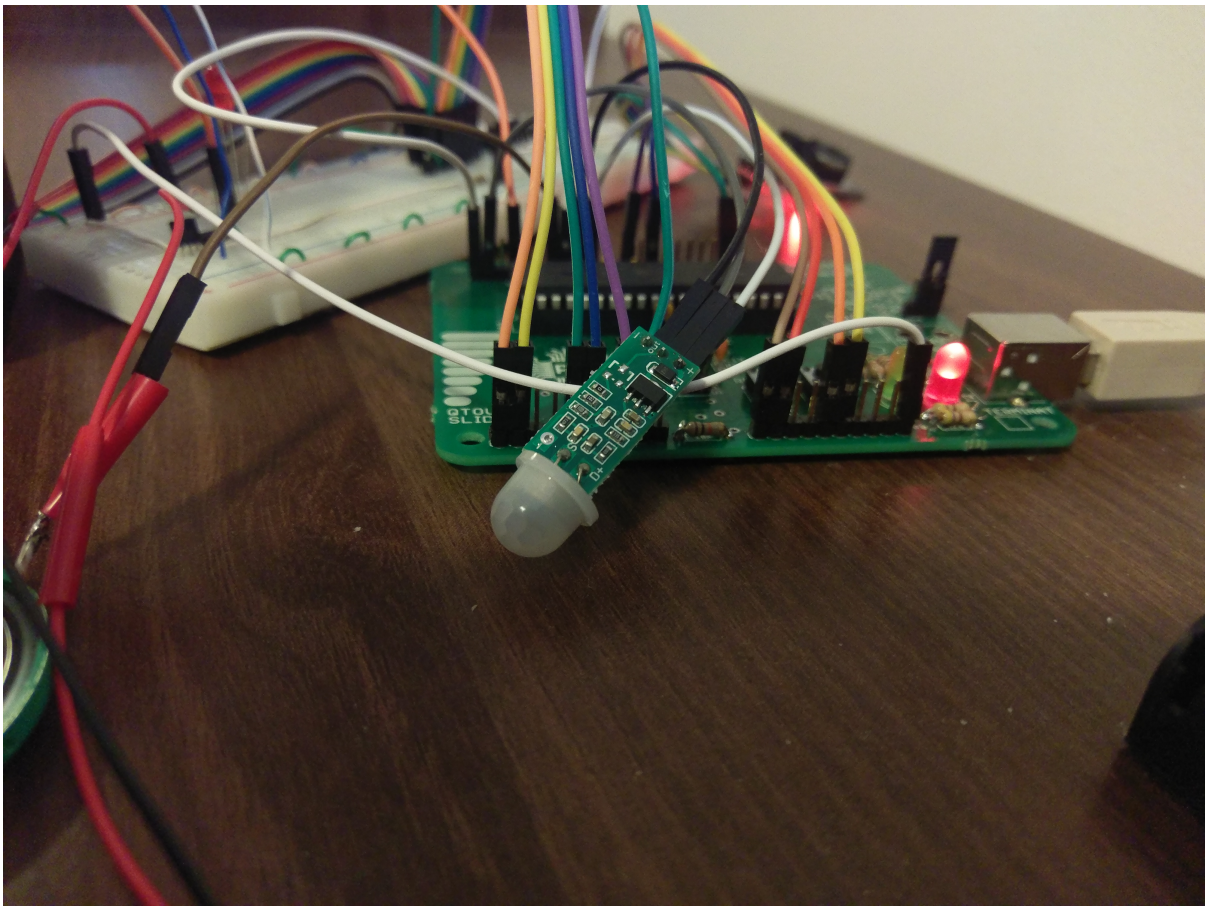
```
ISR(PCINT3_vect)
{
        if ((PIND & (1 << PD5)) == 0)
    {
                if(alarm_set == 0) {
                        OCR1B = 1;
                }
                _delay_ms(250);
        }

        else if ((PIND & (1 << PD4)) == 0)
    {
                h_alarm = (h_alarm + 1) % 24;

                sprintf(buffer1, "%d:%d", h_alarm, min_alarm);
                ST7735R_DrawText(50, 100, spaces, 255, 255, 0, 0, 0, 0);
                ST7735R_DrawText(50, 100, buffer1, 255, 0, 0, 0, 0, 0);
                _delay_ms(250);
        }

        else if ((PIND & (1 << PD1)) == 0)
    {
                min_alarm = (min_alarm + 1) % 60;

                sprintf(buffer1, "%d:%d", h_alarm, min_alarm);
                ST7735R_DrawText(50, 100, spaces, 255, 255, 0, 0, 0, 0);
                ST7735R_DrawText(50, 100, buffer1, 0, 255, 255, 0, 0, 0);
                _delay_ms(250);
        } else if ((PIND & (1 << PD0)) == 0)
    {
                if(alarm_set == 1) {
                        OCR1B = 2;
                }
                _delay_ms(250);

        }
}
```

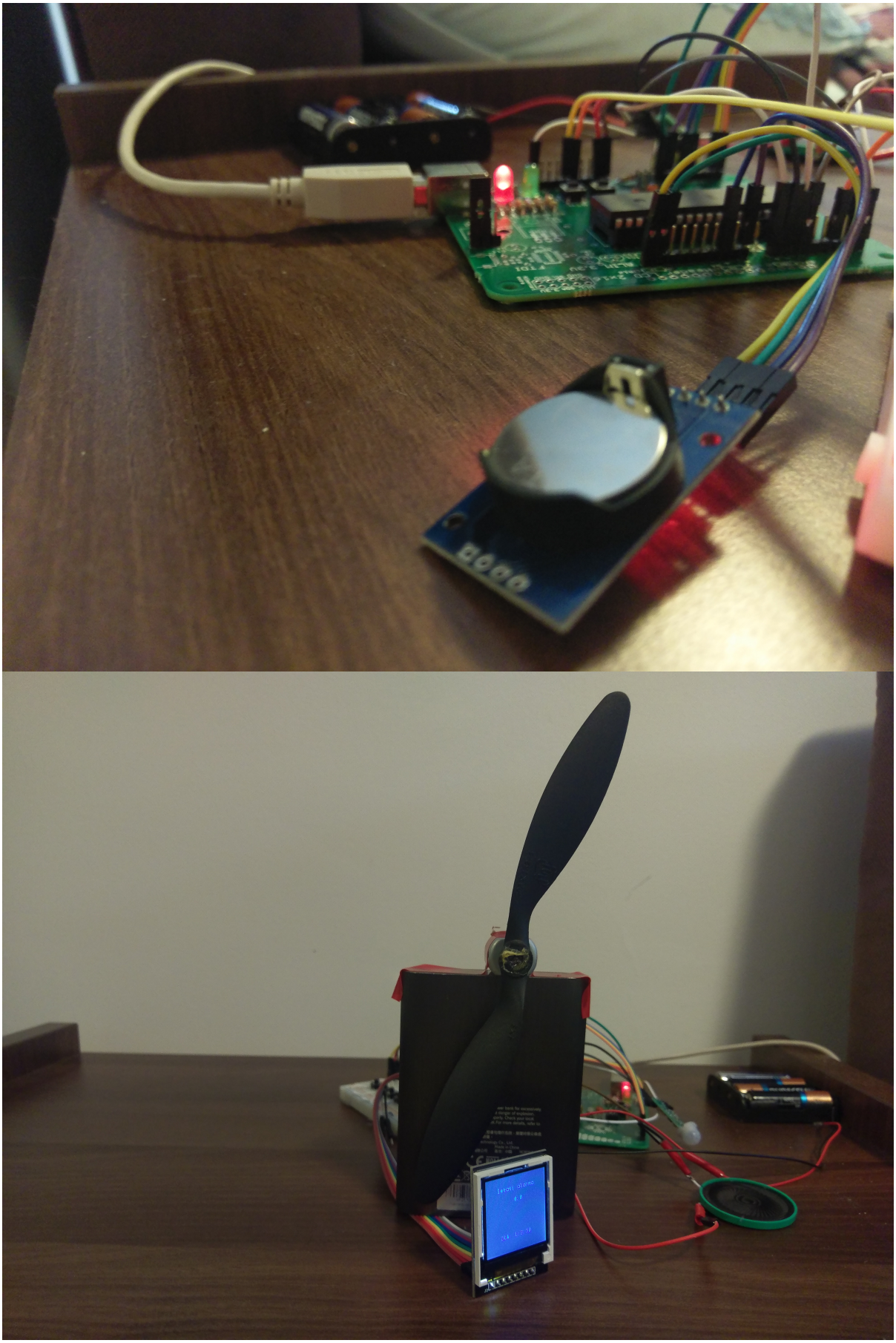
Biblioteci

In implementarea proiectului am folosit cateva biblioteci ajutatoare: una pentru LCD, iar alta pentru RTC.

Rezultate Obţinute

Rezultatele se pot vedea in pozele urmatoare.





Setati alarma
0: 0
ORA: 1: 7: 33
J2

Concluzii

In final, pot sa spun ca sunt fericit ca am reusit sa fac proiectul sa functioneze asa cum am vrut initial. Desi pare foarte simplu, am observat ca apar enorm de multe probleme pe parcurs la care nu m-as fi gandit. Asadar, sper sa il aduc intr-o forma mai frumoasa din punct de vedere vizual si poate chiar sa il folosesc deoarece mi se pare util.

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:prj2017:avoinescu:dumitru_alin**.

daniel_balteanu.zip

Bibliografie/Resurse

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

- SPI si LCD
- RTC
- Datasheet ATmega324A
- Documentația în format PDF

From:
https://ocw.cs.pub.ro/courses/ - **CS Open CourseWare**

Permanent link:
https://ocw.cs.pub.ro/courses/pm/prj2018/amocanu/proiect_danielbalteanu

Last update: **2021/04/14 15:07**