Proiect
Proiectare cu microprocesoare

Universitatea Tehnică din Cluj-Napoca
Brînzoi Ion-Robert
Grupa 30231
Profesor îndrumător: Itu Răzvan

Cuprins

1 Problema propusă

Scopul proiectului constă în simularea unui ATM folosind LCD shield-ul și un keypad. Prin intermediul keypad-ului se va introduce un cod pin, după care se va prezenta o listă de opțiuni. Mai întâi se poate modifica limba curentă, după care se pot vizualiza fondurile, să se efectueze o depunere sau o retragere, sau să se întoarcă la etapa precedentă. Voi folosi și două butoane pentru a încărca date din conturi diferite, și pentru a ilustra salvarea modificărilor făcute.

2 Soluția

Mai întâi am declarat un vector ce va conține tastele ce vor fi apăsate, precum și pinii corespunzători coloanelor și rândurile keypad-ului, urmate de instanțierea acestuia după includerea bibliotecii Keypad.h[1]. Am definit și o structura de tip account care va stoca PIN-ul contului respectiv și suma de bani curentă. Ulterior am instanțiat un vector alcătuit din două structuri de acest tip.

Pentru implementare, am declarat câte o variabilă booleană pentru a-mi specifica starea în care ma aflu. Voi avea următoarele stări:

- **insertPinStage** - starea inițială, în care utilizatorul introduce codul pin
- **chooseLanguageStage** - starea în care utiliatorul poate modifica limba
- **chooseOptionStage** - starea în care utilizatorul alege între a vizualiza fondurile, depunere sau retragere numerar
- **withdrawStage** - starea de retragere numerar
- **depositStage** - starea de depunere numerar
- **balanceStage** - starea de vizualizare a fondurilor curente

Variabile importante:

- char **insertedPIN[5]** - stochează pin-ul inserat în starea inițială
- char **inputAmount[7]** - stochează suma de bani inserată în starea de depunere sau de retragere
- int **viewLanguage**, **viewOption** - alternează între mesajele ce vor fi afișate pe LCD în **chooseLanguageStage**, respectiv **chooseOption-Stage**
- char **messages[][16]** - stochează mesajele ce vor fi afișate pe ecran pentru toate cele trei limbi implementate (engleză, română, franceză)
- int **languageIndex** - semnifică care limbă a fost aleasă
- volatile int **accountToLoad** - variabila ce specifică care cont trebuie încărcat

- Account* **currentAccount** - reprezintă o referință către contul curent, pentru a putea fi modificate datele contului.

Am definit și funcții care vor fi apelate de mai multe ori pentru evitarea repetării codului și modularizarea acestuia:

- **returnToInitState()** - întoarce la starea inițială din orice stare
- **returnToOptions()** - întoarce la **chooseOptionStage** din starea de retragere sau depunere numerar

Pentru setarea string-urilor la zero **insertedPIN** și **inputAmount**, am folosit funcția memset[2].

În funcția **setup()** am inițializat LCD-ul și am setat întreruperile.

```
void setup() {
  lcd.begin(16, 2);
  lcd.print("ENTER PIN");
  pinMode(20, INPUT);
  pinMode(21, INPUT);
  attachInterrupt(digitalPinToInterrupt(20), loadAccount1,RISING);
  attachInterrupt(digitalPinToInterrupt(21), loadAccount2,RISING);
}
```

Am folosit pinii 20 și 21 ai plăcii Arduino Mega pentru a seta întreruperile ce duc la încărcarea datelor primului cont, respectiv celui de-al doilea.

Funcțiile de întrerupere (**loadAccount1()** și **loadAccount2()**) sunt simple, modificând doar variabila **accountToLoad**. Dacă este zero nu facem nimic, dacă este 1, încărcăm primul cont, iar dacă este 2, pe cel de-al doilea. Această verificare se face în funcția **loop()** la început, iar după încărcarea contului, variabila devine zero, și ne întoarcem la prima stare apelând funcția **returnToInitState()**.

```
 if(accountToLoad == 1){
    currentAccount = &accounts[0];
    lcd.clear();
    lcd.setCursor(0,0);
    lcd.print("Account 1 loaded.");
    delay(1000);
    returnToInitState();
  }
```

Implementarea stărilor se face în **loop()** prin verificarea variabilei boolene corespunzătoare.

2.1 Insert Pin State

In funcție de tasta apăsată, pe ecran se va scrie pin-ul introdus de utilizator și stocat în variabila **insertedPIN**, până la maxim 4 caractere. Când se apasă pe tasta *, aceasta este considerată ca un backspace, și se șterge ultima cifră. La apăsarea butonului de confirmare (#), se verifică dacă pin-ul introdus corespunde cu cel corect, și dacă este se afișează mesajul de confirmare și se trece în starea următoare, altfel se afișează mesajul de eroare și trimite utilizatorul din nou la inserarea pin-ului.

2.2 Choose Language State

În funcție de tasta apăsată 1,2 sau 3, are loc selectarea limbii, asignându-se variabila languageIndex cu 0, 1 sau respectiv 2, după care se trece la starea următoare. Dacă se apasă *, se revine la starea inițială.

```
if (key == '1' || key == '2' || key == '3'){
      languageIndex = key - '0' -1;
      chooseLanguageStage = false;
      chooseOptionStage = true;
    } else if (key == '*'){
      viewLanguage = 0;
      chooseLanguageStage = false;
      returnToInitState();
    }
```

Pentru că LCD-ul are decât 2 rânduri, și deoarece dacă aș pune **delay**, nu s-ar mai înregistra input-ul, am folosit funcția **millis()** pentru a salva timpul curent și timpul salvat precedent (inițial zero), după care fac diferența între ele. Dacă diferența este mai mare decât intervalul de timp prestabilit (800ms), se afișează mesajele corespunzătoare pentru ca utilizatorul să înțeleagă care buton selectează ce limbă. Acest lucru se face conform variabilei **viewLanguage**, care se incrementează la fiecare trecere a intervalului.

```
  else if(chooseLanguageStage){
    if (key == '1' || key == '2' || key == '3'){
      languageIndex = key - '0' -1;
      chooseLanguageStage = false;
      chooseOptionStage = true;
    } else if (key == '*'){
      viewLanguage = 0;
      chooseLanguageStage = false;
      returnToInitState();
    }
    if (currentMillis - previousMillis >= interval) {
      previousMillis = currentMillis;

      if (viewLanguage == 0){
        //first screen
        lcd.setCursor(0,0);
        lcd.print("CHANGE LANGUAGE");
        lcd.setCursor(0,1);
        lcd.print("1. ENGLISH     ");
        viewLanguage++;
      }
      else if (viewLanguage == 1){
        //second screen
        lcd.setCursor(0,0);
        lcd.print("2. ROMANA      ");
        lcd.setCursor(0,1);
        lcd.print("3. FRANCAIS    ");
        viewLanguage++;
      }
      else if (viewLanguage == 2){
        //third screen - go back
        lcd.setCursor(0,0);
```

```
33          lcd.print("PRESS * TO GO ");
34          lcd.setCursor(0,1);
35          lcd.print("BACK           ");
36          viewLanguage = 0;
37        }
38      }
39    }
```

2.3 Choose Option Stage

Similar cu stadiul anterior, pe ecran va apărea o listă de opțiuni după fiecare interval trecut, iar mesajul se va prelua din messages după formula MSG_NUMBER * LANGUAGE_NO + languageIndex, unde MSG_NUMBER reprezintă al câtelea mesaj se dorește, LANGUAGE_NO este numărul de limbi, iar languageIndex este cel asignat în starea precedentă. Din această stare se poate face trecerea înapoi (tasta *), la stadiul de vizualizare (tasta 1), retragere (tasta 2) și depunere (tasta 3).

2.4 View Balance Stage

Imediat ce se apasă butonul 1 in starea anterioară, pe LCD se vor afișa fondurile și se activează variabila **balanceStage**, după care, când ne aflăm în starea respectivă, putem apăsa pe tasta * pentru întoarcere.

2.5 Withdraw/Deposit Stage

Aceste stări au logică comună pentru partea de introducere a sumei de bani, așa ca se verifică împreună. După ce este preluată, se fac verificări în funcție de stare, și pentru succes se afișează un mesaj corespunzător, altfel unul de eroare. Dacă suntem in stadiul de depunere, se verifică dacă valoarea introdusă este multiplu de 10. Dacă se face retragere, se verifică dacă sunt fonduri suficiente sau dacă este depășită limita prestabilită.

3 Diagrama de circuit

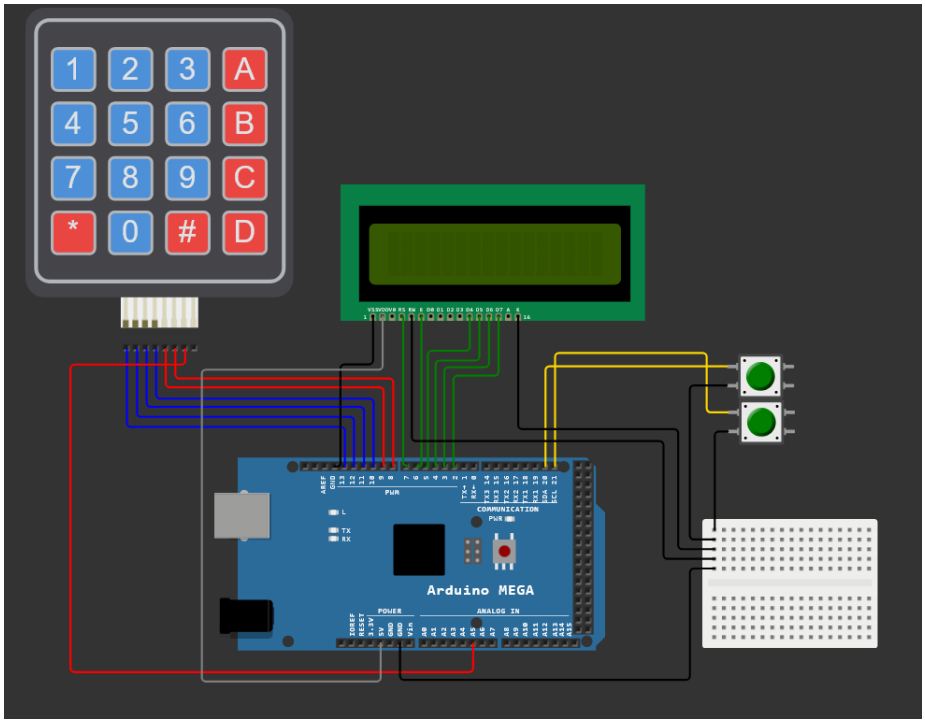

Figura 1: Diagrama de circuit.

Semnificația firelor:

- albastru - randurile keypad-ului (pinii 13, 12, 11, 10)
- roșu - coloanele keypad-ului (pinii 9, 8, A5)
- galben - conectează butoanele la pinii de întrerupere 20 și 21 pentru selectarea contului
- negru - firele ce duc la GND
- gri - firul ce duce spre VCC
- verde - leagă LCD-ul la pinii 7, 6, 5, 4, 3, 2

În realitate voi utiliza un LCD shield, și nu va mai fi nevoie de firele de legătură.

4 Poza proiectului

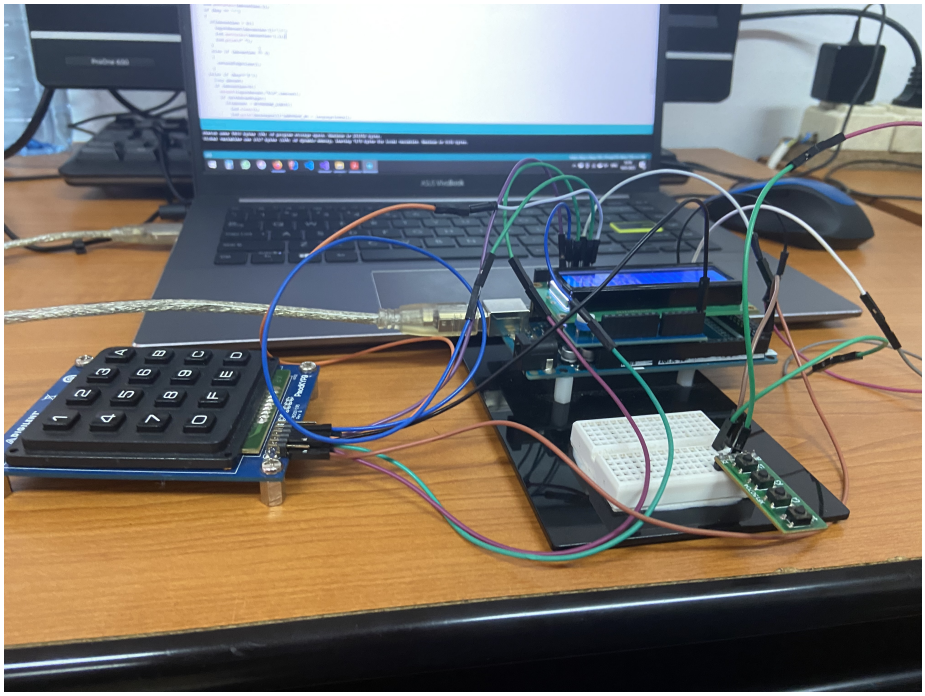

Figura 2: Proiectul realizat cu placa Arduino Mega.

5 Concluzie

În concluzie, am reușit îndeplinirea cerințelor menționate în realizarea acestui proiect și prin intermediul acestuia am dobândit o înțelegere mai bună asupra modului de funcționare al plăcilor Arduino.

Bibliografie

[1] `https://www.circuitbasics.com/how-to-set-up-a-keypad-on-an-arduino/`.

[2] `https://cplusplus.com/reference/cstring/memset/`.