IoT Smart Clock

Course Code	ECE 341
Course Title	Programming IoT
Academic Task no.	2
Date of Allotment	30-03-2023
Date of Submission	21-04-2023
Section	KM052
Max. marks	50

Group Members:

Name	**Reg. no.**	**Roll. No.**
Sudip Kumar Mandal	12019167	15
Sai Kalyan	12004587	4
Gandham Charan	12015334	29
Anurag Gautam	12007475	7

INDEX

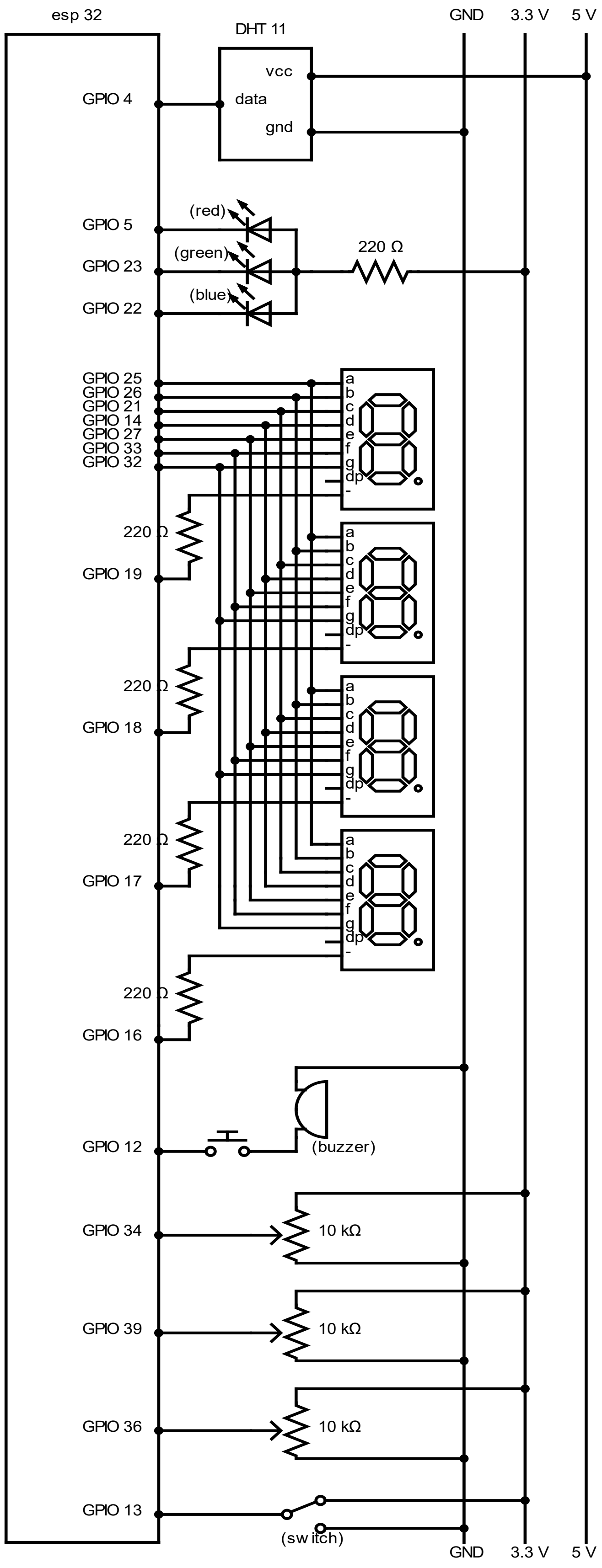
FEATURES

- Turn on/off
    - Turn on or off the 7 segment displays and rgb leds
- Display time offline
    - Displays time without internet connection
- Set time
    - Set hour and minute
- Set alarm
    - Set alarm hour and minute
- Stopwatch
    - Stopwatch with a maximum range of 1 hour
- Display internet synchronized time
    - Connect to Wi-Fi and get time through NTP servers
- Display Temperature and Humidity
    - Display temperature in degree Celsius and humidity in percentage, using the DHT11 sensor
- Cloud
    - Send temperature and humidity data to cloud
    - Turn on/off alarm and set alarm time through cloud

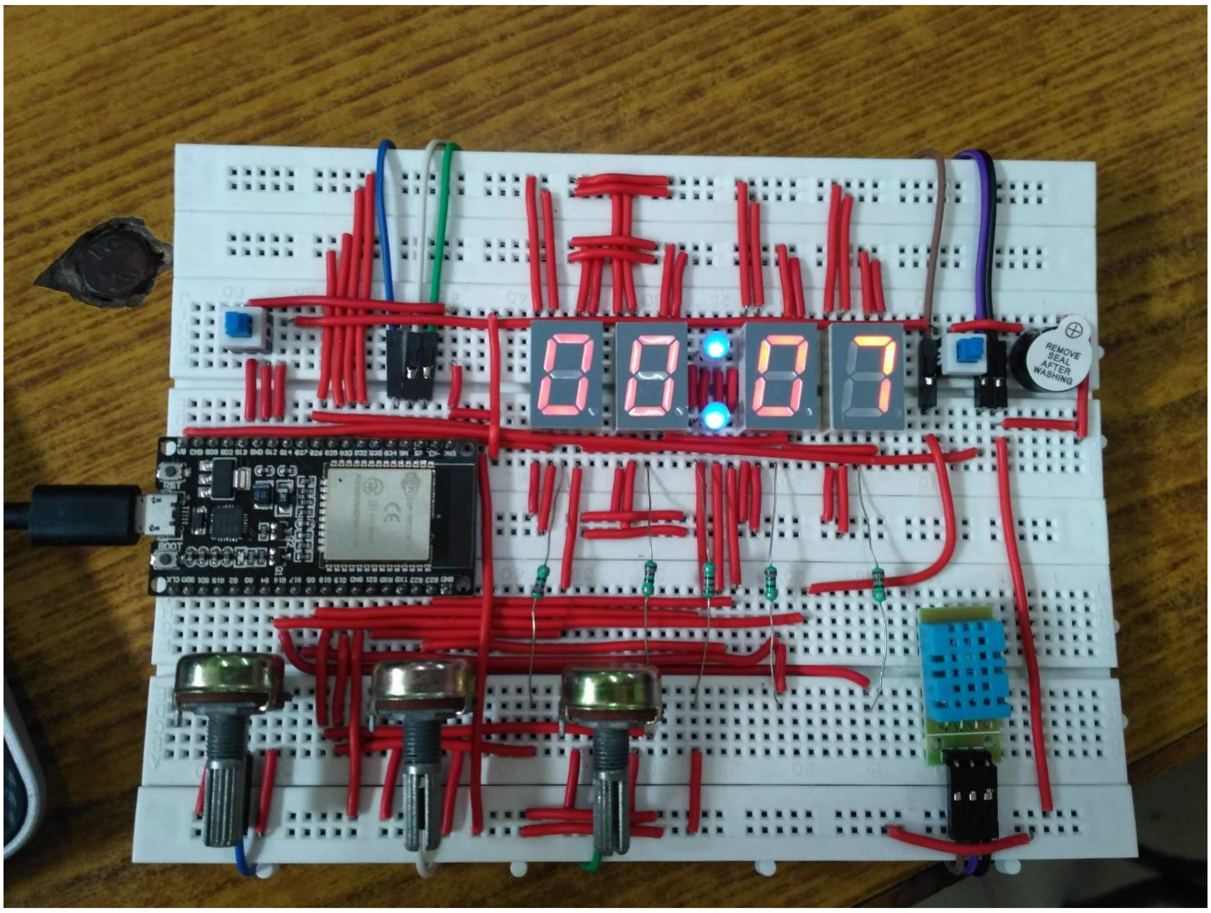
COMPONENTS

Name	**Quantity**
esp wroom 32	1
10k ohm potentiometer	3
Push switch	2
Common anode Rgb led	2
Common cathode 7 segment display	4
Piezo buzzer	1
DHT-11 sensor	1
220 Ohm Resistors	5
830 pins breadboard	2
U shaped jumper wire	~2 meters

CONNECTIONS



CONNECTION IMPLEMENTATOIN:



WORKING

This project implements a smart IoT clock. It displays time, temperature and humidity and integrates with the cloud.

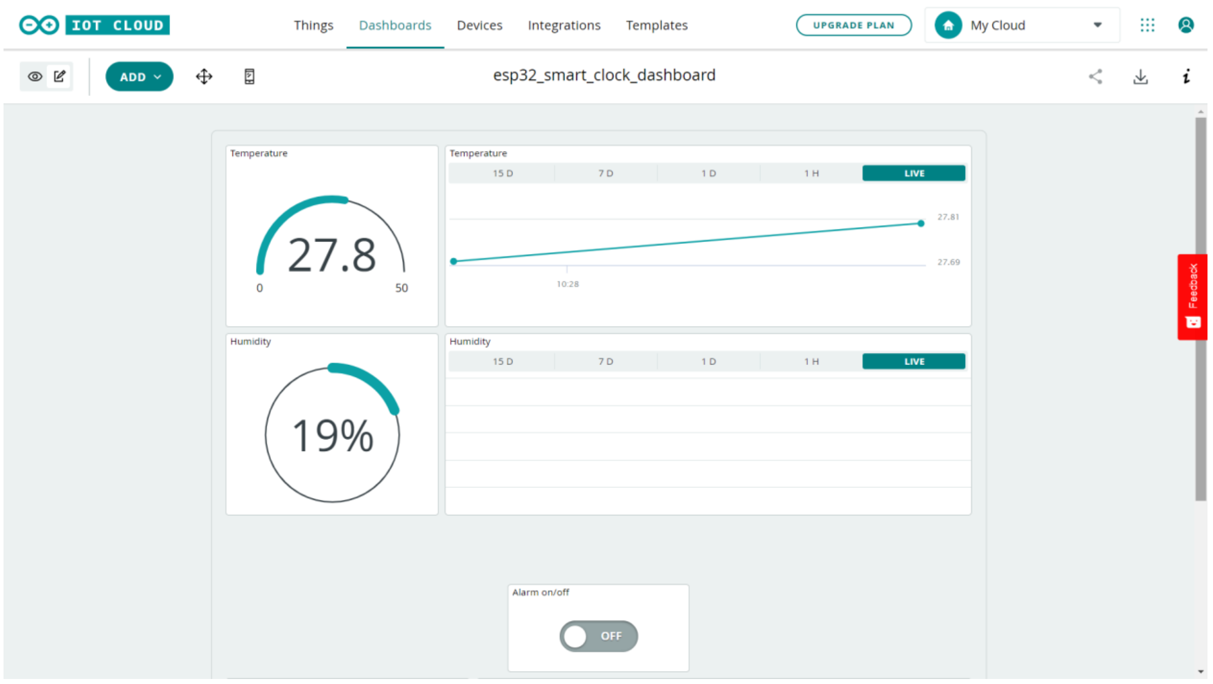
The cloud platform used here is Arduino IoT Cloud Platform. The The potentiometer connected at GPIO pin 34 is used for menu. The menu consists of:

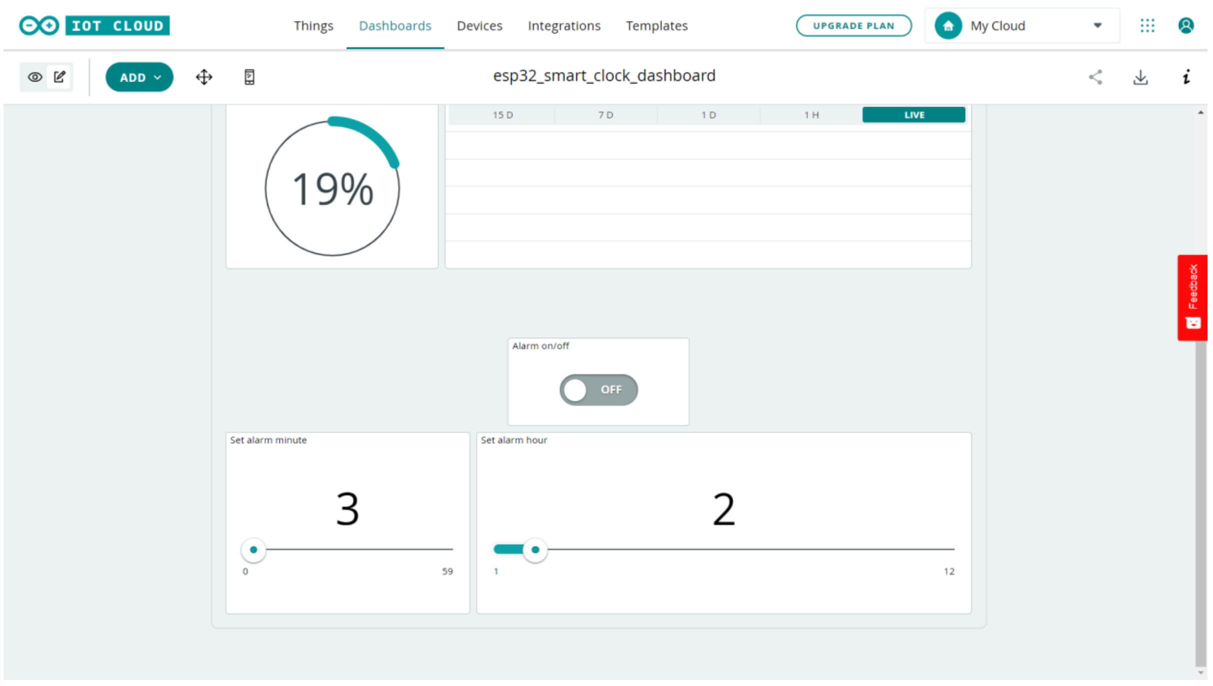
1. **Off**
   - Turn off the 7 segment display and rgb leds
2. **Offline time display**
   - Display time offline. Also display temperature and humidity values
   - In a duration of 6 seconds, displays time for the first 4 seconds and display temperature and humidity in the last 2 seconds
   - Display hour and minute in time
   - Display humidity in the first 2 7-segment displays and temperature in the last 2 7-segment displays
3. **Offline time set**
   - Set time using the potentiometers connected to GPIO pins 39 and 36
   - Select a 7-segment display using the potentiometer connected at pin 39
   - Set a value of the 7-segment using the potentiometer connected at pin 36
   - The first two 7-segments set hour whereas the last two 7-segments set minute
4. **Alarm time set**
   - Set alarm time using the same algorithm used to set time
5. **Stopwatch**
   - Turn off the push switch connected at gpio pin 13
   - Set time using the same algorithm used to set time
   - The first two 7-segments set minute whereas the last two 7-segments set seconds
   - To activate the stopwatch, turn on the push
   - As the timer reaches 0, an alarm tone is played through the buzzer
6. **Internet Time display**
   - Connect the board to a wifi
   - The time is received from the internet using the ntp protocol.

Arduino IoT Cloud

The board connects to Arduino IoT Cloud and uploads data such as Humidity and Temperature.

Also, we can see and set alarm through the cloud.





CODE

File hierarchy:

- esp32_smart_clock.ino
    - setup_dht_rgb_7segment_buzzer_custom.h
    - stopwatch_custom.h
    - arduino_secrets.h
    - thingProperties.h
    - time_display_custom.h
    - time_set_custom.h

esp32_smart_clock.ino

```
#include "setup_dht_rgb_7segment_buzzer_custom.h"
#include "time_display_custom.h"
#include "time_set_custom.h"
#include "stopwatch_custom.h"
#include "thingProperties.h"


void setup() {
  Serial.begin(115200);
  delay(1500);
  initProperties();
  ArduinoCloud.begin(ArduinoIoTPreferredConnection);
  setDebugMessageLevel(2);
  ArduinoCloud.printDebugInfo();

  rgb_led_setup();
  configTime(gmtOffset_sec, daylightOffset_sec, ntpServer1, ntpServer2);

  for(int i=0; i<7; i++)
    pinMode(lpin[i], OUTPUT);
  for(int i=0; i<4; i++)
    pinMode(cpin[i], OUTPUT);
  for(int i=0; i<4; i++)
    digitalWrite(cpin[i], HIGH);


  pinMode(buzzer, OUTPUT);
  pinMode(push_switch, INPUT);

  play_tone(tone_start, sizeof(tone_start));
  dht.begin();
}

void loop() {
  ArduinoCloud.update();

  if(is_alarm_on) {
    if(hour == alarm_hour && minute == alarm_minute) {
      play_tone(tone_alarm, sizeof(tone_alarm));
      is_alarm_on = 0;
    }
  }

  temp_menu = map(analogRead(knob0), 0, 4095, 0, 5);
  if(temp_menu != menu) {
    tone(buzzer, 2000, 100);
    menu = temp_menu;
  }

  Serial.print("menu: ");
  Serial.println(menu);

  switch(menu) {
    case 0:
      delay(500);
        ledcWrite(rgb_led_red, 255);
        ledcWrite(rgb_led_green, 255);
        ledcWrite(rgb_led_blue, 255);
      break;
    case 1:
      time_display();
      break;
    case 2:
      time_set(hour, 12, minute, 59);
      break;
    case 3:
      time_set(alarm_hour, 12, alarm_minute, 59);
      break;
    case 4:
      stopwatch();
      break;
    case 5:
      wifi_time_display();
      break;
  }
}
```

```
void onAlarmHourChange()    {tone(buzzer, 3000, 100);}
void onAlarmMinuteChange()  {tone(buzzer, 3000, 100);}
void onIsAlarmOnChange()    {tone(buzzer, 3000, 100);
```

setup_dht_rgb_7segment_buzzer_custom.h

```
//////////////////////////////////////////// DHT

#include <DHT.h>

#define DHTPIN 4
#define DHTTYPE DHT11

DHT dht(DHTPIN, DHTTYPE);

float temperature = 0.0;
float humidity = 0.0;

//////////////////////////////////////////////// rgb

bool is_led_on = 0;
int rgb_led_red_pin = 5;
int rgb_led_green_pin = 23;
int rgb_led_blue_pin = 22;

int rgb_led_red = 3;
int rgb_led_green = 4;
int rgb_led_blue = 5;

int rgb_led_brightness = 25;  // 0 to 255

void rgb_led_setup() {
  ledcSetup(rgb_led_red, 5000, 8);
  ledcSetup(rgb_led_green, 5000, 8);
  ledcSetup(rgb_led_blue, 5000, 8);

  ledcAttachPin(rgb_led_red_pin, rgb_led_red);
  ledcAttachPin(rgb_led_green_pin, rgb_led_green);
  ledcAttachPin(rgb_led_blue_pin, rgb_led_blue);

  ledcWrite(rgb_led_red, 255);
  ledcWrite(rgb_led_green, 255);
  ledcWrite(rgb_led_blue, 255);
}

/////////////////////////////////////////////// 7 segment display
```

```
int lpin[] = {25,26,21,14,27,33,32};  //LED pins
int cpin[] = {19,18,17,16};           //Control pins



int numbers[10][7] = {
  {1,1,1,1,1,1,0},  //0
  {0,1,1,0,0,0,0},  //1
  {1,1,0,1,1,0,1},  //2
  {1,1,1,1,0,0,1},  //3
  {0,1,1,0,0,1,1},  //4
  {1,0,1,1,0,1,1},  //5
  {1,0,1,1,1,1,1},  //6
  {1,1,1,0,0,0,0},  //7
  {1,1,1,1,1,1,1},  //8
  {1,1,1,1,0,1,1},  //9
};
int fliker = 5;

void digit_display(int p, int num) {      // control pin, number(0-9)
  digitalWrite(cpin[p], LOW);

  for(int i=0; i<7; i++)
    digitalWrite(lpin[i], numbers[num][i]);

  delay(fliker);
  digitalWrite(cpin[p], HIGH);
}

////////////////////////////////////////////////////////// buzzer

int buzzer = 12;

int tone_error[][3] = {
  {1000, 200, 250},
  {2000, 200, 250},
  {3000, 200, 1000}
};

int tone_warning[][3] = {
  {3000, 200, 250},
  {2000, 200, 1000}
};

int tone_success[][3] {
  {3000, 200, 250},
  {2000, 200, 250},
  {1000, 200, 1000}
};

int tone_start[][3] = {
  {2000, 200, 250},
  {3000, 200, 1000}
};

int tone_alarm[][3] {
  {2000, 100, 150},
  {2000, 100, 500},
  {2000, 100, 150},
  {2000, 100, 500},
  {2000, 100, 150},
  {2000, 100, 500},
  {2000, 100, 150},
  {2000, 100, 500},
  {2000, 100, 150},
  {2000, 100, 500}
};

void play_tone(int tone_var[][3], int arr_size) {
  int size_notes = arr_size / sizeof(tone_var[0]);
  for(int i=0; i<size_notes; i++) {
    tone(buzzer, tone_var[i][0], tone_var[i][1]);
    delay(tone_var[i][2]);
  }
}


///////////////////////////////////////////////////// others
////////// ntp, menu, alarm, potentiometers

const char* ntpServer1 = "pool.ntp.org";
const char* ntpServer2 = "time.nist.gov";
const long  gmtOffset_sec = 19800;
const int   daylightOffset_sec = 0;

int menu = 0;
int temp_menu = 0;

int alarm_hour = 0;
int alarm_minute = 0;
bool is_alarm_on = 0;

int knob0 = 34;
int knob1 = 39;
int knob2 = 36;
```

stopwatch_custom.h

```
int push_switch = 13;
int stopwatch_minute = 0;
int stopwatch_second = 0;

void stopwatch() {
  if(digitalRead(push_switch)) {
    if(stopwatch_second == 0) {
      if(stopwatch_minute == 0) {
        play_tone(tone_alarm, sizeof(tone_alarm));
      }
      else {
        stopwatch_minute--;
        stopwatch_second = 60;
      }
    }
    else {
      stopwatch_second--;
    }

    for(int i=0; i<1000/fliker; i+=4) {
      digit_display(0, stopwatch_minute/10);
      digit_display(1, stopwatch_minute%10);
      digit_display(2, stopwatch_second/10);
      digit_display(3, stopwatch_second%10);
    }
  }
  else {
    time_set(stopwatch_minute, 60, stopwatch_second, 60);
  }
}
```

time_display_custom.h

```
int hour = 0;
int minute = 0;
int second = 0;
int temp_counter = 0;

/////////////////////////////////////////////////////////////////////////////

void time_display_combined() {
  int display_part1 = hour;
  int display_part2 = minute;

  temp_counter = (temp_counter + 1) % 6;
  if(temp_counter == 4 || temp_counter == 5) {
    if(temp_counter == 4) {
      temperature = dht.readTemperature();
      humidity = dht.readHumidity();
    }
    display_part1 = (int)humidity;
    display_part2 = (int)temperature;

    if (isnan(display_part1) || isnan(display_part2)) {
      Serial.println(F("Failed to read from DHT sensor!"));
      play_tone(tone_warning, sizeof(tone_warning));
      return;
    }

    ledcWrite(rgb_led_red, 255 - rgb_led_brightness);
    ledcWrite(rgb_led_green, 255);
    ledcWrite(rgb_led_blue, 255);
  }
  else {
    ledcWrite(rgb_led_red, 255);
  }

  for(int i=0; i<1000/fliker; i+=4) {
    digit_display(0, display_part1/10);
    digit_display(1, display_part1%10);
    digit_display(2, display_part2/10);
    digit_display(3, display_part2%10);
  }

  if(is_led_on) {
    ledcWrite(rgb_led_blue, 255 - rgb_led_brightness);
    ledcWrite(rgb_led_green, 255);
  }
  else {
    ledcWrite(rgb_led_green, 255 - rgb_led_brightness);
    ledcWrite(rgb_led_blue, 255);
  }
  is_led_on = !is_led_on;
}

/////////////////////////////////////////////////////////////////////////

void wifi_time_display() {
  struct tm timeinfo;
  if(!getLocalTime(&timeinfo)){
    Serial.println("No time available (yet)");
    play_tone(tone_warning, sizeof(tone_warning));
    return;
  }

  char timeSecond[3];
  strftime(timeSecond, 3, "%S", &timeinfo);
  second = 10 * (timeSecond[0] - '0') + (timeSecond[1] - '0');
  char timeMinute[3];
  strftime(timeMinute, 3, "%M", &timeinfo);
  minute = 10 * (timeMinute[0] - '0') + (timeMinute[1] - '0');
  minute = (minute +  30) % 60;
  char timeHour[3];
  strftime(timeHour, 3, "%I", &timeinfo);
  hour = 10 * (timeHour[0] - '0') + (timeHour[1] - '0');
  hour = (hour + 5) % 12;

  time_display_combined();
}

////////////////////////////////////////////////////////////////////////////

void time_display() {
  if(second >= 60) {
    second = 0;
    minute++;
  }
  if(minute >= 60){
    minute = 0;
    hour++;
  }
  if(hour >= 13) {
    hour = 0;
  }
  second++;
  time_display_combined();
}
```

time_set_custom.h

```
int digit_select= 0;
int digit_set = 0;
int temp_digit_select = 0;
int temp_digit_set = 0;

void time_set(int &digit_part_1, int range_part_1, int &digit_part_2, int
range_part_2) {

  temp_digit_select = map(analogRead(knob1), 0, 4095, 0, 3);
  if(temp_digit_select != digit_select) {
    tone(buzzer, 2000, 100);
    digit_select = temp_digit_select;
  }

  temp_digit_set = map(analogRead(knob2), 0, 4095, 0, 9);
  if(temp_digit_set != digit_set) {
    tone(buzzer, 2000, 100);
    digit_set = temp_digit_set;

    switch(digit_select) {
      case 0:
        digit_part_1 = (digit_set % (range_part_1 / 10 + 1)) * 10;
        break;
      case 1:
        if(digit_part_1 >= 10)
          digit_part_1 = 10 * (digit_part_1 / 10) + digit_set;
        else
          digit_part_1 = digit_set;
        break;
      case 2:
        digit_part_2 = (digit_set % (range_part_2 / 10 + 1)) * 10;
        break;
      case 3:
        if(digit_part_2 >= 10)
          digit_part_2 = 10 * (digit_part_2 / 10) + digit_set;
        else
          digit_part_2 = digit_set;
        break;
    }
  }

  int time_values[] = {digit_part_1/10, digit_part_1%10, digit_part_2/10,
digit_part_2%10};
  digit_display(digit_select, time_values[digit_select]);
  digit_display(0, digit_part_1/10);
  digit_display(digit_select, time_values[digit_select]);
  digit_display(1, digit_part_1%10);
  digit_display(digit_select, time_values[digit_select]);
  digit_display(2, digit_part_2/10);
  digit_display(digit_select, time_values[digit_select]);
  digit_display(3, digit_part_2%10);
}
```

arduino_secrets.h

```
#define SECRET_SSID "(WifiSSID)"
#define SECRET_OPTIONAL_PASS "(WifiPassword)"
#define SECRET_DEVICE_KEY "(SecretDeviceKey)"
```

thingProperties.h

```
// Code generated by Arduino IoT Cloud, DO NOT EDIT.

#include <ArduinoIoTCloud.h>
#include <Arduino_ConnectionHandler.h>


const char DEVICE_LOGIN_NAME[]  = "4c58b4f5-6c8c-495b-92b6-f5f550d61784";

const char SSID[]           = SECRET_SSID;              // Network SSID (name)
const char PASS[]           = SECRET_OPTIONAL_PASS;     // Network password
const char DEVICE_KEY[]     = SECRET_DEVICE_KEY;        // Secret device
password

void onAlarmHourChange();
void onAlarmMinuteChange();
void onIsAlarmOnChange();

void initProperties(){

  ArduinoCloud.setBoardId(DEVICE_LOGIN_NAME);
  ArduinoCloud.setSecretDeviceKey(DEVICE_KEY);
  ArduinoCloud.addProperty(humidity, READ, ON_CHANGE, NULL);
  ArduinoCloud.addProperty(temperature, READ, ON_CHANGE, NULL);
  ArduinoCloud.addProperty(alarm_hour, READWRITE, ON_CHANGE,
onAlarmHourChange);
```

```
  ArduinoCloud.addProperty(alarm_minute, READWRITE, ON_CHANGE,
onAlarmMinuteChange);
  ArduinoCloud.addProperty(is_alarm_on, READWRITE, ON_CHANGE,
onIsAlarmOnChange);

}

WiFiConnectionHandler ArduinoIoTPreferredConnection(SSID, PASS);
```