Incuietoare cu alarma

Introducere

Proiectul implementeaza o incuietoare simpla cu alarma si cateva feature-uri suplimentare.

Folosind un keypad, un client poate introduce o parola (stocata in ROM, adica flash-uita deja in memoria arduino-ului, care poate fi configurata la achizitia incuietorii). In cazul in care parola introdusa este corecta, vom actiona un servomotor care deblocheaza incuietoarea. In caz contrar un buzzer va incepe sa atentioneze o incercare nereusita si cante un sunet ascutit timp de cateva secunde.

Descriere generală

Pentru a introduce un nivel suplimentar de securitate, vom permite deblocarea alarmei doar daca parola a fost introdusa de o persoana (pentru a combate incercarile de deblocare ale unui robot, sau pur si simplu ca un failsafe suplimentar). Acest lucru il vom face cu un senzor de temperatura. Clientul va trebui sa introduca parola in timp ce se aseza in fata senzorului (montat in alarma); iar senzorul va permite deschiderea doar daca va detecta caldura.

Deoarece senzorul de temperatura este inconsistent si nu da rezultate exact de multe ori, am introdus si un feature in care putem da toggle la verificarea temperaturii dupa ce introducem parola. Daca apasam tasta 'A' de pe keypad, feature-ul sa va dezactiva daca era activat (sau va fi activat daca era dezactivat inainte).

Am introdus si 2 mecanisme de timeout in incuietoare:

1. Din momentul in care introducem prima cifra in alarma avem 16 secunde sa introducem restul, sau alarma va porni si progresul se va reseta
2. Dupa ce am introdus corect parola avem din nou 16 secunde sa aducem senzorul de temperatura la valoarea setata (20 de grade Celsius) pentru a deschide incuietoarea

Terminarea timerelor duce la resetarea mecanismului si pornirea alarmei.

Aceasta alarma ar putea fi folosita la o intrare de bloc / intrare intr-un institutie cu usurinta deoarece este foarte usor de produs la un cost redus.

Hardware Design

List componente:

- Arduino UNO
- Keypad 4×4
- LCD cu modul I2C
- Buzzer
- Servomotor
- Senzor umiditate si temperatura

Schema electrica:

Software Design

Am folosit Arduino IDE (TODO Link) pentru a scrie programul. Bibliotecile folosite sunt:

- Keypad library https://github.com/Chris--A/Keypad
- Lcd with I2C https://www.arduino.cc/reference/en/libraries/liquidcrystal-i2c/
- Servo https://www.arduino.cc/reference/en/libraries/servo/
- DHT11 Temperature library https://www.arduino.cc/reference/en/libraries/dht-sensor-library/

Programul foloseste 4 obiecte principale initializate global:

- Ecranul LCD care comunica cu placuta prin I2C
- Servo-ul care foloseste timer 1 pentru a comunica cu arduino
- Senzorul de temperatura DHT11 care comunica cu arduino prin timer 0
- Keypad-ul initializat cu pinii de pe arduino si caracterele desenate fizic pe butoane

Buzzer-ul este controlat printr-o functie “ring_buzzer” care va face buzzerul sa sune timp de o perioada scurta si va afisa un mesaj pe lcd. (nu avem nevoie de obiect special pentru el, il controlam doar prin pin)

Printre celelalte variable globale ale systemului, avem si parola pe care o introduce user-ul si parola corecta cu care este ea verificata. Practic parola de referinta pe care utilizatorul tebuie sa o introduca sa afla in ROM-ul sistemului. (read-only din punctul de vedere al utilizatorului)

Functii implementare:

- **init_lcd()**

Initializeaza ecranul cu informatiile pe care ar trebui sa le vada user-ul la inceput, adica un prompt pentru a-i spune sa introduca parola

- **setup()**

Functia principala de arduino va initializa componentele sistemului: lcd-ul, buzzer-ul (il va opri

deoarece el default face zgomot), servo-ul (si va pozitiona bratul la 0 grade), senzorul de temperatura DHT11, timerul 2 folosit pentru feature-uri de timeout (descrise mai jos). Am folosit timer 2 pentru timeout deoarece celelalte 2 timere erau folosite de bibliotecile servo-ului si a DHT-ului. Am ales sa folosesc un prescaler de 1024 si intreruperea *TIMER2_COMPA_vect*

- **reset_lock()**

Reseteaza progresul facut de user in introducerea parolei. Avem cateva variabile in care se tine minte starea parolei introduse de user, iar apeland aceasta functie vom reseta acest progres.

- **ISR(TIMER2_COMPA_vect)**

Intreruperea setata pe timer2, tot ce face ea este sa decrementeze un contor. Vom folosi aceasta intrerupere pentru a seta niste timeout-uri pentru anumite operatii. Spre exemplu daca dorim ca o anumita funcite/actiune din aplicatie sa aiba un anumit timeout (si nu vrem sa facem busy waiting) vom seta variabila *counter* la o valoare specifica timpului pe care vrem sa il dam ca timeout si vom verifica in fiecare iteratie din loop daca valoarea variabilei a ajuns sub 0 (timpul a expirat). Cand se intampla asta vom declansa un alt eveniment si vom trata situatia de timeout. De asemenea avem un mecanism si pentru cazul in care operatia s-a incheiat inainte ca timerul sa ajunga la 0. In acest caz nu vom mai verifica valoarea variabilei si vom continua flow-ul obisnuit al programului.

- **is_digit(char c)**

O functie triviala care verifica pe baza tabelului ASCII daca caracterul primit este o cifra sau nu.

- **print_password_progress()**

Functia se uita la progresul parolei introduse de user si afiseaza in functie de cate cifre mai are de introdus informatii pe ecran. Daca user-ul mai are de introdus cifre vom afisa ce a introdus pana in momentul respectiv si un prompt pentru a introduce in continuare. Cand am introdus toate cele 4 cifre ale parolei vom afisa parola finala introdusa si vom continua cu validarea acesteia.

- **check_passwd()**

Functia va verifica daca parola introdusa de user este aceasi cu cea memorata in ROM (ca variabila globala in cod).

- **ring_buzzer(char message[])**

Functia va tine buzzerul pornit timp de cateva secunde si va afisa mesajul primit ca parametru pe ecran.

- **open_door()**

Servo-ul se va deschide gradual (de la 0 grade la 130 de grade) pentru a permite cutiei sa se deschida in totalitate.

- **close_door()**

Vom inchide servo-ul gradual (de la 130 la 0 grade). Se apeleaza dupa open_door() si permite cutiei sa se inchida pana la capat.

- **toggle_door()**

Functia va deschide usa, va astepta cu ea ridicata un timp (2 secunde setat de mine), iar apoi o va inchide.

- **reset_lcd()**

Vom sterge orice avem pe ecran in momentul de fata si vom afisa prompt-ul de introducere a parolei.

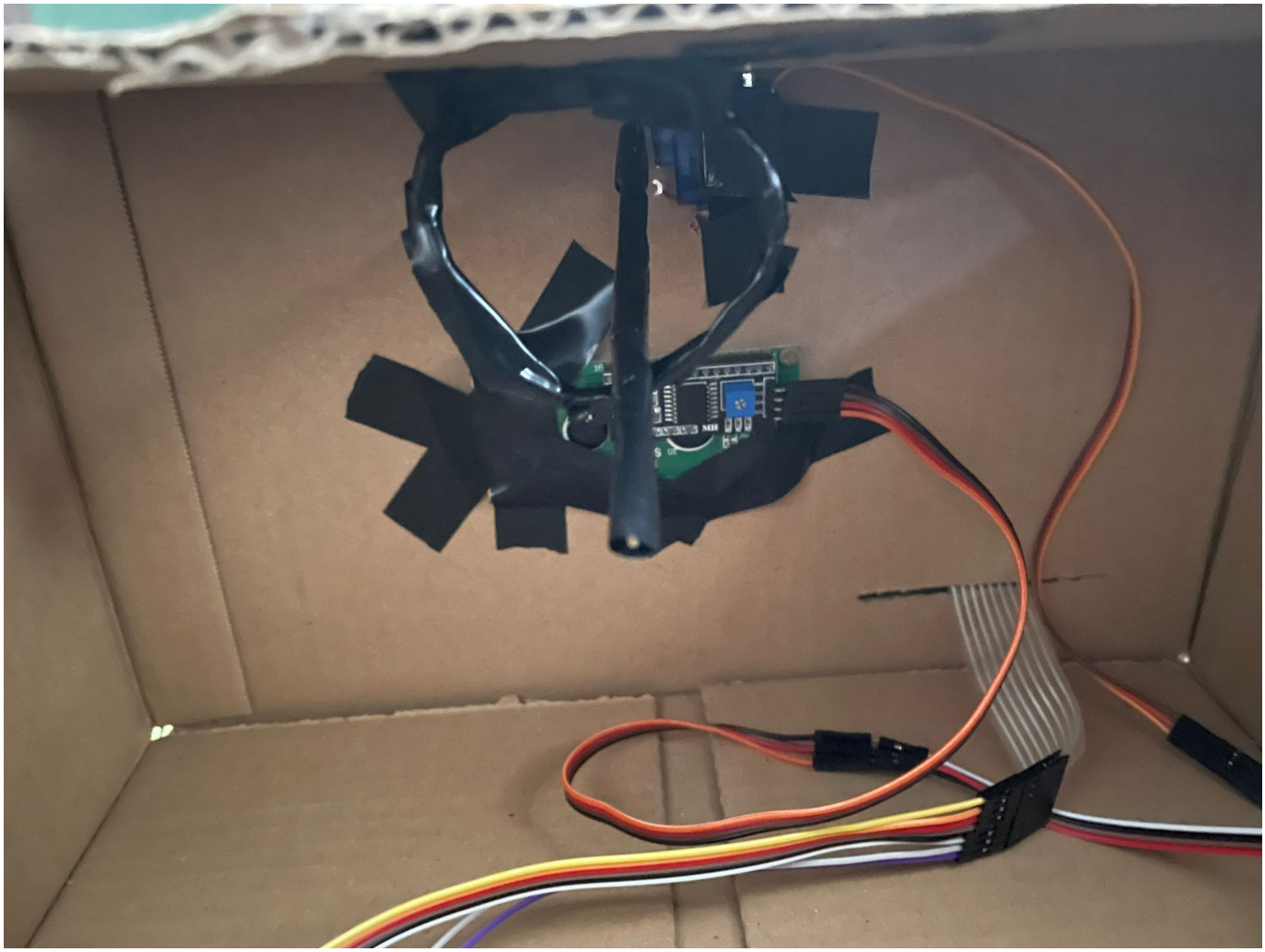
- **loop()**

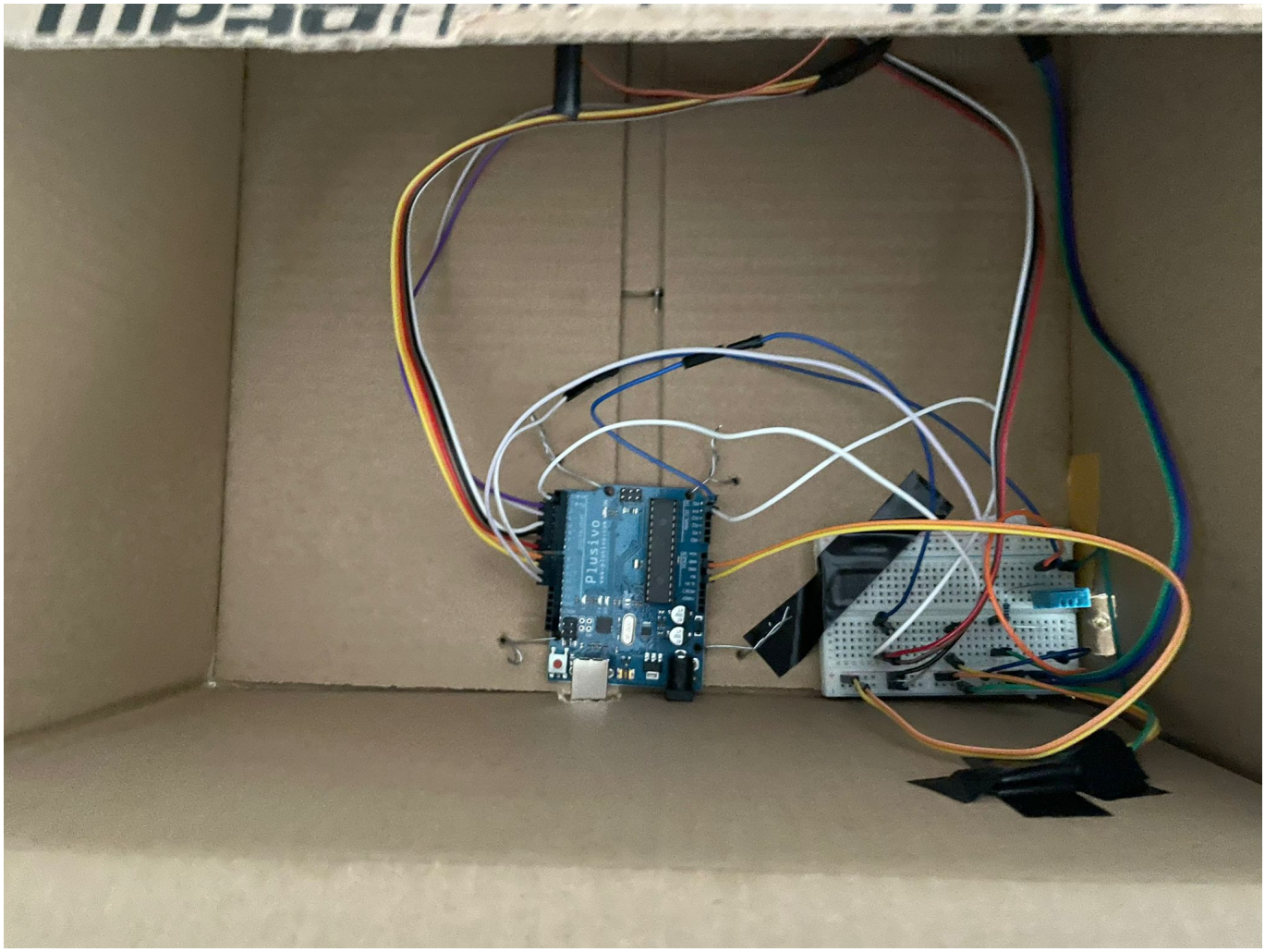
Functia verifica anumite functionalitati cheie ale incuietorii cum ar fi:

1. Odata ce introducem prima tasta de la keypad vom incepe timer-ul cu un timout de *16 secunde*
2. La fiecare cifra introdusa vom schimba prompt-ul pe ecran cu progresul actual al parolei
3. Daca parola finala este incorecta vom suna buzzerul
4. Daca parola finala este corecta verificam senzorul de temperatura, daca timp de 16 secunde acesta nu indica peste o anumita temperatura vom intra in procedura de “fail”, adica vom suna buzzerul si reseta alarma; insa daca ajunge la temperatura in acest timp vom actiona servomotorul si deschide usa
5. Putem de asemenea prin apasarea tastei 'A' sa facem toggle la variabila “will_read_temp”. Aceasta controleaza mecanismul de bypass al verificarii temperaturii. Daca acest feature este oprit, incuietoarea nu va mai verifica temperatura pentru a o deschide, va fi necesara doar introducerea parolei

Rezultate Obţinute


mBeam
1 2 3 A
4 5 6 B
7 8 9 C
* 0 # D




Plusivo


GymBeam
SHARE your unboxing with the hashtag #gymbeam
ALIMENTARE

- **Youtube DEMO**

Concluzii

Am vazut cate feature-uri aditionale poate sa aiba o simpla alarma, plecand de la ideea de o alarma simple cu keypad si buzzer am ajuns sa introduc feature-uri precum: senzor de temperatura pentru detectia persoanei care deblocheaza alarma, timout la introducerea parolei, timeout la detectia persoanei, etc.

Totusi, aceste feature-uri vin cu diferite trade-off uri si requirement-uri, spre exemplu:

- incercand sa introduc feature-ul de timeout, folosind un timer de pe arduino (unul din cele 3 puse la dispozitie) am ales prima data timer-ul 1, insa ceva nu functiona, astfel am vazut ca biblioteca de servo foloseste acest timer pentru anumite functionalitati. Incercand apoi timer0 am vazut ca senzorul de temperatura DHT11 nu merge, afland dupa debug ca biblioteca de DHT foloseste timer 0. Am ajuns astfel sa folosesc timer2 pentru functia de timeout si am luat valorile de prescaling din datasheet-ul **ATmega328P**
- am facut eforturi la nivelul hardware pentru a lega bratul servomotorului de mai mult betisoare de

cafea de la 5 To Go-ul din fata facultatii astfel incat sa se deschida capacul unei cutii.
- senzorul DHT11 este destul de inconsistent, se incalzeste greu la temperatura dorita si are pinii foarte subitiri si fragili. Pentru ca incuietoarea sa fie reliable ar fi necesar un senzor mai performant, mai precis, mai sensibil la temperatura.

Proiectul a fost interesant de realizat si a fost un challenge atat la nivel software si la nivel hardware, am invatat skill-uri noi precum lipire pinilor: am lipit singur (si cu ajutorul laborantului) un driver I2C de un LCD

Download

O arhiva cu codul proiectului poate fi descarcata la acest link
calitescu_mihai-gabriel_331cb_proiect.zip

De asemenea codul proiectului poate fi vazut si pe github

Jurnal

- 2 - 6 Mai

Alegerea tema proiect si dat comanda de piese, asigurat ca piesele primite functioneaza

- 9 - 13 Mai

Conturarea proiectului, a workflow-ului alarmei si a requirement-urilor software pentru fiecare componenta in parte

- 16 - 20 Mai

Lipire piese, asamblare hardware, punerea proiectului in cutie si fixarea lui

- 23-27

Implementarea alarmei in software si terminarea proiectului

Bibliografie/Resurse

- Datasheet
- Lab 2 - Timere
- Lab 3 - Servo
- Folosire senzor DHT11
- Conectare LCD I2C

- Folosire buzzer activ
- Timer in Arduino

Export to PDF