**CODE:**

#include <LiquidCrystal.h>

#include <EEPROM.h>

#include "RTC.h"

#include "Serial.h"

#ifndef RFIDConfig

#define RFIDConfig

#define Card1 "2100203390A2"

#define Card2 "21001BA5EC73"

#define Card3 "210020C05594"

#define Card4 "210020D44693"

#define Card5 "210020BBD06A"

#endif

#ifndef ServerConfig

#define ServerConfig

#define ServerUserName "muthumeenakshi"

#define ServerMailID "projectdkmp@gmail.com"

#define ServerPass "PIC16f877a"

#endif

#ifndef DisplayConfig

#define DisplayConfig

#define RS 13

#define EN 12

#define D4 11

#define D5 10

#define D6 9

#define D7 8

#endif

#ifndef KeyConfig

#define KeyConfig

#define SetKey A0

#define MovKey A1

#define IncKey A2

#define DecKey A3

#define EntKey A4

#endif

#ifndef SensorConfig

#define SensorConfig

#define Lane1IR 7

#define Lane2IR 6

#define Lane3IR 5

#define Lane4IR 4

#endif

#ifndef OutputConfig

#define OutputConig

#define Alarm 3

#endif

#ifndef SecureConfig

#define SecureConfig

#define ClientID "A1E-cC4uturqh2SUBpbvTCv1fYDc7Q17RG"

#define ClientName "MuthuMeena"

#define WiFiName "ESP8266"

#define WiFiPass "PIC16f877a"

#endif

#ifndef IOTConfig

#define IOTConfig

#define Topic "/v1.6/devices/"

#define DeviceLabel "devicelabel"

#define VariableLabel1 "freeslots"

#define VariableLabel2 "regslots"

#define VariableLabel3 "slot1s"

#define VariableLabel4 "slot2s"

#define VariableLabel5 "slot3s"

#define VariableLabel6 "slot4s"

#define SwitchingLabel1 "slot1"

#define SwitchingLabel2 "slot2"

#define SwitchingLabel3 "slot3"

#define SwitchingLabel4 "slot4"

#endif

void SetRTCTime(unsigned char\*, unsigned char\*, unsigned char\*);

void RechargeRFIDCard();

void LCD\_Write(unsigned int);

void LCD\_Write2(unsigned char);

void EntryRFStatus();

void ExitRFStatus();

char ESP\_TalkBack(const char\*, const char\*, unsigned int);

char ESP\_TalkBack(const char\*, unsigned int);

void ESP\_Initialize();

void MQTT\_Connect();

void MQTT\_Publish(const char\*, const char\*, unsigned int);

void MQTT\_Subscribe(const char\*, const char\*);

LiquidCrystal LCD(RS, EN, D4, D5, D6, D7);

char \*sstrstr(const char \*Data, const char \*Find, int Length)

{

char FindLength = strlen(Find);

for(int k=0; k<Length; k++)

{

if(k + FindLength > Length) return 0;

if(strncmp(&Data[k], Find, FindLength)==0) return

(char\*)&Data[k];

}

return NULL;

}

char \*sstrchr(const char \*Data, int Find, int Length)

{

while(\*Data++ != (char)Find)

if(!(Length--)) return 0;

return (char\*)Data;

}

87

char \*sstrrchr(const char \*Data, int Find, int Length)

{

char \*ReturnVariable;

do

{

if(\*Data++ == (char)Find)

ReturnVariable = (char\*)Data;

}while(Length--);

return (char\*)ReturnVariable;

}

unsigned char SerialArray[20], SerialCount;

unsigned char SerialArray1[150], SerialCount1;

unsigned char SerialArray2[15], SerialCount2;

unsigned char SerialArray3[15], SerialCount3;

unsigned char GlobalArray[50];

const char \*Ser2Ptr = (char\*)SerialArray;

const char \*Ser2Ptr1 = (char\*)SerialArray1;

const char \*Ser2Ptr2 = (char\*)SerialArray2;

const char \*Ser2Ptr3 = (char\*)SerialArray3;

const char \*Glo2Ptr = (char\*)GlobalArray;

unsigned int Card1Amount, Card2Amount, Card3Amount;

unsigned int Card4Amount, RegLanes, FreeLanes;

unsigned int Card1InTime, Card2InTime;

unsigned int Card3InTime, Card4InTime;

unsigned int Global1;

unsigned char Hours, Minutes, Seconds;

bool Glo1, Glo2, Glo3, Glo4;

void setup()

{

Serial\_Initialize(9600);

Serial1\_Initialize(9600);

Serial2\_Initialize(9600);

Serial3\_Initialize(9600);

Wire.begin();

LCD.begin(20,4);

Serial\_Write("Arduino Powered...\r\n");

pinMode(SetKey, INPUT\_PULLUP);

pinMode(MovKey, INPUT\_PULLUP);

pinMode(IncKey, INPUT\_PULLUP);

pinMode(DecKey, INPUT\_PULLUP);

pinMode(EntKey, INPUT\_PULLUP);

pinMode(Lane1IR, INPUT);

pinMode(Lane2IR, INPUT);

pinMode(Lane3IR, INPUT);

pinMode(Lane4IR, INPUT);

pinMode(Alarm, OUTPUT);

digitalWrite(Alarm, LOW);

if(EEPROM.read(0) == 0xFF) Card1Amount = 0;

else Card1Amount = EEPROM.read(0) \* 10;

if(EEPROM.read(1) == 0xFF) Card2Amount = 0;

else Card2Amount = EEPROM.read(1) \* 10;

if(EEPROM.read(2) == 0xFF) Card3Amount = 0;

else Card3Amount = EEPROM.read(2) \* 10;

if(EEPROM.read(3) == 0xFF) Card4Amount = 0;

else Card4Amount = EEPROM.read(3) \* 10;

if(EEPROM.read(4) == 0xFF) RegLanes = 0;

else RegLanes = EEPROM.read(4);

FreeLanes = 4;

LCD.setCursor(0,0);

LCD.print("IOT ENABLED PARKING ");

LCD.setCursor(0,1);

LCD.print("SPACE SHARING SYSTEM");

delay(4000); LCD.clear();

ESP\_Initialize();

MQTT\_Connect();

MQTT\_Subscribe(DeviceLabel, SwitchingLabel1);

MQTT\_Subscribe(DeviceLabel, SwitchingLabel2);

MQTT\_Subscribe(DeviceLabel, SwitchingLabel3);

MQTT\_Subscribe(DeviceLabel, SwitchingLabel4);

memset(SerialArray, '\0', sizeof(SerialArray));

SerialCount = 0; memset(GlobalArray, '\0', 50);

}

void loop()

{

Time\_Read(&Hours, &Minutes, &Seconds);

LCD.setCursor(0,0);

LCD.print("TOTAL SLOTS:");

LCD\_Write(4);

LCD.setCursor(0,1);

LCD.print("REGISTER SLOTS:");

LCD\_Write(RegLanes);

LCD.setCursor(0,2);

LCD.print("AVAILABLE SLOTS:");

LCD\_Write(FreeLanes);

LCD.setCursor(0,3);

LCD.print("TIME:");

LCD\_Write2(Hours);

LCD.print("-");

LCD\_Write2(Minutes);

LCD.print("-");

LCD\_Write2(Seconds);

if(!digitalRead(SetKey))

{

while(!digitalRead(SetKey));

SetRTCTime(&Hours, &Minutes, &Seconds);

Time\_Write(Hours, Minutes, Seconds);

LCD.clear();

}

if(!digitalRead(MovKey))

{

while(!digitalRead(MovKey));

LCD.clear();

RechargeRFIDCard();

LCD.clear();

}

if(digitalRead(Lane1IR) && !Glo1)

{

FreeLanes -= 1;

MQTT\_Publish(DeviceLabel, VariableLabel1, FreeLanes);

MQTT\_Publish(DeviceLabel, VariableLabel3, 1);

Glo1 = 1;

}

if(!digitalRead(Lane1IR) && Glo1)

{

FreeLanes += 1;

MQTT\_Publish(DeviceLabel, VariableLabel1, FreeLanes);

MQTT\_Publish(DeviceLabel, VariableLabel3, 0);

Glo1 = 0;

}

if(digitalRead(Lane2IR) && !Glo2)

{

FreeLanes -= 1;

MQTT\_Publish(DeviceLabel, VariableLabel1, FreeLanes);

MQTT\_Publish(DeviceLabel, VariableLabel4, 1);

Glo2 = 1;

}

if(!digitalRead(Lane2IR) && Glo2)

{

FreeLanes += 1;

MQTT\_Publish(DeviceLabel, VariableLabel1, FreeLanes);

MQTT\_Publish(DeviceLabel, VariableLabel4, 0);

Glo2 = 0;

}

if(digitalRead(Lane3IR) && !Glo3)

{

FreeLanes -= 1;

MQTT\_Publish(DeviceLabel, VariableLabel1, FreeLanes);

MQTT\_Publish(DeviceLabel, VariableLabel5, 1);

Glo3 = 1;

}

if(!digitalRead(Lane3IR) && Glo3)

{

FreeLanes += 1;

MQTT\_Publish(DeviceLabel, VariableLabel1, FreeLanes);

MQTT\_Publish(DeviceLabel, VariableLabel5, 0);

Glo3 = 0;

}

if(digitalRead(Lane4IR) && !Glo4)

{

FreeLanes -= 1;

MQTT\_Publish(DeviceLabel, VariableLabel1, FreeLanes);

MQTT\_Publish(DeviceLabel, VariableLabel6, 1);

Glo4 = 1;

}

if(!digitalRead(Lane4IR) && Glo4)

{

FreeLanes += 1;

MQTT\_Publish(DeviceLabel, VariableLabel1, FreeLanes);

MQTT\_Publish(DeviceLabel, VariableLabel6, 0);

Glo4 = 0;

}

if(sstrstr(Ser2Ptr1, SwitchingLabel1, SerialCount1)!='\0')

{

delay(50);

Glo2Ptr = sstrrchr(Ser2Ptr1, '/', SerialCount1);

RegLanes = (sstrstr(Glo2Ptr, "1", 5)!='\0' ?RegLanes+1

:RegLanes-1);

memset(SerialArray1, '\0', sizeof(SerialArray1));

memset(GlobalArray, '\0', sizeof(GlobalArray));

MQTT\_Publish(DeviceLabel, VariableLabel2, RegLanes);

EEPROM.write(0x04, RegLanes);

}

if(sstrstr(Ser2Ptr1, SwitchingLabel2, SerialCount1)!='\0')

{

delay(50);

Glo2Ptr = sstrrchr(Ser2Ptr1, '/', SerialCount1);

RegLanes = (sstrstr(Glo2Ptr, "1", 5)!='\0' ?RegLanes+1

:RegLanes-1);

memset(SerialArray1, '\0', sizeof(SerialArray1));

memset(GlobalArray, '\0', sizeof(GlobalArray));

MQTT\_Publish(DeviceLabel, VariableLabel2, RegLanes);

EEPROM.write(0x04, RegLanes);

}

if(sstrstr(Ser2Ptr1, SwitchingLabel3, SerialCount1)!='\0')

{

delay(50);

Glo2Ptr = sstrrchr(Ser2Ptr1, '/', SerialCount1);

RegLanes = (sstrstr(Glo2Ptr, "1", 5)!='\0' ?RegLanes+1

:RegLanes-1);

memset(SerialArray1, '\0', sizeof(SerialArray1));

memset(GlobalArray, '\0', sizeof(GlobalArray));

MQTT\_Publish(DeviceLabel, VariableLabel2, RegLanes);

EEPROM.write(0x04, RegLanes);

}

if(sstrstr(Ser2Ptr1, SwitchingLabel4, SerialCount1)!='\0')

{

delay(50);

Glo2Ptr = sstrrchr(Ser2Ptr1, '/', SerialCount1);

RegLanes = (sstrstr(Glo2Ptr, "1", 5)!='\0' ?RegLanes+1

:RegLanes-1);

memset(SerialArray1, '\0', sizeof(SerialArray1));

memset(GlobalArray, '\0', sizeof(GlobalArray));

MQTT\_Publish(DeviceLabel, VariableLabel2, RegLanes);

EEPROM.write(0x04, RegLanes);

}

if(SerialCount2 > 11) {LCD.clear(); EntryRFStatus(); LCD.clear();}

if(SerialCount3 > 11) {LCD.clear(); ExitRFStatus(); LCD.clear();}

}

void EntryRFStatus()

{

char LocalHours, LocalMinutes, LocalSeconds;

Time\_Read(&LocalHours, &LocalMinutes, &LocalSeconds);

if(strcmp(Ser2Ptr2, Card1)==0)

{

LCD.clear();

LCD.setCursor(0,0);

LCD.print(" USER - 1 ");

LCD.setCursor(0,1);

LCD.print("TN-59 AB 4567");

LCD.setCursor(0,2);

LCD.print("CB:");

LCD.print(Card1Amount);

LCD.setCursor(0,3);

LCD.print("TIME:");

LCD\_Write(LocalHours);

LCD.print("-");

LCD\_Write(LocalMinutes);

LCD.print("-");

LCD\_Write(LocalSeconds);

if(Card1Amount < 100)

{

digitalWrite(Alarm, HIGH);

delay(1000);

digitalWrite(Alarm, LOW);

}

Card1InTime = LocalHours;

memset(SerialArray2, '\0', sizeof(SerialArray2));

SerialCount2 = 0; delay(1000);

}

if(strcmp(Ser2Ptr2, Card2)==0)

{

LCD.clear();

LCD.setCursor(0,0);

LCD.print(" USER - 2 ");

LCD.setCursor(0,1);

LCD.print("TN-64 AJ 4545");

LCD.setCursor(0,2);

LCD.print("CB:");

LCD.print(Card2Amount);

LCD.setCursor(0,3);

LCD.print("TIME:");

LCD\_Write(LocalHours);

LCD.print("-");

LCD\_Write(LocalMinutes);

LCD.print("-");

LCD\_Write(LocalSeconds);

if(Card2Amount < 100)

{

digitalWrite(Alarm, HIGH);

delay(1000);

digitalWrite(Alarm, LOW);

}

Card2InTime = LocalHours;

memset(SerialArray2, '\0', sizeof(SerialArray2));

SerialCount2 = 0; delay(1000);

}

if(strcmp(Ser2Ptr2, Card3)==0)

{

LCD.clear();

LCD.setCursor(0,0);

LCD.print(" USER - 3 ");

LCD.setCursor(0,1);

LCD.print("TN-58 BA 2205");

LCD.setCursor(0,2);

LCD.print("CB:");

LCD.print(Card3Amount);

LCD.setCursor(0,3);

LCD.print("TIME:");

LCD\_Write(LocalHours);

LCD.print("-");

LCD\_Write(LocalMinutes);

LCD.print("-");

LCD\_Write(LocalSeconds);

if(Card3Amount < 100)

{

digitalWrite(Alarm, HIGH);

delay(1000);

digitalWrite(Alarm, LOW);

}

Card3InTime = LocalHours;

memset(SerialArray2, '\0', sizeof(SerialArray2));

SerialCount2 = 0; delay(1000);

}

if(strcmp(Ser2Ptr2, Card4)==0)

{

LCD.clear();

LCD.setCursor(0,0);

LCD.print(" USER - 4 ");

LCD.setCursor(0,1);

LCD.print("TN-59 CD 1212");

LCD.setCursor(0,2);

LCD.print("CB:");

LCD.print(Card4Amount);

LCD.setCursor(0,3);

LCD.print("TIME:");

LCD\_Write(LocalHours);

LCD.print("-");

LCD\_Write(LocalMinutes);

LCD.print("-");

LCD\_Write(LocalSeconds);

if(Card4Amount < 100)

{

digitalWrite(Alarm, HIGH);

delay(1000);

digitalWrite(Alarm, LOW);

}

Card4InTime = LocalHours;

memset(SerialArray2, '\0', sizeof(SerialArray2));

SerialCount2 = 0; delay(1000);

}

if(strcmp(Ser2Ptr2, Card5)==0)

{

digitalWrite(Alarm, HIGH);

delay(5000);

digitalWrite(Alarm, LOW);

memset(SerialArray2, '\0', sizeof(SerialArray2));

SerialCount2 = 0;

}

}

void ExitRFStatus()

{

char LocalHours, LocalMinutes, LocalSeconds;

Time\_Read(&LocalHours, LocalMinutes, LocalSeconds);

if(strcmp(Ser2Ptr3, Card1)==0)

{

LCD.clear();

LCD.setCursor(0,0);

LCD.print(" USER - 1 ");

LCD.setCursor(0,1);

LCD.print("TN-59 AB 4567");

LCD.setCursor(0,2);

LCD.print("CB:");

LCD.print(Card1Amount);

LCD.setCursor(0,3);

LCD.print("PC:");

Global1 = LocalHours - Card1InTime;

if(Card1Amount < 100)

{

digitalWrite(Alarm, HIGH);

delay(1000);

digitalWrite(Alarm, LOW);

}

else if(!Global1)

{

LCD\_Write(100);

Card1Amount -= 100;

EEPROM.write(0x00, Card1Amount / 10);

}

else

{

LCD\_Write(100 \* Global1);

Card1Amount -= (100 \* Global1);

EEPROM.write(0x00, Card1Amount / 10);

}

memset(SerialArray3, '\0', sizeof(SerialArray3));

SerialCount3 = 0; delay(1000);

}

if(strcmp(Ser2Ptr3, Card2)==0)

{

LCD.clear();

LCD.setCursor(0,0);

LCD.print(" USER - 2 ");

LCD.setCursor(0,1);

LCD.print("TN-64 AJ 4545");

LCD.setCursor(0,2);

LCD.print("CB:");

LCD.print(Card2Amount);

LCD.setCursor(0,3);

LCD.print("PC:");

Global1 = LocalHours - Card2InTime;

if(Card2Amount < 100)

{

digitalWrite(Alarm, HIGH);

delay(1000);

digitalWrite(Alarm, LOW);

}

else if(!Global1)

{

LCD\_Write(100);

Card2Amount -= 100;

EEPROM.write(0x01, Card2Amount / 10);

}

else

{

LCD\_Write(100 \* Global1);

Card2Amount -= (100 \* Global1);

EEPROM.write(0x01, Card2Amount / 10);

}

memset(SerialArray3, '\0', sizeof(SerialArray3));

SerialCount3 = 0; delay(1000);

}

if(strcmp(Ser2Ptr3, Card3)==0)

{

LCD.clear();

LCD.setCursor(0,0);

LCD.print(" USER - 3 ");

LCD.setCursor(0,1);

LCD.print("TN-58 BA 2205");

LCD.setCursor(0,2);

LCD.print("CB:");

LCD.print(Card3Amount);

LCD.setCursor(0,3);

LCD.print("PC:");

Global1 = LocalHours - Card3InTime;

if(Card3Amount < 100)

{

digitalWrite(Alarm, HIGH);

delay(1000);

digitalWrite(Alarm, LOW);

}

else if(!Global1)

{

LCD\_Write(100);

Card3Amount -= 100;

EEPROM.write(0x02, Card3Amount / 10);

}

else

{

LCD\_Write(100 \* Global1);

Card3Amount -= (100 \* Global1);

EEPROM.write(0x02, Card3Amount / 10);

}

memset(SerialArray3, '\0', sizeof(SerialArray3));

SerialCount3 = 0; delay(1000);

}

if(strcmp(Ser2Ptr3, Card4)==0)

{

LCD.clear();

LCD.setCursor(0,0);

LCD.print(" USER - 4 ");

LCD.setCursor(0,1);

LCD.print("TN-59 CD 1212");

LCD.setCursor(0,2);

LCD.print("CB:");

LCD.print(Card1Amount);

LCD.setCursor(0,3);

LCD.print("PC:");

Global1 = LocalHours - Card1InTime;

if(Card4Amount < 100)

{

digitalWrite(Alarm, HIGH);

delay(1000);

digitalWrite(Alarm, LOW);

}

else if(!Global1)

{

LCD\_Write(100);

Card4Amount -= 100;

EEPROM.write(0x03, Card4Amount / 10);

}

else

{

LCD\_Write(100 \* Global1);

Card4Amount -= (100 \* Global1);

EEPROM.write(0x03, Card1Amount / 10);

}

memset(SerialArray3, '\0', sizeof(SerialArray3));

SerialCount3 = 0; delay(1000);

}

if(strcmp(Ser2Ptr3, Card5)==0)

{

digitalWrite(Alarm, HIGH);

delay(5000);

digitalWrite(Alarm, LOW);

memset(SerialArray3, '\0', sizeof(SerialArray3));

SerialCount3 = 0;

}

}

void SetRTCTime(unsigned char \*LHours, unsigned char \*LMinutes,

unsigned char \*LSeconds)

{

LCD.setCursor(0,3);

LCD.print(" ");

unsigned char LocalArray[5], CursorPosition = 0, Value;

LCD.setCursor(0,3);

LCD.print("TIME: - -");

LCD.cursor();

while(digitalRead(EntKey))

{

if(!digitalRead(IncKey))

{

while(!digitalRead(IncKey));

Value++;

if(!CursorPosition && Value > 23) Value = 0;

else if(CursorPosition && Value > 59) Value = 0;

}

if(!digitalRead(DecKey))

{

while(!digitalRead(DecKey));

Value--;

if(!CursorPosition && Value > 23) Value = 23;

else if(CursorPosition && Value > 59) Value = 59;

}

if(!digitalRead(MovKey))

{

while(!digitalRead(MovKey));

CursorPosition++;

if(CursorPosition > 2) CursorPosition = 0;

}

if(CursorPosition == 0)

{

LCD.setCursor(5,3);

LCD\_Write2(Value);

}

if(CursorPosition == 1)

{

LCD.setCursor(8,3);

LCD\_Write2(Value);

}

if(CursorPosition == 2)

{

LCD.setCursor(11,3);

LCD\_Write2(Value);

}

LocalArray[CursorPosition] = Value;

}

while(!digitalRead(EntKey));

LCD.clear(); LCD.noCursor();

\*LHours = LocalArray[0];

\*LMinutes = LocalArray[1];

\*LSeconds = LocalArray[2];

}

void RechargeRFIDCard()

{

unsigned char Counting;

unsigned int QueryBalance;

unsigned int LocalVariable1 = Card1Amount;

unsigned int LocalVariable2 = Card2Amount;

unsigned int LocalVariable3 = Card3Amount;

unsigned int LocalVariable4 = Card4Amount;

do

{

if(strcmp(Ser2Ptr2, Card1) == 0)

{

LCD.setCursor(0,0);

LCD.print(" TN-59 AB 4567 ");

LCD.setCursor(0,1);

LCD.print("RECHARGE:");

LCD.print(LocalVariable1);

LCD.setCursor(13,1);

LCD.print("+");

LCD.print(QueryBalance);

if(!digitalRead(IncKey))

{

while(!digitalRead(IncKey));

Counting++; delay(200);

if(Counting > 250) Counting = 0;

QueryBalance = (Counting \* 10);

LocalVariable1 = Card1Amount + QueryBalance;

}

if(!digitalRead(DecKey))

{

while(!digitalRead(DecKey));

Counting--; delay(200);

if(Counting > 250) Counting = 250;

QueryBalance = (Counting \* 10);

LocalVariable1 = Card1Amount - QueryBalance;

}

}

if(strcmp(Ser2Ptr2, Card2) == 0)

{

LCD.setCursor(0,0);

LCD.print(" TN-64 AJ 4545 ");

LCD.setCursor(0,1);

LCD.print("RECHARGE:");

LCD.print(LocalVariable2);

LCD.setCursor(13,1);

LCD.print("+");

LCD.print(QueryBalance);

if(!digitalRead(IncKey))

{

while(!digitalRead(IncKey));

Counting++; delay(200);

if(Counting > 250) Counting = 0;

QueryBalance = (Counting \* 10);

LocalVariable2 = Card2Amount + QueryBalance;

}

if(!digitalRead(DecKey))

{

while(!digitalRead(DecKey));

Counting--; delay(200);

if(Counting > 250) Counting = 250;

QueryBalance = (Counting \* 10);

LocalVariable2 = Card2Amount - QueryBalance;

}

}

if(strcmp(Ser2Ptr2, Card3) == 0)

{

LCD.setCursor(0,0);

LCD.print(" TN-58 BA 2205 ");

LCD.setCursor(0,1);

LCD.print("RECHARGE:");

LCD.print(LocalVariable3);

LCD.setCursor(13,1);

LCD.print("+");

LCD.print(QueryBalance);

if(!digitalRead(IncKey))

{

while(!digitalRead(IncKey));

Counting++; delay(200);

if(Counting > 250) Counting = 0;

QueryBalance = (Counting \* 10);

LocalVariable3 = Card3Amount + QueryBalance;

}

if(!digitalRead(DecKey))

{

while(!digitalRead(DecKey));

Counting--; delay(200);

if(Counting > 250) Counting = 250;

QueryBalance = (Counting \* 10);

LocalVariable3 = Card3Amount - QueryBalance;

}

}

if(strcmp(Ser2Ptr2, Card4) == 0)

{

LCD.setCursor(0,0);

LCD.print(" TN-59 CD 1212 ");

LCD.setCursor(0,1);

LCD.print("RECHARGE:");

LCD.print(LocalVariable4);

LCD.setCursor(13,1);

LCD.print("+");

LCD.print(QueryBalance);

if(!digitalRead(IncKey))

{

while(!digitalRead(IncKey));

Counting++; delay(200);

if(Counting > 250) Counting = 0;

QueryBalance = (Counting \* 10);

LocalVariable4 = Card4Amount + QueryBalance;

}

if(!digitalRead(DecKey))

{

while(!digitalRead(DecKey));

Counting--; delay(200);

if(Counting > 250) Counting = 250;

QueryBalance = (Counting \* 10);

LocalVariable4 = Card4Amount - QueryBalance;

}

}

}

while(digitalRead(EntKey));

memset(SerialArray2, '\0', sizeof(SerialArray));

SerialCount2 = 0;

Card1Amount = LocalVariable1;

Card2Amount = LocalVariable2;

Card3Amount = LocalVariable3;

Card4Amount = LocalVariable4;

EEPROM.write(0x00, Card1Amount / 10);

EEPROM.write(0x01, Card2Amount / 10);

EEPROM.write(0x02, Card3Amount / 10);

EEPROM.write(0x03, Card4Amount / 10);

Card1Amount = EEPROM.read(0) \* 10;

Card2Amount = EEPROM.read(1) \* 10;

Card3Amount = EEPROM.read(2) \* 10;

Card4Amount = EEPROM.read(3) \* 10;

delay(2000); LCD.clear();

}

void LCD\_Write(unsigned int Data)

{

LCD.write((Data / 100) + 0x30);

LCD.write(((Data % 100) / 10) + 0x30);

LCD.write(((Data % 100) % 10) + 0x30);

}

void LCD\_Write2(unsigned char Data)

{

LCD.write((Data / 10) + 0x30);

LCD.write((Data % 10) + 0x30);

}

char ESP\_TalkBack(const char \*Data, const char \*Feed, unsigned int

Delay)

{

unsigned long int Timer = millis();

memset(SerialArray1, '\0', sizeof(SerialArray1));

SerialCount1 = 0; Serial1\_Write(Data); Serial1\_Write("\r\n");

do

{

if(strstr(Ser2Ptr1, Feed)!='\0') return 1;

if(strstr(Ser2Ptr1, "ERROR")!='\0') return 0;

}

while(millis() < (Timer + Delay));

return 0;

}

char ESP\_TalkBack(const char \*Feed, unsigned int Delay)

{

unsigned long int Timer = millis();

do

{

if(sstrstr(Ser2Ptr1, Feed, SerialCount1)!='\0') return 1;

}

while(millis() < (Timer + Delay));

return 0;

}

void ESP\_Initialize()

{

LCD.clear(); LCD.setCursor(0,0);

if(ESP\_TalkBack("AT","OK",2500))

{

Serial\_Write("ESP Module Found...\r\n");

LCD.print("ESP MODEL FOUND");

}

else

{

Serial\_Write("ESP Module Not Found...\r\n");

LCD.print("ESP NOT FOUND");

}

delay(1000);

LCD.clear(); LCD.setCursor(0,0);

if(ESP\_TalkBack("ATE0","OK",2500))

{

Serial\_Write("Echo Cleared...\r\n");

LCD.print("ECHO CLEARED...");

}

else

{

Serial\_Write("Echo Not Cleared...\r\n");

LCD.print("ECHO NOT CLEAR...");

}

delay(1000);

LCD.clear(); LCD.setCursor(0,0);

if(ESP\_TalkBack("AT+CWMODE=3","OK",2500))

{

Serial\_Write("Soft + Ap Mode...\r\n");

LCD.print("SOFT + AP MODE");

}

else

{

Serial\_Write("Configuration Error...\r\n");

LCD.print("CONFIG ERROR....");

}

delay(1000);

LCD.clear(); LCD.setCursor(0,0);

memset(GlobalArray, '\0', sizeof(GlobalArray));

strcat(GlobalArray, "AT+CWJAP=\"");

strncat(GlobalArray, WiFiName, strlen(WiFiName));

strcat(GlobalArray, "\",\"");

strncat(GlobalArray, WiFiPass, strlen(WiFiPass));

strcat(GlobalArray, "\"");

if(ESP\_TalkBack(GlobalArray, "OK", 25000))

{

Serial\_Write("WiFi Connected...\r\n");

LCD.print("WiFi CONFIGURED");

}

else

{

Serial\_Write("WiFi Not Connected...\r\n");

LCD.print("WiFi NOT CONN");

}

delay(1000);

LCD.clear(); LCD.print(0,0);

if(ESP\_TalkBack("AT+CIPSTART=\"TCP\",\"things.ubidots.com\",1883","C

ONNECT",10000))

{

Serial\_Write("TCP Connection Established...\r\n");

LCD.print("SERVER CONNECTED");

}

else

{

Serial\_Write("TCP Connection Not Fixed...\r\n");

LCD.print("SERVER ERROR");

}

delay(1000);

memset(GlobalArray, '\0', sizeof(GlobalArray));

}

void MQTT\_Connect()

{

const char \*LocalArray = "\x00\x04\x4D\x51\x54\x54\x04\x82";

unsigned char Length = 2 + 8 + 2 + 2 + strlen(ClientName) + 2 +

strlen(ClientID);

Serial1\_Write("AT+CIPSEND="); Serial1\_Decimal(Length);

ESP\_TalkBack("",">",3000);

Serial1\_Send(0x10); Serial1\_Send(Length - 2);

Serial1\_Write(LocalArray, 8);

Serial1\_Send(0x00); Serial1\_Send(0x00);

Serial1\_Send(strlen(ClientName) >> 8);

Serial1\_Send(strlen(ClientName) & 0xFF);

Serial1\_Write(ClientName);

Serial1\_Send(strlen(ClientID) >> 8);

Serial1\_Send(strlen(ClientID) & 0xFF);

Serial1\_Write(ClientID);

LCD.clear(); LCD.setCursor(0,0);

if(ESP\_TalkBack("\x02\x00\x00",5000))

{

Serial\_Write("MQTT Connection Established...\r\n");

LCD.print("MQTT CONNECTED");

}

else

{

Serial\_Write("MQTT Not Connected...\r\n");

LCD.print("MQTT NOT CONNECT");

}

delay(1000); LCD.clear();

}

void MQTT\_Publish(const char \*DevLab, const char \*VarLab, unsigned

int PayLoad)

{

unsigned char LocalArray[6];

itoa(PayLoad, LocalArray, 10);

unsigned int TopicLength = strlen(Topic) + strlen(DevLab);

unsigned int PayLoadLength = strlen(VarLab) + strlen(LocalArray) +

15;

unsigned char Length = 2 + 2 + TopicLength + 2 + PayLoadLength;

Serial1\_Write("AT+CIPSEND="); Serial1\_Decimal(Length);

ESP\_TalkBack("",">",3000);

Serial1\_Send(0x32); Serial1\_Send(Length - 2);

Serial1\_Send(TopicLength >> 8);

Serial1\_Send(TopicLength & 0xFF);

Serial1\_Write(Topic);

Serial1\_Write(DevLab);

Serial1\_Send(0x00);

Serial1\_Send(0x01);

Serial1\_Write("{\""); Serial1\_Write(VarLab);

Serial1\_Write("\":{\"value\":");

Serial1\_Write(LocalArray); Serial1\_Write("}}");

LCD.clear(); LCD.setCursor(0,0);

if(ESP\_TalkBack("\x40\x02\x00\x01",5000))

{

Serial\_Write("Data Published...\r\n");

LCD.print("DATA PUBLISHED");

}

else

{

Serial\_Write("Data Not Published...\r\n");

LCD.print("DATA NOT PUBLISH");

}

delay(1000); LCD.clear();

}

void MQTT\_Subscribe(const char \*DevLab, const char \*VarLab)

{

static unsigned int SubscribeMessageID;

unsigned int TopicLength = strlen(Topic) + strlen(DevLab) +

strlen(VarLab) + 4;

unsigned char Length = 2 + 2 + 2 + TopicLength + 1;

SubscribeMessageID++; //Don't Change or Delete

Serial1\_Write("AT+CIPSEND="); Serial1\_Decimal(Length);

ESP\_TalkBack("",">",3000);

Serial1\_Send(0x82); Serial1\_Send(Length - 2);

Serial1\_Send(SubscribeMessageID >> 8);

Serial1\_Send(SubscribeMessageID & 0xFF);

Serial1\_Send(TopicLength >> 8);

Serial1\_Send(TopicLength & 0xFF);

Serial1\_Write(Topic); Serial1\_Write(DevLab);

Serial1\_Send('/'); Serial1\_Write(VarLab);

Serial1\_Write("/lv"); Serial1\_Send(0x01);

Glo2Ptr = "\x90\x03\x00\x00\x01";

GlobalArray[3] = SubscribeMessageID & 0xFF;

LCD.clear(); LCD.setCursor(0,0);

if(ESP\_TalkBack(Glo2Ptr, 5000))

{

Serial\_Write("Topic ");

Serial\_Decimal(SubscribeMessageID);

Serial\_Write(" Subscribed...\r\n");

LCD.print("TOPIC ");

LCD.print(SubscribeMessageID, DEC);

LCD.print(" SUBSCRI");

}

else

{

Serial\_Write("Topic ");

Serial\_Decimal(SubscribeMessageID);

Serial\_Write(" Not Subscribed..\r\n");

LCD.print("TOPIC ");

LCD.print(SubscribeMessageID, DEC);

LCD.print(" ERROR");

}

delay(1000); LCD.clear();

}

ISR(USART0\_RX\_vect)

{

SerialArray[SerialCount] = Serial\_Receive();

SerialCount = (SerialCount < 20 ?SerialCount+1 :0);

}

ISR(USART1\_RX\_vect)

{

SerialArray1[SerialCount1] = Serial1\_Receive();

SerialCount1 = (SerialCount1 < 150 ?SerialCount1+1 :0);

}

ISR(USART2\_RX\_vect)

{

SerialArray2[SerialCount2] = Serial2\_Receive();

SerialCount2 = (SerialCount2 < 15 ?SerialCount2+1 :0);

}

ISR(USART3\_RX\_vect)

{

SerialArray3[SerialCount3] = Serial3\_Receive();

SerialCount3 = (SerialCount3 < 15 ?SerialCount3+1 :0);

}