Alarm System

Autor

Oprea Andrei-Dorin 331CB

Introducere

Am ales realizarea unei alarme de apartament. Aceasta permite activarea/dezactivarea ei cu ajutorul unei parole numerice. In cazul in care se detecteaza miscare si alarma este activata va porni un semnal sonor.

Descriere generală

- Introducerea parolei se realizeaza cu ajutorul unui keypad
- Alte instructiuni se trimit catre alarma tot prin keypad:
  - Activeaza alarma - Necesita apasarea tastei * cand alarma este dezactivatata
  - Schimba parola - Necesita apasarea tastei # cand alarma este dezactivata. Se afiseaza un prompt pentru introducerea noii parole.
- Instructiunile si starea alarmei se afiseaza pe un display LCD (activat/dezactivat, prompt pentru parola)
- Semnalul sonor va fi genereat de un buzzer cand este detectata miscare si alarma este activata. Acesta se va opri doar dupa introducerea corecta a parolei
- Detectarea unei persoane va fi realizata cu ajutorul unui senzor PIR
- In cazul introducerii unei parole gresite exista un timp de asteptare pana la urmatoarea incercare. Acesta creste pentru incercari succesive gresite de la 2s pana la 90s.
- Verificarea parolei se realizeaza automat cand lungimea inputului este 4 (parola are lungime 4)
- Intr-un prompt de parola tasta # este folosita pentru a sterge ultimul caracter.

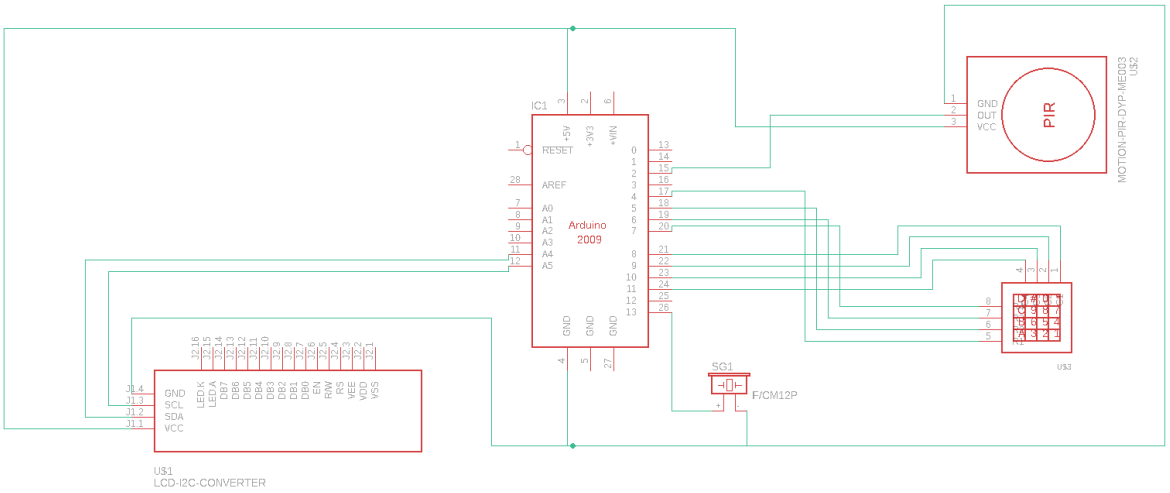
Schema bloc:

Hardware Design

Lista componente:

- Arduino Uno
- Display LCD I2C
- Keypad 4×4
- Senzor PIR HC-SR501
- Buzzer
- Conectori
- Breadboard

Schema electrica:



Software Design

Am folosit Arduino IDE ca mediu de dezvoltare. Biblioteci externe folosite:

- Keypad.h pentru interfatarea key padului
- LiquidCrystal_I2C.h pentru comunicarea folosind protocolul I2C cu LCD-ul

Alarma poate fi considerata un fel de state machine. Variabilele globale movementDetected, alarmArmed si changePassword denota in principiu aceste schimbari.

Alte **variabile globale**:

- **ROWS, COLS , keys[][], rowPins, colPins** - folosite pentru initializarea tastaturii;
- **keypad** - obiect de tip Keypad (referinta catre tastatura);
- **lcd** - obiect de tip LiquidCrystal_I2C (referinta catre display);
- **password** - buffer pentru parola alarmei;
- **inputPassword** - stocheaza inputul utilizatorului cand ii este oferit promptul de parola;
- **setup()** - In setup se initializeaza LCD-ul, pinul pentru buzzer si se ataseaza functia ce se executa in

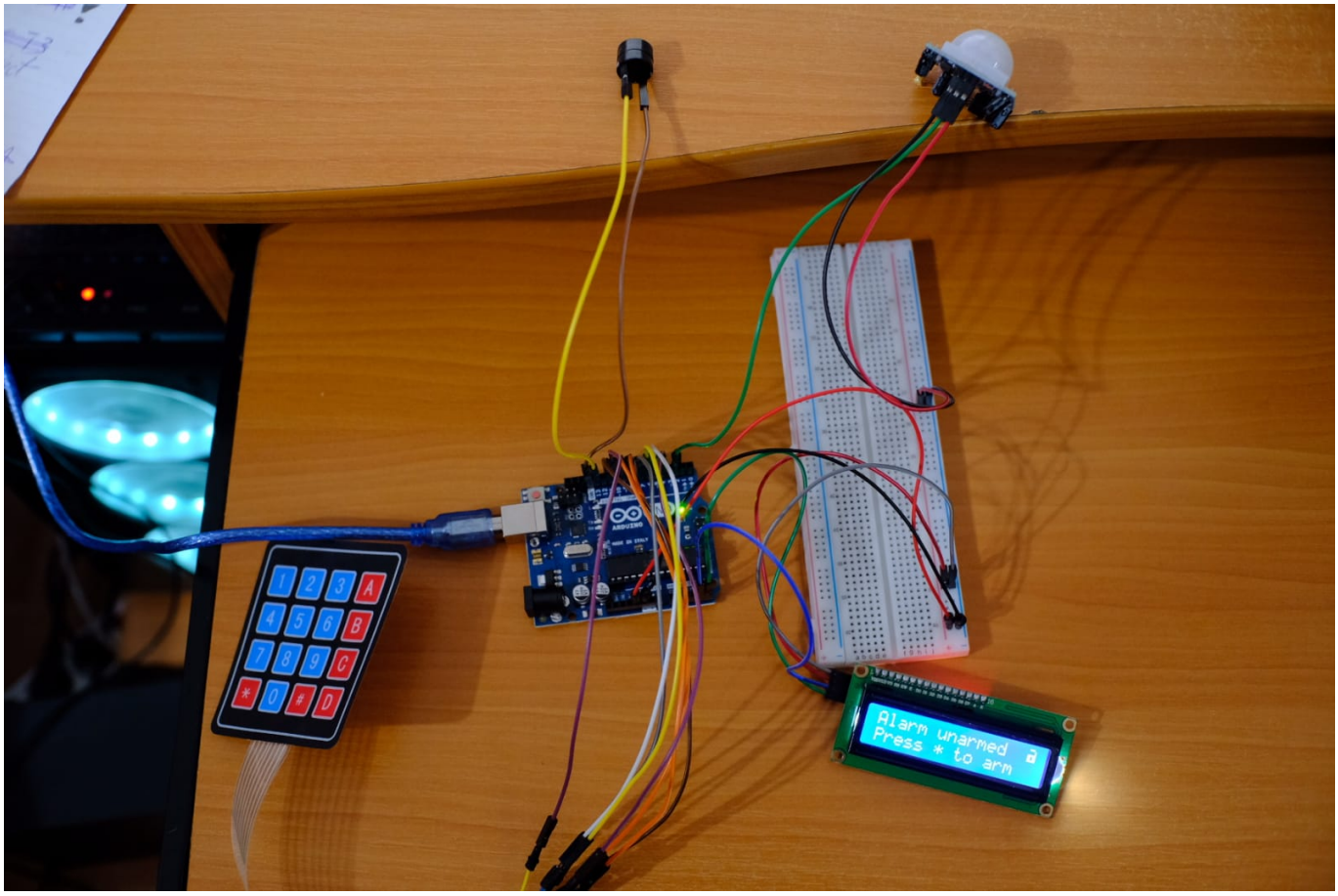
cazul unei intreruperi primite de la senzor. Intreruperea este declansata de frontul crescator al semnalului;
- **resetInput()** - seteaza inputIdx la inceputul bufferului si suprascrie password cu caractere dummy care nu sunt pe tastatura;
- **renderLcd()** - apeleaza displayUnarmed sau diplayPrompt in functie de starea in care se afla alarma;
- **inputIdx** - pe care trebuie sa scriui in inputPassword cand userul apasa pe o tasta;
- **waitTime** - vector cu timpii de asteptare in caz de inserare parola gresita(2s, 2s, 2s, 5s, 15s, 30s, 60s, 90s);
- **waitIdx** contor pentru a retine cate incercari gresite consecutive are userul;
- **char_lock** si **char_unlock** - caractere custom facute sa arate ca un lacat inchis respectiv deschis;
- **reRender** - boolean setat cand trebuie rerandat ecranul (dupa apasarea unei taste pe tastatura).

Functii:

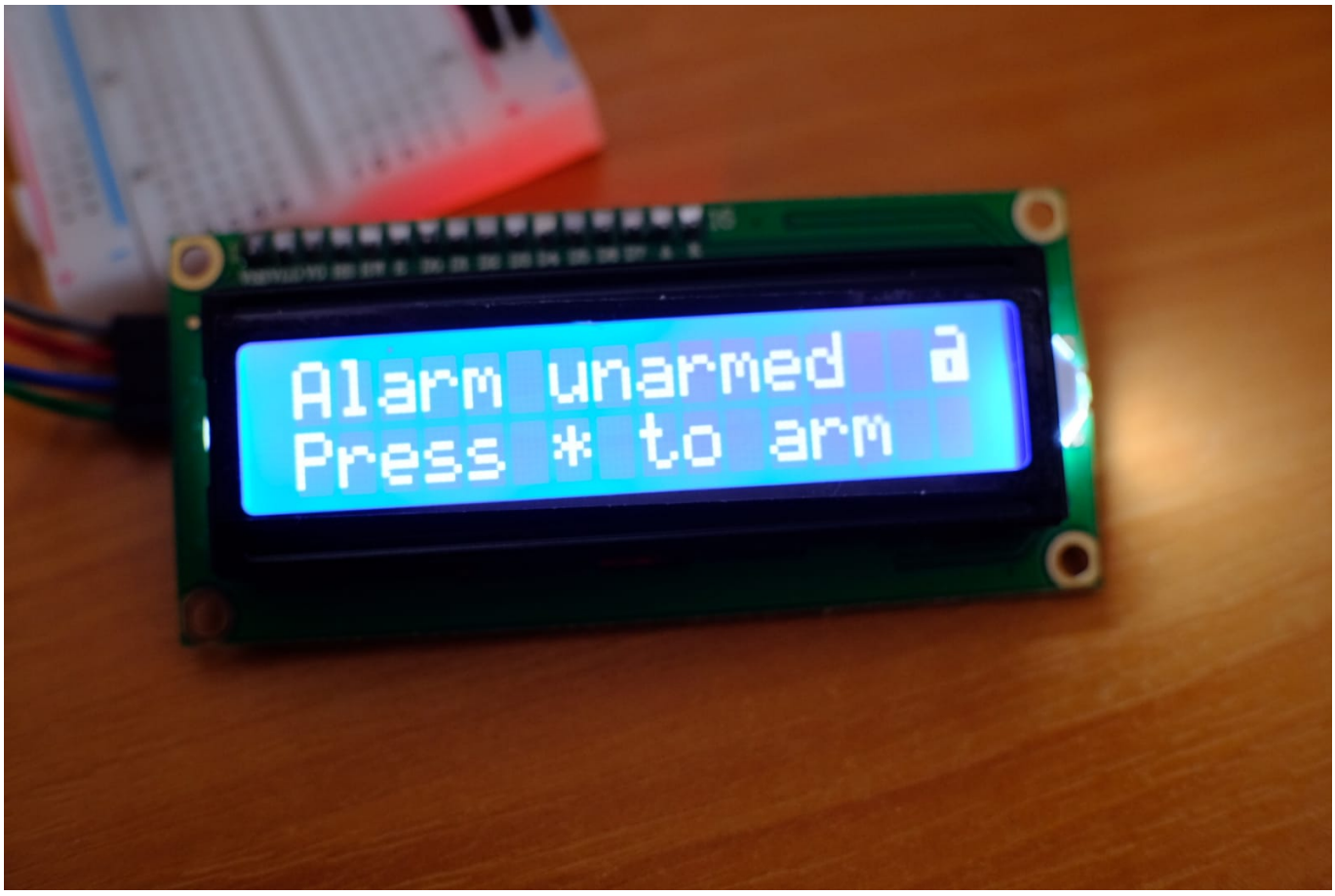
- **initLcd()** - intializare display, curatare, aprindere lumina de fundal, creeaza caracterele custom;
- **initKeypad()** - seteaza timpul de debounce al tastaturii;
- **displayUnarmed()** - afisare pentru alarma dezactivata;
- **diplayPrompt()** - afisare prompt pentru parola. Promptul difera in functie de variabila chagePassword;
- **handleBuzzer()** - seteaza pinul pentru buzzer pe HIGH sau LOW in functie de variabila movementDetected;
- **interrupt_routine()** - codul pentru intreruperea generata de senzorul PIR. Seteaza variabila movement detected in functie de starea alarmei;
- **wrongTry()** - Afiseaza un mesaj de eroare pe ecran si obliga userul sa astepte un timp definit de waitTime in functie de numarul de incercari gresite. In acest timp se afiseaza timpul ramas de asteptare si se verifica daca buzzerul trebuie pornit. Timpul ramas este calculat prin intermediul functiei milis;
- **setup()** - se initializeaza LCD-ul, pinul pentru buzzer si se ataseaza functia ce se executa in cazul unei intreruperi primite de la senzor. Intreruperea este declansata pe frontul crescator al semnalului primit de la senzor;
- **loop()**
  - preia input de la keypad;
  - rerandeaza displayul daca este cazul;
  - modifica starea buzzerului;
  - daca inputul de la keypad exista (nu este null) in functie de starea alarmei control flowul este diferit:
    - daca alarma este armata atunci verific daca caracterul este #. Daca da sterg ultimul caracter din buffer. Daca nu il adaug in buffer. Apoi verific daca parola este corecta si daca da resetez toate variabilele globale corespunzator. Altfel apelez wrongTry si resetez bufferul;
    - daca alarma nu este armata atunci verific daca inputul este * si atunci schimb variabila globala care semnalizeaza activarea ei. Daca este # atunci semnalizez ca trebuie sa schimb parola. Pentru schimbarea parolei preiau 4 caractere ca input si le copiez in bufferul password;
    - semnalizez ca ecranul trebuie actualizat.

Rezultate obtinute

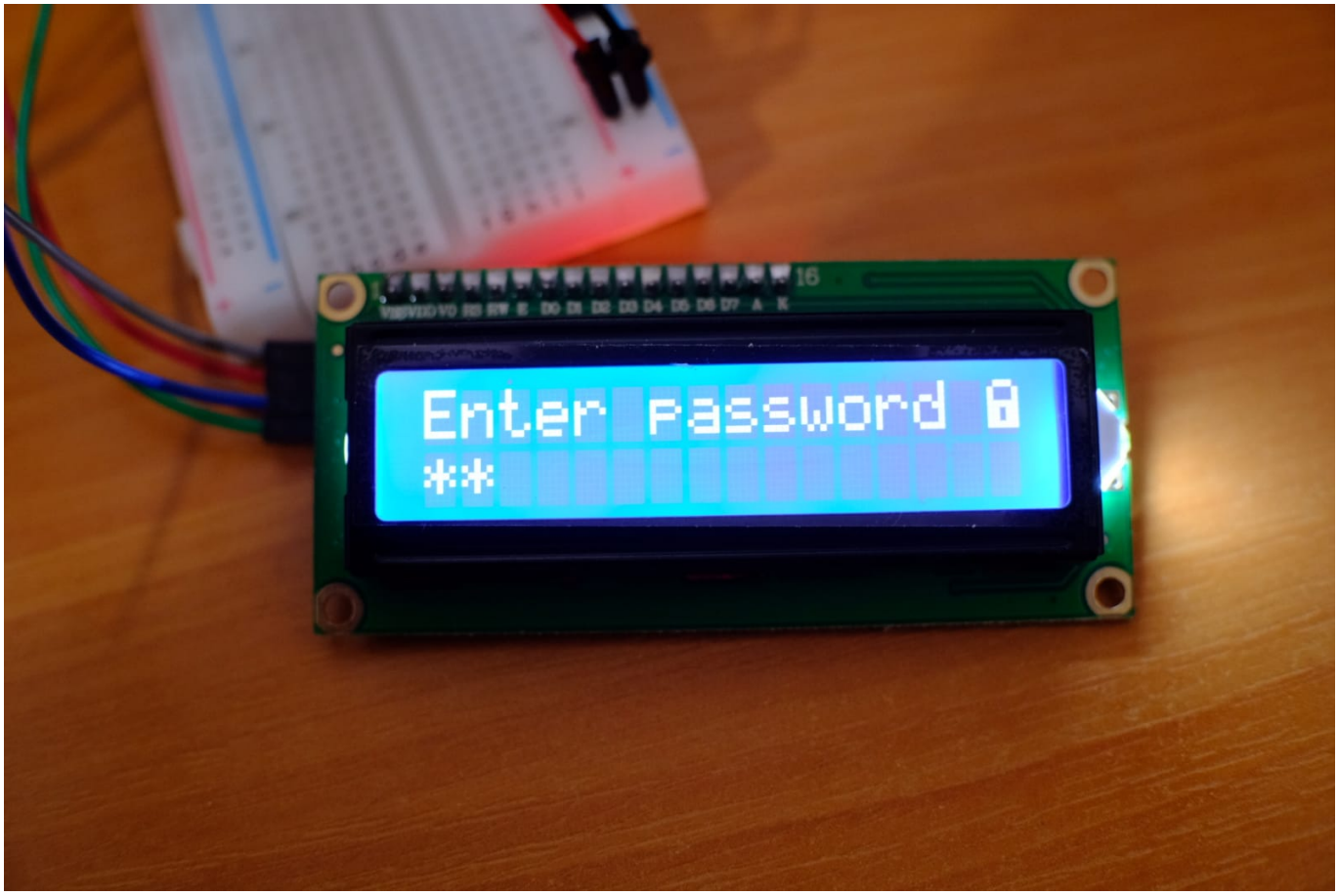
Overview al proiectului:



Stare initiala (alarma dezactivata):



Introducerea parolei(alarma activata):



Introducerea gresita a parolei:

Schimbarea parolei:

Demo pentru functionalitatea proiectului. Sound warning secunda 45.

Concluzii

- Am reusit sa implementez tot ce mi-am propus si sunt multumit de rezultat.
- Am invatat cum sa lucrez cu partea hardware a unui proiect. Faptul ca am proiectat singur designul proiectului m-a ajutat sa inteleg mult mai bine informatia din laboratoare.
- Dificultai:
  - rabdarea (maini mari, componente mici)
  - debugging (30 de minute sa imi dau seama ca LCD-ul functioneaza cum trebuie si nu exista nici un bug software, ci contrastul nu era calibrat + multe altele)

Download

Arhiva cu codul sursa si readme: alarm_system.zip.

Jurnal

- 25.04: alegerea temei de proiect si creearea pagini de wiki.

- 26.04 - 16.05: lucru la partea hardware a proiectului (conectare + verificare componente).
- 17.05 - 30.05: lucru la partea software a proiectului (implementare software efectiva si testare).
- 31.05: finalizarea paginii de wiki.

Bibliografie/Resurse

PDF

https://www.makerguides.com/hc-sr501-arduino-tutorial/

https://playground.arduino.cc/Code/Keypad/

https://create.arduino.cc/projecthub/arduino_uno_guy/i2c-liquid-crystal-displays-5b806c

https://lastminuteengineers.com/i2c-lcd-arduino-tutorial/