ATM MACHINE PROJECT

Contents

Project Description

Prototype that mimics the ATM Card and Machine interactions, consists of two parts:

1. CARD ECU:
   An ECU connected to EEPROM that contains the card data, it has two modes:
   A. Programming Mode:
      A mode where we set the data of the card
      - Card holder name.
      - Primary account number.
      - PIN

   B. User Mode:
      A mode where the card is being used by the end user for money transactions.
2. ATM ECU:
   An ECU connected to keypad, LCD and temperature sensor, manipulates the data of accounts stored in the bank server according to transaction processes.
   Bank servers are represented by an EEPROM.
   The ATM ECU has two modes:
   A. Programming Mode:
      Used for setting transaction amount limits and for accessing and modifying of the stored data on the bank servers.
   B. Operating Mode:
      Has two main choices:
      - Display the current temperature.
      - Insert the card and starts the transaction.

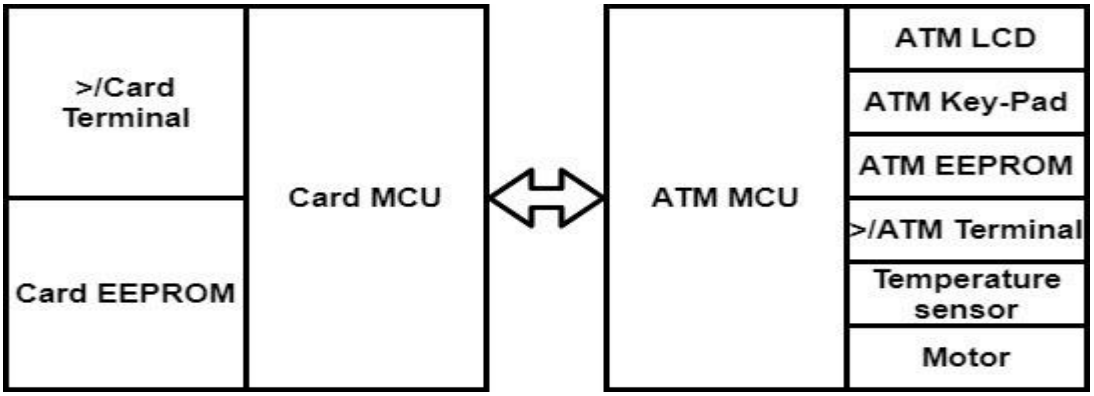

Figure 1 ATM Machine Components

Hardware Requirements

- Two Atmega32 MCUs.
- Two 24C16B EEPROM.
- Two Terminals.
- One Push Button.
- One 3 X 4 keypad.
- Four pull-up resistors.
- One pull-down resistor.
- One LM35 temperature sensor.
- One 16x2 LCD.
- One DC Motor.

Software Requirements

- ### CARD ECU

  There are two modes for the card:

  - #### Programming Mode.
    - The card MCU entered this mode when it receives from the terminal the “ADMIN” command.
    - In this mode the MCU asks for some data:
      1. Card Holder Name (9-characters string).
      2. Primary Account Number (9-characters string).
      3. Personal Identification Number (PIN), it is a 4-numeric characters string.
    - All these data will be stored in the EEPROM.
  - #### User Mode.
    - The Card entered this mode if there is data stored into the EEPROM.
    - From this mode you can go to Programming mode using the “ADMIN” command.
    - The card is ready to make transactions in this mode.

- ### ATM ECU

  The ATM has two modes:

  - #### Programming Mode.
    - It will enter this mode when the “ADMIN” is send by the terminal.
    - It will asks for the admin password, which is stored in the EEPROM.
      - Admin Password is entered by the Terminal.
    - Data related to the Accounts are stored in the EEPROM.
      - These data are entered by the terminal.
      - PAN.
      - Balance.
      - Max Amount.
      - These data are entered by the terminal.
  - #### Operating Mode.
    - The ATM entered this mode when “USER” command is sent through the terminal.
    - The ATM can read the card only when it is in the user mode and when it is in the operating mode.
    - The ATM will start to read the card when the button is pressed.
    - The terminal will display messages and the user will reply with the appropriate data.
      - The ATM will ask about the PIN.
      - It will check the PIN from the CARD ECU.
      - If the PIN is correct:
        - It will ask for the Amount (6-digits string in the format “0000.00”)
        - It will check the amount:
          - If it exceeds the MAX limit:
            - “Max Amount Exceeded” message will be displayed on the terminal.
          - Else if it exceeds the Balance stored in the EEPROM:
            - “Insufficient Fund” message will be displayed on the terminal.
          - Else:
            - “APPROVED” message will be displayed
            - The motor rotates for 1s.
      - Else:
        - “INCORRECT PIN” message will be displayed.

Project Static Design

- CARD ECU
  - Layered Architecture

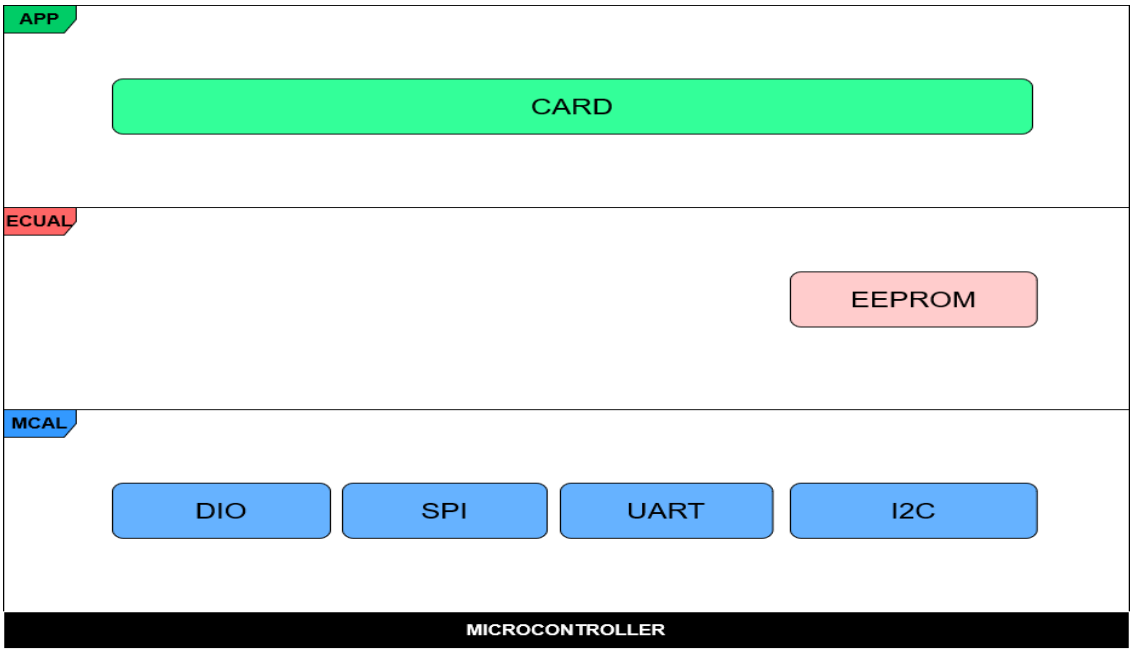

Figure 2 Card Layered Architecture Design

  - Layer Modules

1. MCAL Layer Modules:

1. DIO Module.
2. SPI Module.
3. I2C Module.
4. UART Module.

2. HAL Layer Modules:

1. EEPROM Module.

3. Application Layer Modules:

1. Card Module.

- Module APIs

1. DIO APIs

- uint8_t DIO_SetPinDirection(uint8_t PortName , uint8_t PinNo ,uint8_t PinDirection);
- uint8_t DIO_WritePin(uint8_t PortName , uint8_t PinNo ,uint8_t PinValue);
- uint8_t DIO_TogglePin(uint8_t PortName,uint8_t PinNo);
- uint8_t DIO_ReadPin(uint8_t PortName,uint8_t PinNo,uint8_t * PinData);
- uint8_t DIO_EnablePinPullup(uint8_t PortName,uint8_t PinNo);

2. SPI APIs

- uint8_t SPI_Init(uint8_t SpiNumber);
- uint8_t SPI_TransmitChar(uint8_t SpiNumber,uint8_t TxChar,uint8_t slave_CH);
- uint8_t SPI_ReceiveChar(uint8_t SpiNumber,ptr_uint8_t RxData,uint8_t slave_CH);
- uint8_t SPI_DataExchange (uint8_t SpiNumber, uint8_t TxChar, ptr_uint8_t RxData, uint8_t slave_CH);
- uint8_t SPI_TransmitString(uint8_t SpiNumber,ptr_uint8_t TxString,uint8_t slave_CH);
- uint8_t SPI_ReceiveString(uint8_t SpiNumber,ptr_uint8_t RxString,uint8_t slave_CH);
- uint8_t SPI_EnableInterrupt(uint8_t SpiNumber);
- uint8_t SPI_DisableInterrupt(uint8_t SpiNumber);
- uint8_t SPI_Set_TX_CompleteCallback(uint8_t SpiNumber,void(*callBack)(void));

3. I2C APIs

- uint8_t I2C_Init(uint8_t I2C_CH);
- uint8_t I2C_SetSlaveAddress(uint8_t I2C_CH, uint8_t SlaveAddr);
- uint8_t I2C_Start(uint8_t I2C_CH);
- uint8_t I2C_RepeatedStart(uint8_t I2C_CH);
- uint8_t I2C_Write(uint8_t I2C_CH, uint8_t Data);
- uint8_t I2C_ReadAck(uint8_t I2C_CH, uint8_t * Data);
- uint8_t I2C_ReadNoAck(uint8_t I2C_CH, uint8_t * Data);
- uint8_t I2C_Stop(uint8_t I2C_CH);
- uint8_t I2C_Status(uint8_t I2C_CH, uint8_t * Status);
- uint8_t I2C_EnableInterrupt(uint8_t I2C_CH);
- uint8_t I2C_DisableInterrupt(uint8_t I2C_CH);
- uint8_t I2C_SetCallback(uint8_t I2C_CH, Ptr_VoidFuncVoid_t Callback);

4. UART APIs

- uint8_t UART_Init(uint8_t UartNumber);
- uint8_t UART_TransmitChar(uint8_t UartNumber,uint8_t TxChar);
- uint8_t UART_TransmitString(uint8_t UartNumber,ptr_uint8_t TxString);
- uint8_t UART_ReceiveChar(uint8_t UartNumber,ptr_uint8_t RxChar);
- uint8_t UART_ReceiveString(uint8_t UartNumber,ptr_uint8_t RxString);
- uint8_t UART_EnableInterrupt(uint8_t UartNumber,uint8_t UartInterruptType);
- uint8_t UART_DisableInterrupt(uint8_t UartNumber,uint8_t UartInterruptType);
- uint8_t UART_SetCallback(uint8_t UartNumber,uint8_t UartInterruptType, Ptr_VoidFuncVoid_t Callback);
- uint8_t UART_GetData(uint8_t UartNumber, ptr_uint8_t RxChar);
- void UART_FlushReceiveBuffer(void);

5. EEPROM APIs

- uint8_t EEPROM_Init(uint8_t EEPROM_CH);
- uint8_t EEPROM_Read(uint8_t EEPROM_CH, uint8_t memoryBlock, uint8_t address, uint8_t * data);
- uint8_t EEPROM_Write(uint8_t EEPROM_CH, uint8_t memoryBlock, uint8_t address, uint8_t data);
- uint8_t EEPROM_ReadBytes(uint8_t EEPROM_CH, uint8_t memoryBlock, uint8_t start_address, uint8_t * data, uint8_t bytes_num);
- uint8_t EEPROM_WriteBytes(uint8_t EEPROM_CH, uint8_t memoryBlock, uint8_t start_address, uint8_t * data, uint8_t bytes_num);

6. Card APIs

- void CARD_Init(void);
- void CARD_GetData(void);
- void CARD_SetData(void);
- void CARD_Send(void);
- void CARD_Receive(void);

- ATM ECU
  - Layered Architecture

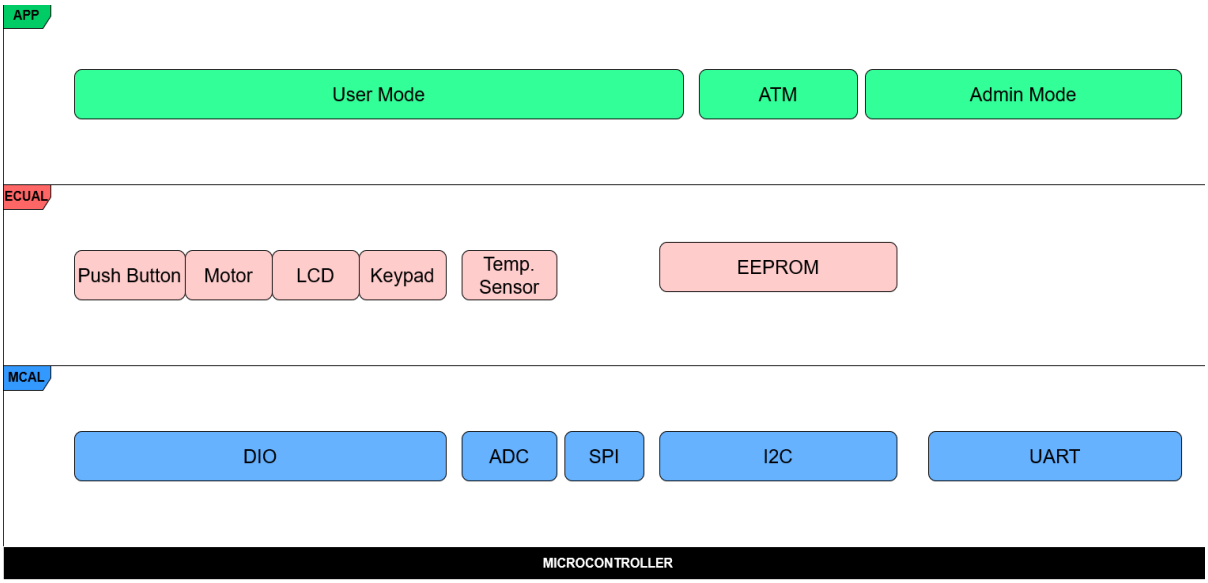

Figure 3 ATM Layered Architecture Design

  - Layer Modules

1. MCAL Layer Modules:

1. DIO Module.
2. ADC Module.
3. SPI Module.
4. I2C Module.
5. UART Module.

2. HAL Layer Modules:

1. Push Button Module.
2. Motor Module.
3. LCD Module.
4. Keypad Module.
5. Temperature Sensor Module.
6. EEPROM Module.

3. Application Layer Modules:

1. User Mode Module.
2. Admin Mode Module.
3. ATM Module.

- Module APIs

1. DIO APIs

Click to follow link *DIO APIs*

2. ADC APIs

- uint8_t ADC_Init(uint8_t ACD_CH);
- uint8_t ADC_StartSingleConversion(uint8_t ADC_Ch);
- uint8_t ADC_Read(uint8_t ADC_Ch, uint16_t*ADC_DATA);
- uint8_t ADC_EnInterrupt(void);
- uint8_t ADC_DisInterrupt(void);
- uint8_t ADC_SetCallback(Ptr_VoidFuncVoid_t Callback);

3. SPI APIs

Click to follow link *SPI APIs*

4. I2C APIs

Click to follow link *I2C APIs*

5. UART APIs

Click to follow link *UART APIs*

6. Push Button APIs

- uint8_t PSHBTTN_Init (DIO_PORT_ID_t port, DIO_PIN_ID_t pin, PSHBTTN_PULLUP_Status_t status);
- uint8_t PSHBTTN_EnablePullUp (DIO_PORT_ID_t port, DIO_PIN_ID_t pin);
- uint8_t PSHBTTN_Status (DIO_PORT_ID_t port, DIO_PIN_ID_t pin);

7. Motor APIs

- uint8_t MOTOR_INIT(void);
- uint8_t MOTOR_START(void);
- uint8_t MOTOR_STOP(void);
- uint8_t MOTOR_STATUS(uint8_t * status);

8. LCD APIs

- uint8_t LCD_Init(void);
- uint8_t LCD_SendCommand(uint8_t Cmd);
- uint8_t LCD_SendData(uint8_t Data);
- uint8_t LCD_SendString(ptr_int8_t String);
- uint8_t LCD_SendNumber(uint32_t Number);

9. Keypad APIs

- uint8_t KEYPAD_Init(void);
- uint8_t KEYPAD_ReadKey(ptr_uint8_t Key);
- uint8_t KEYPAD_WhichRow(uint8_t RowsVal,ptr_uint8_t RowNumber);
- uint8_t KEYPAD_MustPressed(ptr_uint8_t Key);

10. Temperature Sensor APIs

- uint8_t TEMP_Init(uint8_t Temp_ch);
- uint8_t TEMP_GetTemp(uint8_t Temp_ch, uint8_t* temp);

11. EEPROM APIs

Click to follow link *EEPROM APIs*

12. User Mode APIs

- void USER_Mode();
- uint8_t USER_Interface();
- uint8_t USER_PrintTemp(uint8_t Temp_Ch);
- uint8_t USER_GetCardData(void);
- EN_inServer_t USER_GetServerData(void);
- uint8_t User_Authenticate(ptr_uint8_t PINState);
- uint8_t User_CheckAmount(ptr_uint8_t AmountState);
- uint8_t User_UpdateBalance(void);

13. Admin Mode APIs

- void ADMIN_Mode(void);
- void ADMIN_Authenticate(void);
- void ADMIN_Interface(ptr_uint8_t pu8_choice);
- void ADMIN_SetAccount(void);
- void ADMIN_GetAccount(void);
- void ADMIN_SetMaxBalance(void);
- void ADMIN_GetMaxBalance(void);

14. ATM APIs

- uint8_t ATM_Init();
- uint8_t ATM_SelectMode();

Flow Charts

- CARD APIs Flow Chart

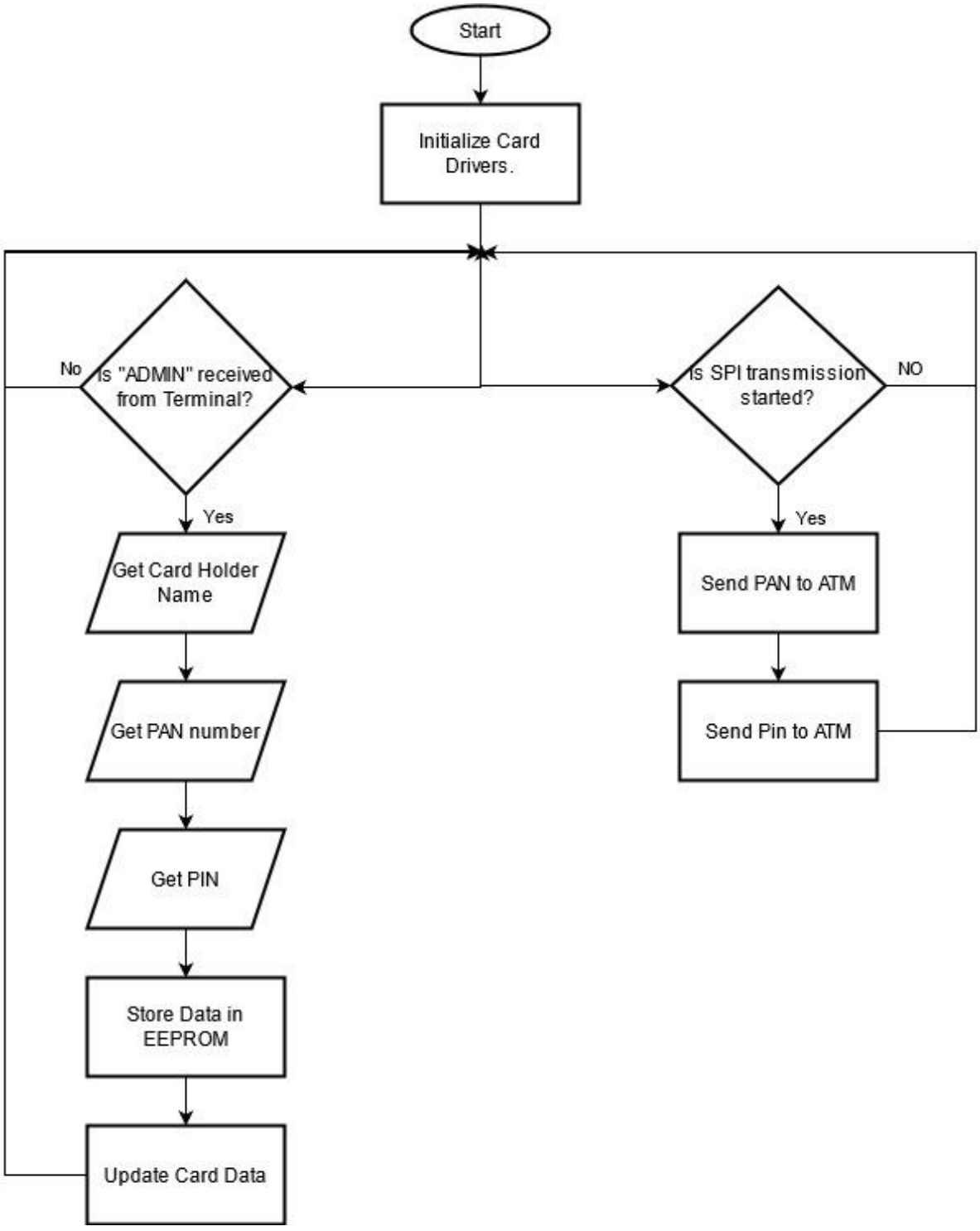

Figure 4 Card APIs Flow Chart

ATM APIs Flow Chart

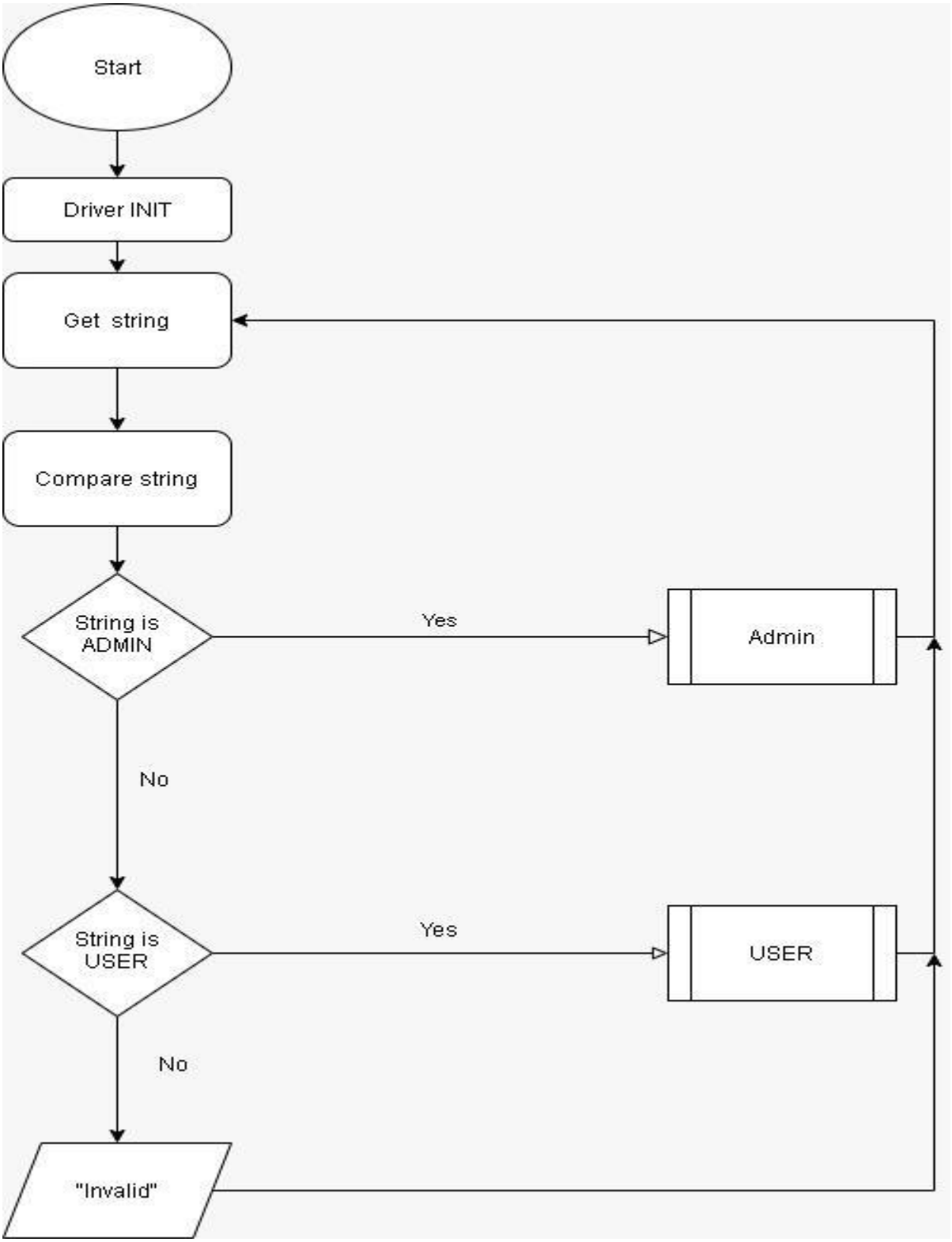

Figure 5 ATM APIs Flow Chart

- Admin APIs Flow Chart

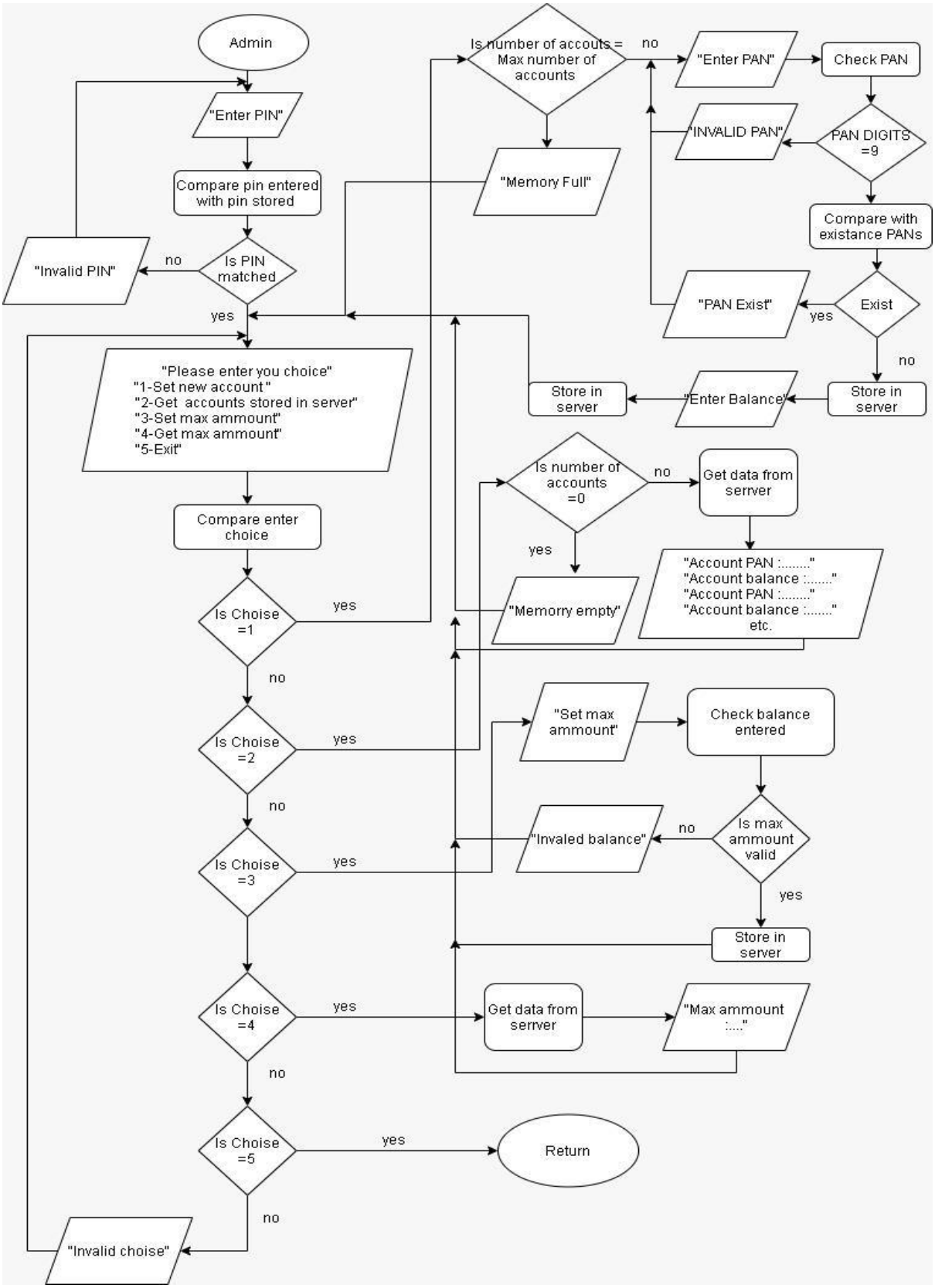

Figure 6 Admin APIs Flow Chart

- User APIs Flow Chart

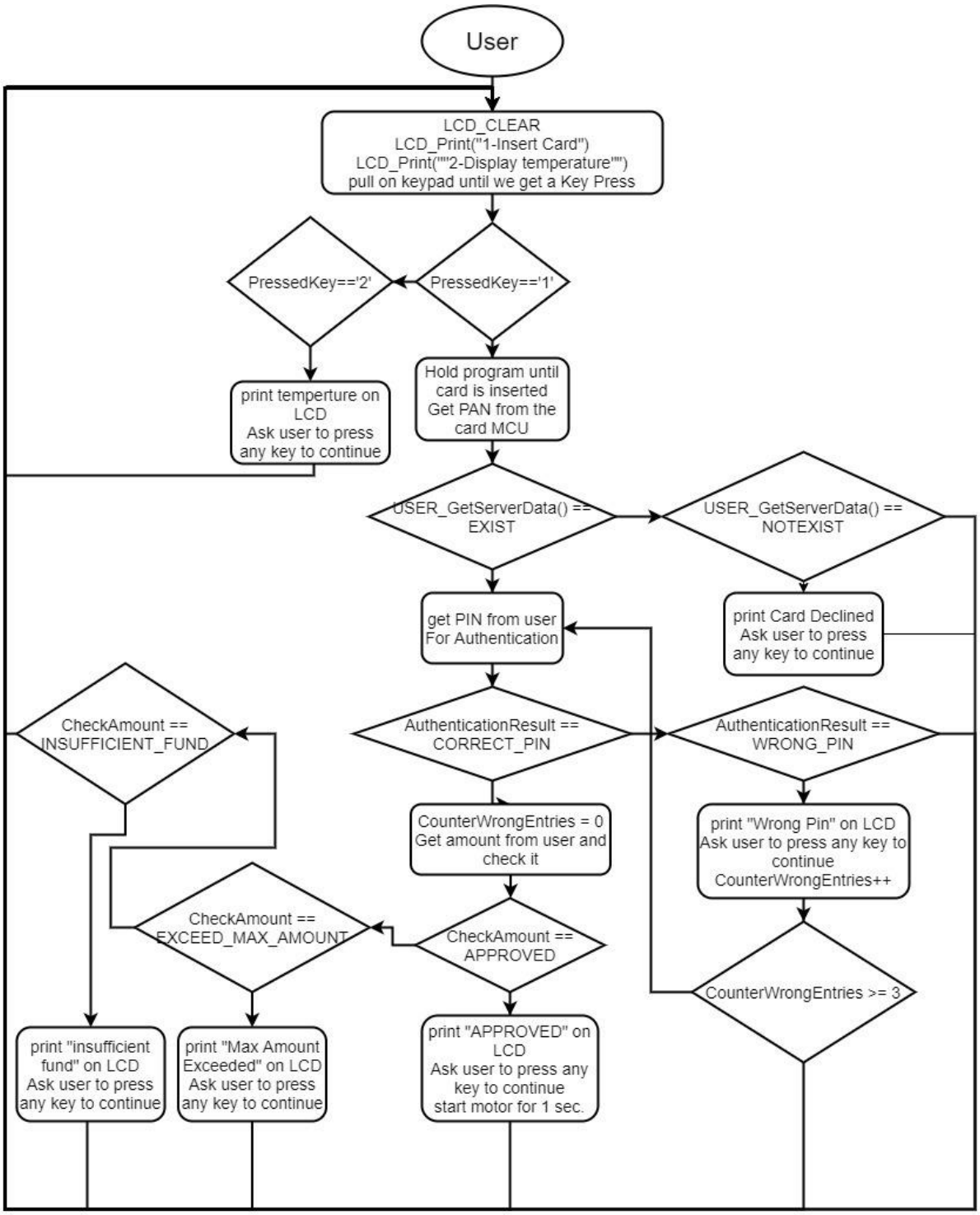

Figure 7 User APIs Flow Chart

Project Simulation

- Schematic

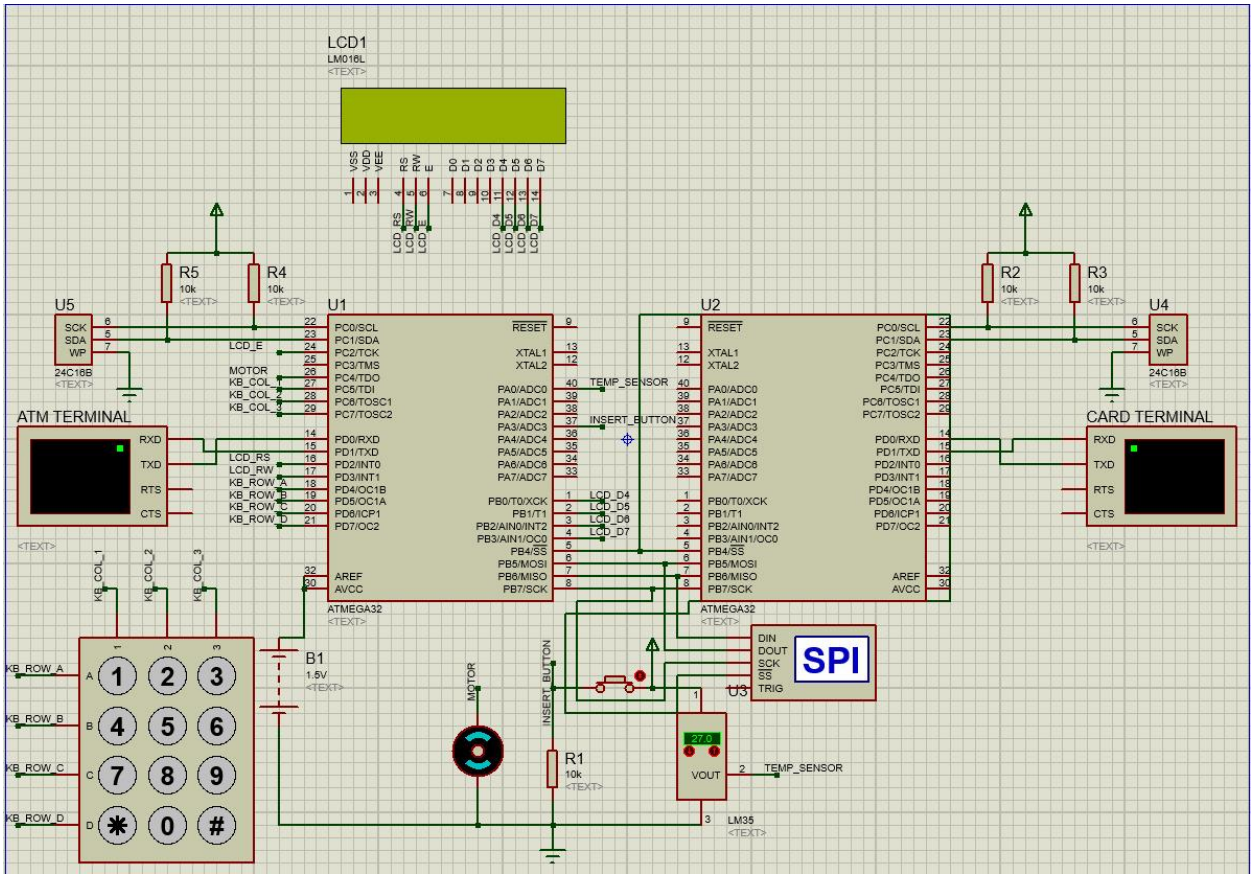

Figure 8 Proteus Simulation Circuit

- Simulation

  Click to follow link [Simulation Video]

Project Testing

Click to follow link [Project Test Cases]

Project Demo Videos

Click to follow link [Project Overview Video]

Click to follow link [Hardware Demo Video]