WORKSHEET
2.4

Class: CSE 26**(B)** **Group No.: 05**

Group Members Details

S. No.	**Name**	**UID**
1.	**RAJDEEP JAISWAL**	**20BCS2761**
2.	**ADARSH SHARMA**	**20BCS2762**
3.	**VISHAL CHOUDHARY**	**20BCS2848**
4.	**SOUMYA SHUBHAM NAYAK**	**20BCS2781**

Task:

Develop a human vitals monitoring and alert system using IoT analytics platform.

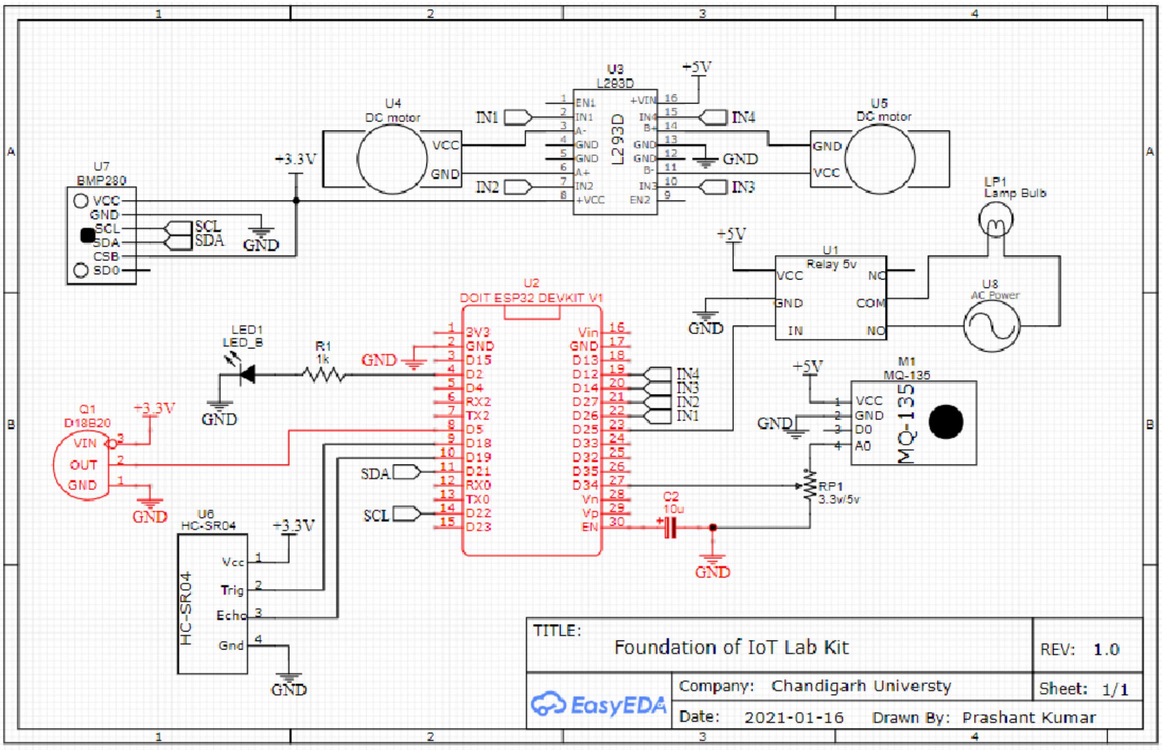
Requirements:

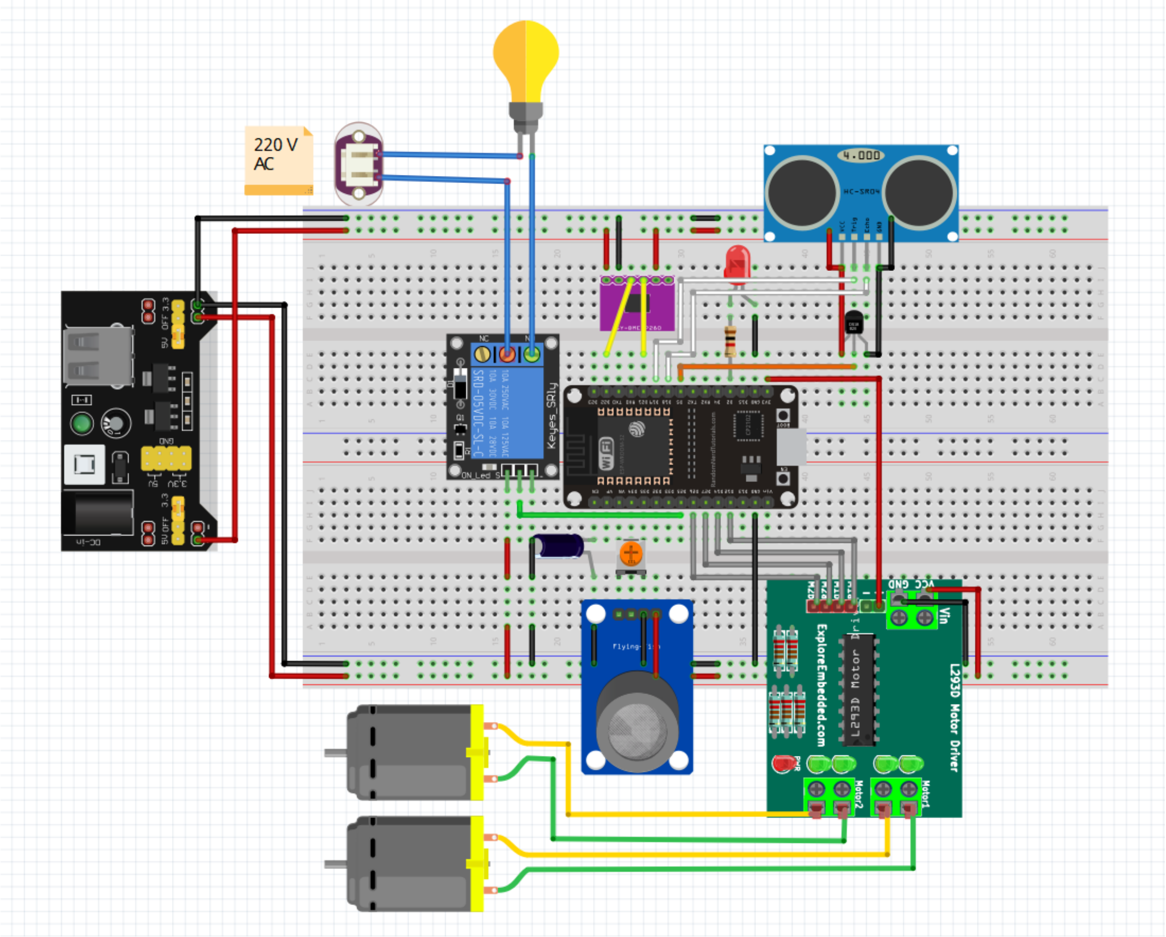
- PC with Arduino
- Connecting Wires
- Breadboard
- DOIT ESP32 DEVKIT V1
- 10uF Electrolytic Capacitor
- Wire Clipper
- USB Type A to Micro USB Cable
- DC 5V Power Supply
- DC 3.3V Power Supply
- DS18B20

THEORY

The DS18B20 is a 1-wire programmable Temperature sensor from maxim integrated. It is widely used to measure temperature in hard environments like in chemical solutions, mines or soil etc. The constriction of the sensor is rugged and also can be purchased with a waterproof option making the mounting process easy. It can measure a wide range of temperature from -55°C to +125° with a decent accuracy of ±5°C.

Circuit Diagram:

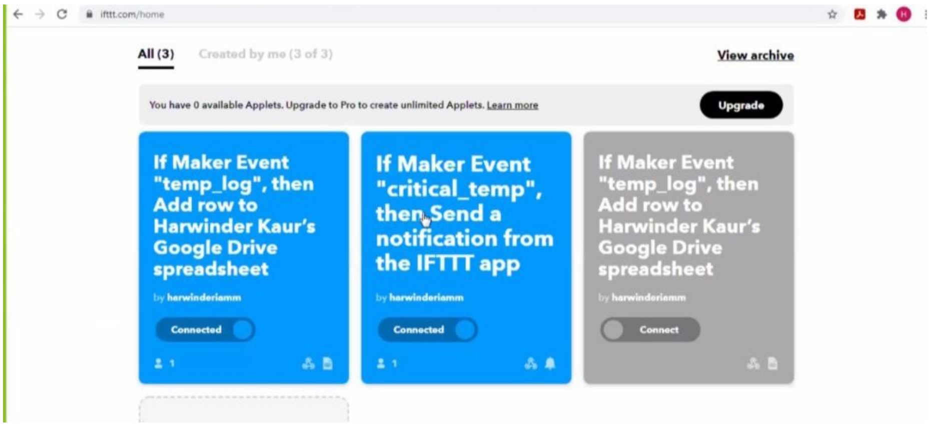

U7
BMP280
VCC
GND
SCL
SDA
CSB
SD0
SCL
SDA
GND
+3.3V
U4
DC motor
VCC
GND
IN1
IN2
U3
L293D
EN1
IN1
A-
GND
GND
A+
IN2
+VCC
+VIN
IN4
B+
GND
GND
B-
IN3
EN2
+5V
IN4
GND
IN3
U5
DC motor
GND
VCC
LP1
Lamp Bulb
+5V
U1
Relay 5v
VCC
GND
IN
NC
COM
NO
U8
AC Power
GND
U2
DOIT ESP32 DEVKIT V1
3V3
GND
D15
D2
D4
RX2
TX2
D5
D18
D19
D21
RX0
TX0
D22
D23
Vin
GND
D13
D12
D14
D27
D26
D25
D33
D32
D35
D34
Vn
Vp
EN
IN4
IN3
IN2
IN1
LED1
LED_B
R1
1k
GND
GND
Q1
D18B20
+3.3V
VIN
OUT
GND
GND
SDA
SCL
U6
HC-SR04
+3.3V
Vcc
Trig
Echo
Gnd
GND
+5V
M1
MQ-135
VCC
GND
D0
A0
MQ-135
GND
RP1
3.3v/5v
C2
10u
GND
TITLE:
Foundation of IoT Lab Kit
REV: 1.0
EasyEDA
Company: Chandigarh Universty
Sheet: 1/1
Date: 2021-01-16
Drawn By: Prashant Kumar


220 V
AC
HC-SR04
Keyes_SRly
SRD-05VDC-SL-C
L293D Motor Driver
ExploreEmbedded.com
L293D Motor Dri
Vin
VCC GND
Motor1
Motor2

Code (if any):

```
/*
  Board: DOIT ESP32 DEVKIT V1
*/
#include <WiFi.h>
#include <IFTTTWebhook.h>
#include <OneWire.h>
#include <DallasTemperature.h>

#define WIFISSID "Joker" // Your WiFi Name
#define PASSWORD "Joker@tenda" // Your WiFi Password

#define CRITICAL_API_KEY "itp_xKwcmnWcTvsLARkH9xrSRsixs34g_ZTPrItqkGT"
#define CRITICAL_EVENT_NAME "critical_temp"

#define LOG_API_KEY "itp_xKwcmnWcTvsLARkH9xrSRsixs34g_ZTPrItqkGT"
#define LOG_EVENT_NAME "temp_log"

#define MAX_TEMP_THRESHOLD 99.5 // F
#define MIN_TEMP_THRESHOLD 97.7 // F

// Data wire
#define ONE_WIRE_BUS 5

// Setup a oneWire instance to communicate with any OneWire devices (not just Maxim/Dallas temperature ICs)
OneWire one_wire(ONE_WIRE_BUS);

// Pass our oneWire reference to Dallas Temperature.
DallasTemperature temp_sensor(&one_wire);

IFTTTWebhook critical_webhook(CRITICAL_API_KEY, CRITICAL_EVENT_NAME);
IFTTTWebhook log_webhook(LOG_API_KEY, LOG_EVENT_NAME);


bool last_normal_temp_state = true;


void setup() {
 // Initializing Serial communication.
 Serial.begin(9600);
 Serial.println("Init... T9_Human_Vitals");


 // Start up the library
 temp_sensor.begin();


 // Setup up WiFi and Connecting to an active hotspot.
 Serial.print("\n\nCnonnecting to ");
 Serial.println(WIFISSID);


 WiFi.begin(WIFISSID, PASSWORD);
 while (WiFi.status() != WL_CONNECTED) { // Waiting for successful connection
    delay(500);
    Serial.print(".");
 }


 Serial.print("\nRSSI: ");
 Serial.println(WiFi.RSSI());


 Serial.print("WiFi connected. IP address: ");
 Serial.println(WiFi.localIP());


}


// the loop function runs over and over again forever
void loop() {
 // call sensors.requestTemperatures() to issue a global temperature
 // request to all devices on the bus
 Serial.print("Requesting temperatures...");
 temp_sensor.requestTemperatures(); // Send the command to get temperatures
 Serial.println("Done");
 // After we got the temperatures, we can print them here.
 // We use the function ByIndex, and as an example get the temperature from the first sensor only.
 float temp = temp_sensor.getTempFByIndex(0); // temp in F

 // Check if reading was successful
 if(temp != DEVICE_DISCONNECTED_C)
 {
  Serial.print("Temperature for the device 1 (index 0) is: ");
  Serial.print(temp);
  Serial.println(" F");

        bool current_normal_temp_state = (temp <= MAX_TEMP_THRESHOLD) && (temp >=
MIN_TEMP_THRESHOLD);
  if(current_normal_temp_state == false && last_normal_temp_state == true){
   Serial.println("Temp out of normal range.");
   critical_webhook.trigger(String(temp).c_str());
  }
  last_normal_temp_state = current_normal_temp_state;
  log_webhook.trigger(String(temp).c_str());
 }
 else
 {
  Serial.println("Error: Could not read temperature data");
 }
 delay(5000);
}
```

Dashboard Snippet (if any):

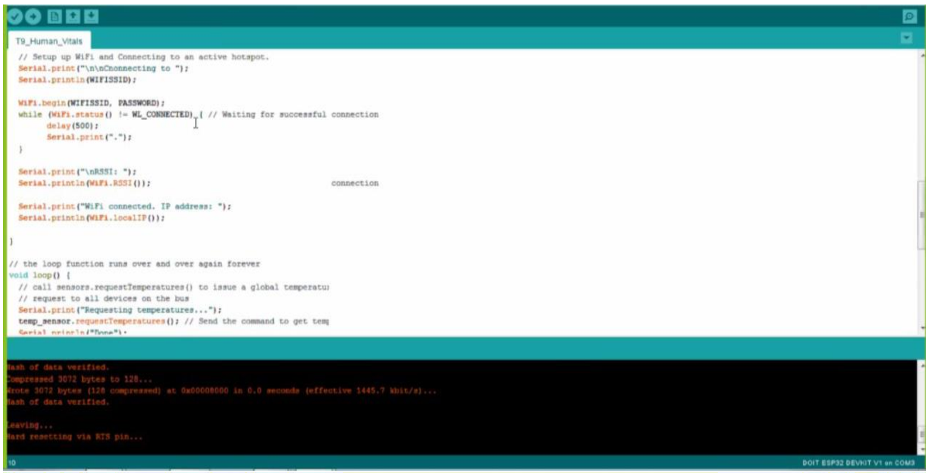


You have 0 available Applets. Upgrade to Pro to create unlimited Applets. Learn more

If Maker Event "temp_log", then Add row to Harwinder Kaur's Google Drive spreadsheet
by harwinderiamm
Connected

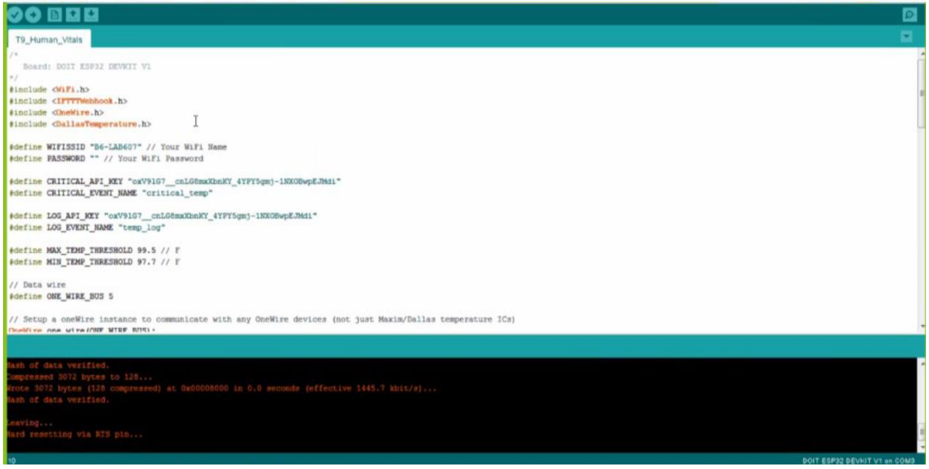
If Maker Event "critical_temp", then Send a notification from the IFTTT app
by harwinderiamm
Connected

If Maker Event "temp_log", then Add row to Harwinder Kaur's Google Drive spreadsheet
by harwinderiamm
Connect



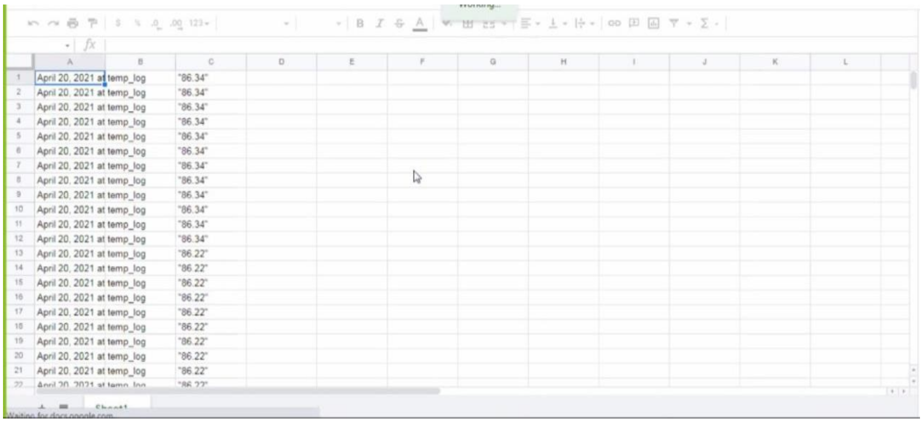
```
  // Setup up WiFi and Connecting to an active hotspot.
  Serial.print("\n\nConnecting to ");
  Serial.println(WIFISSID);

  WiFi.begin(WIFISSID, PASSWORD);
  while (WiFi.status() != WL_CONNECTED) { // Waiting for successful connection
        delay(500);
        Serial.print(".");
  }

  Serial.print("\nRSSI: ");
  Serial.println(WiFi.RSSI());                              connection

  Serial.print("WiFi connected. IP address: ");
  Serial.println(WiFi.localIP());

}

// the loop function runs over and over again forever
void loop() {
  // call sensors.requestTemperatures() to issue a global temperatu
  // request to all devices on the bus
  Serial.print("Requesting temperatures...");
  temp_sensor.requestTemperatures(); // Send the command to get tem
  Serial.println("Done");
```

```
Hash of data verified.
Compressed 3072 bytes to 128...
Wrote 3072 bytes (128 compressed) at 0x00008000 in 0.0 seconds (effective 1445.7 kbit/s)...
Hash of data verified.

Leaving...
Hard resetting via RTS pin...
```

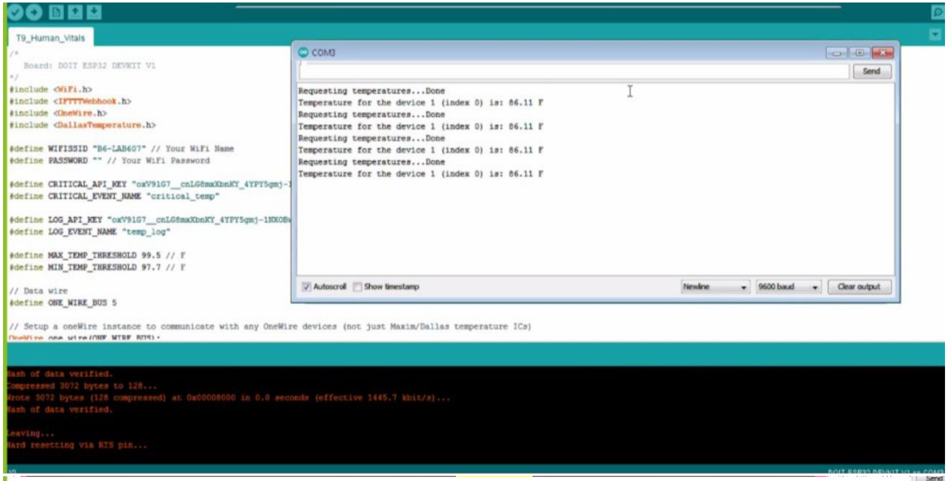


```
/*
   Board: DOIT ESP32 DEVKIT V1
*/
#include <WiFi.h>
#include <IFTTTWebhook.h>
#include <OneWire.h>
#include <DallasTemperature.h>

#define WIFISSID "B6-LAB607" // Your WiFi Name
#define PASSWORD "" // Your WiFi Password

#define CRITICAL_API_KEY "oxV91G7__cnLG8mxXbnKY_4YPY5gmj-1NXOBwpEJMdi"
#define CRITICAL_EVENT_NAME "critical_temp"

#define LOG_API_KEY "oxV91G7__cnLG8mxXbnKY_4YPY5gmj-1NXOBwpEJMdi"
#define LOG_EVENT_NAME "temp_log"

#define MAX_TEMP_THRESHOLD 99.5 // F
#define MIN_TEMP_THRESHOLD 97.7 // F

// Data wire
#define ONE_WIRE_BUS 5

// Setup a oneWire instance to communicate with any OneWire devices (not just Maxim/Dallas temperature ICs)
OneWire one_wire(ONE_WIRE_BUS);
```

```
Hash of data verified.
Compressed 3072 bytes to 128...
Wrote 3072 bytes (128 compressed) at 0x00008000 in 0.0 seconds (effective 1445.7 kbit/s)...
Hash of data verified.

Leaving...
Hard resetting via RTS pin...
```



	A	B	C
1	April 20, 2021 at	temp_log	"86.34"
2	April 20, 2021 at	temp_log	"86.34"
3	April 20, 2021 at	temp_log	"86.34"
4	April 20, 2021 at	temp_log	"86.34"
5	April 20, 2021 at	temp_log	"86.34"
6	April 20, 2021 at	temp_log	"86.34"
7	April 20, 2021 at	temp_log	"86.34"
8	April 20, 2021 at	temp_log	"86.34"
9	April 20, 2021 at	temp_log	"86.34"
10	April 20, 2021 at	temp_log	"86.34"
11	April 20, 2021 at	temp_log	"86.34"
12	April 20, 2021 at	temp_log	"86.34"
13	April 20, 2021 at	temp_log	"86.22"
14	April 20, 2021 at	temp_log	"86.22"
15	April 20, 2021 at	temp_log	"86.22"
16	April 20, 2021 at	temp_log	"86.22"
17	April 20, 2021 at	temp_log	"86.22"
18	April 20, 2021 at	temp_log	"86.22"
19	April 20, 2021 at	temp_log	"86.22"
20	April 20, 2021 at	temp_log	"86.22"
21	April 20, 2021 at	temp_log	"86.22"
22	April 20, 2021 at	temp_log	"86.22"

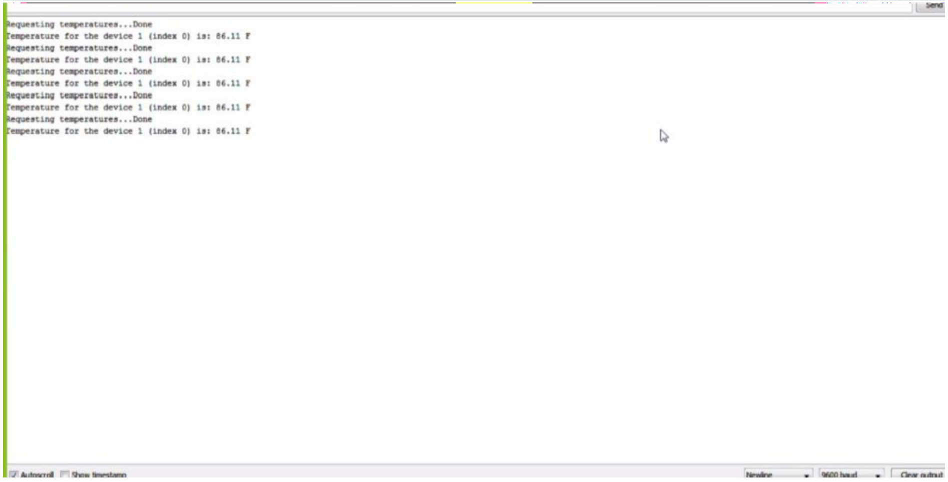


```
/*
  Board: DOIT ESP32 DEVKIT V1
*/
#include <WiFi.h>
#include <IFTTTWebhook.h>
#include <OneWire.h>
#include <DallasTemperature.h>

#define WIFISSID "B6-LAB607" // Your WiFi Name
#define PASSWORD "" // Your WiFi Password

#define CRITICAL_API_KEY "oxV91G7__cnLG8mxXbnKY_4YPY5gmj-1
#define CRITICAL_EVENT_NAME "critical_temp"

#define LOG_API_KEY "oxV91G7__cnLG8mxXbnKY_4YPY5gmj-1NXOBw
#define LOG_EVENT_NAME "temp_log"

#define MAX_TEMP_THRESHOLD 99.5 // F
#define MIN_TEMP_THRESHOLD 97.7 // F

// Data wire
#define ONE_WIRE_BUS 5

// Setup a oneWire instance to communicate with any OneWire devices (not just Maxim/Dallas temperature ICs)
OneWire one_wire(ONE_WIRE_BUS);
```

```
Requesting temperatures...Done
Temperature for the device 1 (index 0) is: 86.11 F
Requesting temperatures...Done
Temperature for the device 1 (index 0) is: 86.11 F
Requesting temperatures...Done
Temperature for the device 1 (index 0) is: 86.11 F
Requesting temperatures...Done
Temperature for the device 1 (index 0) is: 86.11 F
```

☑ Autoscroll □ Show timestamp Newline 9600 baud Clear output

```
Hash of data verified.
Compressed 3072 bytes to 128...
Wrote 3072 bytes (128 compressed) at 0x00008000 in 0.0 seconds (effective 1445.7 kbit/s)...
Hash of data verified.

Leaving...
Hard resetting via RTS pin...
```



```
Requesting temperatures...Done
Temperature for the device 1 (index 0) is: 86.11 F
Requesting temperatures...Done
Temperature for the device 1 (index 0) is: 86.11 F
Requesting temperatures...Done
Temperature for the device 1 (index 0) is: 86.11 F
Requesting temperatures...Done
Temperature for the device 1 (index 0) is: 86.11 F
Requesting temperatures...Done
Temperature for the device 1 (index 0) is: 86.11 F
```

☑ Autoscroll □ Show timestamp Newline 9600 baud Clear output

Outcome:

- Establish an interface between embedded IoT system and the physical world through sensors, to read the state of the world, and actuators, to change the state of the world.
- Establish connectivity of IoT modules with cloud for sensor data collection and management.