College of Engineering
Mechatronics and mechanical engineering department
MECA442 microcontrollers for mechatronics
Fall 2023

Smart Plug

Report Submitted by:

Ahmad Hammoudeh – 20210482

Ahmad Achaji – 20210313

Semester: Fall 2023

Report Submitted to:

Ahmad koubeissi

Section 1: Project Description

The objective of this Project is to create a smart plug that goes beyond traditional outlets, offering users the ability to precisely schedule the operating hours of connected devices, set specific limits on the maximum allowable current, ensure safe and efficient energy use, and allow users to establish both maximum and minimum voltage thresholds, adding an extra layer of customization to accommodate diverse electrical needs.

Section 2: Functional Specifications

- **Scheduling and Timing:** To schedule the period of operation the user must use the keypad on top of the plug and choose option B. Option B will allow the user to enter (dd-hh-mm) which means the day- hour-min of the start and the end of the period. To create the time base we used a high-accuracy RTC (Real Time Clock): **(DS3231).**
  Note: The minimum period allowed is 1 minute.
- **Current Control:** The smart plug was designed to withstand a current of up to 10 Amps, but the user can only draw a maximum of 5 Amps. Using the keypad, the user can choose the maximum current allowed in the circuit, but if the user tries to enter a maximum value of current higher than 5 Amps the program will not accept this number and will use the old valid value of I max. The plug is equipped with a current sensor that can measure up to 20 Amps **(ACS712 20A),** and a 3 Amps circuit breaker which can allow the passage of 7 - 10 Amps before turning off.
- **Voltage Parameters:** By using the **(ZMPT101B)** we were able to monitor the value of the voltage to ensure that it does not go beyond the allowed range. The user can choose the voltage range of operation, but the program will not allow any voltages above 250 or under 200 for the safety of the user and the components.
- **User Interface and Interaction:** The plug's interface is made of 3 parts: first, the main screen shows the current, voltage, and power usage of the connected load. If the user presses **(A)** the current and voltage choosing menu will appear. If the user presses **(B)** the user can choose the start and the end of the period**.** Pressing the **(C)** key to confirm values. And the **(D)** key to return to the main screen.
- **Safety Measures:** The plug is equipped with a circuit breaker which will protect the circuit from any short circuits. The circuit breaker is rated for 4.5KA which means if a short circuit happens causing thousands of amps to flow, the breaker is certified to be able to interrupt it if it's less than 4,500 Amps. The program is designed in a way that if the current is higher than 5Amps the circuit will turn off to protect the Relay from high current values.

Section 3: Block Diagram

To illustrate the internal architecture and interconnections of the Smart Plug, a detailed block diagram of the circuit is provided below (Figure 1).

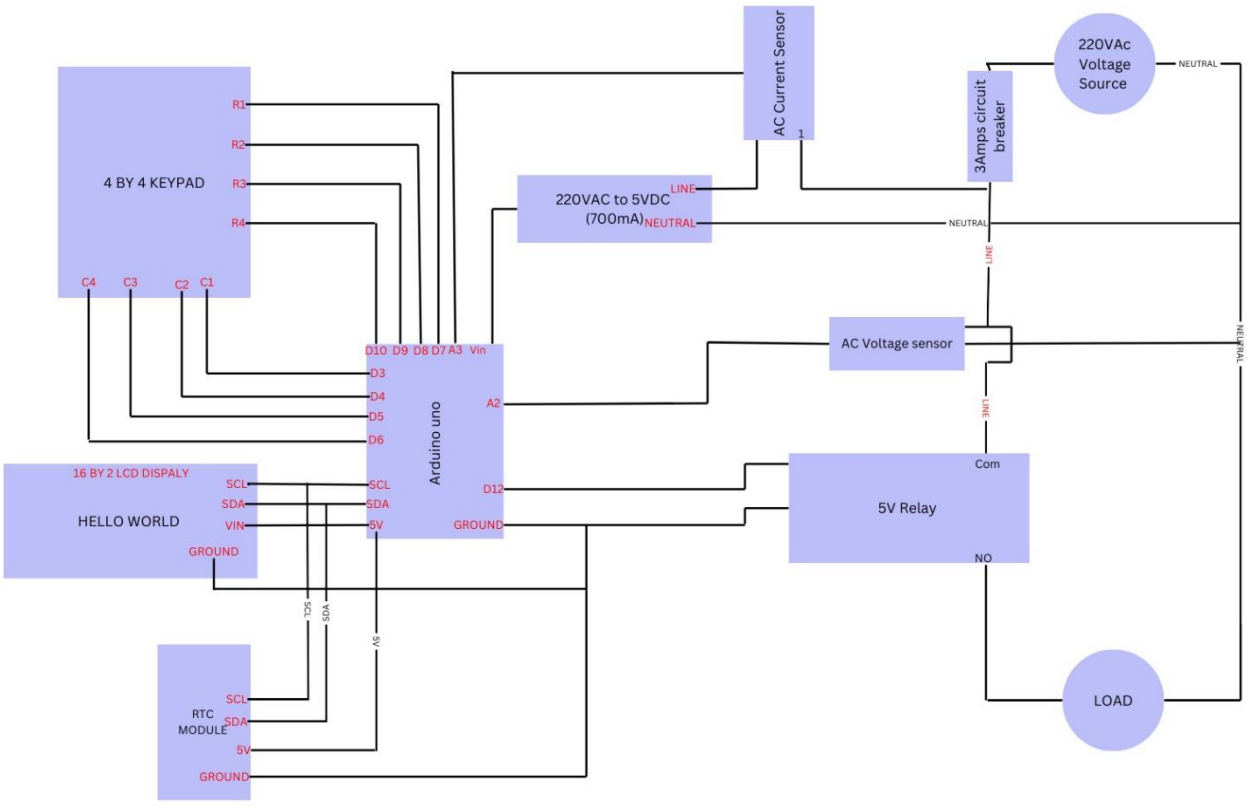


Figure 1: Block Diagram of the internal components

Section 4: Principle of Operation:

1- When the plug is turned on, the display will turn on, display "HELLO WORLD" for 2 seconds, and then display the voltage, current, and power consumed by the load, and show if the event in the plug is On or Off. As shown in figure 2.

Figure 2: Main Screen

2- The user has 2 other screens he can access after the main screen, by pressing 'A' the user can access the max/min voltage and current selection menu, and the start/end time menu by pressing 'B'. As shown in Figures 3 and 4.

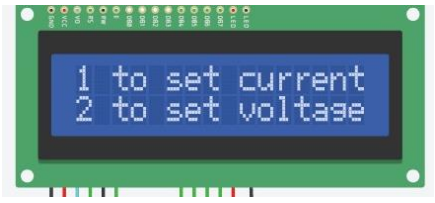


Figure 3: I and V menu

Figure 4: Start/ End period menu.

3- by pressing 1 the user can choose the maximum allowable current to pass trough the load. Note: The maximum current that the is allowed to pass is 5Amps due to safety measures if the user tries to enter a value higher than 5 a error screen will pop up and the new values will not be saved. As shown in figures 5 and 6. **Note:** The program will also not allow the minimum V to be higher than the max V, nor the start time to be after the end time and will notify the user about the error as logical error. As shown in figure 7, 8.

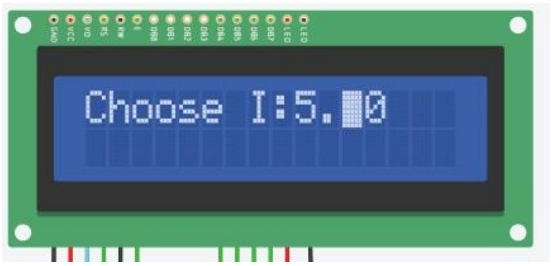


Figure 5: Choosing value of I

Figure 6: unsupported value error.

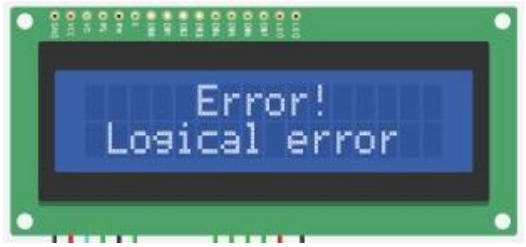


Figure 7: Logical error screen.

Figure 8: Start/end time menu.

4- If the user wants to add an event, he can access the start and end time as shown in Figure 8, when the time is equal to the start time the relay will be ON until the time reaches the end time and it will remain OFF until the user presses the option of ending the event. When the user chooses to end the event, the plug will get back to its normal operation. Note: The user can end the event at any time even if he is in the start time and the relay is ON.

Figure 9: Event is OFF

Figure 10: Event is ON

Section 5: Complete Schematic Diagram:

Note: the screen used in the schematic is a normal 16 by 2 instead of the I2C version used in the real circuit, and the voltage / current sensors are replaced by potentiometers due to them being unavailable in the Software.

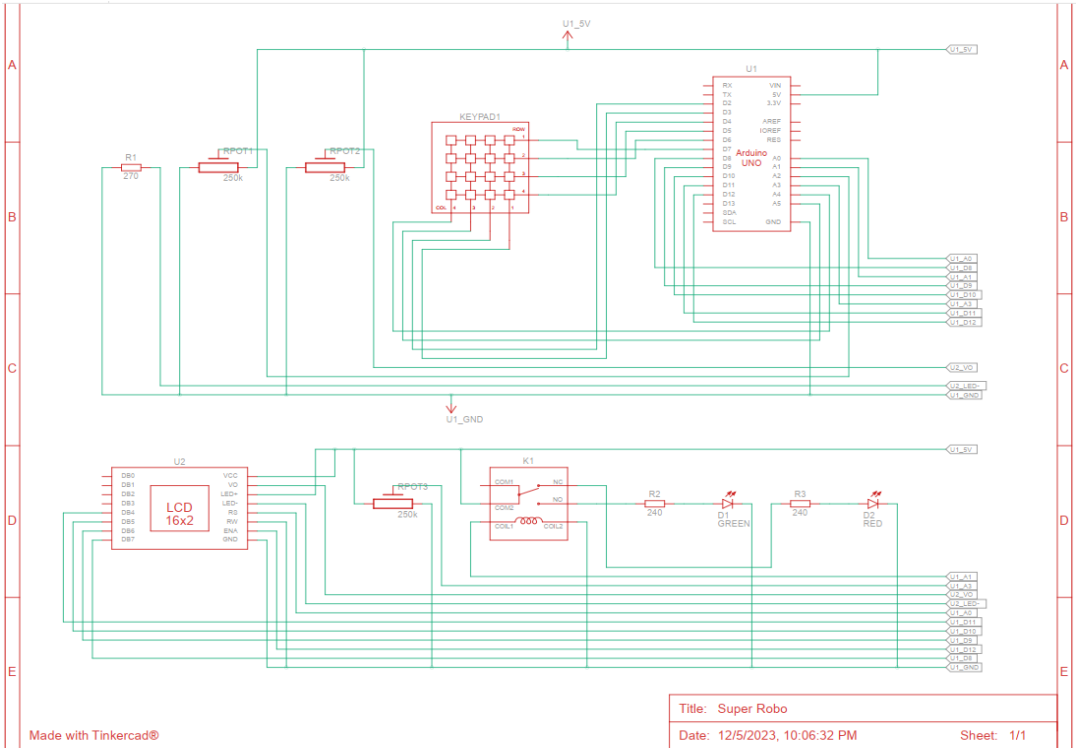


Figure 11: schematic of the circuit using Tinker cad

Section 6: Simulation

Figure 12 and Figure 13 represents the simulation of the experiment using TinkerCad and Proteus respectively.

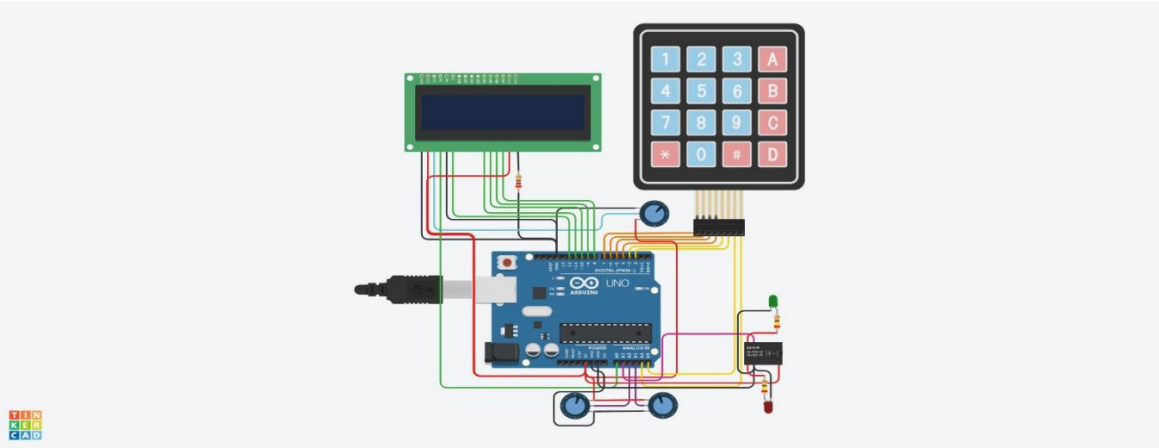


Figure 12: Tinker-CAD Simulation

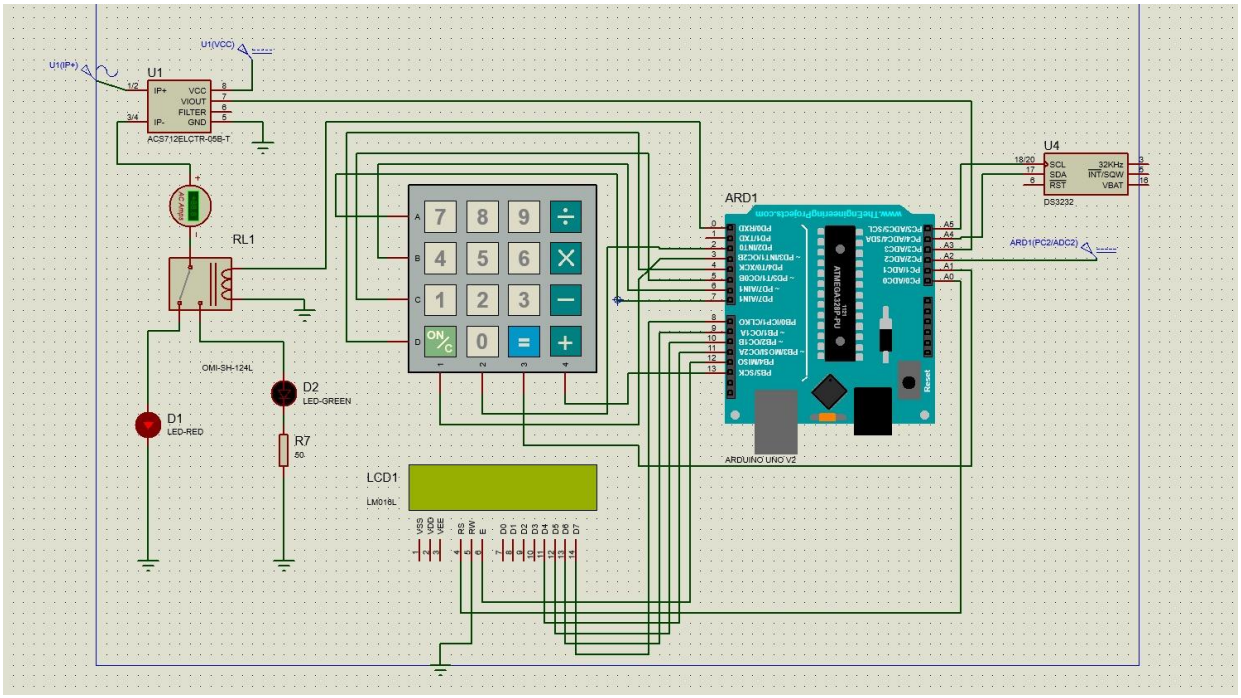


Figure 13: Proteus Simulation

Section 7: Bill of Materials

Table 1 represents the components used in the experiment as well as the quantities.

Device	Quantity
Arduino UNO	x1
Ds3231 RTC module	x1
1 Channel 5V Relay	x1
220VAC To 5VDC Converter	x1
4 By 4 Keypad	x1
ZMPT101B	x1
ACS712 20Amps	x1
LCD I2C 16 by 2 Screen	x1
220V Small lamps	x2
Circuit breaker 3 Amps	x1

Table 1: Bills of Materials

Section 8: Results:

Note: For the Smart plug to function correctly it needs to stay connected to the laptop USB port or any other external power supply. Because the converter used although rated for 700mAmps is not delivering the full amount specified so there is a lack of power. One of the solutions was to use buck booster to specify the amount of current needed (700 -750mAmps), but we didn't find any version compatible with our application.

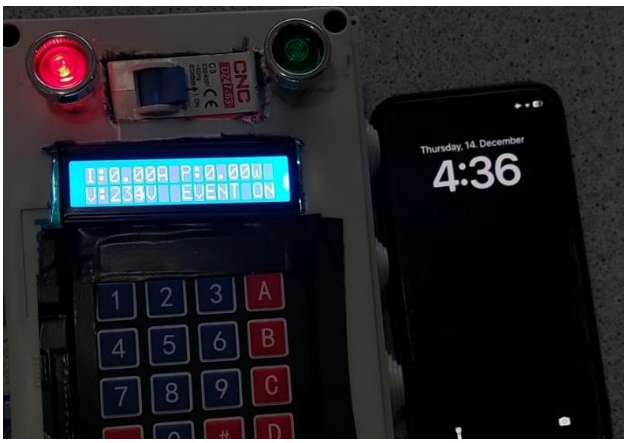
1. Voltage Accuracy: during opeartion voltage accuracy is failry high a maximum diffrence of +5V or -5V comapred to a multimeter. As shown in figures 13 and 14.

Figure 14-15: Voltage readings comapred to clamp meter value

2. Current Accuracy: the current measured by the Sensor is the one passing only trough the load, excluding the consupmtion of the arduino borad, lcd, sensors. We compared the values of the sesnors by the values of the clamp meter, and deduced a + or – 0.08Amps maximum error. As shown in figures 15 and 16.

Figure 15-16: Current Readings comapred to clamp meter value

3. Start and end period: suing the keypad as shown in the interface part we made the pulg turn on for 1 minute (from 4:37 to 4:38). Note: there is a 10 seconds delay compared to the real world clock. When the clock hit the 37 min mark the relay will turn on and deliver power to the plug, and when the clock cross 4:38 the relay turns off and cut the electricity of the pulg. As shown in figures 17, 18, and 19.



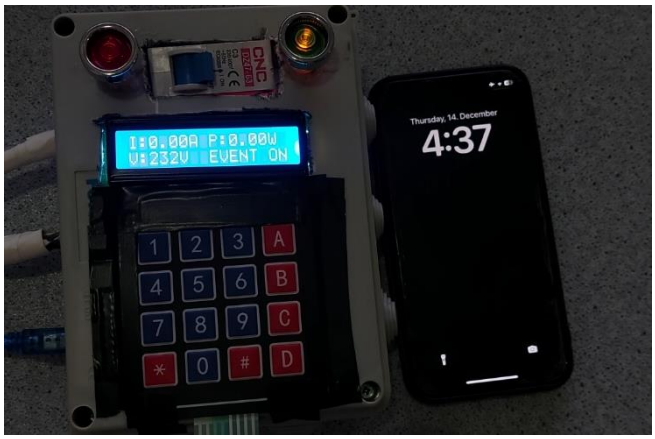


Figure 17-18-19: The start and the end of the speicfied working period.

4. The user can also choose to use the plug at anytime by disabling the events mode. All the protective and measuremnts are still working in both modes.

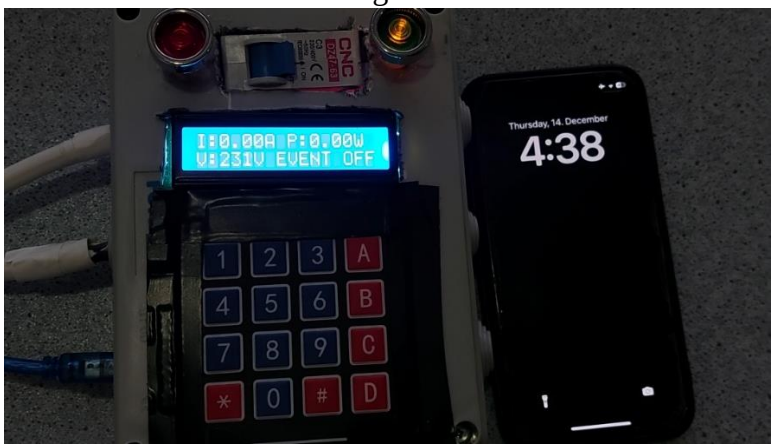


Figure 20: Event Off Mode

Conclusion:

We have succesfuly achieved our goal of creating a smart plug that has a friendly interface and that can protect the load from:1- Any voltage outside the acceptable range (can be specified by the user). 2-Any load current that exceeds the current limit which can be specified by the user. While also having the ability to choose the working period of the plug which is also specified by the user. “Please read the results before using the plug” through.

Appendix A:

Arduino code**: (please use the Arduino file included with the submission)**

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

LiquidCrystal_I2C lcd(0x27, 16, 2);

const byte ROWS = 4;
const byte COLS = 4;
char hexaKeys[ROWS][COLS] = {
  { '1', '2', '3', 'A' },
  { '4', '5', '6', 'B' },
  { '7', '8', '9', 'C' },
  { '.', '0', '#', 'D' }
};
byte rowPins[ROWS] = { 10, 9, 8, 7 };
byte colPins[COLS] = { 6, 5, 4, 3 };
Keypad customKeypad = Keypad(makeKeymap(hexaKeys), rowPins, colPins, ROWS, COLS);

//(start of rtc defining) Defining days and months
#define SUNDAY 0
#define MONDAY 1
#define TUESDAY 2
```

```
#define WEDNESDAY 3
#define THURSDAY 4
#define FRIDAY 5
#define SATURDAY 6

#define JANUARY 1
#define FEBRUARY 2
#define MARCH 3
#define APRIL 4
#define MAY 5
#define JUNE 6
#define JULY 7
#define AUGUST 8
#define SEPTEMBER 9
#define OCTOBER 10
#define NOVEMBER 11
#define DECEMBER 12

RTC_DS3231 rtc;

char daysOfTheWeek[7][12] = {
  "Sunday",
  "Monday",
  "Tuesday",
  "Wednesday",
  "Thursday",
  "Friday",
  "Saturday"
```

```
};


//(end of rtc defining)
////////////////////////////////////////////////////////  lcd_variabels_control_inputs
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
///
int temp1 = -1, temp2 = -1;                                                    //switch between frames
int temp_choosing_I_max = -1, temp_chossing_V_M_m = -1, temp_set_events = -1, temp_start_events = -1;  // set i value , set v max and vmin value , set the event time, break event
int temp_calculating_I = -1, temp_calculating_v = -1, temp_calculating_time = -1;          //save the values of i , save the value of vmax and min, save event time


float final_value_of_I = 3.50;
int final_value_of_Vmax = 240, final_value_of_Vmin = 200;
int final_value_of_day_start = 12, final_value_of_minit_start = 26, final_value_of_hour_start = 10, final_value_of_day_end = 18, final_value_of_minit_end = 30, final_value_of_hour_end = 16;
float calulate_i[]{ 2, 5, 0 };
int calulate_V_max[]{ 2, 4, 0 }, calulate_V_min[]{ 2, 1, 0 };
int calulate_day_start[]{ 00, 00 }, calulate_minit_start[]{ 00, 00 }, calulate_hour_start[]{ 00, 00 };
int calulate_day_end[]{ 00, 00 }, calulate_minit_end[]{ 00, 00 }, calulate_hour_end[]{ 00, 00 };


int intconvertor = 0;       //to change from character to int
int i = 0;                  //to control and move the cursor
int b = 0;                  // to move in the array whyle calculation the value
int v = 0;                  //  to move the value in the array while choosing the value
float old_final_value_of_I;  // to save the value of i because if we have some error we will need this value
int old_final_value_of_Vmax, old_final_value_of_Vmin;
int old_final_value_of_day_start, old_final_value_of_minit_start, old_final_value_of_hour_start, old_final_value_of_day_end, old_final_value_of_minit_end, old_final_value_of_hour_end;
```

```
int removeeror1 = 0;                                                    // rempve error that give us value 1

int error_screen_1 = 0, error_screen_2 = 0, error_screen_2_1 = 0, error_screen_3 = 0, error_screen_3_1 = 0;  // error when the user use a value that is out of the range

//////////////////////////////////////////////                    voltage_sensor_variabels ////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

double sensorValue1 = 0, sensorValue2 = 0;

int crosscount = 0;

int climb_flag = 0;

int val[120];  // Array to store sensor values

int max_v = 0;

double VmaxD = 0, VeffD = 0, Veff = 0;

int voltage_value = 0;

int Old_value_of_V = 0;

//////////////////////////////////////////////                    current_sensor_variabels ////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

const int sensorIn = A3;  // pin where the OUT pin from sensor is connected on Arduino

int mVperAmp = 100;       // this the 5A version of the ACS712 -use 100 for 20A Module and 66 for 30A Module

int Watt = 0;

double Voltage = 0;

double VRMS = 0;

double AmpsRMS = 0;

float curent;

float current_value;

//////////////////////////////////////////////////////general_variabels////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

int start_year, end_year;

int start_month, end_month;

int relay_pin = 11;
```

```
int x = 0;
void setup() {
  Serial.begin(9600);


  pinMode(relay_pin, OUTPUT);
  digitalWrite(relay_pin, HIGH);  // relay is active LOWWWWWW
  // SETUP RTC MODULE
  if (!rtc.begin()) {
    Serial.println("Couldn't find RTC");
    Serial.flush();
    while (1)
      ;
  }
  rtc.adjust(DateTime(F(_DATE), F(TIME_)));
  lcd.init();
  lcd.init();
  lcd.backlight();
  lcd.setCursor(5, 0);
  lcd.print("hello!");
  delay(500);
  lcd.clear();
}


void loop() {
  char customKey = customKeypad.getKey();


  if (customKey == 'A') {  // seting of i and  V
    lcd.clear();
```

```
    lcd.setCursor(0, 0);
    lcd.print("1 to set current");
    lcd.noCursor();
    lcd.noBlink();
    lcd.setCursor(0, 1);
    lcd.print("2 to set voltage");
    lcd.noCursor();
    lcd.noBlink();
    temp1 = 1;
    temp2 = -1;
    temp_choosing_I_max = -1;
    temp_calculating_I = -1;
    temp_chossing_V_M_m = -1;
    temp_calculating_v = -1;
    temp_set_events = -1;
    temp_calculating_time = -1;
  } else if (customKey == 'B') {  //seting of time and events
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("1 to set events");
    lcd.setCursor(0, 1);
    lcd.print("2 to end events");
    lcd.noCursor();
    lcd.noBlink();
    temp2 = 1;
    temp1 = -1;
    temp_choosing_I_max = -1;
    temp_calculating_I = -1;
```

```
    temp_chossing_V_M_m = -1;
    temp_calculating_v = -1;
    temp_set_events = -1;
    temp_calculating_time = -1;
  } else if (customKey == 'D') {  // back to main value
    lcd.clear();
    temp1 = -1;
    temp2 = -1;
    temp_choosing_I_max = -1;
    temp_calculating_I = -1;
    temp_chossing_V_M_m = -1;
    temp_calculating_v = -1;
    temp_calculating_time = -1;
    temp_set_events = -1;
  }

  if (temp1 == 1 && customKey == '1') {  //choosing  the value of I
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Choose I:");
    lcd.print(final_value_of_I);
    lcd.cursor();
    lcd.blink();
    temp_choosing_I_max = 1;  // entering and saving the value of I
    removeeror1 = 1;          // to remove a error
    temp1 = -1;
    i = 0;
    v = 0;
```

```
} else if (temp1 == 1 && customKey == '2') {  //choosing vmax/min
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Choose V max:");
  lcd.print(final_value_of_Vmax);
  lcd.setCursor(0, 1);
  lcd.print("Choose V min:");
  lcd.print(final_value_of_Vmin);
  lcd.setCursor(13, 0);
  lcd.cursor();
  lcd.blink();
  temp1 = -1;              // to close this frame
  temp_chossing_V_M_m = 1;  // to start next frame
  removeeror1 = 1;
  i = 0;
  v = 0;

} else if (temp2 == 1 && customKey == '1') {  //set a events time
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Start T:");
  lcd.print(final_value_of_day_start);
  lcd.print("/");
  lcd.print(final_value_of_hour_start);
  lcd.print(":");
  lcd.print(final_value_of_minit_start);
  lcd.setCursor(0, 1);
  lcd.print("End T:");
```

```
    lcd.setCursor(8, 1);
    lcd.print(final_value_of_day_end);
    lcd.print("/");
    lcd.print(final_value_of_hour_end);
    lcd.print(":");
    lcd.print(final_value_of_minit_end);
    lcd.setCursor(8, 0);
    lcd.cursor();
    lcd.blink();
    temp2 = -1;
    temp_set_events = 1;
    removeeror1 = 1;
    i = 0;
    v = 0;
  } else if (temp2 == 1 && customKey == '2') temp_start_events = -1;  // end events and back to normal operation


  if (temp_calculating_v == 1 && customKey == 'C') {  // save the value of vmax vmin
    temp_chossing_V_M_m = -1;
    int k = 100;  // to multiply the array to havedecimal value
    int summing_Vmin = 0, summing_Vmax = 0;
    b = 0;
    old_final_value_of_Vmax = final_value_of_Vmax;
    old_final_value_of_Vmin = final_value_of_Vmin;
    for (b; b < 3; b++) {
      summing_Vmax = calulate_V_max[b] * k + summing_Vmax;
      summing_Vmin = calulate_V_min[b] * k + summing_Vmin;
      k = k / 10;
    }
```

```
    final_value_of_Vmax = summing_Vmax;
    final_value_of_Vmin = summing_Vmin;
    Serial.print("MAX v:");
    Serial.println(final_value_of_Vmax);
    Serial.print("MIN v:");
    Serial.println(final_value_of_Vmin);
    if (final_value_of_Vmax >= 260 || final_value_of_Vmax <= 200 || final_value_of_Vmin < 200 || final_value_of_Vmin >= 260) error_screen_2 = 1;  // thoose are the value that the user cant enter not good for the real world
    else if (final_value_of_Vmax <= final_value_of_Vmin) error_screen_2_1 = 1;
    lcd.noCursor();
    lcd.noBlink();
    temp_calculating_v = -1;
    lcd.clear();
  } else if (temp_chossing_V_M_m == 1 && customKey) {  //choosing v max and min  input user
    if (removeeror1 == 1) {                       //having error
      lcd.setCursor(12, 0);
      lcd.print(":");
      removeeror1 = 0;
    } else if (removeeror1 == 0) {  //no error
      intconvertor = customKey - '0';

      if (intconvertor >= 0 && intconvertor <= 9 && i < 3) {  // value is a number between 0 and 9
        lcd.print(customKey);
        i++;
        Serial.print("v:  ");
        Serial.println(v);
        calulate_V_max[v] = intconvertor;
        int l = 0;
```

```
  Serial.print("value of V max ");
  Serial.print("{ ");
  Serial.print(calulate_V_max[l]);
  for (int l = 1; l < 3; l++) {
    Serial.print(", ");
    Serial.print(calulate_V_max[l]);
  }
  Serial.println(" } ");
  v++;
  if (v >= 3) v = 0;
  if (i == 3) lcd.setCursor(13, 1);  // to go to the next line
} else if (intconvertor >= 0 && intconvertor <= 9 && i > 2) {
  lcd.print(customKey);
  i++;
  calulate_V_min[v] = intconvertor;
  int l = 0;
  Serial.print("value of V min  ");
  Serial.print("{ ");
  Serial.print(calulate_V_min[l]);
  for (int l = 1; l < 3; l++) {
    Serial.print(", ");
    Serial.print(calulate_V_min[l]);
  }
  Serial.println(" } ");
  v++;
  if (v >= 3) v = 0;
  if (i == 6) {  // to back to the first value first cursor
    lcd.setCursor(13, 0);
```

```
        i = 0;
       }
     }
   }
   temp_calculating_v = 1;
 }

 if (temp_calculating_I == 1 && customKey == 'C') {  //save the value of I
   temp_choosing_I_max = -1;
   float k = 1.000;      // to multiply the array to havedecimal value
   float summing_i = 0;  // to summ the value of the array
   b = 0;
   old_final_value_of_I = final_value_of_I;
   for (b; b < 3; b++) {
     summing_i = calulate_i[b] * k + summing_i;
     k = k * 0.100;
   }
   final_value_of_I = summing_i;
   Serial.print("I value: ");
   Serial.println(final_value_of_I);
   if (final_value_of_I > 5.00 || final_value_of_I < 0) { error_screen_1 = 1; }
   k = 1.000;
   lcd.noCursor();
   lcd.noBlink();
   temp_calculating_I = -1;
   lcd.clear();
 } else if (temp_choosing_I_max == 1 && customKey) {  //choosung i input user
   if (removeeror1 == 1) {                          //having error
```

```
    lcd.setCursor(8, 0);
    lcd.print(":");
    removeeror1 = 0;
  } else if (removeeror1 == 0) {  //no error
    intconvertor = customKey - '0';
    if (intconvertor >= 0 && intconvertor <= 9) {  // value is a number between 0 and 9
      lcd.print(customKey);
      i++;
      if (i == 1) {  // to put a automatic camma
        lcd.print('.');
      } else if (i >= 3) {
        i = 0;
        lcd.setCursor(9, 0);
      }
      calulate_i[v] = intconvertor;
      int l = 0;
      Serial.print("value of the array: ");
      Serial.print("{ ");
      Serial.print(calulate_i[l]);
      for (int l = 1; l < 3; l++) {
        Serial.print(", ");
        Serial.print(calulate_i[l]);
      }
      Serial.println(" } ");
      v++;
      if (v >= 3) v = 0;
    }
    temp_calculating_I = 1;
```

```
    }
  }


  if (temp_calculating_time == 1 && customKey == 'C') {
    temp_set_events = -1;
    int k = 10;  // to multiply the array to havedecimal value
    int summing_day_start = 0, summing_minit_start = 0, summing_hour_start = 0, summing_day_end = 0, summing_minit_end = 0, summing_hour_end = 0;
    b = 0;
    old_final_value_of_day_start = final_value_of_day_start;
    old_final_value_of_minit_start = final_value_of_minit_start;
    old_final_value_of_hour_start = final_value_of_hour_start;
    old_final_value_of_day_end = final_value_of_day_end;
    old_final_value_of_minit_end = final_value_of_minit_end;
    old_final_value_of_hour_end = final_value_of_hour_end;

    for (b; b < 2; b++) {
      Serial.print("K: ");
      Serial.println(k);
      Serial.print("b: ");
      Serial.println(b);
      summing_day_start = calulate_day_start[b] * k + summing_day_start;
      summing_minit_start = calulate_minit_start[b] * k + summing_minit_start;
      summing_hour_start = calulate_hour_start[b] * k + summing_hour_start;
      summing_day_end = calulate_day_end[b] * k + summing_day_end;
      summing_minit_end = calulate_minit_end[b] * k + summing_minit_end;
      summing_hour_end = calulate_hour_end[b] * k + summing_hour_end;
      k = k / 10;
      Serial.print(" day start array value  ");
```

```
Serial.println(calulate_day_start[b]);
Serial.print("day start summ value: ");
Serial.println(summing_day_start);

Serial.print(" day start array value  ");
Serial.println(calulate_hour_start[b]);
Serial.print("hour start summ value: ");
Serial.println(summing_hour_start);

Serial.print(" day start array value  ");
Serial.println(calulate_minit_start[b]);
Serial.print("minute start summ value: ");
Serial.println(summing_minit_start);

Serial.print(" day start array value  ");
Serial.println(calulate_day_end[b]);
Serial.print("day stop summ value: ");
Serial.println(summing_day_end);

Serial.print(" day start array value  ");
Serial.println(calulate_hour_end[b]);
Serial.print("hour stop summ value: ");
Serial.println(summing_hour_end);

Serial.print(" day start array value  ");
Serial.println(calulate_minit_end[b]);
Serial.print("minute stop summ value: ");
Serial.println(summing_minit_end);
```

```
    }
    final_value_of_day_start = summing_day_start;
    final_value_of_minit_start = summing_minit_start;
    final_value_of_hour_start = summing_hour_start;
    final_value_of_day_end = summing_day_end;
    final_value_of_minit_end = summing_minit_end;
    final_value_of_hour_end = summing_hour_end;


    Serial.print("start date:");
    Serial.print(final_value_of_day_start);
    Serial.print("/");
    Serial.print(final_value_of_hour_start);
    Serial.print(":");
    Serial.println(final_value_of_minit_start);
    Serial.print("end date :");
    Serial.print(final_value_of_day_end);
    Serial.print("/");
    Serial.print(final_value_of_hour_end);
    Serial.print(":");
    Serial.println(final_value_of_minit_end);


    if (final_value_of_day_start > 31 || final_value_of_minit_start > 60 || final_value_of_hour_start
> 24 || final_value_of_day_end > 31 || final_value_of_minit_end > 60 || final_value_of_hour_end
> 24 || final_value_of_day_start == 0 || final_value_of_day_end == 0) error_screen_3 = 1;  // out
range value

    else if (final_value_of_day_start == final_value_of_day_end && final_value_of_hour_start >
final_value_of_hour_end) error_screen_3_1 = 1;

    else if (final_value_of_day_start == final_value_of_day_end && final_value_of_minit_start
>= final_value_of_minit_end && final_value_of_hour_start == final_value_of_hour_end)
error_screen_3_1 = 1;
```

```
  else if (final_value_of_day_start > final_value_of_day_end) error_screen_3_1 = 1;
  if (error_screen_3 == 0 && error_screen_3_1 == 0) temp_start_events = 1;
  lcd.noCursor();
  lcd.noBlink();
  temp_calculating_time = -1;
  lcd.clear();
 } else if (temp_set_events == 1 && customKey) {  // choosing start and end time of the event
  if (removeeror1 == 1) {                    //having error
   lcd.setCursor(7, 0);
   lcd.print(":");
   removeeror1 = 0;
  } else if (removeeror1 == 0) {  //no error
   intconvertor = customKey - '0';
   if (intconvertor >= 0 && intconvertor <= 9) {
    lcd.print(customKey);
    i++;
    if (i < 3) {  //start day
     calulate_day_start[v] = intconvertor;
     v++;
     if (v >= 2) {
      v = 0;
     }
     if (i == 2) lcd.print("/");
    }

    else if (i > 2 && i < 5) {  // start hour
     calulate_hour_start[v] = intconvertor;
     v++;
```

```
  if (v >= 2) {
    v = 0;
  }
  if (i == 4) lcd.print(":");
}

else if (i > 4 && i < 7) {  // start minute
  calulate_minit_start[v] = intconvertor;
  v++;
  if (v >= 2) {
    v = 0;
  }
  if (i == 6) lcd.setCursor(8, 1);
}

else if (i > 6 && i < 9) {  // end day
  calulate_day_end[v] = intconvertor;
  v++;
  if (v >= 2) v = 0;
  if (i == 8) lcd.print("/");
} else if (i > 8 && i < 11) {  // end hour
  calulate_hour_end[v] = intconvertor;
  v++;
  if (v >= 2) v = 0;
  if (i == 10) lcd.print(":");
} else if (i > 10 && i < 13) {  // end minute
  calulate_minit_end[v] = intconvertor;
```

```
  v++;
  if (v >= 2) {
    v = 0;
  }
  if (i == 12) {
    lcd.setCursor(8, 0);
    i = 0;
  }
}
Serial.print("V: ");
Serial.println(v);
Serial.print("I: ");
Serial.println(i);

Serial.print("value of start  day ");
Serial.print("{ ");
Serial.print(calulate_day_start[0]);
Serial.print(", ");
Serial.print(calulate_day_start[1]);
Serial.println(" } ");
Serial.print("value of start hour");
Serial.print("{ ");
Serial.print(calulate_hour_start[0]);
Serial.print(", ");
Serial.print(calulate_hour_start[1]);
Serial.println(" } ");
Serial.print("value of start minute");
Serial.print("{ ");
```

```
      Serial.print(calulate_minit_start[0]);
      Serial.print(", ");
      Serial.print(calulate_minit_start[1]);
      Serial.println(" } ");
      Serial.print("value of end day");
      Serial.print("{ ");
      Serial.print(calulate_day_end[0]);
      Serial.print(", ");
      Serial.print(calulate_day_end[1]);
      Serial.println(" } ");
      Serial.print("value of end hour");
      Serial.print("{ ");
      Serial.print(calulate_hour_end[0]);
      Serial.print(", ");
      Serial.print(calulate_hour_end[1]);
      Serial.println(" } ");
      Serial.print("value of end minute");
      Serial.print("{ ");
      Serial.print(calulate_minit_end[0]);
      Serial.print(", ");
      Serial.print(calulate_minit_end[1]);
      Serial.println(" } ");
    }
  }
  temp_calculating_time = 1;
}

if (error_screen_1 == 1) {  // value not in the required range error
```

```
    lcd.clear();
    final_value_of_I = old_final_value_of_I;
    lcd.setCursor(5, 0);
    lcd.print("Error!");
    lcd.setCursor(0, 1);
    lcd.print("Out of the range");
    error_screen_1 = 0;
    delay(1500);
    lcd.clear();
  } else if (error_screen_2 == 1) {
    lcd.clear();
    final_value_of_Vmax = old_final_value_of_Vmax;
    final_value_of_Vmin = old_final_value_of_Vmin;
    lcd.setCursor(5, 0);
    lcd.print("Error!");
    lcd.setCursor(0, 1);
    lcd.print("Out of the range");
    error_screen_2 = 0;
    delay(1500);
    lcd.clear();
  } else if (error_screen_2_1 == 1) {
    lcd.clear();
    final_value_of_Vmax = old_final_value_of_Vmax;
    final_value_of_Vmin = old_final_value_of_Vmin;
    lcd.setCursor(5, 0);
    lcd.print("Error!");
    lcd.setCursor(1, 1);
    lcd.print("Logical error");
```

```
    error_screen_2_1 = 0;
    delay(1500);
    lcd.clear();
  } else if (error_screen_3 == 1) {
    lcd.clear();
    final_value_of_day_start = old_final_value_of_day_start;
    final_value_of_minit_start = old_final_value_of_minit_start;
    final_value_of_hour_start = old_final_value_of_hour_start;
    final_value_of_day_end = old_final_value_of_day_end;
    final_value_of_minit_end = old_final_value_of_minit_end;
    final_value_of_hour_end = old_final_value_of_hour_end;
    lcd.setCursor(5, 0);
    lcd.print("Error!");
    lcd.setCursor(0, 1);
    lcd.print("Out of the range");
    error_screen_3 = 0;
    delay(1500);
    lcd.clear();
  } else if (error_screen_3_1 == 1) {
    lcd.clear();
    final_value_of_day_start = old_final_value_of_day_start;
    final_value_of_minit_start = old_final_value_of_minit_start;
    final_value_of_hour_start = old_final_value_of_hour_start;
    final_value_of_day_end = old_final_value_of_day_end;
    final_value_of_minit_end = old_final_value_of_minit_end;
    final_value_of_hour_end = old_final_value_of_hour_end;
    lcd.setCursor(5, 0);
    lcd.print("Error!");
```

```
    lcd.setCursor(1, 1);
    lcd.print("Logical error");
    error_screen_3_1 = 0;
    delay(1500);
    lcd.clear();
  }


  if (temp1 == -1 && temp2 == -1 && temp_choosing_I_max == -1 && temp_chossing_V_M_m
== -1 && temp_set_events == -1 && temp_calculating_I == -1 && temp_calculating_v == -1
&& temp_calculating_time == -1)  //if nothing happening start this screen
  {
    lcd.setCursor(0, 0);
    lcd.print("I:");
    lcd.print(current_value);
    lcd.print("A");
    lcd.setCursor(0, 1);
    lcd.print("V:");
    lcd.print(voltage_value);
    lcd.print("V");
    lcd.setCursor(8, 0);
    lcd.print("P:");
    lcd.print(current_value * voltage_value);
    lcd.print("W");
    if (temp_start_events == 1) {
      lcd.setCursor(7, 1);
      lcd.print(" EVENT ON ");
    } else {
      lcd.setCursor(7, 1);
      lcd.print("EVENT OFF");
```

```
  }
  delay(50);
 }
 if (x = 1) Old_value_of_V = Veff;
 if (x = 0) x = 1;
 for (int i = 0; i < 120; i++) {
  sensorValue1 = analogRead(A2);  // Read analog sensor value from A0
  if (analogRead(A2) > 511) {
    val[i] = sensorValue1;  // Store sensor value in the array if it's greater than 511
  } else {
    val[i] = 0;  // Otherwise, set the value to 0
  }
  delay(1);  // Short delay for stability
 }

 // Find the maximum sensor value in the array
 max_v = 0;
 for (int i = 0; i < 120; i++) {
  if (val[i] > max_v) {
    max_v = val[i];  // Update max_v if a higher value is found
  }
  val[i] = 0;  // Reset the array element to 0
 }

 // Calculate effective voltage based on the maximum sensor value
 if (max_v != 0) {
  VmaxD = max_v;                          // Set VmaxD to the maximum sensor value
  VeffD = VmaxD / sqrt(2);                // Calculate effective voltage (RMS) from VmaxD
```

```
  Veff = (((VeffD - 420.76) / -90.24) * -210.2) + 210.2;  // Apply calibration and scaling to Veff
} else {
  Veff = 0;  // If no maximum value, set Veff to 0
}
if (Veff <= 100) Veff = 0;
Veff = (Old_value_of_V + Veff) / 2;
// Print the calculated voltage to the serial monitor
Serial.print("Voltage: ");
Serial.println(Veff);

VmaxD = 0;              // Reset VmaxD for the next iteration
voltage_value = (Veff);  // voltage value

//////////////////////////////////////////////////////////////////current sens
float result;
int readValue;        // value read from the sensor
int maxValue = 0;     // store max value here
int minValue = 1024;  // store min value here
for (int i = 0; i < 120; i++) {
  readValue = analogRead(sensorIn);
  // see if you have a new maxValue
  if (readValue > maxValue) {
    /record the maximum sensor value/
    maxValue = readValue;
  }
  if (readValue < minValue) {
    /record the minimum sensor value/
    minValue = readValue;
```

```
  }
 }
 // Subtract min from max
 result = ((maxValue - minValue) * 5.0) / 1024.0;
 float curent = analogRead(sensorIn);
 Voltage = result;
 // Serial.println(result);
 VRMS = (Voltage / 1.78 ) * 0.707;  //root 2 is 0.707
 AmpsRMS = (VRMS * 1000) / mVperAmp;
 current_value = AmpsRMS - 0.3;
 if (current_value < 0) current_value = 0;
 Serial.print(current_value);  // current value
 Serial.println(" Amps RMS  ---  ");


 ////////////////////////////////////////////////////////////////////////////
 DateTime EVENT_START(start_year, start_month, final_value_of_day_start, final_value_of_hour_start, final_value_of_minit_start);
 DateTime EVENT_END(start_year, start_month, final_value_of_day_end, final_value_of_hour_end, final_value_of_minit_end);
 DateTime now = rtc.now();
 //  first check that current and voltage are okay , than see if user start a event or not, if yes relay will be off untill event start and return off when the event end, if the user didnt start a event or stop it the relay will be ON
if (voltage_value >= 260 || voltage_value <= 200 || voltage_value >= final_value_of_Vmax || voltage_value <= final_value_of_Vmin) {
 digitalWrite(relay_pin, HIGH);  //  relay OFF
  delay(250);
 }
 else  if (current_value > 5 || final_value_of_I <= current_value) {  //    //|| voltage_value >= final_value_of_Vmax ||voltage_value <= final_value_of_Vmin
```

```
    digitalWrite(relay_pin, HIGH);                          //  relay OFF
    delay(5000);
  } else if (temp_start_events == -1 ) {
    digitalWrite(relay_pin, LOW);  // relay ON
  } else if (temp_start_events == 1) {
    if (now.secondstime() >= EVENT_START.secondstime() && now.secondstime() < EVENT_END.secondstime()) {
      Serial.println("It is on scheduled time");
      digitalWrite(relay_pin, LOW);  // relay ON
    } else {
      Serial.println("It is NOT on scheduled time");
      digitalWrite(relay_pin, HIGH);  //  relay OFF
    }
  }



  printTime(now);  //read time
}

void printTime(DateTime time) {
  // Serial.print("TIME: ");
  Serial.print(time.year(), DEC);
  start_year = time.year();
  Serial.print('/');
  Serial.print(time.month(), DEC);
  start_month = time.month();
  Serial.print('/');
  Serial.print(time.day(), DEC);
```

```
  Serial.print('/');  // added
  //Serial.print(" (");
  //Serial.print(daysOfTheWeek[time.dayOfTheWeek()]);
  //Serial.print(") ");
  Serial.print(time.hour(), DEC);
  Serial.print('/');  // added
  //Serial.print(':');
  Serial.print(time.minute(), DEC);
  Serial.print('/');  // added
  //Serial.print(':');
  Serial.println(time.second(), DEC);
}
```