Alexandru MOSCU (78414) - Cooler laptop

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

Introducere

In general un cooler are doua moduri de functionare:

- ventilatoarele se rotesc cu o viteza constanta indiferent de temperaturile laptop-ului
- ventilatoarele se pot roti cu mai multe viteze alese de producator (2-3 ca numar), putand fi schimbate prin intermediul unor butoane

Proiectul ales consta in implementarea unui cooler pentru laptop care va putea fi controlat prin intermediul unei aplicatii software. Daca laptopul se afla sub stres ridicat (temperaturi ridicate) atunci ventilatoarele cooler-ului se vor roti cat se poate de repede, iar daca laptopul e in standby ventilatoarele se vor roti la o viteza minima pentru a nu scoate un sunet deranjant.

Descriere generală

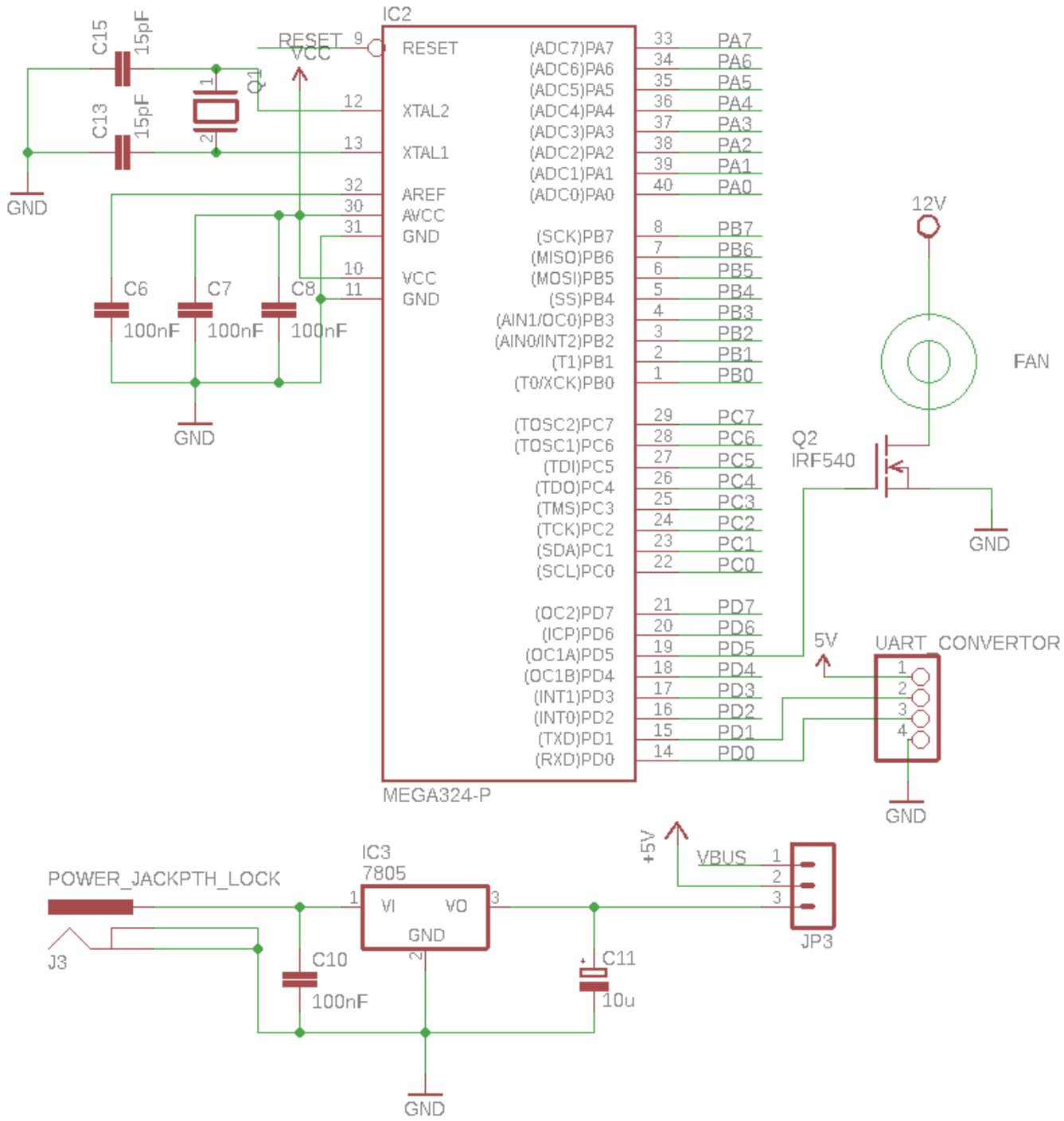
Aplicatia software va transmite informatii legate de tempraturile interne ale CPU-ului si astfel microcontroller-ul va decide o viteza de rotatie optima a ventilatoarelor.

Hardware Design

1. Lista de piese

- Placa de baza
- Microcontroller ATmega324
- Cablu Convertor USB la UART PL2303HX
- Ventilator 12V
- Stabilizator de tensiune 5V L7805CV
- Tranzistor MOSFET N-MOS IRF540N
- Fire, rezistente, condensatoare etc.

2. Schema electrica



Software Design

1. Mediu de dezvoltare

- **Sistem de operare:** Windows
- **Schema electrica:** Autodesk Eagle 9.0.0
- **Editor:** Sublime Text 3

- **Incarcare:** HID Boot Flash

2. Librarii si surse 3rd-party

- **jSensors:** jSensors
- **jSerialComm:** jSerialComm

3. Implementare

Microcontroller

- **cooler.c, cooler.h:** punctul de start al aplicatiei. Are loc receptionarea datelor prin USART si stabilirea vitezei de rotatie a ventilatorului.

```
/* modurile de functionare a cooler-ului */
#define MODE_STOP 0
#define MODE_MANUAL 1
#define MODE_TEMP 2

/* primeste setarile prin seriala */
void receive_settings();

/* initializeaza timer1 pe module FastPWM */
void FastPWM_init();

int main();
```

- **usart.c, usart.h:** implementarea functiilor pentru initializare si transmitere/receptionare USART.

```
/* functie de initializare a controllerului USART */
void USART0_init();

/* functie ce transmite un caracter prin USART */
void USART0_transmit(unsigned char data);

/* functie ce primeste un caracter prin USART */
unsigned char USART0_receive();
```

Aplicatie desktop

Aplicatia de desktop am facut-o folosind API-ul de Swing-uri pus la dispozitie de Java pentru GUI,

jSensors pentru detectarea temperaturii procesorului si jCommSerial pentru transmiterea comenzilor catre microcontroller.

- **Main:** punctul de start al aplicatiei. Se deseneaza GUI-ul.
- **Sensors:** folosit pentru identificarea procesorului si obtinerea temepraturii pentru fiecare core.

```
  /* obtinerea efectiva a informatiilor oferite de senzorii sistemului */
  static Sensors getSystemInfo();

  /* obtinerea numelui procesorului sistemului */
  String getCPUName();

  /* obtinerea temperaturilor asociate CPU-ului */
  List<Temperature> getCPUTemperature();
```

- **Serial:** initializeaza convorbirea seriala cu microcontroller-ul si transmite/receptioneaza setarile pentru cooler.

```
  /* modurile de functionare a cooler-ului */
  static final int MODE_STOP =     0b00;
  static final int MODE_MANUAL =   0b01;
  static final int MODE_TEMP =     0b10;

  /* constructorul clasei in care se initializeaza interfata USART (baud
rate, biti paritate, stop etc.) */
  Serial();

  /* trimite modul de functionare si valoarea asociata catre microcontroller
*/
  void send(int mode, byte value);

  /* receptioneaza modul de functionare si valoarea de la microcontroller
(folosita pentru debugging) */
  void receive();
```

- **UI:** contine initializarea elementelor interfetei si logica acesteia.

```
  /* constructorul interfetei; initializeaza elementele si obiectele
folosite */
  public UI(JFrame frame);

  /* un worker care este executat la fiecare secunda si se ocupa de
update-ul GUI-ului */
  public class SensorsWorker extends SwingWorker<Sensors, Sensors> {

    /* afla noile temperaturi ale CPU-ului folosind o instanta a clasei
Sensors intr-un
       thread propriu */
    protected Sensors doInBackground();
```

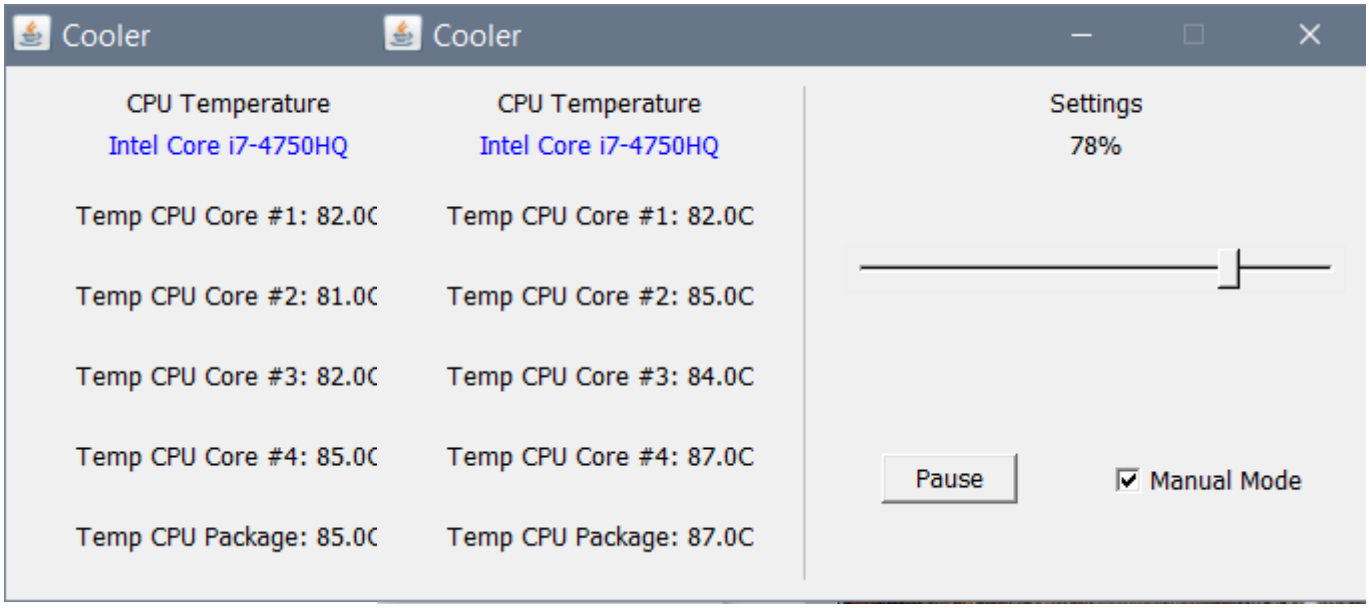
```
    /* updateaza GUI-ul pe baza informatiilor gasite in doInBackground() */
    protected void done();

  }

  /* se ocupa de transmiterea setarilor cooler-ului catre microcontroller */
  public class SerialRunnable implements Runnable {
    public void run();
  }
```

4. Logica de functionare

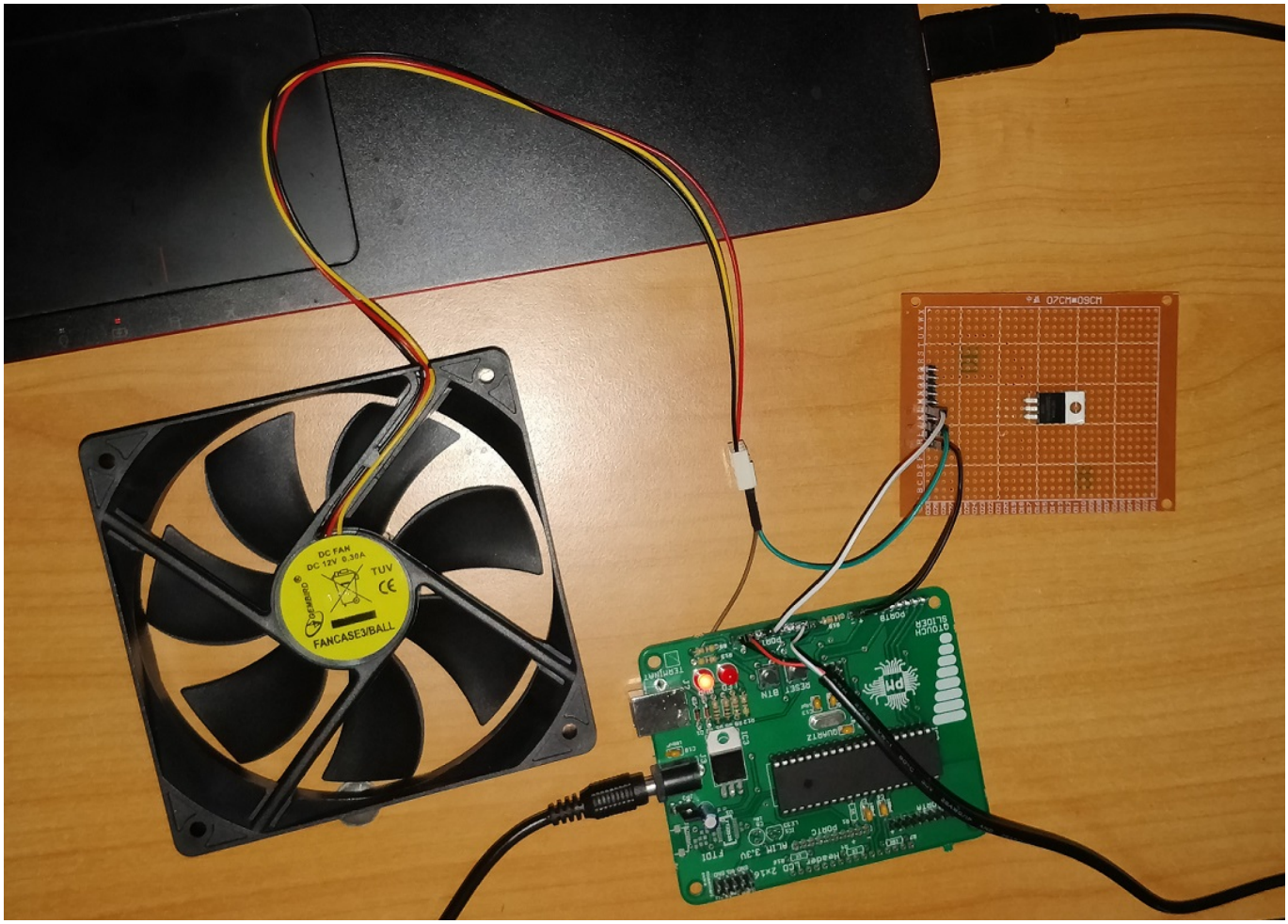
La pornirea cooler-ului acesta este in modul STOP si nu se invarte deloc. Se asteapa apasarea butonului Start din cadrul aplicatiei.
Odata apasat butonul de Start se va transmite sub modul de temperatura valori catre microcontroller. Valoarea va fi o medie a valorilor afisate in partea stanga a aplicatiei. Daca se doreste sa se controleze manual viteza de rotatie (procentual) a cooler-ului se poate bifa casuta “Manual Mode” si se va putea controla procentul prin intermediul slider-ului.
Daca se doreste intreruperea functionarii cooler-ului se poate apasa pe Pause.

Rezultate Obţinute

Aplicatia software:



Hardware:



Concluzii

A fost un proiect in urma caruia am castigat puina experienta cu sistemele embedded, in special pe partea de hard. Consider ca ideea proiectului este una foarte practica si as dori ca pe viitor sa revin

asupra lui si sa aduc anumite imbunatatiri precum: sa folosesc API-ul pus la dispozitie de catre NVIDIA pentru aflarerea temperaturii GPU-ului si sa calculez viteza de rotatie a cooler-ului in functie de aceasta. De asemenea, la inceputul proiectului nu mi-am dat seama ca voi avea nevoie de o sursa de 12V (adica de un fir in plus pe langa USB), astfel ca o alta imbunatatire ar fi folosirea unui modul de bluetooth in locul convertorului USART-USB.

Download

moscualexandru_331cb.zip

Bibliografie/Resurse

- Datasheet ATMega doc8272.pdf
- jSerialComm package-summary.html
- Documentația în format PDF