ATM Machine Project

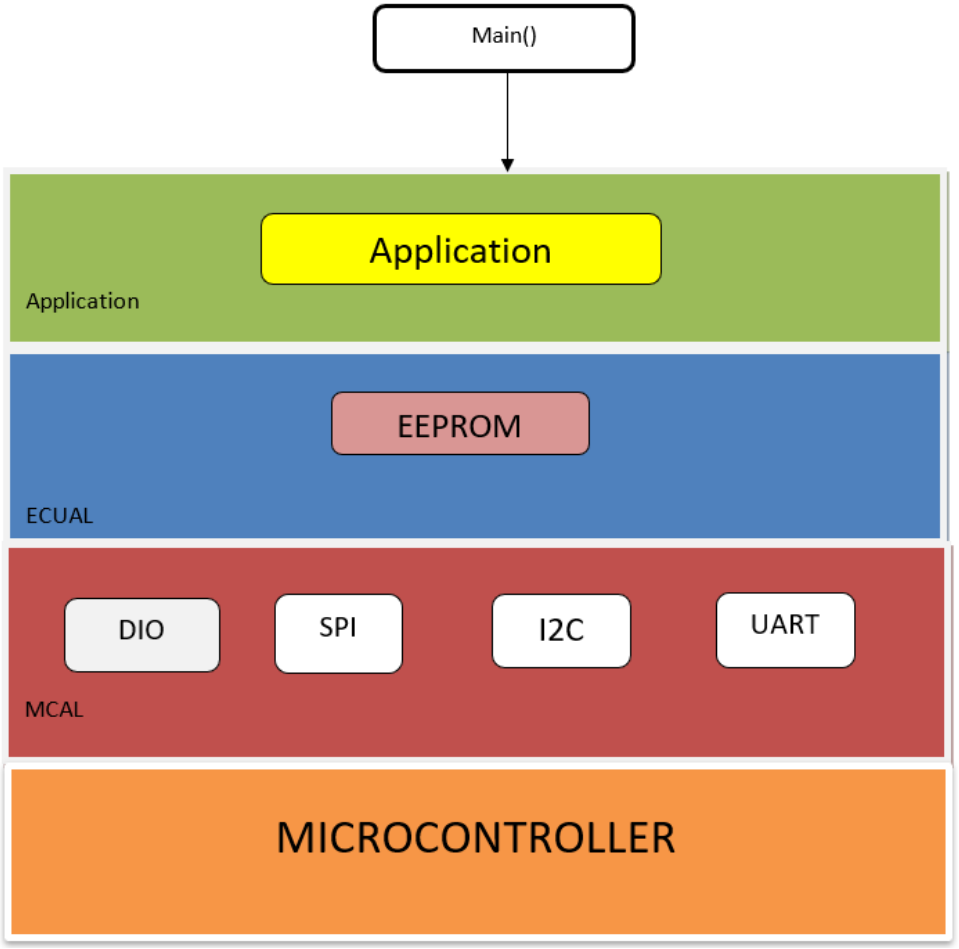
A project By:

SAKE TEAM (Team 2)

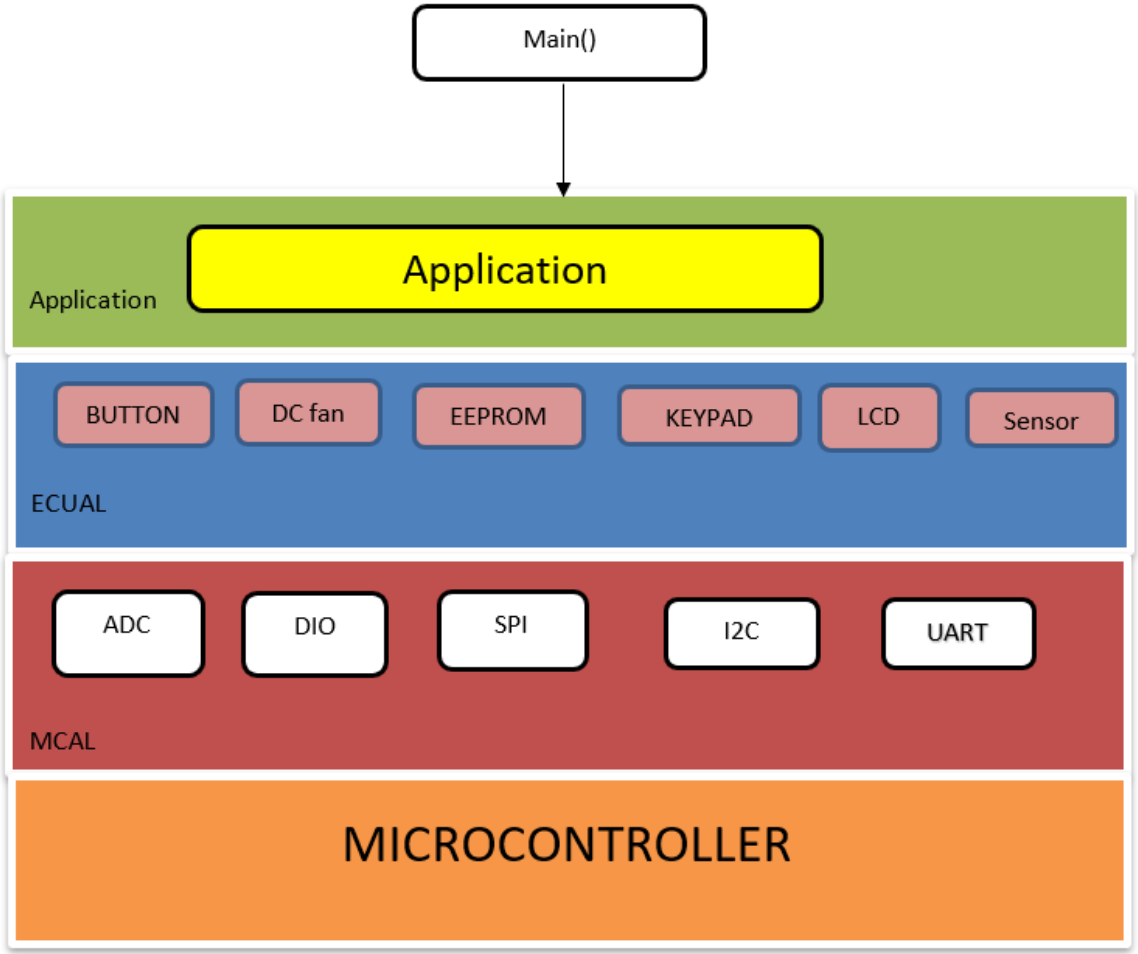
- **Mohamed Samy**
- **Amr Mohamed**
- **Kariman Karam**
- **Mohamed Ehab**

I) Static Design:.

CARD



ATM



- ENUMS

```
typedef enum {E_ERROR=0, E_OK=1,ERROR=0,SUCCESS=1}          enuErrorStatus_t;
typedef enum {LOW, HIGH}                  enuDigitalStates_t;
```

- Application Layer APIs:

```
void APP_Init(void);
void APP_Update(void);
```

- ECUAL layer:
  - EEPROM APIs:

```
enuErrorStatus_t EEPROM_Init       (void);
enuErrorStatus_t EEPROM_WriteByte (uint16_t u16ByteAddress,uint8_t u8Data);
enuErrorStatus_t EEPROM_ReadByte  (uint16_t u16ByteAddress,uint8_t *pu8Data);
enuErrorStatus_t EEPROM_WriteString (uint16_t u16ByteAddress,uint16_t u16ByteCount,uint8_t *pu8Data);
enuErrorStatus_t EEPROM_ReadString  (uint16_t u16ByteAddress,uint16_t u16ByteCount,uint8_t *pu8Data);
```

- ECUAL layer:
  - Button APIs:

```
typedef enum {BUTTON_ACTIVE_HIGH, BUTTON_ACTIVE_LOW} enuButtonConnectionType_t;
typedef enum {BUTTON_OFF, BUTTON_PRESSED, BUTTON_RELEASED, BUTTON_ON} enuButtonStates_t;
```

```
enuErrorStatus_t Button_Init(enuDIOPinNo_t enuDIOPinNo, enuButtonConnectionType_t enuButtonConnectionType);
```

```
enuErrorStatus_t Button_GetState(uint8_t enuDIOPinNo_t, enuButtonStates_t* penuButtonStates);
```

  - DC Fan APIs:

```
void DCFan_Init(enuDIOPinNo_t enuDIOPinNo);
```

```
enuErrorStatus_t DCFan_Start(enuDIOPinNo_t enuDIOPinNo);
```

```
enuErrorStatus_t DCFan_Stop(enuDIOPinNo_t enuDIOPinNo);
```

- ECUAL layer:
  - Keypad APIs:

```
typedef enum {KEYPAD_4X4, KEYPAD_4X3} enuKeypadType_t;

enuErrorStatus_t Keypad_Init(enuKeypadType_t enuKeypadType);

enuErrorStatus_t Keypad_GetChar(uint8_t* pu8Data);
```

  - Temp Sensor APIs:

```
void LM35_Init(void);

enuErrorStatus_t LM35_ReadTemperature(uint8_t* pu8Temperature);
```

- ECUAL layer:
  - LCD APIs:

```
typedef enum {LCD_DATA_FOUR_BIT_MODE = 4, LCD_DATA_EIGHT_BIT_MODE = 8} enuLCDDataMode_t;
typedef enum {LCD_CURSOR_OFF, LCD_CURSOR_ON} enuLCDCursorMode_t;
```

```
enuErrorStatus_t LCD_Init(enuLCDDataMode_t enuLCDDataMode, enuLCDCursorMode_t enuLCDCursorMode);
```

```
enuErrorStatus_t LCD_Clear(void);
```

```
enuErrorStatus_t LCD_SetCursorPosition(uint8_t u8Horizontal, uint8_t u8Vertical);
```

```
enuErrorStatus_t LCD_DisplayChar(uint8_t u8Char);
```

```
enuErrorStatus_t LCD_DisplayString(uint8_t* pu8String);
```

- MCAL layer:
  - ADC APIs:

```
typedef enum {ADC_AREF, ADC_AVCC, ADC_INTERNAL_VOLTAGE_REFERNECE = 3} enuADCReferenceSelectionBits_t;
typedef enum {ADC_DISABLE_LEFT_ADJUST_RESULT, ADC_ENABLE_LEFT_ADJUST_RESULT} enuADCLeftAdjustResult_t;
typedef enum {ADC_DIVISOR_FACTOR_IS_2 = 1, ADC_DIVISOR_FACTOR_IS_4, ADC_DIVISOR_FACTOR_IS_8, ADC_DIVISOR_FACTOR_IS_16,
              ADC_DIVISOR_FACTOR_IS_32, ADC_DIVISOR_FACTOR_IS_64, ADC_DIVISOR_FACTOR_IS_128} enuADCPrescalerSelectBits_t;
typedef enum
{
    ADC0, ADC1, ADC2, ADC3, ADC4, ADC5, ADC6, ADC7,
    ADC0_ADC0_10x, ADC1_ADC0_10x, ADC0_ADC0_200x, ADC1_ADC0_200x,
    ADC2_ADC2_10x, ADC3_ADC2_10x, ADC2_ADC2_200x, ADC3_ADC2_200x,
    ADC0_ADC1_1x, ADC1_ADC1_1x, ADC2_ADC1_1x, ADC3_ADC1_1x, ADC4_ADC1_1x, ADC5_ADC1_1x, ADC6_ADC1_1x, ADC7_ADC1_1x,
    ADC0_ADC2_1x, ADC1_ADC2_1x, ADC2_ADC2_1x, ADC3_ADC2_1x,ADC4_ADC2_1x,ADC5_ADC2_1x,
    VBG, GND
} enuADCAnalogChannel_t;
```

```
enuErrorStatus_t ADC_Init(enuADCReferenceSelectionBits_t enuADCReferenceSelectionBits, enuADCLeftAdjustResult_t enuADCLeftAdjustResult, enuADCPrescalerSelectBits_t enuADCPrescalerSelectBits);
```

```
enuErrorStatus_t ADC_ReadChannel(enuADCAnalogChannel_t enuAnalogChannel, uint16_t* pu16Data);
```

- MCAL layer:
  - DIO APIs:

```
typedef enum {PORTA, PORTB, PORTC, PORTD} enuDIOPortNo_t;
typedef enum {PA0, PA1, PA2, PA3, PA4, PA5, PA6, PA7,
              PB0, PB1, PB2, PB3, PB4, PB5, PB6, PB7,
              PC0, PC1, PC2, PC3, PC4, PC5, PC6, PC7,
              PD0, PD1, PD2, PD3, PD4, PD5, PD6, PD7} enuDIOPinNo_t;
typedef enum {INPUT, OUTPUT} enuDIOPinDirection_t;
typedef enum {NO_CONNECTION, PULL_UP_ENABLE} enuDIOPinPullupResistorEnable_t;
```

```
enuErrorStatus_t DIO_PinInit(enuDIOPinNo_t enuDIOPinNo, enuDIOPinDirection_t enuDIOPinDirection, enuDIOPinPullupResistorEnable_t enuDIOPinPullupResistorEnable);
```

```
enuErrorStatus_t DIO_PortInit(enuDIOPortNo_t enuDIOPortNo, uint8_t u8DIOPortDirection, uint8_t u8DIOPortPullupResistorEnable);
```

```
enuErrorStatus_t DIO_PinWrite(enuDIOPinNo_t enuDIOPinNo, uint8_t u8PinData);
```

```
enuErrorStatus_t DIO_PinWrite(enuDIOPinNo_t enuDIOPinNo, uint8_t u8PinData);
```

```
enuErrorStatus_t DIO_PortWrite(enuDIOPortNo_t enuDIOPortNo, uint8_t u8PortData);
```

```
enuErrorStatus_t DIO_PinRead(enuDIOPinNo_t enuDIOPinNo, uint8_t* pu8PinData);
```

```
enuErrorStatus_t DIO_PortRead(enuDIOPortNo_t enuDIOPortNo, uint8_t* pu8PortData);
```

```
enuErrorStatus_t DIO_PinToggle(enuDIOPinNo_t enuDIOPinNo);
```

```
enuErrorStatus_t DIO_PortToggle(enuDIOPortNo_t enuDIOPortNo);
```

- MCAL layer:
  - SPI APIs:

```
void SPI_init(void);
```

```
void SPI_sendData(const uint8_t u8Data);
```

```
void SPI_receiveData(uint8_t* pu8Data);
```

```
void SPI_sendString(const uint8_t *pu8String);
```

```
void SPI_receiveString(uint8_t *pu8Data, uint8_t u8BufferMaxSize);
```

- MCAL layer:
  - I2C (TWI) APIs:

```c
typedef enum{
   ACK,
   NACK
}enuI2CAck_t;
//data direction enum
typedef enum{Write, Read}enuI2CMode_t;
```

```c
enuErrorStatus_t I2C_Init(void);
```

```c
enuErrorStatus_t I2C_Start(void);
```

```c
enuErrorStatus_t I2C_Repeated_Start(void);
```

```c
enuErrorStatus_t I2C_Stop(void);
```

```c
enuErrorStatus_t I2C_SendAddressFrame(uint8_t u8SlaveAddress, enuI2CMode_t enuMode);
```

```c
enuErrorStatus_t I2C_SendDataFrame(uint8_t u8Data);
```

```c
enuErrorStatus_t I2C_RecieveDataFrame(uint8_t * pu8Data, enuI2CAck_t enuACK);
```

- MCAL layer:
  - I2C (TWI) APIs:

```
enuErrorStatus_t I2C_MASTER_SendData(uint8_t u8SlaveAddress,uint8_t u8Data);
```

```
enuErrorStatus_t I2C_MASTER_RecieveData(uint8_t u8SlaveAddress,uint8_t  *pu8Data);
```

```
enuErrorStatus_t I2C_SLAVE_SendData(uint8_t u8Data);
```

```
enuErrorStatus_t I2C_SLAVE_ReceiveData(uint8_t *pu8Data);
```

```
enuErrorStatus_t I2C_Enable_Interrupt(void);
```

```
enuErrorStatus_t I2C_Disable_Interrupt(void);
```

```
enuErrorStatus_t I2C_SetCallBack(void(*local_fptr)(void));
```

- MCAL layer:
  - UART APIs:

```
enuErrorStatus_t UART_Init(void);
enuErrorStatus_t UART_SendData(uint16_t u16Data);
enuErrorStatus_t UART_ReceiveData(uint16_t* pu16Data);
enuErrorStatus_t UART_RecieveDataNoBLOCK(uint16_t* pu16Data);
enuErrorStatus_t UART_SendString(uint8_t* pu8String);
enuErrorStatus_t UART_ReceiveString(uint8_t* pu8String,uint8_t max_length);
enuErrorStatus_t UART_SendDataNoBLOCK(uint16_t u16Data);
enuErrorStatus_t UART_TX_Enable_Interrupt(void);
enuErrorStatus_t UART_TX_Disable_Interrupt(void);
enuErrorStatus_t UART_TX_SetCallBack(void(*local_fptr)(void));
enuErrorStatus_t UART_RX_Enable_Interrupt(void);
enuErrorStatus_t UART_RX_Disable_Interrupt(void);
enuErrorStatus_t UART_RX_SetCallBack(void(*local_fptr)(void));
```

II) Tasks and Responsibilities:

- ### DRIVERS:

Mohamed Samy : ADC , DIO , Keypad ,
LCD , DC fan , Motor , Sensor , Button

- **Amr Mohamed :** I2C , EEPROM
- **Kariman Karam :** SPI
- **Mohamed Ehab :** UART

- ### Application Software:

- **Card :** Amr Mohamed
- **ATM :** Mohamed Samy , Kariman Karam , Mohamed Ehab

II) Tasks and Responsibilities:

- Hardware :

  Mohamed Samy

  Kariman Karam

- Static Design :

  Mohamed Ehab

- Simulation Video :

  Amr Mohamed

III) Time Estimate:

From July 29 to August 1

IV) Test Cases:

A) CARD Application:

TC ID	TEST CASE	EXPECTED OUTPUT	ACTUAL OUTPUT	Output
1	- Initialize with empty/corrupted EEPROM Data	Go into programming mode and initialize EEPROM data	Card got into programming mode to initialize card PAN+PIN	PASS
2	- Initialize with valid EEPROM data - Input “ADMIN” In Terminal	Go to admin screen and reinitialize PAN and PIN	Card got into programming mode to initialize card PAN+PIN	PASS
3	- Initialize with valid EEPROM data - Enable SPI Communication	Send PAN+PIN frame over SPI	Card sent PAN+PIN Frame over SPI	PASS

IV) Test Cases:

B) ATM Application: (To be continued)

TC ID	TEST CASE	EXPECTED OUTPUT	ACTUAL OUTPUT	Output
1	- Input “ADMIN” in terminal - Input invalid password	Returning “Invalid Password” on the terminal and ask for the used mode	“Invalid Password” returned on the terminal and asked about the used mode	PASS
2	Initializing with valid EEPROM data but the PAN of the card is not stored in the EEPROM	Returning “Invalid card , card is ejected” on the LCD	“Invalid card , card is ejected” returned on the LCD and go to the first screen	PASS
3	Initializing with valid EEPROM data and the PAN of the card is stored in the EEPROM but the user input incorrect PIN	Returning “Incorrect PIN, card is ejected” on the LCD	“Incorrect PIN, card is ejected” returned on the LCD and go to the first screen	PASS

IV) Test Cases:

B) ATM Application

TC ID	TEST CASE	EXPECTED OUTPUT	ACTUAL OUTPUT	Output
4	Initializing with valid EEPROM data and the PAN of the card is stored in the EEPROM and the user input the PIN correctly but the user entered amount greater than the maximum amount	Returning “Maximum amount exceeded” on the LCD	“Maximum amount exceeded” returned on the LCD and go to the first screen	PASS
5	Input invalid data but the user entered amount greater than the his balance	Returning “Insufficient fund” on the LCD	“Insufficient fund” returned on the LCD and go to the first screen	PASS
6	Input invalid data but the user entered amount less than or equal to the his balance	Returning “Transaction approved” on the LCD	“Transaction approved” returned on the LCD and go to the first screen	PASS