ATMega328P-Based Banking System Project Report

Author - João Moreira

Project Overview

The project implements a basic banking system using an ATMega328P microcontroller. The system provides core banking functionalities through a user interface consisting of an LCD display, a 4x3 numeric keypad, and four control buttons.

System Assumptions

Hardware Components

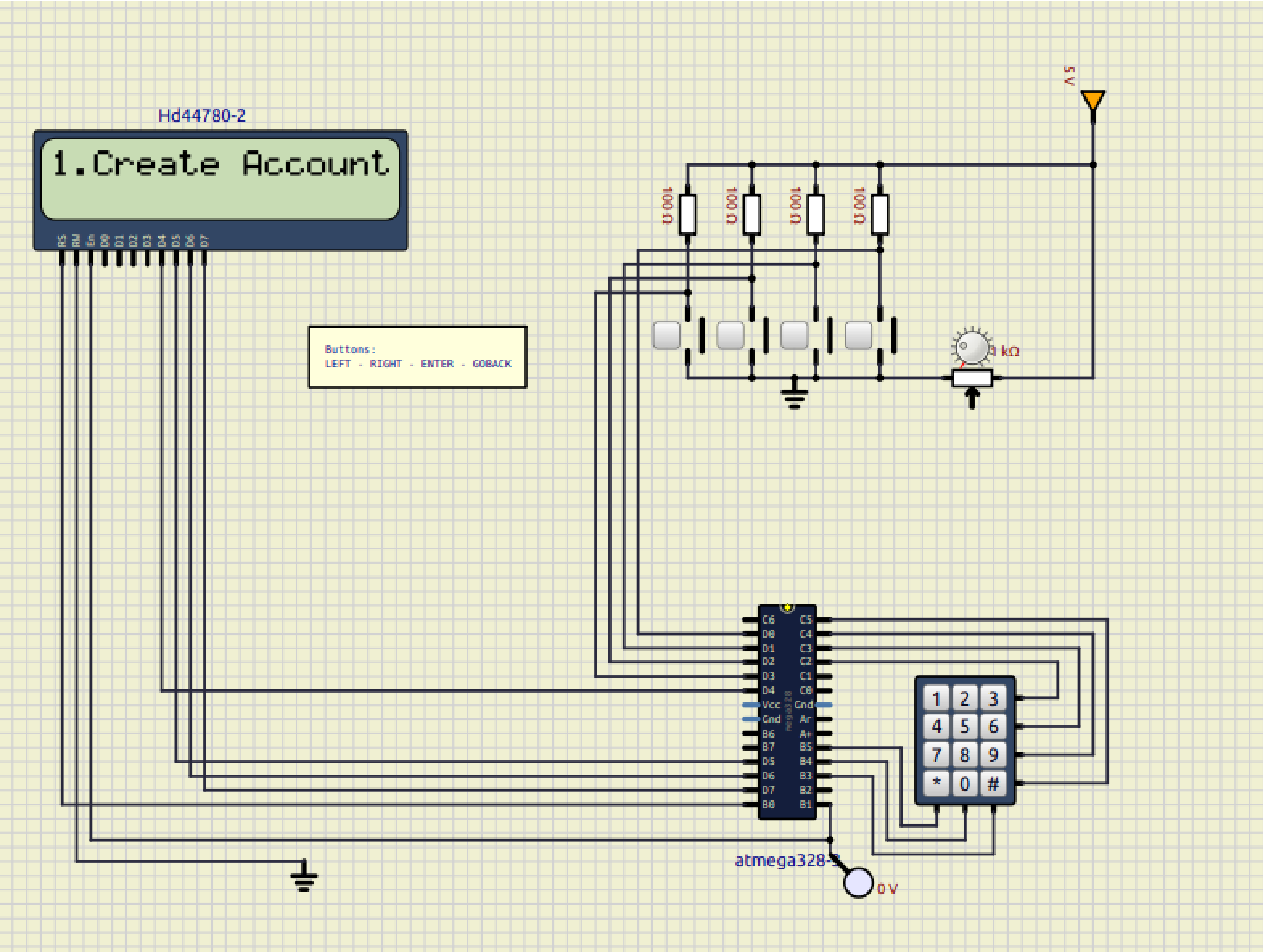
- ATMega328P microcontroller
- LCD display (16x2 characters)
- 4x3 numeric keypad
- Four control buttons (LEFT, RIGHT, ENTER, BACK)
- Pull-up resistors for buttons (100kΩ)

Software Constraints

- Maximum of 10 user accounts
- Account ID: 4 digits
- PIN: 4 digits
- Balance stored as unsigned 32-bit integer (maximum as 9 digits)
- Menu-driven interface with 4 main options

System Architecture

Hardware Schematic


Hd44780-2
1.Create Account
RS RW En D0 D1 D2 D3 D4 D5 D6 D7
Buttons:
LEFT - RIGHT - ENTER - GOBACK
5 V
100 Ω
100 Ω
100 Ω
100 Ω
kΩ
C6 D0 D1 D2 D3 D4 Vcc Gnd B6 B7 D5 D6 D7 B0
C5 C4 C3 C2 C1 C0 Gnd Ar A+ B5 B4 B3 B2 B1
1 2 3
4 5 6
7 8 9
* 0 #
atmega328-1
0 V

The provided schematic shows the following connections:

- HD44780 LCD connected to digital pins
- 4x3 matrix keypad interface
- Four buttons with pull-up resistors
- Power supply connection (5V)

Software Architecture

The system is organized into several key components:

1. Core Components:
   - Main control loop (`main.c`)
   - LCD interface (`HD44780.hpp`)
   - Keypad (`keypad.hpp`)
   - Button (`buttons.hpp`)
2. Data Structures:

```
struct Account {
    char id[ID_LENGTH + 1];
    char pin[PIN_LENGTH + 1];
    uint32_t balance;
};
```

3. Main Features:
    - Account creation
    - Balance checking
    - Money deposit
    - Money withdrawal

Custom Libraries

Buttons

The buttons library provides a simple way for handling four push buttons connected to Port D (pins 0-3).

1. Hardware Configuration:
    - Four independent push buttons
    - Connected to PORTD (PD0-PD3)
    - Functions:
        - BUTTON_BACK (PD0): Delete/return
        - BUTTON_ENTER (PD1): Confirm selections
        - BUTTON_RIGHT (PD2): Menu navigation
        - BUTTON_LEFT (PD3): Menu navigation

2. Key Functions:

```
void buttons_init() {
     // Configure pins 0-3 on Port D for button inputs

     // Set pins as inputs by clearing bits 0-3 in DDRD (Data
Direction Register)
     DDRD &= ~((1 << PD0) | (1 << PD1) | (1 << PD2) | (1 << PD3));

     // Enable internal pull-up resistors on these pins by setting
bits in PORTD
     // This ensures pins read high when buttons are not pressed
     PORTD |= (1 << PD0) | (1 << PD1) | (1 << PD2) | (1 << PD3);
 }
```

    - Configures pins 0-3 on Port D as inputs
    - Enables internal pull-up resistors
    - Called once during system initialization

```
uint8_t read_buttons() {
     // Check status of 4 buttons connected to Port D pins 0-3
     // Returns the index (0-3) of the first pressed button, or 255 if no button is pressed
     for(uint8_t i = 0; i < 4; i++) {
         // Check if button i is pressed by checking if its bit in PIND is 0
         // PIND & (1 << i) creates a mask to isolate the button's pin
         // Buttons are active low, meaning a pressed button reads as 0
         if(!(PIND & (1 << i))) {
             _delay_ms(50);
             return i;
         }
     }
     return 255;
 }
```

  - Polls all four buttons
  - Returns index (0-3) of pressed button
  - Returns 255 if no button is pressed
  - Includes 50ms debounce delay

Keypad Interface

The keypad library manages a 4x3 matrix keypad for numeric input.

- 4x3 matrix keypad configuration

- Hardware Configuration:

  - Rows connected to PORTC (PC2-PC5)
  - Columns connected to PORTB (PB3-PB5)
  - Matrix layout:

```
const char keypad[4][3] = {
    {'1', '2', '3'},    // PC2
    {'4', '5', '6'},    // PC3
    {'7', '8', '9'},    // PC4
    {'*', '0', '#'}     // PC5
//   PB5  PB4  PB3
};
```

2. Key Functions:

```
void keypad_init() {
     // Set row pins as outputs
     DDRC |= (1 << PC5) | (1 << PC4) | (1 << PC3) | (1 << PC2);
     // Set column pins as inputs
     DDRB &= ~((1 << PB5) | (1 << PB4) | (1 << PB3));

     // Enable internal pull-up resistors for column pins
     PORTB |= (1 << PB5) | (1 << PB4) | (1 << PB3);
 }
```

- Configures row pins as outputs (PORTC)
- Configures column pins as inputs (PORTB)
- Enables pull-up resistors on column pins

```
char scan_keypad() {
     uint8_t row;
     for(row = 0; row < 4; row++) {
         // Set all rows high
         PORTC |= (1 << PC5) | (1 << PC4) | (1 << PC3) | (1 << PC2);
         // Set current row low (active)
         // PC2 is top row (row 0), PC5 is bottom row (row 3)
         PORTC &= ~(1 << (PC2 + row));
         _delay_us(10);

         // Check each column for the current row
         if(!(PINB & (1 << PB5))) return keypad[row][0];  // Left
column
         if(!(PINB & (1 << PB4))) return keypad[row][1];  // Middle
column
         if(!(PINB & (1 << PB3))) return keypad[row][2];  // Right
column
     }
     return 0;
 }
```

3. Implementation Details:

    - Uses row-scanning technique:
        - For each row:
        - All rows are first set HIGH (inactive)
        - The current row is set LOW (active)
        - Each column is checked for a LOW state
        - If a LOW is detected, the corresponding key is returned
    - Direct port manipulation for performance

Implementation Details

User Interface

The system implements a menu-driven interface with the following navigation:

- LEFT/RIGHT buttons: Navigate through menu options
- ENTER: Select current option
- BACK: Delete input during data entry

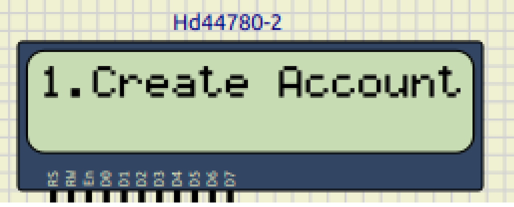
Menu Structure

1. Create Account
   - Enter 4-digit ID
   - Enter 4-digit PIN
   - Account created with 0 balance
2. Check Balance
   - Enter ID and PIN
   - Display current balance
3. Deposit Money
   - Enter ID and PIN
   - Enter amount
   - Update balance
4. Withdraw Money
   - Enter ID and PIN
   - Enter amount
   - Check sufficient funds
   - Update balance if possible

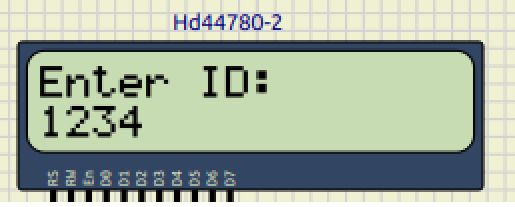
System Operation

Account Creation

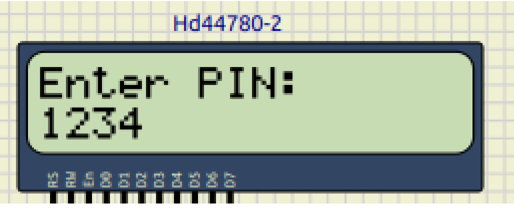
1. System checks for available account slots
2. User enters 4-digit ID
3. System verifies ID uniqueness
4. User enters 4-digit PIN
5. Account created with 0 balance



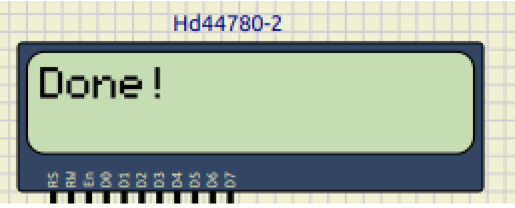
```
1.Create Account
```



```
Enter ID:
1234
```



```
Enter PIN:
1234
```

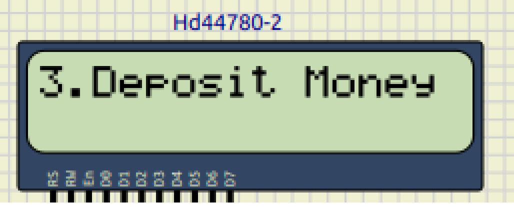


```
Done!
```

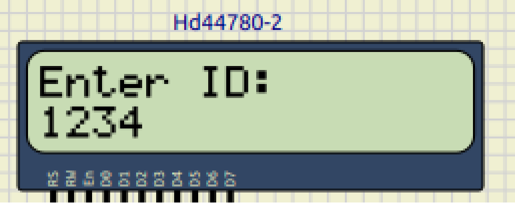
Transaction Flow (Deposit/Withdrawal)

1. User selects operation from menu
2. Enters account credentials
3. System validates credentials
4. User enters transaction amount (for deposit/withdrawal)
5. System performs operation and displays result

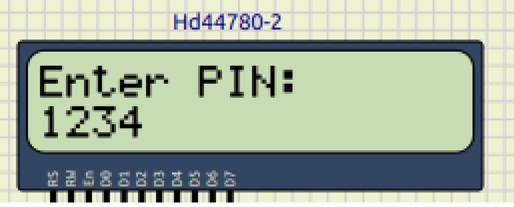
Deposit



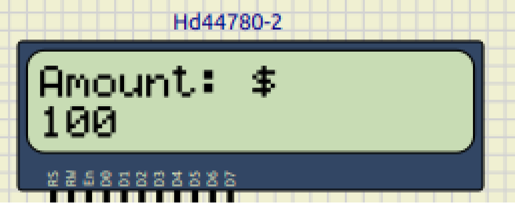
```
3.Deposit Money
```



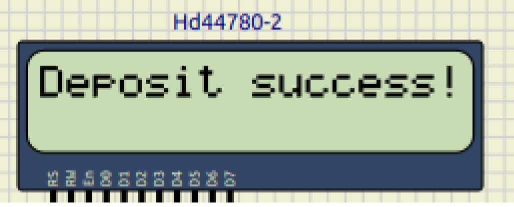
```
Enter ID:
1234
```



```
Enter PIN:
1234
```

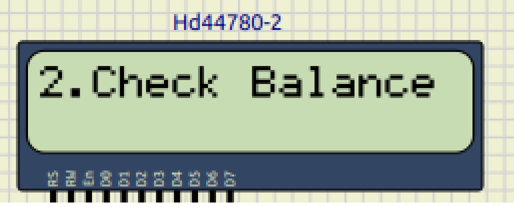


```
Amount: $
100
```

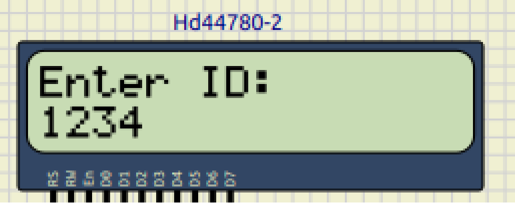


```
Deposit success!
```

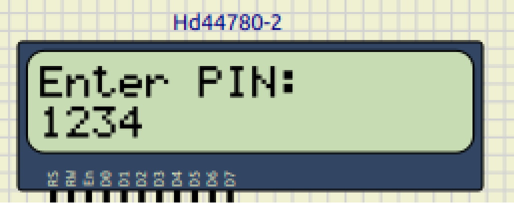
Check Balance



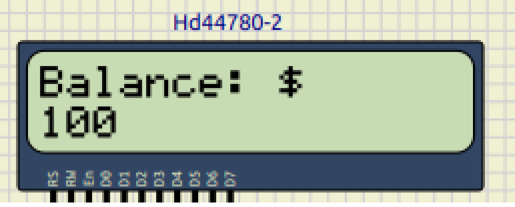
```
2.Check Balance
```



```
Enter ID:
1234
```

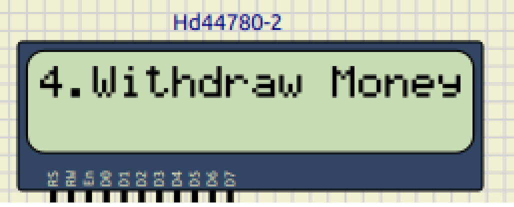


```
Enter PIN:
1234
```

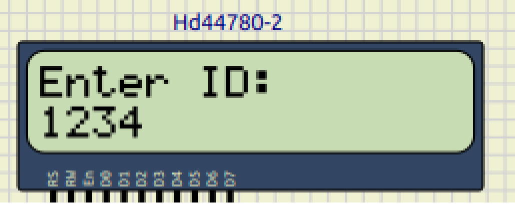


```
Balance: $
100
```

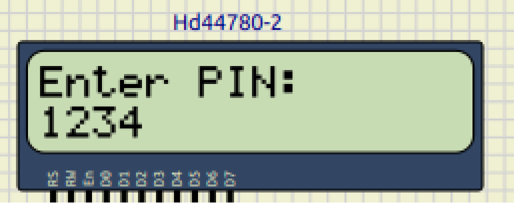
Withdrawal



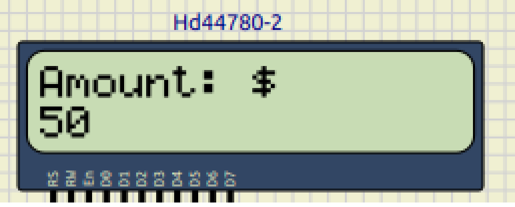
```
4.Withdraw Money
```



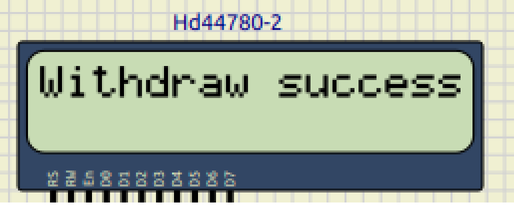
```
Enter ID:
1234
```



```
Enter PIN:
1234
```



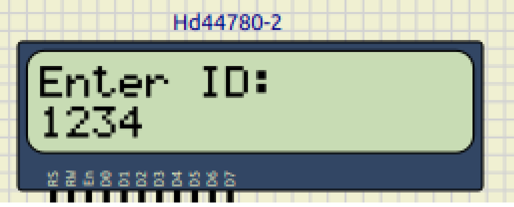
```
Amount: $
50
```



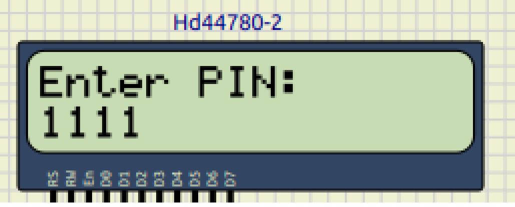
```
Withdraw success
```

Error Handling

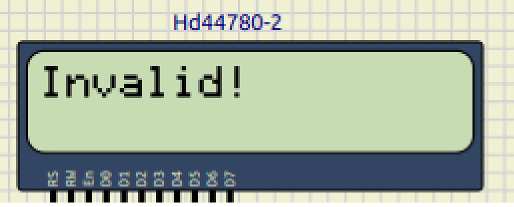
Invalid login



```
Enter ID:
1234
```

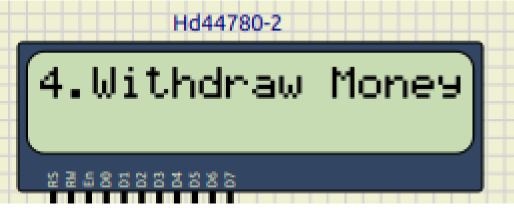


```
Enter PIN:
1111
```

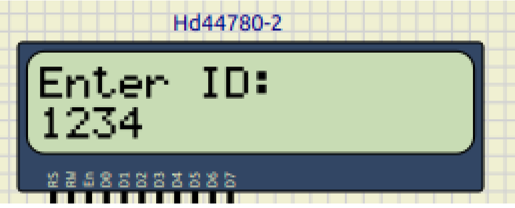


```
Invalid!
```

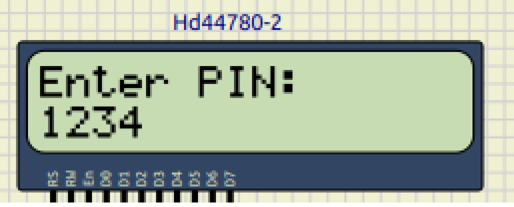
Insufficient funds



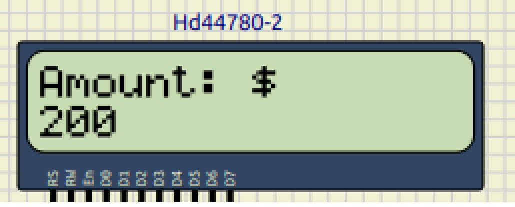
```
4.Withdraw Money
```



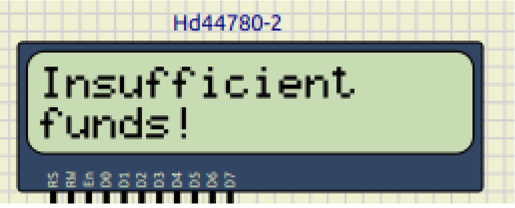
```
Enter ID:
1234
```



```
Enter PIN:
1234
```



```
Amount: $
200
```



```
Insufficient
funds!
```

Github Repo

The code for this project is available at:

- https://github.com/Moreira-26/Arduino