**CODE EXPLANATION IN BRIEF:**

In the beginning all necessary libraries are imported and functions are defined.

The pin numbers of all components connected are declared.

/\*

**Step by Step Approach:-**

**importing necessary libraries**

**Naming of csv file**

**EEPROM declaration**

**Function Prototyping**

**LCD Pin declaration**

**pin declaration of Rotary Encoder**

**MODBUS Declaration**

**Reading from EEPROM at F\_Min\_Addr location**

**MODBUS Initialization**

**SETUP:**

**Setting Baud rate for serial communication**

**SD Card Initialization**

**LED Init**

**LCD Init**

**Setting Mode Check**

**LOOP:**

**if State == 0, go to print\_modbus();**

**if State == 1, go to States\_Menu();**

**if the latest Force value is greater than Previous value = Continue reading**

**if the latest Force value is less than Previous value = Assume the Previous value as Maximum Force**

**and if the Maximum Force value is greater than F\_Min = PASS**

**else it is FAIL**

**and finaly save the result in Micro SD Card**

\*/

**ENTIRE CODE is given below:**

// include the library code: //importing necessary libraries

#include <LiquidCrystal.h>

#include <Arduino.h>

#include <RotaryEncoder.h>

#include <SoftwareSerial.h>

#include <ModbusMaster.h>

#include <EEPROM.h>

#include <SPI.h>

#include <SD.h>

File testfile;

String fileName = "test.csv"; // Name of csv file

/\*==================== EEPROM ==========================\*/

int F\_Min\_Addr = 4;

void writeIntIntoEEPROM(int address, int number)

{

EEPROM.write(address, number >> 8);

EEPROM.write(address + 1, number & 0xFF);

}

int readIntFromEEPROM(int address)

{

return (EEPROM.read(address) << 8) + EEPROM.read(address + 1);

}

/\*======================================================\*/

void print\_measure(); // Function Prototyping

void print\_setting();

void SD\_Results\_Fail();

void SD\_Results\_Pass();

void print\_modbus();

void States\_Menu();

const int rs = 8, en = 9, d4 = 4, d5 = 5, d6 = 6, d7 = 7; //LCD Pin declaration

LiquidCrystal lcd(rs, en, d4, d5, d6, d7);

#define PIN\_IN1 A3 // pin declaration of Rotary Encoder

#define PIN\_IN2 A4

#define PUSHB A5

RotaryEncoder encoder(PIN\_IN1, PIN\_IN2, RotaryEncoder::LatchMode::TWO03);

/\*============MODBUS Declaration ==================\*/

#define MAX485\_DE 3

#define MAX485\_RE\_NEG 2

#define Pass\_LED A1

#define Fail\_LED A0

int cellStatus = 0;

volatile int result;

float temp = 0;

float Force = 0;

/\*===================================\*/

static int pos = 0;

static int newPos = 0;

int State = 0;

volatile int F\_Min = readIntFromEEPROM(F\_Min\_Addr); // Reading from EEPROM at F\_Min\_Addr location

/\*==================MODBUS Init ===================\*/

// instantiate ModbusMaster object

ModbusMaster node;

void preTransmission()

{ //

digitalWrite(MAX485\_RE\_NEG, 1);

digitalWrite(MAX485\_DE, 1);

}

void postTransmission()

{ //

digitalWrite(MAX485\_RE\_NEG, 0);

digitalWrite(MAX485\_DE, 0);

}

/\*========================================\*/

void setup()

{

// Modbus communication runs at 19200 baud

Serial.begin(19200);

while (! Serial);

/\*=============== SD Card Init =====================\*/

// wait for SD module to start

if (!SD.begin(10)) {

Serial.println("No SD Module Detected");

}

delay(1000);

/\*============================================\*/

pinMode(PUSHB, INPUT); //Encoder button

/\*===================MODBUS==================\*/

pinMode(MAX485\_RE\_NEG, OUTPUT);

pinMode(MAX485\_DE, OUTPUT);

pinMode(Pass\_LED, OUTPUT);

pinMode(Fail\_LED, OUTPUT);

// Init in receive mode

digitalWrite(MAX485\_RE\_NEG, 0);

digitalWrite(MAX485\_DE, 0);

// Modbus slave ID 1

node.begin(1, Serial);

// Callbacks allow us to configure the RS485 transceiver correctly

node.preTransmission(preTransmission);

node.postTransmission(postTransmission);

/\*===============================================\*/

/\*============= LED Init================\*/

digitalWrite(Pass\_LED, HIGH);

digitalWrite(Fail\_LED, LOW);

delay(500);

digitalWrite(Pass\_LED, LOW);

digitalWrite(Fail\_LED, HIGH);

delay(500);

digitalWrite(Pass\_LED, LOW);

digitalWrite(Fail\_LED, LOW);

/\*======================================\*/

/\*=============== LCD Init ====================\*/

lcd.begin(16, 2);

lcd.clear();

lcd.print("LOAD CELL");

delay(1000);

/\*===============================================\*/

/\*================ Setting Mode Check ===========\*/

if (digitalRead(PUSHB) == LOW ) // If Encoder button is pressed

{

State = 1;

Serial.println(" loop State =1");

print\_setting();

encoder.setPosition(F\_Min);

delay(1000);

}

else

{

State = 0;

print\_measure();

}

/\*==========================================\*/

}//setup end

void loop()

{

if (State == 0)

{

print\_modbus();

}

if (State == 1) {

encoder.tick();

newPos = encoder.getPosition();

if (pos != newPos || digitalRead(PUSHB) == LOW) {

pos = newPos;

States\_Menu();

}

}

} // loop end

void States\_Menu()

{

delay(100);

if (pos >= 0) // If rotary encoder position is greater than zero

{

Serial.println(pos);

F\_Min = pos;

print\_setting();

if (digitalRead(PUSHB) == LOW )

{

Serial.println("Button Pressed");

State = 0;

encoder.setPosition(0);

/\*============ EEPROM ===========\*/

writeIntIntoEEPROM(F\_Min\_Addr, F\_Min);

/\*===============================\*/

}

}

if (pos < 0) {

encoder.setPosition(0);

}

} // States\_Menu end

void print\_modbus()

{

delay(100); // Very very important for proper working of modbus

if (cellStatus == 0)

{

// Read 16 registers starting at 0x07)

result = node.readHoldingRegisters(0x07, 2);

Force = float(node.getResponseBuffer(0x00)) / 10; //Like 12 =>1.2

if (result == node.ku8MBSuccess) // If proper data is received

{

print\_measure();

Serial.print("Fmax: ");

Serial.println(Force);

if (Force >= temp) //If the new incoming Force value is greater than the previous one

{

temp = Force;

Serial.println("Reading");

digitalWrite(Pass\_LED, HIGH);

digitalWrite(Fail\_LED, HIGH);

}

else if (Force < temp) // If the new incoming Force value is less than the previous one

{

if (temp >= F\_Min) // If Previous force value is greater than the Minimum force limit

{

Serial.println(" ");

Serial.println("PASS");

Serial.print("Fmax: ");

Serial.println(temp);

lcd.clear();

lcd.setCursor(1, 0);

lcd.print("PASS");

lcd.setCursor(1, 1);

lcd.print("Fmax: ");

lcd.setCursor(7, 1);

lcd.print(temp);

lcd.setCursor(13, 1);

lcd.print("Kg");

cellStatus = 1;

digitalWrite(Pass\_LED, HIGH);

digitalWrite(Fail\_LED, LOW);

SD\_Results\_Pass();

}

else if (temp < F\_Min) // If Previous force value is less than the Minimum force limit

{

Serial.println(" ");

Serial.println("Fail");

Serial.print("Fmax: ");

Serial.println(temp);

lcd.clear();

lcd.setCursor(1, 0);

lcd.print("Fail");

lcd.setCursor(1, 1);

lcd.print("Fmax: ");

lcd.setCursor(7, 1);

lcd.print(temp);

lcd.setCursor(13, 1);

lcd.print("Kg");

cellStatus = 1;

digitalWrite(Pass\_LED, LOW);

digitalWrite(Fail\_LED, HIGH);

SD\_Results\_Fail(); //Store the results to SD Card

}

}

}

if (result == node.ku8MBResponseTimedOut) //If Communication Breaks between Weight Transmitter and System(Arduino)

{

digitalWrite(Pass\_LED, HIGH);

digitalWrite(Fail\_LED, HIGH);

lcd.clear();

lcd.setCursor(5, 0); //(col, row)

lcd.print("Meter");

lcd.setCursor(1, 1); //(col, row)

lcd.print("Disconnected");

}

}

}

void print\_measure()

{

lcd.clear();

lcd.setCursor(1, 0); //(col, row)

lcd.print("Measuring Mode:");

lcd.setCursor(1, 1); //(col, row)

lcd.print("F\_Min: ");

lcd.setCursor(8, 1); //(col, row)

lcd.print(F\_Min);

lcd.setCursor(13, 1); //(col, row)

lcd.print("Kg");

}

void print\_setting()

{

lcd.clear();

lcd.setCursor(1, 0);

lcd.print("Setting Mode");

lcd.setCursor(1, 1);

lcd.print("Fmin: ");

lcd.setCursor(8, 1); //(col, row)

lcd.print(F\_Min);

lcd.setCursor(13, 1); //(col, row)

lcd.print("Kg");

}

/\*============== Saving to SD Card ==================\*/

void SD\_Results\_Pass()

{

SD.begin(10);

// save new integer every loop and then wait 1s

testfile = SD.open(fileName, FILE\_WRITE);

delay(100);

if (testfile) {

testfile.println(String(F\_Min) + "," + String(temp) + "," + "PASS");

Serial.println(String(F\_Min) + "," + String(temp) + "," + "PASS");

testfile.close();

} else {

Serial.println("error opening file");

}

}

void SD\_Results\_Fail()

{

SD.begin(10);

// save new integer every loop and then wait 1s

testfile = SD.open(fileName, FILE\_WRITE);

delay(100);

if (testfile) {

testfile.println(String(F\_Min) + "," + String(temp) + "," + "FAIL");

Serial.println(String(F\_Min) + "," + String(temp) + "," + "FAIL");

testfile.close();

} else {

Serial.println("error opening file");

}

}

/\*=====================================================\*/