VENDING MACHINE

Introducere

Nume: Dumitrașcu Mihaela
Grupa: 332AB
Proiectul presupune construirea unui automat de vânzare automatizată a băuturilor răcoritoare. Acesta dispune de o interfaţă cu utilizatorul alcătuită dintr-un ecran pe care sunt afişate opţiunile de produse. Selectarea băuturii dorite se face prin intermediul unei tastaturi. Plata va putea fi făcută asemănător plăţii cu cardul contactless.

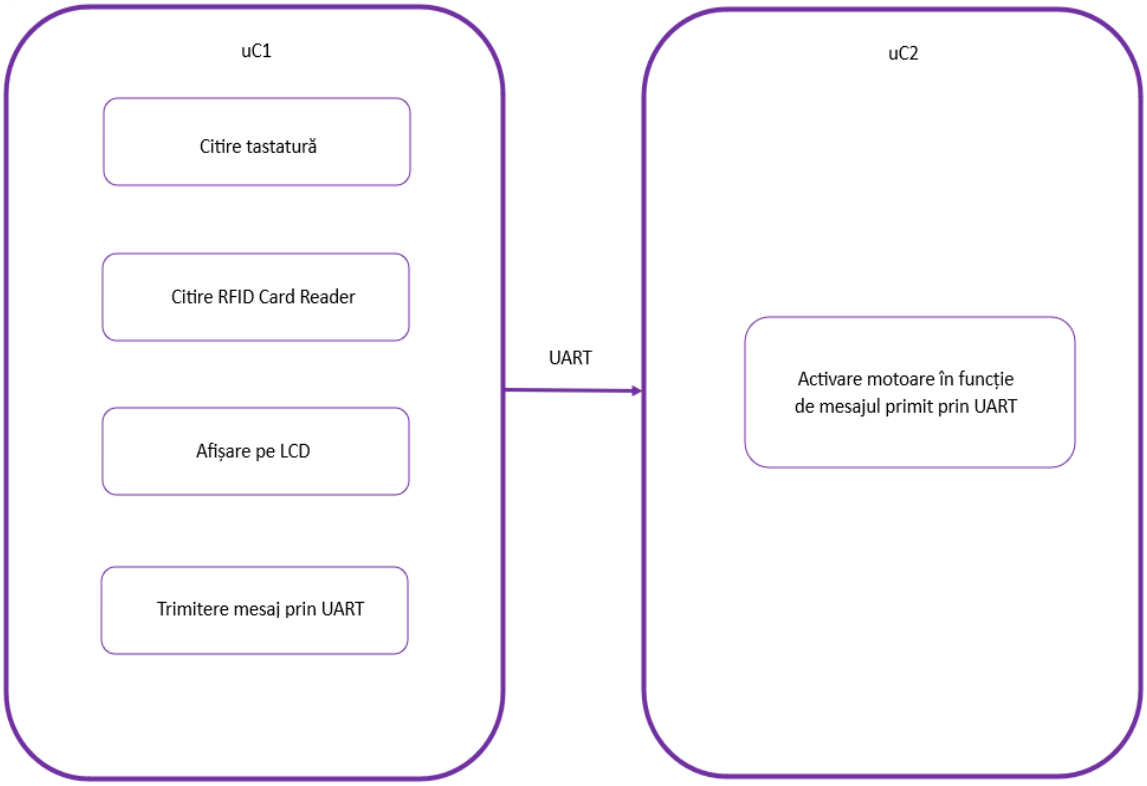
Utilizatorul selectează băutura pe care o doreşte conform informaţiilor de pe ecran, după care efectuează plata cu cardul. În urma aprobării plăţii se activează motorul corespunzător băuturii selectate; astfel, aceasta este împinsă spre tava de la baza automatului. În cazul în care clientul selectează o băutură care nu este în stoc sau are fonduri insuficiente pe card, este anunţat printr-un mesaj afişat pe LCD.

Pentru aprovizionarea automatului se apasă pe tasta 0, apoi se selectează băutura ce urmează a fi adăugată, iar motorul corespunzător se va roti în sens invers. La finalizarea aprovizionării se apasă din nou tasta 0 şi automatul este pregătit pentru un nou utilizator.

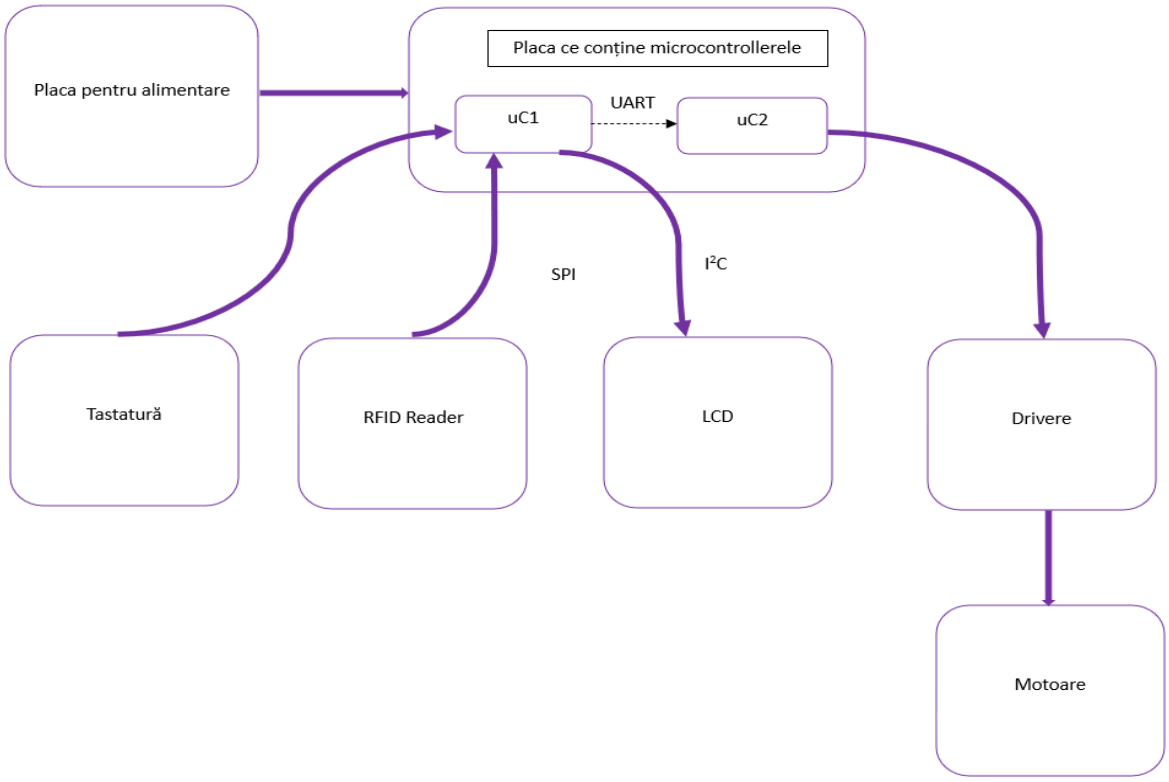
Am ales acest proiect deoarece am căutat o provocare în ceea ce privește partea hardware, motiv pentru care am folosit 2 microcontrollere ce comunică prin UART și am renunțat la placa Arduino Uno, realizând montajul pe o plăcuță de prototipare. În plus, având în vedere numărul mare de automate de acest tip care nu funcționează corect, mi s-a părut un experiment bun să testez dacă pot face unul în care nu se blochează produsele.

Descriere generală

Schemă bloc software



Schemă bloc hardware



Hardware Design

Listă de componente:

- 4 motoare de curent continuu, cu reductor 1:70 care pun în mişcare 4 spirale ce au rolul de a împinge dozele de suc spre tava de unde clientul le poate ridica; de asemenea, când sensul de

rotaţie al motoarelor se schimbă, acestea facilitează aprovizionarea automatului
- o tastatură 4×4 pentru selectarea uneia din cele 4 băuturi, dar şi pentru activarea opţiunii de aprovizionare
- un LCD pe care sunt afişate opţiunile alături de numerele corespunzătoare (voi utiliza protocolul I2C pentru a limita numărul de pini utilizați)
- un RFID reader împreună cu un card pentru simularea plăţii contactless
- 2 microcontrollere ATmega328P (varianta through-hole), întrucât unul singur nu ar fi avut suficienţi pini pentru a suporta toate componentele utilizate
- 2 cristale de cuarţ
- polistiren + plexiglas + plăci de lemn pentru susținere
- alimentator 12V ce se poate conecta la priză pentru alimentarea driverelor şi a microcontrollerelor
- 2 drivere pentru controlul motoarelor
- un level shifter pentru tastatură
- componente adiţionale de protecţie: rezistenţe, diode, siguranţe (250mA, 1A)
- stabilizatoare de tensiune de 5V, 3V3
- sursă în comutație 12V → 7V
- amplificator operațional în montaj de comparator

Schema electrică
schema_electrica_vending_machine.pdf

Software Design

Medii de dezvoltare:

- Arduino IDE - pentru scrierea codului pentru ambele microcontrollere
- Microchip Studio - pentru setarea fuse bits
- Kicad - pentru schema electrică

Biblioteci utilizate:

- LiquidCrystal_I2C.h - pentru LCD-ul care comunică prin I^2C
- MFRC522.h (include SPI.h)- pentru RFID (Radio-Frequency Identification) Card Reader care comunică prin SPI
- Keypad.h - pentru tastatura

Surse şi funcţii implementate:
La unul dintre microcontrollere sunt conectate LCD-ul, RFID Reader-ul și tastatura. Logica este implementată într-un switch case în funcție de tasta apăsată, deoarece fiecare determină altă acțiune în continuare.

```
char key = keypad.getKey(); // stocheaza caracterul apasat pe tastatura
if (key){
  switch(key){ // in functie de tasta apasata, operatiile efectuate de tonomat sunt diferite
  case '0': refillCounter++;
            if(refillCounter % 2 == 0) // la o apasare impara a lui 0 se cere refill, la una para se dezactiveaza refill
              refill = 0;
            else
              refill = 1;
            LCDPrintChoices();
            break;
  case '1': chooseAction(START_M1, REVERSE_M1, 1, 1);
            break;
  case '2': chooseAction(START_M2, REVERSE_M2, 2, 2);
            break;
  case '3': chooseAction(START_M3, REVERSE_M3, 3, 3);
            break;
  case '4': chooseAction(START_M4, REVERSE_M4, 4, 4);
            break;
  default:  lcd.clear();
            lcd.setCursor(1, 1);
            lcd.print("Invalid option!");
            delay(1000);
            LCDPrintChoices();
            break;
  }
}
```

- Mesajele ce sunt afișate pe LCD sunt structurate în funcții, apelându-se cea potrivită pentru fiecare scenariu de funcționare în parte (void LCDPrintChoices(), void LCDPrintChoiceError(), void LCDReset(), void LCDPrintNotEnoughMoney()).

```
void LCDPrintChoices(){
  if(refill != 0){ // functie ce afiseaza pe LCD un mesaj care indica utilizatorului sa aleaga bautura pentru care doreste sa faca refill
    lcd.clear();
    lcd.setCursor(0, 1);
    lcd.print("What drink do you");
    lcd.setCursor(0, 2);
    lcd.print("want to refill?");
  }
  else{ // functie ce afiseaza pe LCD un mesaj care indica utilizatorului ce tasta sa apese in funtie de operatia dorita
    lcd.clear();
    lcd.setCursor(1, 0);
    lcd.print("Choose your drink!");
    lcd.setCursor(2, 2);
    lcd.print("Or press 0 for");
    lcd.setCursor(7, 3);
    lcd.print("refill");
  }
}
```

Instrucțiunile afișate sunt diferite în funcție de scenariul de funcționare (de exemplu, refill activat sau nu ^)

- Citirea și verificarea cardului prin modulul RFID se fac într-o funcție separată (bool validateCard()).

```
bool validateCard()
{
  startWait = millis();
  /* inspirat din https://www.electronicshub.org/arduino-rc522-rfid-module-based-access-control-system/#Code */
  while(true){
    if (!cardReader.PICC_IsNewCardPresent() && (millis() - startWait) > 10000) { // daca nu se apropie niciun card in 10 secunde, automatul revine
      lcd.clear();                                                                // in starea initiala
      lcd.setCursor(0, 1);
      lcd.print("Timeout");
      delay(2000);
      LCDPrintChoices();
      return false;
    }

    if(cardReader.PICC_IsNewCardPresent()){ // daca este identificat un card
      if (!cardReader.PICC_ReadCardSerial()) // se verifica daca se poate citi cardul
      {
        return false;
      }
      readCard = "";
      for (int i = 0; i < cardReader.uid.size; i++) { // se formeaza un String ce contine ID-ul cardului
        readCard += String(cardReader.uid.uidByte[i], HEX);
      }
      readCard.toUpperCase();
      cardReader.PICC_HaltA(); // se opreste citirea cardului
```

Pentru validarea cardului ^ se verifică dacă a fost apropiat în maxim 10 secunde, dacă se poate citi.

```
      if(readCard.equals(ID)) // daca ID-ul identificat este bun, se accepta plata, altfel se respinge
      {
        return true;
      }
      else
      {
        return false;
      }
```

Dacă verificările menționate anterior sunt trecute se citește ID-ul și se compară cu singurul ID care permite continuarea funcționării ^.

- Trimiterea informațiilor despre motorul care trebuie să fie pus în mișcare și sensul lui de rotație, cât și actualizarea datelor precum stocul sau soldul de pe card se realizează printr-o funcție dedicată (void chooseAction(byte messageVend, byte messageRefill, int s, int p))

```
void chooseAction(byte messageVend, byte messageRefill, int s, int p){ // functie ce determina mesajul ce va fi trimis pe seriala si actualizarea stocurilor
  if(sold < price[p-1]){ // se trateaza cazul in care clientul are fonduri insuficiente
    LCDPrintNotEnoughMoney();
    delay(2000);
    LCDPrintChoices();
    return;
  }
  else
  {
    if(stock[s-1] == 0 && refill == 0) // se trateaza cazul in care nu exista produsul in stoc si nu se doreste refill
    {
      LCDReset();
    }
    else
    {
      if(refill) // daca se activeaza refill-ul
      {
        stock[s-1]++; // se incrementeaza stocul produsului ales
        serialMessage = messageRefill; // catre celalalt microcontroller se va transmite comanda de refill
      }
      else if(pay(p)) // daca s-a putut efectua plata
          {
            stock[s-1]--; // se decrementeaza stocul produsului ales
            serialMessage = messageVend; // catre celalalt microcontroller se va transmite comanda de livrare
          }
          else return;

      Serial.write(serialMessage); // se trimite mesajul catre celalalt microcontroller prin comunicatie seriala
    }
  }
}
```

În funcție de scenariul de funcționare, selectat anterior prin apăsarea sau nu a tastei 0, fie se transmite mesaj de refill, fie de vend către celălalt microcontroller, actualizându-se stocul de fiecare dată.

```
bool pay(int p) // functie care determina daca plata a fost acceptata sau nu
{
    lcd.clear();
    lcd.setCursor(6,1);
    lcd.print("Tap card");
    if(validateCard()){  // daca este valid cardul
      sold = sold - price[p-1]; // actualizare sold
      lcd.clear();
      lcd.setCursor(1,1);
      lcd.print("Payment accepted");
      delay(2000);
      lcd.clear();
      lcd.setCursor(0,1);
      lcd.print("Remaining balance: "); // afisare sold actualizat
      lcd.setCursor(1,2);
      lcd.print(sold);
      delay(2000);
      LCDPrintChoices(); // se revine la starea initiala
      return true;
    }
    else{
      lcd.clear();
      lcd.setCursor(1,1);
      lcd.print("Payment declined");
      delay(2000);
      LCDPrintChoices();
      return false;
    }
}
```

Dacă se livrează băutura se verifică soldul și dacă plata a putut fi efectuată ^.
Pentru microcontroller-ul ce controlează motoarele, întreaga logică este bazată pe un switch case în funcție de mesajul primit prin UART de la celălalt microcontroller (prin funcția chooseAction menționată mai sus).

```
if (Serial.available() > 0) { // daca microcontroller-ul 1 a trimis un mesaj pe seriala
  command = Serial.read(); // se citesc
  switch(command){ // in functie de mesajul receptionat se alege sensul de rotatie (polaritatea bornelor motorului)
    case START_M1:  defineRotation(M1_plus, M1_minus);
                    break;
    case REVERSE_M1:  defineRotation(M1_minus, M1_plus);
                      break;
    case START_M2:  defineRotation(M2_plus, M2_minus);
                    break;
    case REVERSE_M2:  defineRotation(M2_minus, M2_plus);
                      break;
    case START_M3:  defineRotation(M3_plus, M3_minus);
                    break;
    case REVERSE_M3:  defineRotation(M3_minus, M3_plus);
                      break;
    case START_M4:  defineRotation(M4_plus, M4_minus);
                    break;
    case REVERSE_M4:  defineRotation(M4_minus, M4_plus);
                      break;
    default:  digitalWrite(M1_plus, LOW);
              digitalWrite(M2_plus, LOW);
              digitalWrite(M3_plus, LOW);
              digitalWrite(M4_plus, LOW);
              digitalWrite(M1_minus, LOW);
              digitalWrite(M2_minus, LOW);
              digitalWrite(M3_minus, LOW);
              digitalWrite(M4_minus, LOW);
              break;
  }
```

Se verifică dacă pe serială s-a primit un mesaj, iar dacă da, în funcție de acesta este apelată funcția void defineRotation(byte highTerminal, byte lowTerminal) cu anumiți parametri astfel încât să se activeze motorul potrivit în sensul de rotație corespunzător.

```
void defineRotation(byte highTerminal, byte lowTerminal){
  digitalWrite(highTerminal, HIGH);
  digitalWrite(lowTerminal, LOW);
}
```

- Tot aici am implementat rutinele de tratare ale întreruperilor provenite de la buton și de la senzorul (fotorezistența) care semnalează căderea dozei.

```
ISR(INT1_vect){
    PORTC &= ~(1 << PC0); // digitalWrite(M1_plus, LOW);
    PORTC &= ~(1 << PC2); // digitalWrite(M2_plus, LOW);
    PORTD &= ~(1 << PD4); // digitalWrite(M3_plus, LOW);
    PORTD &= ~(1 << PD6); // digitalWrite(M4_plus, LOW);
    PORTC &= ~(1 << PC1); // digitalWrite(M1_minus, LOW);
    PORTC &= ~(1 << PC3); // digitalWrite(M2_minus, LOW);
    PORTD &= ~(1 << PD5); // digitalWrite(M3_minus, LOW);
    PORTD &= ~(1 << PD7); // digitalWrite(M4_minus, LOW);
}
```

Rezultate Obținute

Rezultatele obținute sunt foarte bune, întrucât am reușit să implementez toate funcționalitățile enunțate inițial și, în urma testelor făcute, pot spune că cea mai mare provocare, anume să nu se mai blocheze produsele în interior mai ales după ce plata a fost făcută, a fost depășită.

Concluzii

Automatul funcționează conform target-urilor setate inițial. Totuși, există și câteva îmbunătățiri ulterioare pe care le-aș putea aduce, precum:

- constuirea unei platforme care se deplasează pe 2 axe cu rolul de a ajunge în dreptul produsului selectat pentru a-l aduce la tava de colectare, evitând astfel ca acesta să rămână blocat în produse care se află mai jos
- aprinderea luminii care cade pe fotorezistența ce determină oprirea motoarelor doar după efectuarea plății pentru a reduce din consumul energetic
- implementarea opțiunii de încărcare cu puncte a unui card de fidelitate și achitarea produselor cu acestea

Download

resurse_dumitrascumihaela_332ab.zip

Bibliografie/Resurse

Surse de inspirație:
- https://www.youtube.com/watch?v=BHQBsswUeT0, accesat la 07.03.2023
- https://www.youtube.com/watch?v=Ou-VgY3DKlQ, accesat la 07.03.2023
- https://www.youtube.com/watch?v=-gdm71P1k9c, accesat la 07.03.2023
- https://www.youtube.com/watch?v=7gW1hmYqdwo, accesat la 07.03.2023

Resurse software:
- https://arduinogetstarted.com/tutorials/arduino-keypad, accesat la 20.04.2023
- https://www.arduino.cc/reference/en/language/functions/communication/serial/read/, accesat la 25.04.2023
- https://www.electronicshub.org/arduino-rc522-rfid-module-based-access-control-system/#Code, accesat la 27.04.2023
- https://forum.arduino.cc/t/rfid-reader-mfrc-522-uid-vs-picc/261212, accesat la 30.04.2023

Resurse hardware:
- https://www.circuito.io/blog/arduino-uno-pinout/, accesat la 15.04.2023
- http://electronoobs.com/eng_arduino_tut31_sch3.php, accesat la 18.04.2023

Export to PDF