Project Report
Programming in C-Lab
Title: ATM System Using Arduino, Keypad, LCD, Mini Servo, and LEDs

Submitted By

VINAYAK BANSAL-24070122233

VEDANT SHARMA-24070122230

UTKARSH SINGH-24070122224

KUSH PATEL-24070122276

AMOGH SINGLA-24070122239

Under Guidance of

DR. RUPALI GANGARDE

SYMBIOSIS INSTITUTE OF TECHNOLOGY

(A CONSTITUENT OF SYMBIOSIS INTERNATIONAL UNIVERSITY)

Pune

2024-25

Introduction

This project simulates an Automated Teller Machine (ATM) using an Arduino microcontroller. It integrates an LCD, a 4x4 keypad, LEDs, a servo motor, and a buzzer to mimic the core functionalities of an ATM. The user can insert a virtual card (button press), authenticate with an account number and PIN, and perform actions such as withdrawing, depositing, checking balance, or creating a new account.

Objective

To build a mini-prototype of an ATM machine using embedded system components and Arduino. The system should allow:

- Account authentication.
- Cash withdrawal and deposit simulation.
- Real-time balance checking.
- Account creation with secure PIN setup.
- Visual and audio feedback for different operations.

Architecture & Flowchart

Architecture Overview:

- **Arduino UNO**: Controls the system logic.
- **16x2 I2C LCD**: Displays messages and menus.
- **Keypad (4x4)**: Input interface for account number, PIN, and operations.
- **Servo Motor**: Simulates cash dispensing.
- **Buzzer**: Indicates success or error.
- **LEDs**: Green for success, red for error, yellow for processing.
- **Push Buttons**: Simulate card insertion and cancellation.

Demo Accounts for Testing:

1. **Account 1234 / PIN 1111** → Withdraw $100 → New Balance: $8900.
2. **Account 5678 / PIN 2222** → Deposit $500 → New Balance: $9000.
3. **New Account** → PIN ****→ Account Number: 1002 → Balance: $0.

Flowchart:

1. Display welcome screen.
2. Wait for card button press.
3. Prompt for account number and PIN.
4. Authenticate:
    - If correct, proceed to main menu.
    - If incorrect, display error.
5. Menu options:
    - Withdraw: Validate and dispense.
    - Deposit: Add amount to balance.
    - Check balance: Show current balance.
    - Create account: Setup new account if slots available.
6. Logout via cancel button.

Code

```
#include <LiquidCrystal_I2C.h>
#include <Keypad.h>
#include <Servo.h>

// LCD setup (I2C)
LiquidCrystal_I2C lcd(0x27, 16, 2); // Set address to 0x27 for a 16x2 display

// Keypad setup
const byte ROWS = 4;
const byte COLS = 4;
char keys[ROWS][COLS] = {
  {'1', '2', '3', 'A'},
  {'4', '5', '6', 'B'},
  {'7', '8', '9', 'C'},
  {'*', '0', '#', 'D'}
};
byte rowPins[ROWS] = {9, 8, 7, 6}; // Connect to the row pinouts of the keypad
byte colPins[COLS] = {5, 4, 3, 2}; // Connect to the column pinouts of the keypad
Keypad keypad = Keypad(makeKeymap(keys), rowPins, colPins, ROWS, COLS);

// Define pins
#define SERVO_PIN 10
#define BUZZER_PIN 11
#define GREEN_LED 12  // Success LED
#define RED_LED 13    // Error LED
```

```
#define YELLOW_LED A0 // Processing LED
#define CARD_BUTTON A1
#define CANCEL_BUTTON A2

// Servo setup
Servo cashDispenser;

// Bank account structure
struct BankAccount {
  int accNumber;
  int pin;
  float balance;
  bool isActive;
};

// Bank account database (maximum 5 accounts)
#define MAX_ACCOUNTS 5
struct BankAccount accounts[MAX_ACCOUNTS];
int currentAccounts = 2; // Start with 2 pre-defined accounts

// System state variables
int currentAccountIndex = -1;
bool isAuthenticated = false;
char inputBuffer[5]; // For PIN input
int bufferIndex = 0;

// Function prototypes
void displayWelcomeScreen();
bool authenticateUser();
void displayMainMenu();
void createAccount();
void withdrawCash();
void depositCash();
void checkBalance();
void successBeep();
void errorBeep();
void dispenseCash();
int getAccountIndex(int accNumber);
void resetSystem();

void setup() {
  // Initialize serial communication
  Serial.begin(9600);

  // Initialize LCD
  lcd.init();
  lcd.backlight();

  // Initialize outputs
  pinMode(BUZZER_PIN, OUTPUT);
```

```
  pinMode(GREEN_LED, OUTPUT);
  pinMode(RED_LED, OUTPUT);
  pinMode(YELLOW_LED, OUTPUT);

  // Initialize inputs
  pinMode(CARD_BUTTON, INPUT_PULLUP);
  pinMode(CANCEL_BUTTON, INPUT_PULLUP);

  // Initialize servo
  cashDispenser.attach(SERVO_PIN);
  cashDispenser.write(0); // Initial position

  // Create sample accounts
  accounts[0].accNumber = 1234;
  accounts[0].pin = 1111;
  accounts[0].balance = 9000.0;//change
  accounts[0].isActive = true;

  accounts[1].accNumber = 5678;
  accounts[1].pin = 2222;
  accounts[1].balance = 8500.0;//change
  accounts[1].isActive = true;

  // Display welcome message
  displayWelcomeScreen();

  Serial.println("ATM System Initialized");
  Serial.println("Pre-defined accounts:");
  Serial.println("Account #1: Number=1234, PIN=1111, Balance=$1000");
  Serial.println("Account #2: Number=5678, PIN=2222, Balance=$2500");
}
void loop() {
  // Check if cancel button is pressed
  if (digitalRead(CANCEL_BUTTON) == LOW) {
    resetSystem();
    delay(500); // Debounce
    return;
  }

  // Wait for card insertion (button press)
  if (!isAuthenticated && digitalRead(CARD_BUTTON) == LOW) {
    delay(200); // Debounce
    lcd.clear();
    lcd.print("Card inserted");
    lcd.setCursor(0, 1);
    lcd.print("Processing...");

    digitalWrite(YELLOW_LED, HIGH); // Processing
    delay(1000);
```

```
    isAuthenticated = authenticateUser();

    if (isAuthenticated) {
      digitalWrite(YELLOW_LED, LOW);
      digitalWrite(GREEN_LED, HIGH);
      successBeep();
      delay(1000);
      digitalWrite(GREEN_LED, LOW);
      displayMainMenu();
    } else {
      digitalWrite(YELLOW_LED, LOW);
      digitalWrite(RED_LED, HIGH);
      errorBeep();
      delay(1000);
      digitalWrite(RED_LED, LOW);
      displayWelcomeScreen();
    }
  }

  // Handle menu selection if authenticated
  if (isAuthenticated) {
    char key = keypad.getKey();

    if (key) {
      digitalWrite(YELLOW_LED, HIGH); // Processing

      switch (key) {
        case '1': // Withdraw
          withdrawCash();
          break;
        case '2': // Deposit
          depositCash();
          break;
        case '3': // Check balance
          checkBalance();
          break;
        case '4': // Create account
          createAccount();
          break;
        case 'D': // Cancel/Logout
          resetSystem();
          break;
        default:
          lcd.clear();
          lcd.print("Invalid option");
          delay(1000);
          displayMainMenu();
          break;
      }
```

```
      digitalWrite(YELLOW_LED, LOW);
    }
  }
}
void displayWelcomeScreen() {
  lcd.clear();
  lcd.print("Welcome to Bank");
  lcd.setCursor(0, 1);
  lcd.print("Press button");

  isAuthenticated = false;
  currentAccountIndex = -1;
}

bool authenticateUser() {
  // Ask for account number
  lcd.clear();
  lcd.print("Account Number:");
  lcd.setCursor(0, 1);

  // Reset buffer
  memset(inputBuffer, 0, sizeof(inputBuffer));
  bufferIndex = 0;

  // Get account number input
  int accNumber = 0;
  while (bufferIndex < 4) {
    char key = keypad.getKey();
    if (key >= '0' && key <= '9') {
      inputBuffer[bufferIndex++] = key;
      lcd.print(key);

      // Convert to integer when done
      if (bufferIndex == 4) {
        accNumber = atoi(inputBuffer);
      }
    }
  }

  delay(1000);

  // Find the account
  currentAccountIndex = getAccountIndex(accNumber);
  if (currentAccountIndex == -1) {
    lcd.clear();
    lcd.print("Account not found");
    delay(2000);
    return false;
  }
```

```
  // Ask for PIN
  lcd.clear();
  lcd.print("Enter PIN:");
  lcd.setCursor(0, 1);

  // Reset buffer
  memset(inputBuffer, 0, sizeof(inputBuffer));
  bufferIndex = 0;

  // Get PIN input
  while (bufferIndex < 4) {
    char key = keypad.getKey();
    if (key >= '0' && key <= '9') {
      inputBuffer[bufferIndex++] = key;
      lcd.print("*"); // Show asterisk for security
    }
  }

  int enteredPin = atoi(inputBuffer);

  // Validate PIN
  if (enteredPin == accounts[currentAccountIndex].pin) {
    return true;
  } else {
    lcd.clear();
    lcd.print("Invalid PIN");
    delay(2000);
    return false;
  }
}
void displayMainMenu() {
  lcd.clear();
  lcd.print("1:Withdraw 2:Dep");
  lcd.setCursor(0, 1);
  lcd.print("3:Balance 4:New");
}

void createAccount() {
  // Check if we've reached maximum accounts
  if (currentAccounts >= MAX_ACCOUNTS) {
    lcd.clear();
    lcd.print("Maximum accounts");
    lcd.setCursor(0, 1);
    lcd.print("reached");
    delay(2000);
    displayMainMenu();
    return;
  }

  // Generate new account number (simple implementation)
```

```
int newAccNumber = 1000 + currentAccounts;

// Ask for new PIN
lcd.clear();
lcd.print("Create new PIN:");
lcd.setCursor(0, 1);

// Reset buffer
memset(inputBuffer, 0, sizeof(inputBuffer));
bufferIndex = 0;

// Get PIN input
while (bufferIndex < 4) {
  char key = keypad.getKey();
  if (key >= '0' && key <= '9') {
    inputBuffer[bufferIndex++] = key;
    lcd.print("*"); // Show asterisk for security
  }
}

int newPin = atoi(inputBuffer);

// Create the new account
accounts[currentAccounts].accNumber = newAccNumber;
accounts[currentAccounts].pin = newPin;
accounts[currentAccounts].balance = 0.0;
accounts[currentAccounts].isActive = true;

// Print info to serial for debugging
Serial.print("New account created: Number=");
Serial.print(newAccNumber);
Serial.print(", PIN=");
Serial.print(newPin);
Serial.println(", Balance=$0");

// Increment account counter
currentAccounts++;

// Show success
lcd.clear();
lcd.print("Account created!");
lcd.setCursor(0, 1);
lcd.print("ACC#: ");
lcd.print(newAccNumber);

digitalWrite(GREEN_LED, HIGH);
successBeep();
delay(2000);
digitalWrite(GREEN_LED, LOW);
```

```
  displayMainMenu();
}
void withdrawCash() {
  // Show current balance
  lcd.clear();
  lcd.print("Balance: $");
  lcd.print(accounts[currentAccountIndex].balance, 2);
  lcd.setCursor(0, 1);
  lcd.print("Amount: $");

  // Reset buffer for amount input
  memset(inputBuffer, 0, sizeof(inputBuffer));
  bufferIndex = 0;

  // Get amount
  int amount = 0;
  char key;
  do {
    key = keypad.getKey();
    if (key >= '0' && key <= '9' && bufferIndex < 4) {
      inputBuffer[bufferIndex++] = key;
      lcd.print(key);
    }
  } while (key != '#' && key != 'D'); // # to confirm, D to cancel

  if (key == 'D') {
    displayMainMenu();
    return;
  }

  amount = atoi(inputBuffer);

  // Check if amount is valid
  if (amount <= 0) {
    lcd.clear();
    lcd.print("Invalid amount");
    delay(1500);
    displayMainMenu();
    return;
  }

  // Check if sufficient balance
  if (amount > accounts[currentAccountIndex].balance) {
    lcd.clear();
    lcd.print("Insufficient");
    lcd.setCursor(0, 1);
    lcd.print("balance");

    digitalWrite(RED_LED, HIGH);
    errorBeep();
```

```
    delay(2000);
    digitalWrite(RED_LED, LOW);

    displayMainMenu();
    return;
  }

  // Process withdrawal
  accounts[currentAccountIndex].balance -= amount;

  // Print to serial monitor
  Serial.print("Withdrawal: $");
  Serial.print(amount);
  Serial.print(" from Account #");
  Serial.print(accounts[currentAccountIndex].accNumber);
  Serial.print(". New balance: $");
  Serial.println(accounts[currentAccountIndex].balance, 2);

  // Display success
  lcd.clear();
  lcd.print("Withdrawal: $");
  lcd.print(amount);
  lcd.setCursor(0, 1);
  lcd.print("New bal: $");
  lcd.print(accounts[currentAccountIndex].balance, 2);

  // Actuate the servo to simulate cash dispensing
  dispenseCash();

  digitalWrite(GREEN_LED, HIGH);
  successBeep();
  delay(2000);
  digitalWrite(GREEN_LED, LOW);

  displayMainMenu();
}

void depositCash() {
  // Ask for deposit amount
  lcd.clear();
  lcd.print("Deposit amount:");
  lcd.setCursor(0, 1);
  lcd.print("$");

  // Reset buffer for amount input
  memset(inputBuffer, 0, sizeof(inputBuffer));
  bufferIndex = 0;

  // Get amount
  int amount = 0;
```

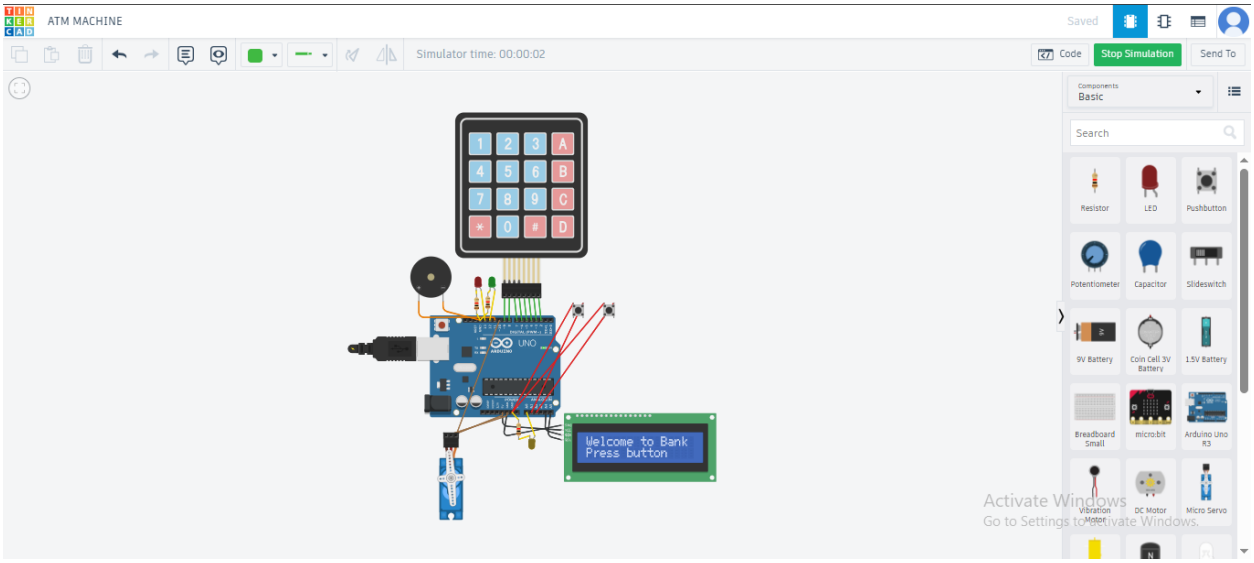
```
  char key;
  do {
    key = keypad.getKey();
    if (key >= '0' && key <= '9' && bufferIndex < 4) {
      inputBuffer[bufferIndex++] = key;
      lcd.print(key);
    }
  } while (key != '#' && key != 'D'); // # to confirm, D to cancel

  if (key == 'D') {
    displayMainMenu();
    return;
  }

  amount = atoi(inputBuffer);

  // Check if amount is valid
  if (amount <= 0) {
    lcd.clear();
    lcd.print("Invalid amount");
    delay(1500);
    displayMainMenu();
    return;
  }

  // Process deposit
  accounts[currentAccountIndex].balance += amount;

  // Print to serial monitor
  Serial.print("Deposit: $");
  Serial.print(amount);
  Serial.print(" to Account #");
  Serial.print(accounts[currentAccountIndex].accNumber);
  Serial.print(". New balance: $");
  Serial.println(accounts[currentAccountIndex].balance, 2);

  // Display success
  lcd.clear();
  lcd.print("Deposit: $");
  lcd.print(amount);
  lcd.setCursor(0, 1);
  lcd.print("New bal: $");
  lcd.print(accounts[currentAccountIndex].balance, 2);

  digitalWrite(GREEN_LED, HIGH);
  successBeep();
  delay(2000);
  digitalWrite(GREEN_LED, LOW);

  displayMainMenu();
```

```
}
void checkBalance() {
  lcd.clear();
  lcd.print("Account: ");
  lcd.print(accounts[currentAccountIndex].accNumber);
  lcd.setCursor(0, 1);
  lcd.print("Balance: $");
  lcd.print(accounts[currentAccountIndex].balance, 2);

  delay(3000);
  displayMainMenu();
}

void successBeep() {
  tone(BUZZER_PIN, 1000);
  delay(200);
  noTone(BUZZER_PIN);
}

void errorBeep() {
  tone(BUZZER_PIN, 400);
  delay(200);
  noTone(BUZZER_PIN);
  delay(100);
  tone(BUZZER_PIN, 400);
  delay(200);
  noTone(BUZZER_PIN);
}

void dispenseCash() {
  // Simulate cash dispensing with servo
  cashDispenser.write(90);  // Open dispenser
  delay(1000);
  cashDispenser.write(0);   // Close dispenser
}

int getAccountIndex(int accNumber) {
  for (int i = 0; i < currentAccounts; i++) {
    if (accounts[i].accNumber == accNumber && accounts[i].isActive) {
      return i;
    }
  }
  return -1; // Not found
}

void resetSystem() {
  isAuthenticated = false;
  currentAccountIndex = -1;
  lcd.clear();
  lcd.print("Session ended");
```

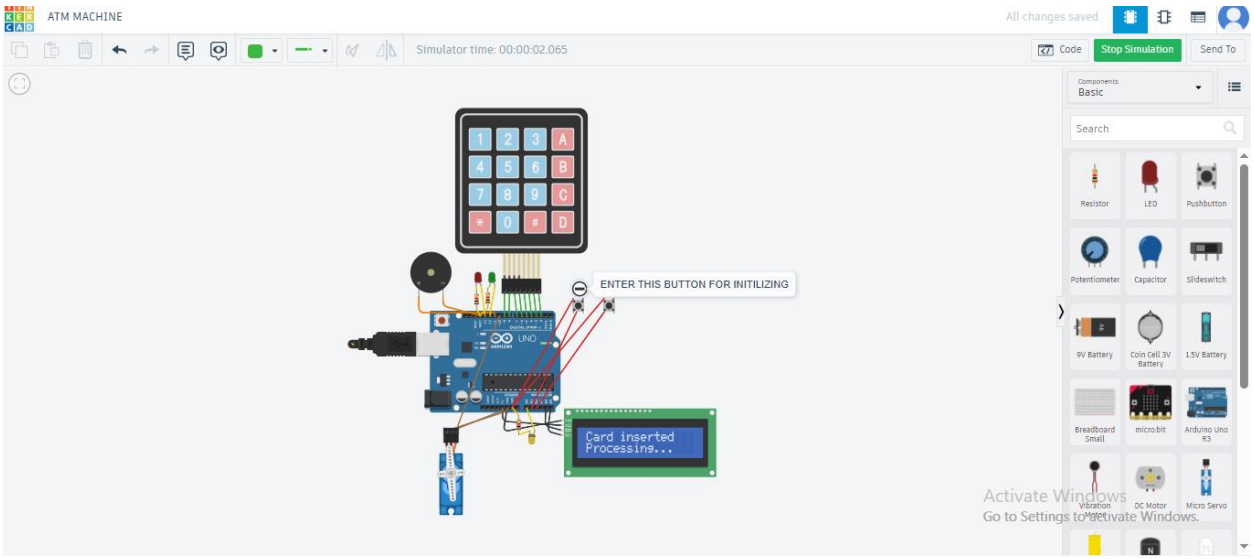
```
  delay(1500);
  displayWelcomeScreen();
}
```

Output

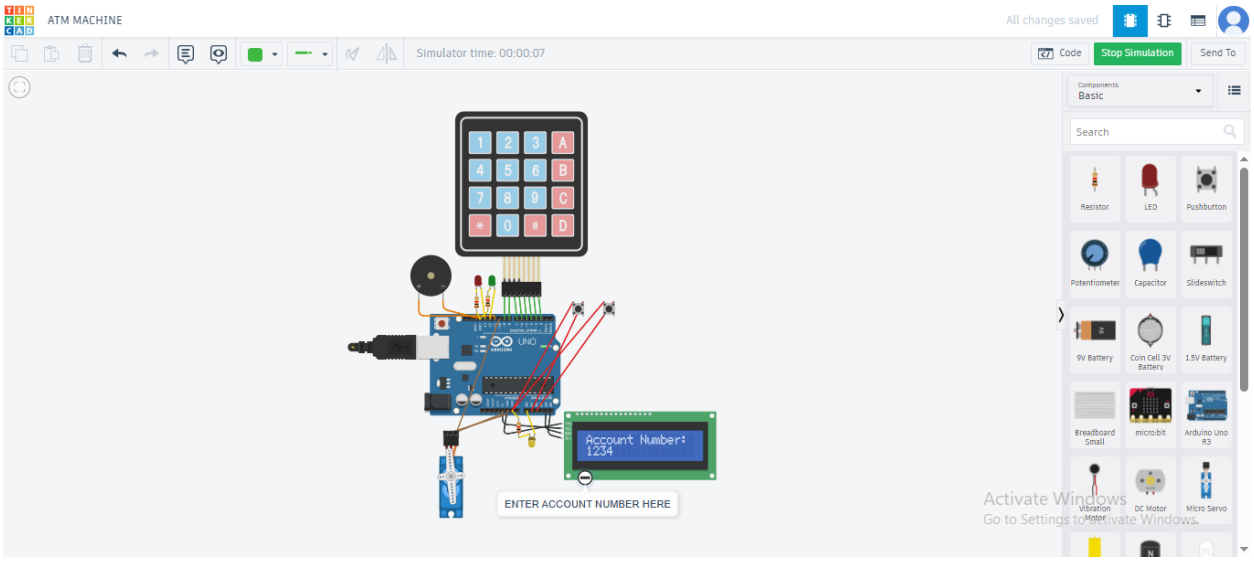
STEP-1: WELCOME TO BANK



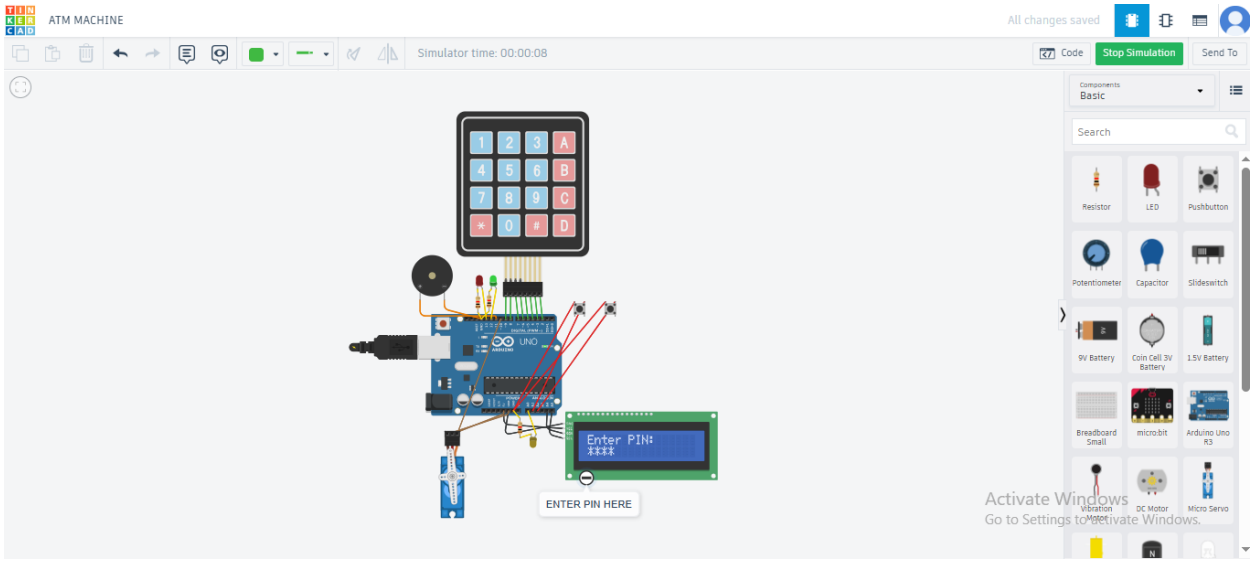
STEP-2: PUSH START BUTTON



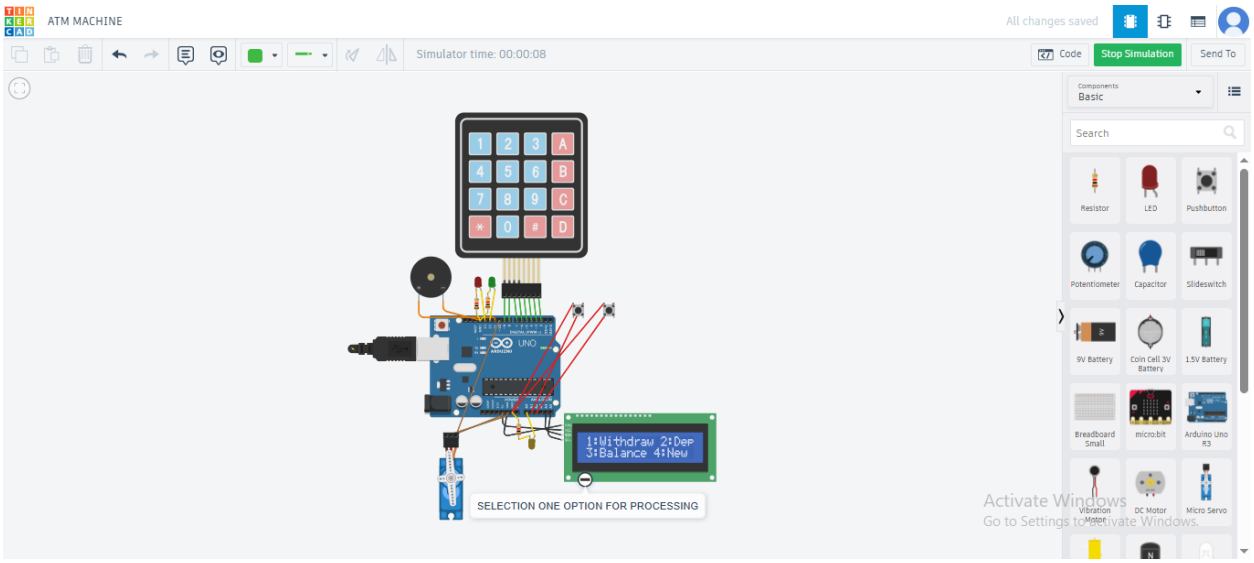
STEP-3: ENTER ACCOUNT NUMBER



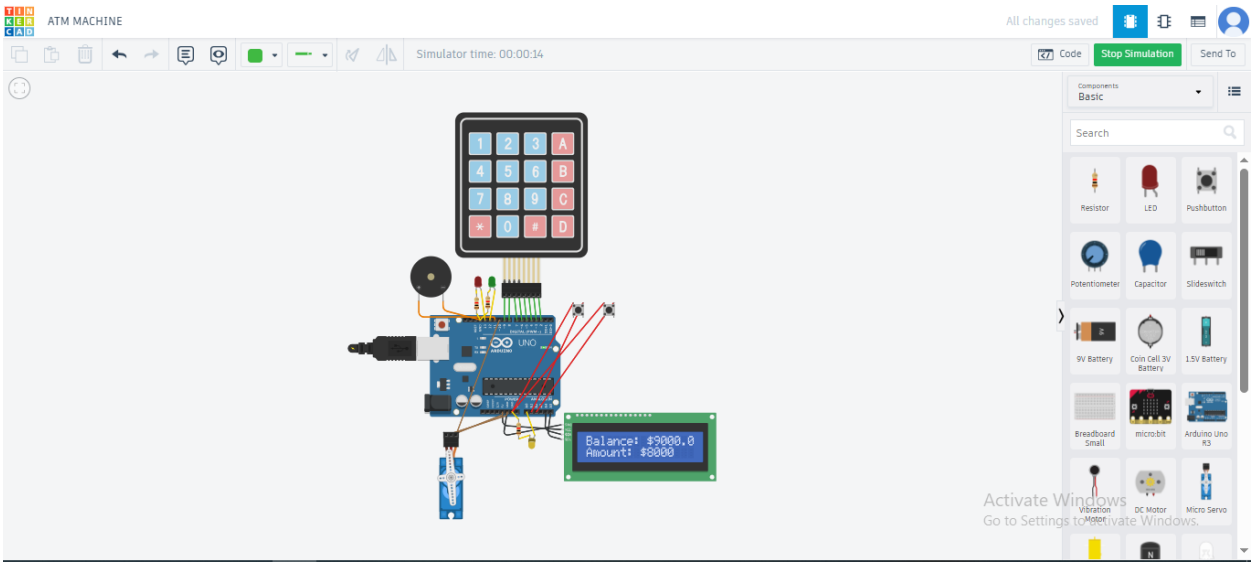
STEP-4: ENTER PIN

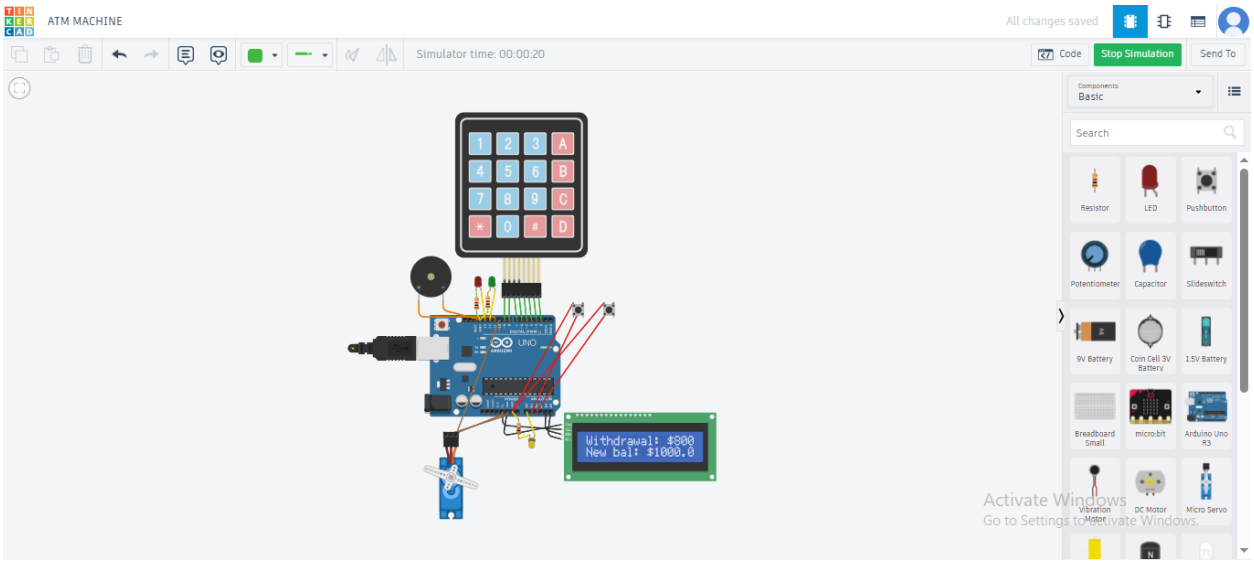


STEP-5: SELECT ANY ONE OPTION FROM MENU

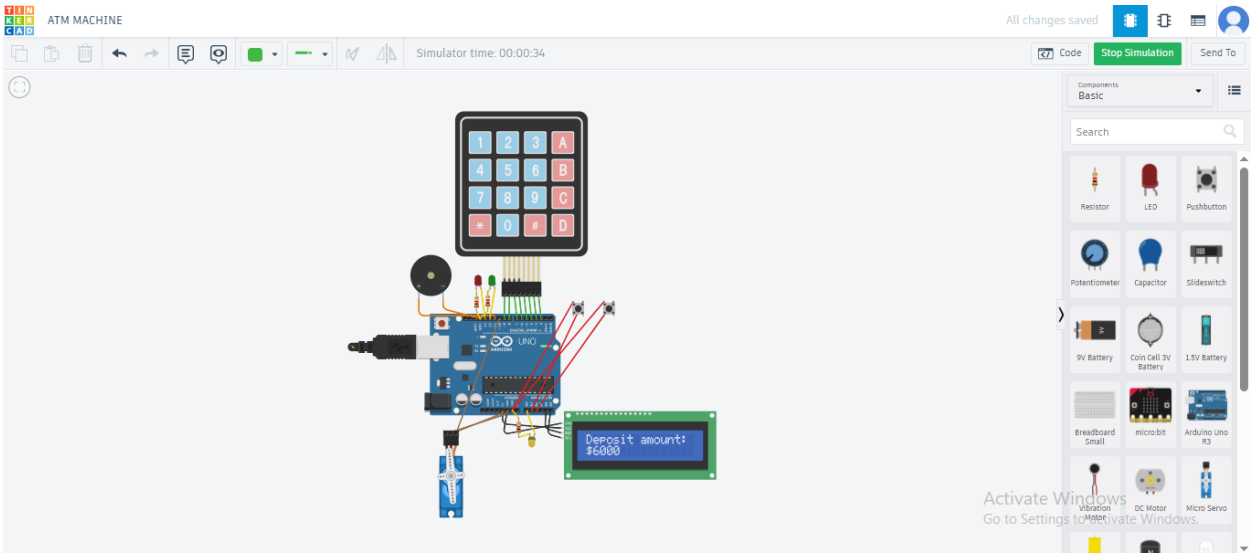


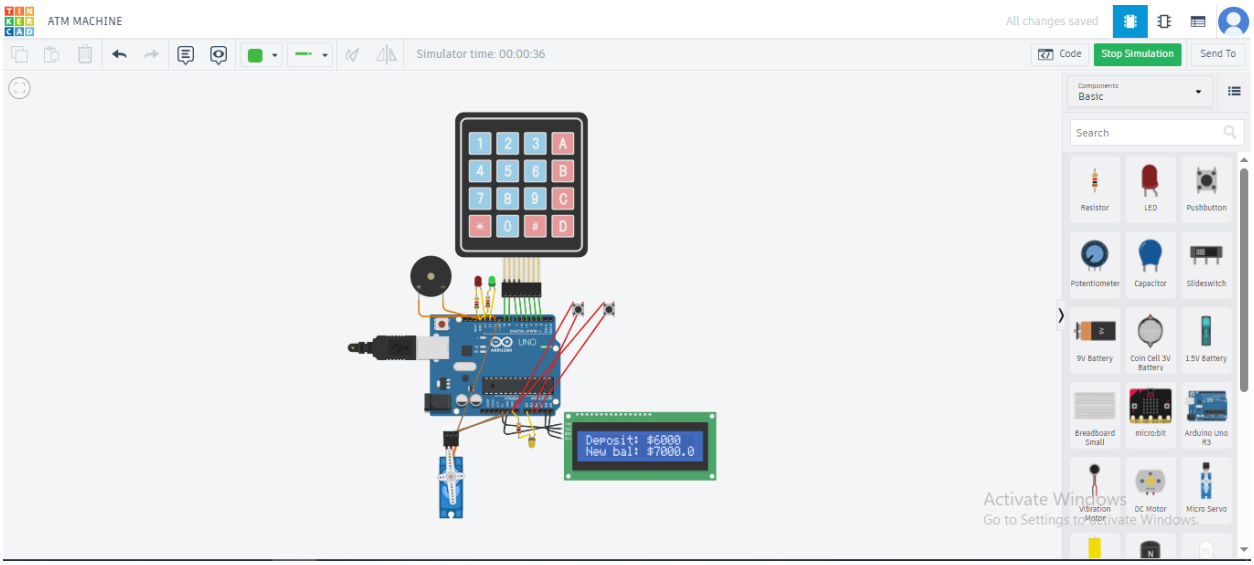
STEP-6: IF YOU SELECT OPTION 1



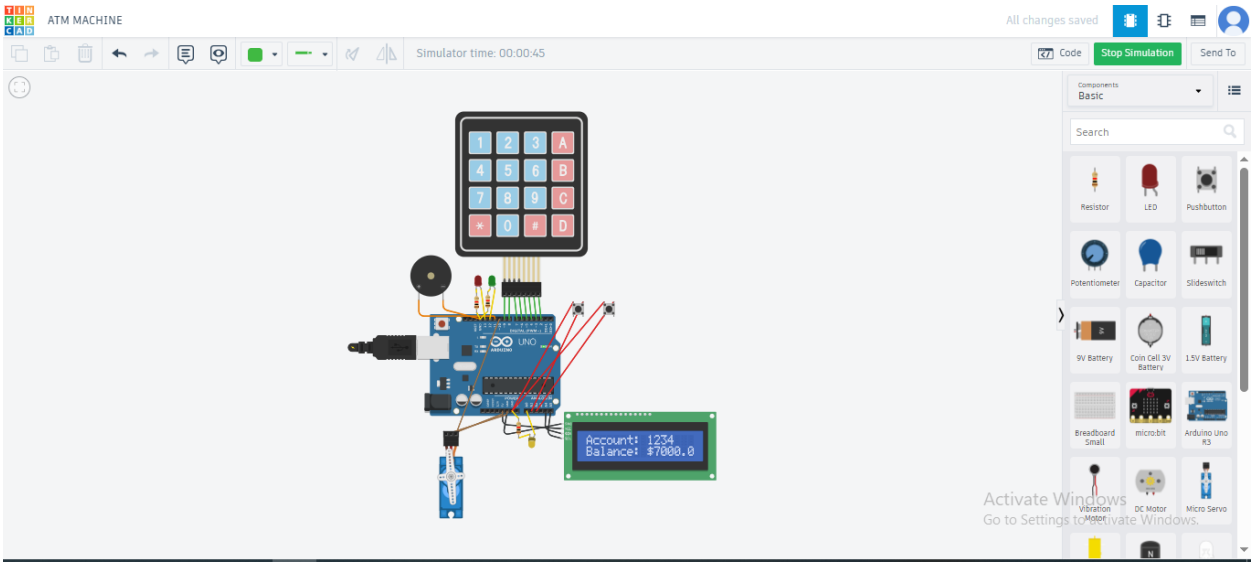


STEP-7: IF YOU SELECT OPTION 2

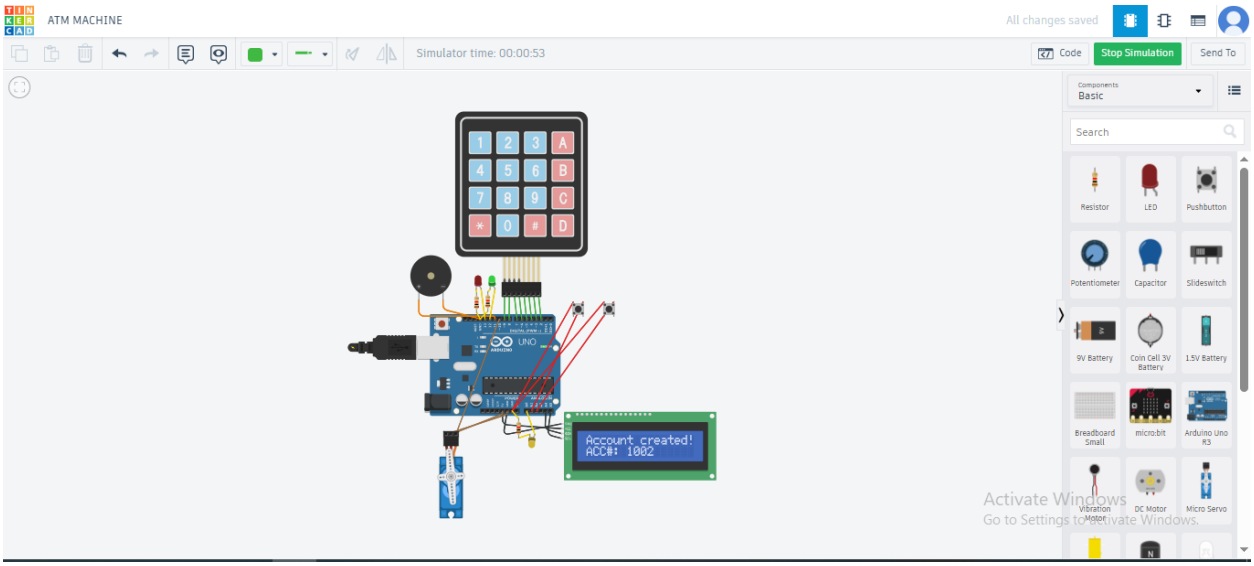




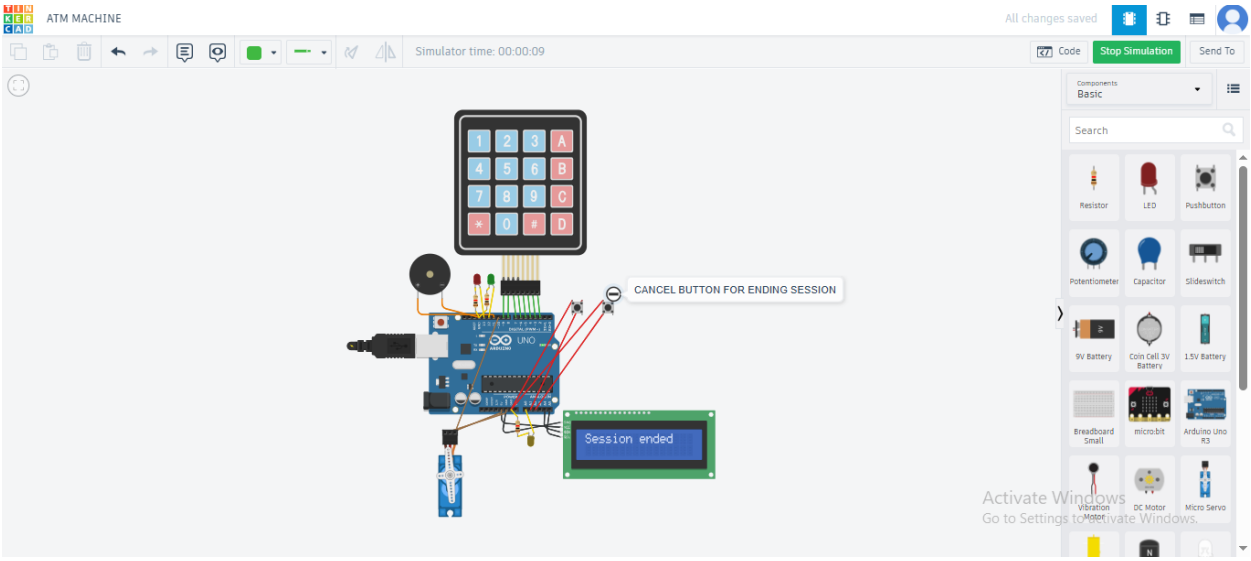
STEP-8: IF YOU SELECT OPTION 3



STEP-9: IF YOU SELECT OPTION 4



STEP-10: PRESS CANCEL BUTTON FOR ENDING SESSION



Conclusion

THIS RESULT COMES FROM COMBINING THE LOGIC OF DIFFERENT FUNCTIONS WE IMPLEMENTED**: -**

- Successful login grants access to ATM functions.
- Withdrawals simulate cash dispensing via servo.
- Deposits and balance checks are updated and displayed correctly.
- Creating a new account adds it to memory with a new number and PIN.
- Proper audio/visual cues for all operations (success, error, processing).