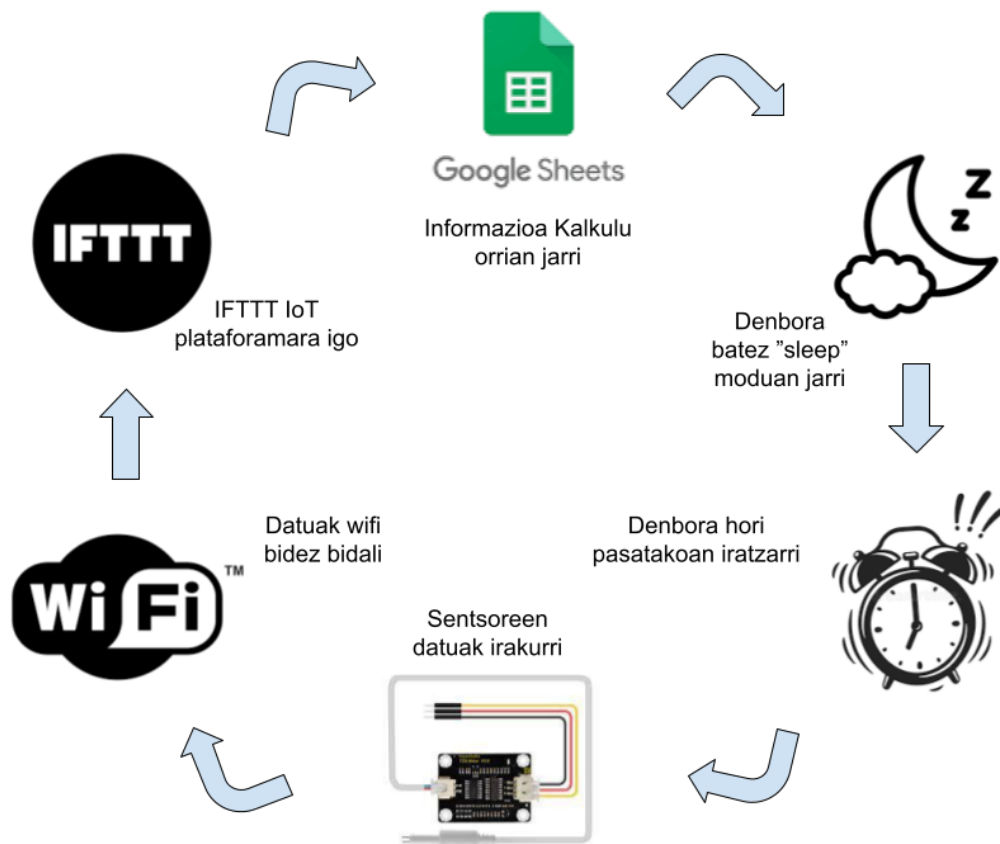


Datu hauek lortu ahal izateko, wifi antenadun mikrokontrolagailua eta IoT plataformak erabiliz, sentsoreen datuak (tenperatura, hezetasuna eta ongarriaren kontzentrazioa) googleko kalkulu horrian ikusi ahal izango dira. Horretarako, mikrokontrolagailuak wifi sarea duen ingurunea beharko du. Wifi ingurunerik ezean, tarteko komunikazioa beste modu batekoa izan beharko litzateke, LoRa adibidez. Proiektu hau aurrerago ikus dezakegu. Horrela, ongarri nahikoa duen edo ez ikusteko aukera izango dugu; baita tenperaturaren arabera ere zer nolako aldaketan erakusten dituen.



## 2. aterialen zerrenda

Zenbakia	Izena	Deskripzioa
1	Esp32	Mikrokontrolagailua
2	DHT11	Tenperatura eta hezetasun sentsorea (airean)
3	DS18B20	Tenperatura sentsorea (uretan)
4	EC sensor	Elektrizitatearen eroakortasun sentsorea
5	USB kablea	USB A-tik micro-rako kablea. Kotralagailua eta ordenagailua konektatzeko

6	Erresistentzia	4.7K $\Omega$ erresistentzia
7	Board	Konexioak egiteko plaka
8	Konexio kableak	Board barruan eta kontrolagailuaren artean konexioak egiteko

### 3. Arduino IDE eta liburutegiak

Mikrokontrolagailua programatzeko Arduino IDE erabiliko dugu. IDEak, garapen-ingurune integratuak, aplikazioa sortzeko softwareak dira. Arduino IDEa oso erabilia da mikrokontrolagailuen programazioan, mikrokontrolagailu txartel mota asko onartzen baititu eta ingurune irekia izatearekin komunitate oso zabala dauka, foro eta liburutegi ugari dituen.

Arduino IDE softwarea lortzeko: [Arduino IDE instalazioa](#).

ESP32 motako kontrolagailuak programatu ahal izateko, hasieraketa [hemen](#).

Arduino IDE barruan, programazioko kodea idazterako garaian gure programak funtzionatu ahal izateko funtsezko liburutegi batzuk beharko ditugu. Liburutegi hoiak instalazioa nola egiten den ikusteko [hemen](#).

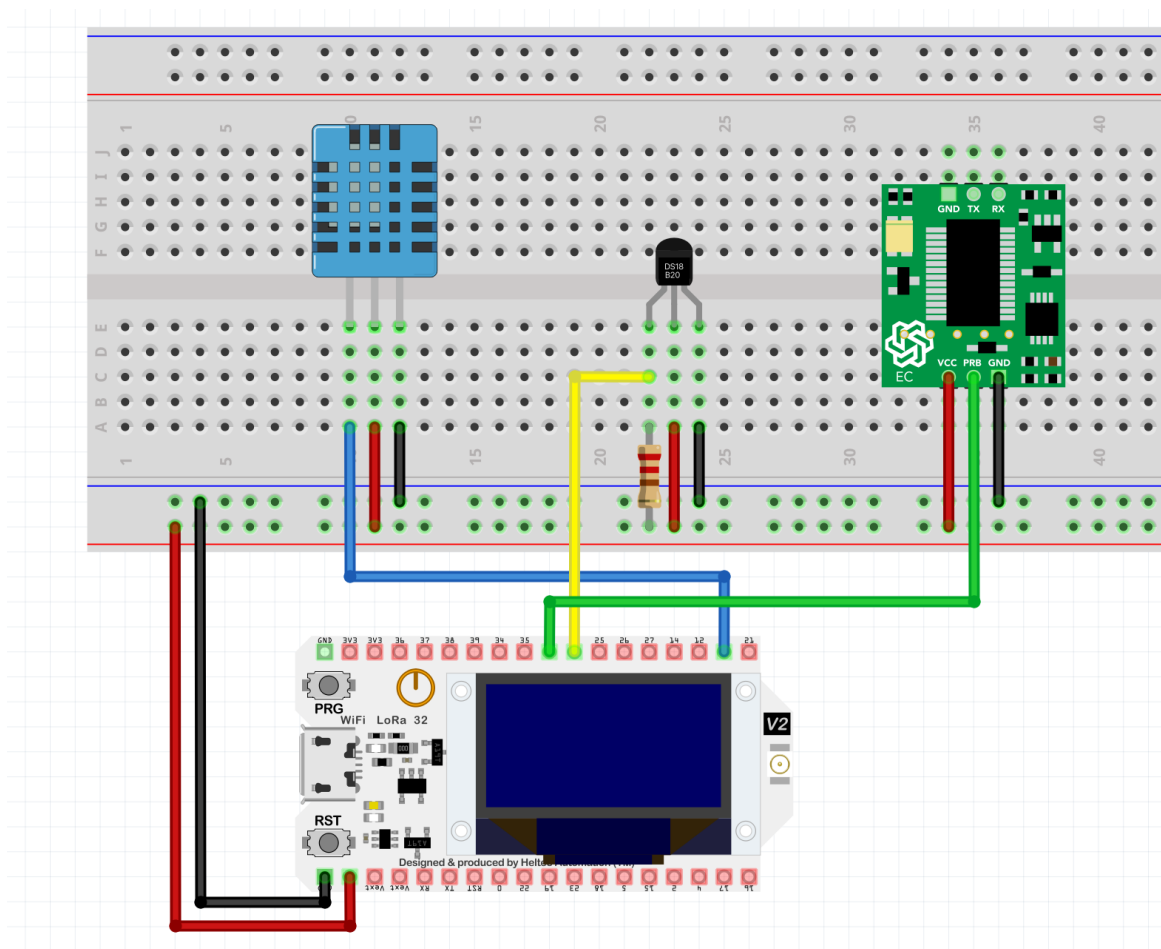
Proiektu honen kasuan, honako liburutegi zerrenda eduki behar ditugu instalatuta (Liburutegi batzuk iada instalaturik egongo dira):

- OneWire.h (Arduinoko liburutegi kudeatzailetik, egilea: Paul Stoffregen)
- DallasTemperature.h (Arduinoko liburutegi kudeatzailetik, egilea: Miles Burton)
- DHT.h (Arduinoko liburutegi kudeatzailetik, egilea: Adafruit)
- Wifi.h (Arduinoko liburutegi kudeatzailetik, egilea: Arduino)

### 4. Eskematikoa eta diagrama

Sentsoreen irakurketa egokiak egiteko, honako konexioak egin behar dira:

- Heltec Wifi Lora 32 txartelaren 5v, GND, 13, 33 eta 32 pinak erabiliko ditugu ([pinout](#))
- DHT 11 sentsoreak seinalea 13. pinean + 5v, GND
- DS18B20 sentsoreak seinalea 33. pinean + 5v, GND + 4.7K $\Omega$  (seinalearen pinetik 5v)
- EC TDS sentsoreak seinalea 32. pinean + 5v, GND



## 5. Kodearen azalpena eta IoT konfigurazioa

### 5.1. Sortu IFTTT kontua

Proiektu honetarako IFTTT erabiliko dugu Google-ko kalkulu horriarekin integratzeko. Beraz, lehenengo urratsa IFTTT kontua sortzea, aurrez ez baduzu. IFTT kontu bat sortzea doakoa da, baina erabilerara mugatua dauka doako bertsioan.

Joan web orrialde ofizialera: [ifttt.com](https://ifttt.com) eta sartu zure emaila hasiera emateko.

## Get started with IFTTT



Continue with Apple



Continue with Google



Continue with Facebook

Or use your email to [sign up](#) or [log in](#)

Behin IFTTT kontua sortuta dukazula, sortu aplikazioa. Horretarako **Create** sakatu.



## Explore

All

Applets

Services

Stories



Search Applets or services

Ondoren, baldintzako gertaera zehaztu behar dugu. Horretarako **Add** sakatu **If This** atalean.

Cancel

## Create your own



Upgrade for more, faster, better Applets with advanced features. Upgrade

You're using 0 of 5 Applets


If This

Add

Then That

Jarraian, **Webhooks** zerbitzua bilatu eta "Receive a web request" aukeratu.

## Choose a service


  
Webhooks

### Receive a web request

This trigger fires every time the Maker service receives a web request to notify it of an event. For information on triggering events, go to your Maker service settings and then the listed URL (web) or tap your username (mobile)

Jarri izena proiektuari eta **Create trigger** eman. Ondoren izen hau beharko dugu arduinoko kodea jartzeko.

## Complete trigger fields



### Receive a web request

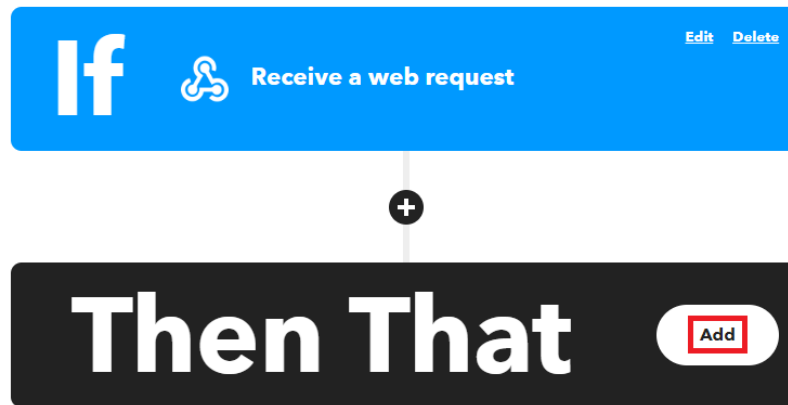
This trigger fires every time the Maker service receives a web request to notify it of an event. For information on triggering events, go to your Maker service settings and then the listed URL (web) or tap your username (mobile)

**Event Name**

The name of the event, like "button\_pressed" or "front\_door\_opened". Use only letters, numbers, and underscores


**Create trigger**

Orain baldintzako akzioa zehaztea geratzen da. Horretarako **Then That** atalean **Add** sakatu.



Ondoren, gure datuak Google-ko kalkulu orri batea azaltzeko, bilatu **Google Sheets** zerbitzua eta **Add row to spreadsheet** aukeratu.

### Choose a service




Google Sheets

#### Add row to spreadsheet

This action will add a single row to the bottom of the first worksheet of a spreadsheet you specify. Note: a new spreadsheet is created after 2000 rows.


Googleko zein kontutan gehitu behar den adierazi eta gero, kalkulu orriaren izena zehaztuko da.



## Add row to spreadsheet

This action will add a single row to the bottom of the first worksheet of a spreadsheet you specify. Note: a new spreadsheet is created after 2000 rows.

**Google Sheets account**



Free **Add new account**

**Spreadsheet name**

Will create a new spreadsheet if one with this title doesn't exist **Add ingredient**

Beherago, kalkulu orriaren formatua zehazten da. Kalkuluko lehenengo zutabeen gertaeraren ordua eta eguna agertuko dira. Bigarrenean, proiektuaren izena. Ondorengo hiru zutabeetan sentsoeren datuak. Datu hauen ordena arduino kodean zehaztuko dugu. Jarraian, kalkulu orri hau sortuko den karpeterari izena jarri. Aurreko konfigurazioan aldaketarik egin nahi izanez gero, **Add ingredient** sakatu; bestela **Create action**.

**Formatted row**

OccurredAt	EventName
Value1	Value2

Use "|||" to separate cells **Add ingredient**

**Drive folder path**

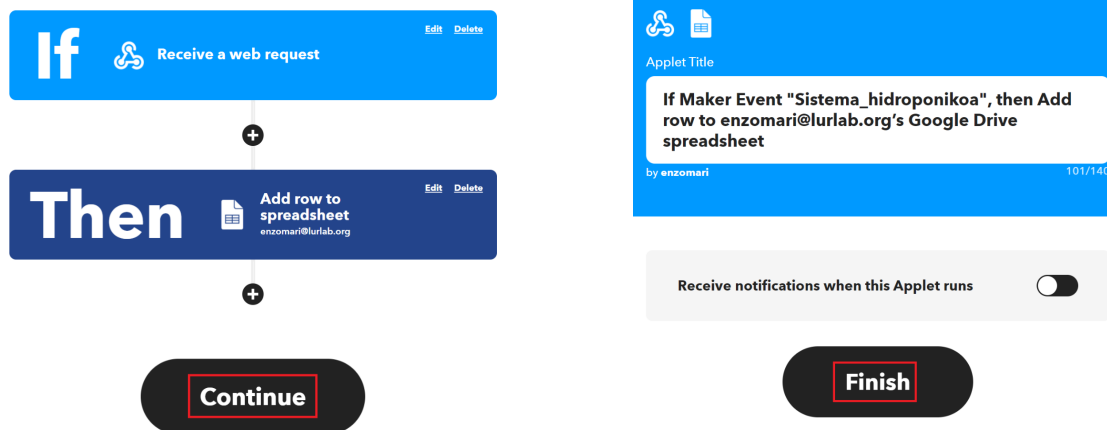
IFTTT/MakerWebbooks/  
EventName

Format: some/folder/path (defaults to "IFTTT") **Add ingredient**

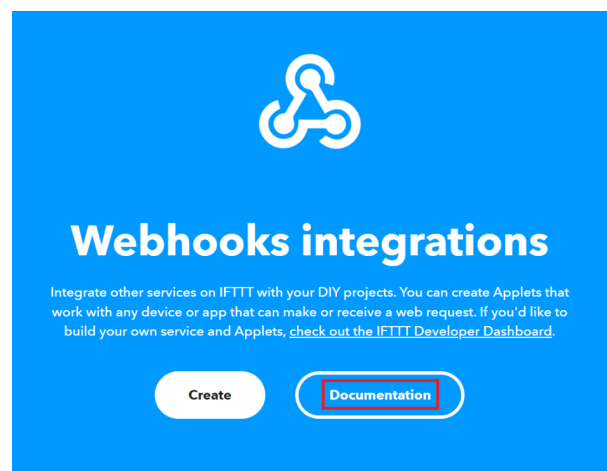
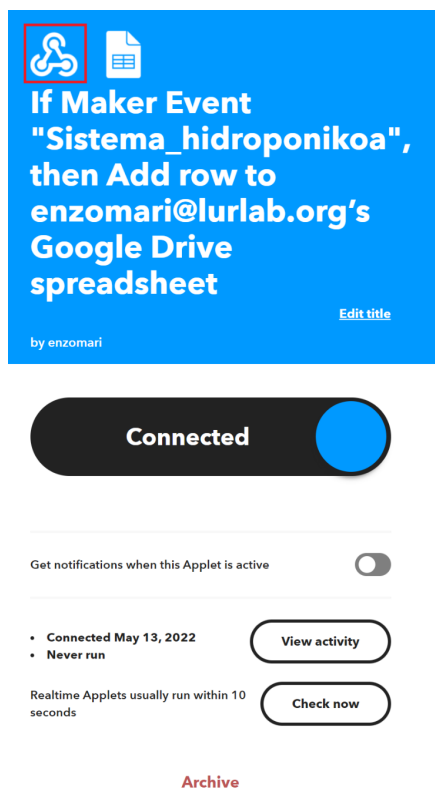
**Create action**



Azkenik, **Continue** eta **Finish** sakatu.



Finish sakatu ondoren, beste orri bat agertuko zaigu. Orri hori itxi aurretik, webhooks logoa sakatuko dugu eta dokumentaziora joango gara. Bertan, aurrerago beharko dugun informazioa egongo da.



**Applets** My Applets Details



[◀ Back to service](#)

Make a POST or GET web request to:

\* Note the extra `/json` path element in this trigger.

```
{ "this" : [ { "is": { "some": [ "test", "data" ] } } ] }
```

```
curl -X POST -H "Content-Type: application/json" -d '{"this":{"is":{"some":["test","data"]}}}'  
https://maker.ifttt.com/trigger/{event}/json/with/key/{key}
```

## Test It

Make a POST or GET web request to:

With an optional JSON body of:

The data is completely optional, and you can also pass `value1`, `value2`, and `value3` as query parameters or form variables. This content will be passed on to the action in your Applet.

```
curl -X POST -H "Content-Type: application/json" -d '{"value1": "10", "value2": "20", "value3": "30"}' https://maker.ifttt.com/trigger/Sistema_hidroponikoa/with/key/
```

**Test It**

Beraz, zure Google Drive kontuan kalkulu orri bat sortu baldin bada, IFTTT konfigurazioarekin amaitu dugu. Orain Arduino kodea zehaztea geratzen da. Kodea eskuragarri dago gure [Github](#) karpetan edo dokumentu honen amaieran eranskinetan. Hala ere, kode honetan norbere parametroak jartzeko zenbait aldaketa egin behar dira.

## 5.2. Kodearen azalpena pausoz pauso

Lehenik eta behin, gure wifi sarearen izena eta pasahitza sartu behar dira (kakotxen barruan). Include komandoaren laguntzaz liburutegiak sartzen dizkiogu. Pauso honekin, internet konexioaren datuak sartzen dira

```
#include <WiFi.h> //WiFi-ra konektatzeko liburutegia

//Zure WiFi sarean izena eta pasahitza jarri kakotxen barruan
const char* ssid = "WiFi sarearen izena"; //Aldatu lerro hau zure wifiaren izenarekin
const char* password = "pasahitza"; //Aldatu lerro hau zure wifiaren pasahitzarekin
```

IFTTT.com > my applets > Sistema hidroponikoa > webhooks logoa > documentation

Aurretik aipatutako dokumentaziotik esteka bat hartu eta kodean jarri behar da.



Your key is:



[Back to service](#)

### To trigger an Event with an arbitrary JSON payload

Make a POST or GET web request to:

[https://maker.ifttt.com/trigger/Sistema\\_hidroponikoa/json/with/key/REDACTED](https://maker.ifttt.com/trigger/Sistema_hidroponikoa/json/with/key/REDACTED)

*\* Note the extra /json path element in this trigger.*

```
//Aldatu lerro hau zure IFTTT-ko gertakizunaren izenarekin (event) eta giltza-zenbakiarekin (key)
const char* resource = "https://maker.ifttt.com/trigger/EVENT_NAME/with/key/dhJdzqY3Hdikay5JfjMnm8"; //IFTTT - webhooks - documentation
const char* server = "maker.ifttt.com"; //Erabiliko den zerbitzaria IoT
```

Zenbat denbora pasako da (segundotan) sentsoreen irakurketa batetik hurrengora?. Hori finkatzen da ondorengo kode zatian.

```
uint64_t uS_TO_S_FACTOR = 1000000; //Mikrosegunduetatik segunduetara aldatzeko
uint64_t TIME_TO_SLEEP = 300; //Jarri lerro honetan zenbateko tartea nahi den datuak zenbat segunduro jaso nahi diren
```

Sentsoreen sarrerarekin hasi gaitezke puntu honetan. Lehenengoa DHT11 sentsorea izango da. Hemen, lehenik liburutegia sartzen diogu eta sentsorea mikrokontrolagailuaren zenbagarren pinera konektatuta dagoen esan.

```
#include <DHT.h>      //Airearen tenperatura eta hezetasuna lortzeko liburutegia
#define DHTPIN 13      //Mikrokontrolagailuan DHT11 sentsorea konektatzeko erabiliko den hanka
#define DHTTYPE DHT11  //Erabiliko dugun sentsore mota
DHT dht(DHTPIN, DHTTYPE); //DHT11 sentsorearen konfigurazioa
```

Bigarren sentsorea DS18B20 sentsorea izango da. Honek ere aurreko sentsorearen konfigurazioaberdina behar du, baina komandoak desberdinak dira.

```
//Uraren tenperatura lortzeko liburutegiak
#include <OneWire.h>
#include <DallasTemperature.h>
#define oneWireBus 33 //Mikrokontrolagailuan DS18B20 sentsorea konektatzeko erabiliko den hanka
//DS18B20 sentsorearen konfigurazioa
OneWire oneWire(oneWireBus);
DallasTemperature sensors(&oneWire);
```

Azkeneko sentsorea elektrizitatearen eroankortasunarena da. Honek ez du liburutegirik, baina konfigurazioa jartzea behar du. Sentsorea mikrokontrolagailuaren zein pinetan joango den, erreferentziako tentsioa (esp32 txarteletan 3.3V da) eta kodean beharrezkoak izango ditugun aldagaiak sortu.

```
#define TdsSensorPin 32 //Mikrokontrolagailuan EC Tds sentsorea konektatzeko erabiliko den hanka
#define VREF 3.3        // Mikrokontrolagailuaren AnalogicDigitalConverter (ADC) erreferentziako tentsioa
float averageVoltage = 0, tdsValue = 0, tdsValueA = 0, ZuzenketaKoeff = 0.15; //Zuzenketa koefizientea beharren arabera egokitu
```

Konfigurazio guztiak egin ostean hasieraketan sartzen da kodea. Hemen, datuen transferentziako abiadura eta sentsoreen hasieraketan jartzen dira.

```
void setup()
{
    Serial.begin(115200); //Datuen trasferentziazako abiadura
    dht.begin();          //DHT11 sentsorearen hasieraketa
    sensors.begin();      //DS18B20 sentsorearen hasieraketa
    pinMode(TdsSensorPin, INPUT); //EC Tds sentsorearen konfigurazioa
}
```

Hurrengo pausoa, sentsoreen irakurketak egitea da. DHT11 sentsoreak, airearen tenperatura neurtzeaz gain, hezetasuna ere neurtu dezake.

```
//Sentsoreen datuak lortu
float t = dht.readTemperature(); //airearen tenperatura
//float h = dht.readHumidity();   //airearen hezetasuna
sensors.requestTemperatures();
float temperatureC = sensors.getTempCByIndex(0); //uraren tenperatura
```

DHT11 eta DS18B20 sentsoareak, liburutegiek dakarten komandoekin zuzenean irakurri daiteke tenperaturaren balioa. Elektrizitatearen eroankortasunarekin ordea, zenbait kalkulu egin behar dira.

```
int ADCval = analogRead(TdsSensorPin); //EC Tds sentsoaretik jasotako tentsioaren irakurketa ADC
averageVoltage = (ADCval* (float)VREF / 4095.0) - ZuzenketaKoeff; // ADC baliotik tentsioaren baliora pasatzeko eragiketa
float compensationCoefficient=1.0+0.02*(t-25.0); //tenperaturaren arabera moldaketa
float compensationVoltage=averageVoltage/compensationCoefficient; //tenperaturaren arabera moldaketa
//Tentsioaren baliotik EC (ppm) baliora pasatzeko eragiketa
tdsValue=(133.42*compensationVoltage*compensationVoltage*compensationVoltage - 255.86*compensationVoltage*compensationVoltage + 857.39*compensationVoltage)*0.5;
```

Behin sentsoaren balioak lortutakoan, Google Driveko kalkulu orrira IFTTT bidez bidaltzeko prest jartzen dira (hemen erabakitzen dugu Google Driveko kalkulu orrian zutabe bakoitzean zein datu edukiko dugun).

```
//Balioak IFTTT plataformara bidaltzeko prestatu
float value1 = t;
float value2 = temperatureC;
float value3 = tdsValue;
```

Ondoren, sentsoarek balioa ondo irakurtzen duten ikusteko, ordenagailuan bertan eta internetez bidali aurretik, serie-portuan ikusteko aukera dago (kontuan izan, serie-portuko irakurketa abiadura, bidalketa abiaduraren berdina izan behar duela, aurrerago ikusko dugu).

```
//Serie monitorean sentsoareetatik jasotako balioak erakutsi
Serial.print("Temperatura (Airea): ");
Serial.print(value1);
Serial.print("°C Temperatura (Ura): ");
Serial.print(value2);
Serial.print("°C Eroankortasuna: ");
Serial.print(value3);
Serial.println("ppm");
```

Azkenik, datu hauek internet bidez IFTTT plataformara internet bidaltzeko zatia geratzen da. Kode zati honetan ez dugu zerta ezer aldatu beharrik, dagoen bezela utzi daiteke.

Zati honetan, internet konexioa bilatzen du eta bilaketaren emaitza azalduko du serie-portuan. Internet sarea bilatuz gero, **Wifi connected** mezua ikusiko dugu serie-portuan, datuak bidaliko ditu eta kalkulu orrian ikusteko moduan izango gara. Baina denbora tarte horretan wifi sarerik bilatzen ez badu, serie-portuan **failed to connect...** mezua azalduko da eta “sleep” egoeran sartuko da, hurrengo irakurketa egin arte.

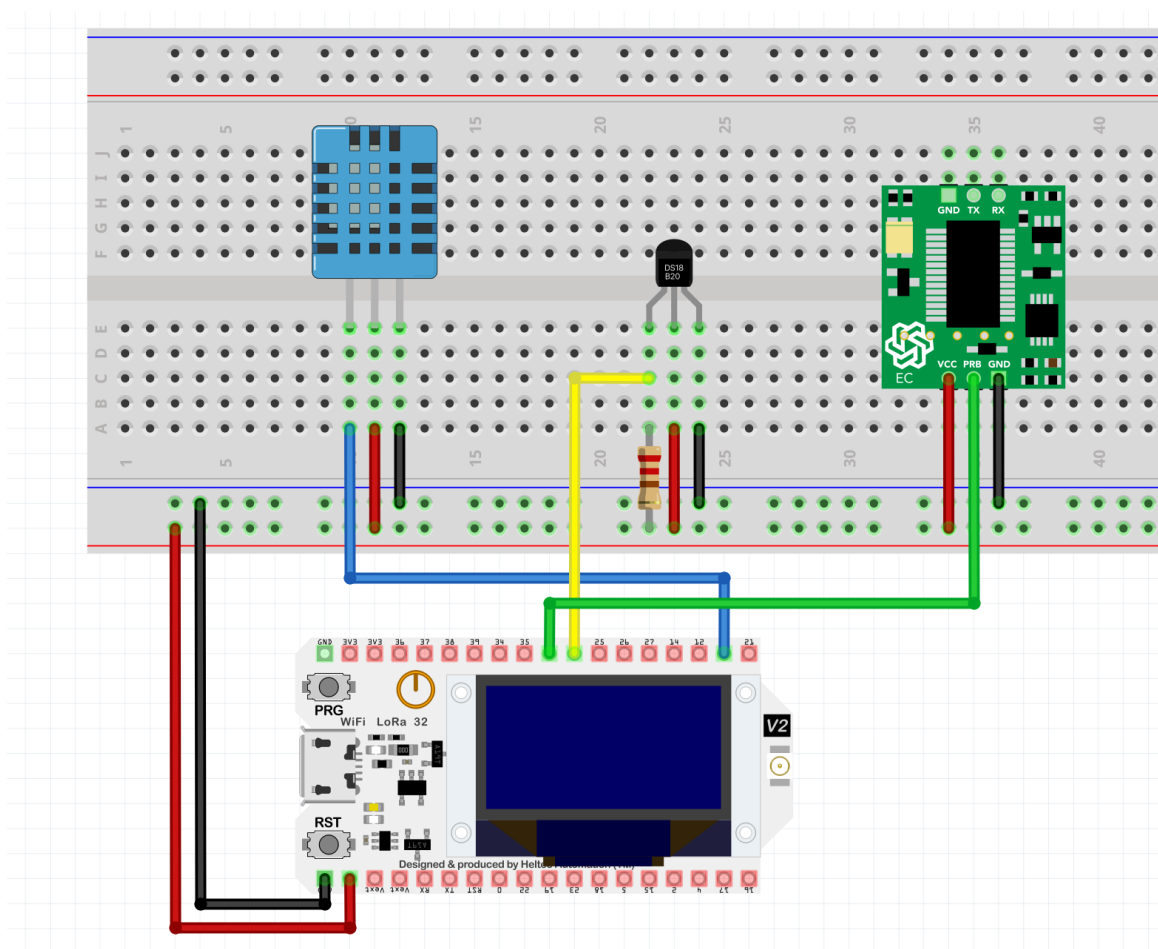
Gure kode honetan, IoT sarearekin kontaktuan gaudenez eta “sleep” egoera kodearen zatia denez, **void loop** barruan ez daukagu ezer jarri beharrik.

## 5.3. Kodea Probatu

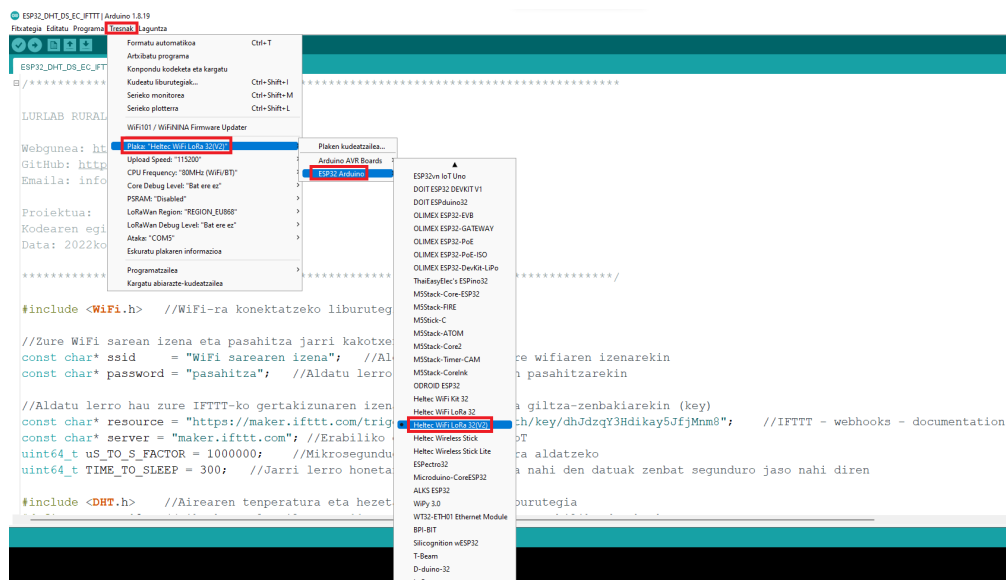
Behin kodearen azalpena eta bakoitzaren egokitzapenak egin ondoren, martxan jartzeko garaia da. Zati honetan, mikrokontrolagailua aukeratu, konexioak egin eta kodea kargatzen erakutsiko da.

Lehenik eta behin, aurreko zatian kodearen azalpenean aipatutako aldaketak egin behar dira, pauso hau egin aurretik zati hori eginda egon behar du; baita liburutegien instalazioa eta esp32 txartelaren hasieraketa ere.

Behin pauso horiek egindakoa, kodea txartelean kargatzeko moduan dago. Lehenengo pausoa, konexio elektrikoak egitea. Gogoratu sentsoak eta mikrokontrolagailuaren arteko konexioak.

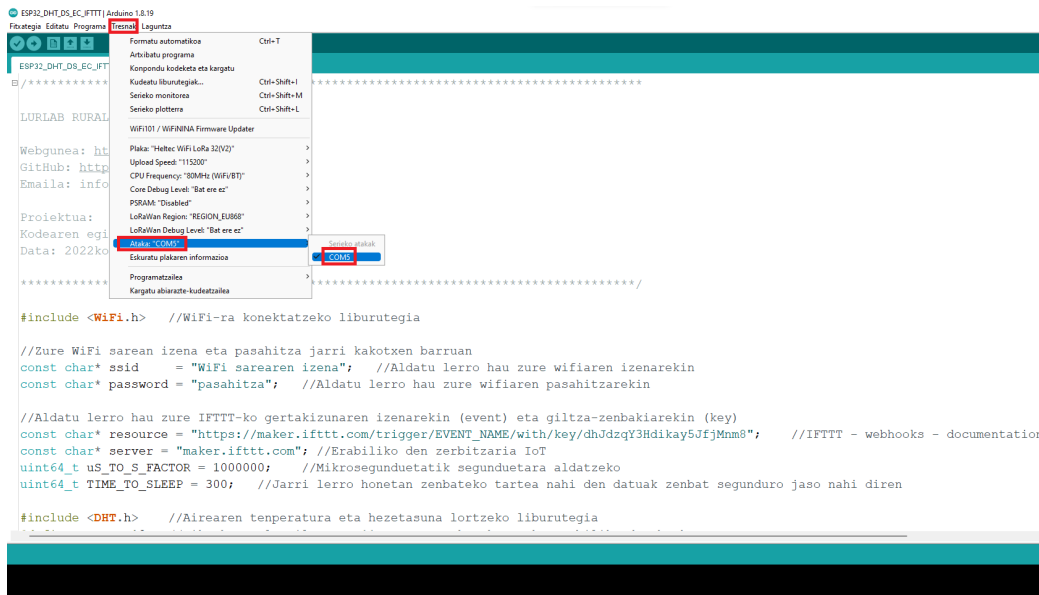


Ondoren, usb kablea ordenagailuan sartu eta serieko ataka aukeratu. Horretarako, **Tresnak > Plaka > Aukeratu zure mikrokontrolagailua** (nire kasuan: Heltec Wifi LoRa 32 V2).



Behin txartela aukeratuta **Tresnak** barruan ere beste zenbait aldaketa egin behar dira. Upload speed: "115200" eta CPU frequency: "80MHz (WiFi/BT)".

Azkenik, serie ataka aukeratu behar da. Nire kasuan COM 5.



Puntu honetan, guztia prest dago kodea txartelean kargatzeko. Horretarako, Kargatu botoiari eman eta kodea mikrokontrolagailura sartzen hasiko da. Ondoren, Serie Monitoreari eman eta mikrokontrolagailuan gertatzen ari dena ikus dezakezu.



```
ESP32_DHT_DS_EC_IFTTT | Arduino 1.8.19
File Edit View Tools Help
ESP32_DHT_DS_EC_IFTTT
//*****
LURLAB RURAL MAKERSPACE

Webgunea: https://www.lurlab.org/
GitHub: https://github.com/lurlab
Email: info@lurlab.org

Proiektua:
Kodearen egilea: Eñaut Erkurdia Mañoa
Data: 2022ko apirilaren 28a

//*****

#include <WiFi.h> //WiFi-ra konektatzeko liburutegia

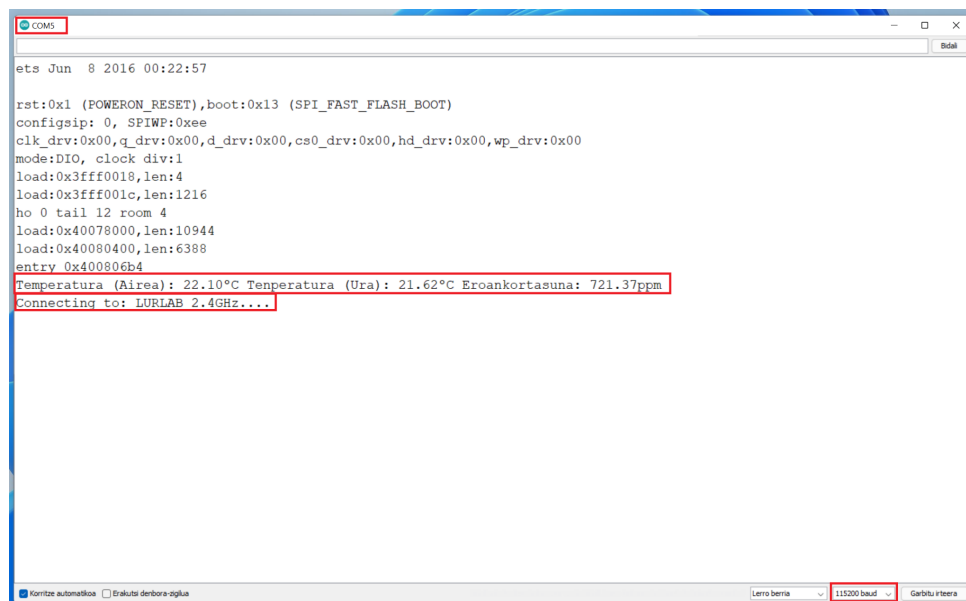
//Zure WiFi sarearen izena eta pasahitza jarri kakotxen barruan
const char* ssid = "WiFi sarearen izena"; //Aldatu lerro hau zure wifiaren izenarekin
const char* password = "pasahitza"; //Aldatu lerro hau zure wifiaren pasahitzarekin

//Aldatu lerro hau zure IFTTT-ko gertakizunaren izenarekin (event) eta giltza-zenbakiarekin (key)
const char* resource = "https://maker.ifttt.com/trigger/EVENT_NAME/with/key/dhJdzgY3Hdikay5JfjMms8"; //IFTTT - webhooks - documentation
const char* server = "maker.ifttt.com"; //Erabiliko den zerbitzaria IoT
uint64_t us_TO_S_FACTOR = 1000000; //Mikrosegunduetatik segunduetara aldatzeko
uint64_t TIME_TO_SLEEP = 300; //Jarri lerro honetan zenbateko tartea nahi den datuak zenbat segunduro jaso nahi diren

#include <DHT.h> //Airearen temperatura eta hezetasuna lortzeko liburutegia

Programa konpilatzen.
```

Serie monitoreak irekitakoan, lehenengo kargako abiadura berean jarri leihoa: 115200 baud. Abiadura berean jarrita, sentsoreek jasotako datuak erakutsiko dira kalkulu orrian agertuko diren orden berean. Ondoren, mikrokontrolagailua kodean adierazitako WiFi sarearen bila hasiko dela ikusiko dugu.



```
COM5
ets Jun  8 2016 00:22:57

rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
configsip: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:1
load:0x3fff0018,len:4
load:0x3fff001c,len:1216
ho 0 tail 12 room 4
load:0x40078000,len:10944
load:0x40080400,len:6388
entry 0x400806b4
Temperatura (Airea): 22.10°C Temperatura (Ura): 21.62°C Eroankortasuna: 721.37ppm
Connecting to: LURLAB 2.4GHz....
```

Behin WiFi sarea aurkitzen duenean, hori adierazteko mezua ikusiko da eta baita IP helbidea ere. Ondoren IFTTT zerbitzarira konektatuko da eta konexioko esteka erakutsiko du. Azkenik, sentsoreen datuak zein proiektutara bidaliko dituen adierazi eta "sleep" egoeran sartuko da.



```
COM5
load:0x3fff001c,len:1216
ho 0 tail 12 room 4
load:0x40078000,len:10944
load:0x40080400,len:6388
entry 0x400806b4
Temperatura (Airea): 22.10°C Temperatura (Ura): 21.62°C Eroankortasuna: 721.37ppm
Connecting to: LURLAB 2.4GHz.....
WiFi connected in: 3844, IP address: 192.168.2.126
Connecting to maker.ifttt.com
Request resource: https://maker.ifttt.com/trigger/Lurlab_hidroponikoa/with/key/
HTTP/1.1 200 OK
Content-Type: text/html; charset=utf-8
Content-Length: 59
Connection: close
Date: Thu, 19 May 2022 09:59:21 GMT
X-Powered-By: Sad Unicorns
X-Robots-Tag: none
X-Top-Secret: VG9vIGVhc3k/IElmlHlvdSBjYW4gcmVhZCB0aGlzLCBFbWVpYCB1cyBhdCBqb2JzK3NlY3JldEBpZnR0dC5jb20uIFdlIHdhbnQgTWI
ETag: W/"3b-kMwVgJpQ3IUtC9hQ0ln3VFwyl5A"
X-Cache: Miss from cloudfront
Via: 1.1 e4b8d81d5f13e1c05d52108e75ecf23c.cloudfront.net (CloudFront)
X-Amz-Cf-Pop: MAD51-C2
X-Amz-Cf-Id: FByG90Z2yREIdmwIoQneIYRa2XVj-XCZcElds4-ZK0U8021r0fAjA==

Congratulations! You've fired the Lurlab_hidroponikoa event
closing connection
Going to sleep now
```

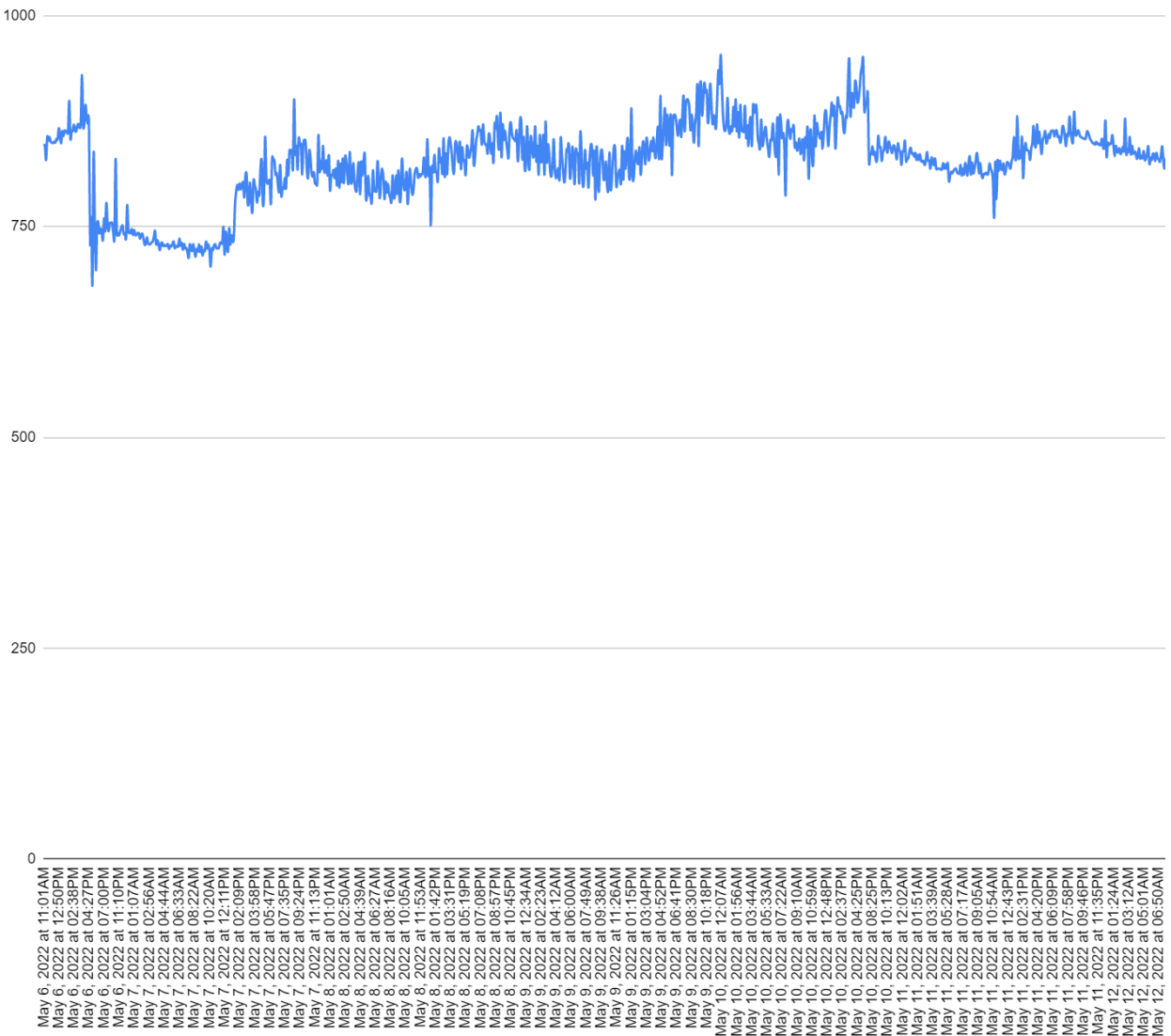
Datuak kalkulu orrian jartzen ditu, datuak jasotako ordu eta egunarekin batera.

IFTTT\_Maker\_Webhooks\_Events Lurlab\_hidroponikoa (1) ☆ 📁 ☁

Fitxategia Editatu Ikusi Txertatu Formatua Datuak Tresnak Luzapenak Laguntza

E1205	fx					
	A	B	C	D	E	
1193	May 19, 2022 at 11:08AM	Lurlab_hidroponikoa	21.9	23.25	716.56	
1194	May 19, 2022 at 11:17AM	Lurlab_hidroponikoa	21.9	22.62	776.69	
1195	May 19, 2022 at 11:25AM	Lurlab_hidroponikoa	22.1	22.37	765.94	
1196	May 19, 2022 at 11:34AM	Lurlab_hidroponikoa	22.2	22.13	770.02	
1197	May 19, 2022 at 11:42AM	Lurlab_hidroponikoa	22.1	21.87	761.37	
1198	May 19, 2022 at 11:43AM	Lurlab_hidroponikoa	22	21.87	744.34	
1199	May 19, 2022 at 11:43AM	Lurlab_hidroponikoa	22	21.87	715.61	
1200	May 19, 2022 at 11:43AM	Lurlab_hidroponikoa	21.9	21.87	704.21	
1201	May 19, 2022 at 11:59AM	Lurlab_hidroponikoa	22.1	21.62	668.28	
1202	May 19, 2022 at 11:59AM	Lurlab_hidroponikoa	22.1	21.62	721.37	
1203	May 19, 2022 at 12:07PM	Lurlab_hidroponikoa	22.1	21.5	747.82	
1204	May 19, 2022 at 12:16PM	Lurlab_hidroponikoa	22.2	21.37	739.4	
1205						
1206						

Behin datu hauek edukita, kalkulu orrian edo zer gauza egin daiteke: diagramak, grafikak, *heat map*-ak... eta beste zenbait gauza.



Tutorial hau gomendagarria eta baliagarria dela iruditzen bazaizu, ikusi bertan dauzkagun beste proiektuak. Edozein duda, zalantza edo gomendio baduzu, idatzi lasai gure posta elektronikora. Honelako proiektuak egitea gustoko baduzu, aprobeztatu eta animatu zaitez gurera etortzeko!

[www.lurlab.org](http://www.lurlab.org)

[info@lurlab.org](mailto:info@lurlab.org)

[github.com](https://github.com)

/\*\*\*\*\*

## LURLAB RURAL MAKERSPACE

Webgunea: <https://www.lurlab.org/>

GitHub: <https://github.com/lurlab>

Emaila: [info@lurlab.org](mailto:info@lurlab.org)

Proiektua:

Kodearen egilea: Eñaut Ezkurdia Muñoa

Data: 2022ko apirilaren 28a

\*\*\*\*\*/

```
#include <WiFi.h> //WiFi-ra konektatzeko liburutegia
```

```
//Zure WiFi sarean izena eta pasahitza jarri kakotxen barruan
```

```
const char* ssid = "WiFi sarearen izena"; //Aldatu lerro hau zure wifiaren izenarekin
```

```
const char* password = "pasahitza"; //Aldatu lerro hau zure wifiaren pasahitzarekin
```

```
//Aldatu lerro hau zure IFTTT-ko gertakizunaren izenarekin (event) eta giltza-zenbakiarekin (key)
```

```
const char* resource =
```

```
"https://maker.ifttt.com/trigger/EVENT_NAME/with/key/dhJdzqY3Hdikay5JfjMnm8"; //IFTTT - webhooks - documentation
```

```
const char* server = "maker.ifttt.com"; //Erabiliko den zerbitzaria IoT
```

```
uint64_t uS_TO_S_FACTOR = 1000000; //Mikrosegunduetatik segunduetara aldatzeko
```

```
uint64_t TIME_TO_SLEEP = 300; //Jarri lerro honetan zenbateko tartea nahi den datuak zenbat segunduro jaso nahi diren
```

```
#include <DHT.h> //Airearen tenperatura eta hezetasuna lortzeko liburutegia
```

```
#define DHTPIN 13 //Mikrokontrolagailuan DHT11 sentsorea konektatzeko erabiliko den hanka
```

```
#define DHTTYPE DHT11 //Erabiliko dugun sentsore mota
```

```
DHT dht(DHTPIN, DHTTYPE); //DHT11 sentsorearen konfigurazioa
```

```
//Uraren tenperatura lortzeko liburutegiak
```

```
#include <OneWire.h>
```

```
#include <DallasTemperature.h>
```

```
#define oneWireBus 33 //Mikrokontrolagailuan DS18B20 sentsorea konektatzeko erabiliko den hanka
```

```
//DS18B20 sentsorearen konfigurazioa
```

```
OneWire oneWire(oneWireBus);
```

```
DallasTemperature sensors(&oneWire);
```

```
#define TdsSensorPin 32 //Mikrokontrolagailuan EC Tds sentsorea konektatzeko erabiliko den hanka
```

```
#define VREF 3.3 // Mikrokontrolagailuaren AnalogicDigitalConverter (ADC)
```

```

erreferentziako tentsioa
float averageVoltage = 0,tdsValue = 0,tdsValueA = 0, ZuzenketaKoef = 0.15; //Zuzenketa
koefizientea beharren arabera egokitu

void setup()
{
    Serial.begin(115200); //Datuen trasferentziarako abiadura
    dht.begin(); //DHT11 sentsorearen hasieraketa
    sensors.begin(); //DS18B20 sentsorearen hasieraketa
    pinMode(TdsSensorPin,INPUT); //EC Tds sentsorearen konfigurazioa

    //Sentsoreen datuak lortu
    float t = dht.readTemperature(); //airearen tenperatura
    //float h = dht.readHumidity(); //airearen hezetasuna
    sensors.requestTemperatures();
    float temperatureC = sensors.getTempCByIndex(0); //uraren tenperatura

    int ADCval = analogRead(TdsSensorPin); //EC Tds sentsoretik jasotako tentsioaren
    irakurketa ADC
    averageVoltage = (ADCval* (float)VREF / 4095.0) - ZuzenketaKoef; // ADC baliotik
    tentsioaren baliora pasatzeko eragiketa
    float compensationCoefficient=1.0+0.02*(t-25.0); //tenperaturaren araberako moldaketa
    float compensationVolatge=averageVoltage/compensationCoefficient; //tenperaturaren
    araberako moldaketa
    //Tentsioaren baliotik EC (ppm) baliora pasatzeko eragiketa
    tdsValue=(133.42*compensationVolatge*compensationVolatge*compensationVolatge -
    255.86*compensationVolatge*compensationVolatge + 857.39*compensationVolatge)*0.5;
    //convert voltage value to tds value

    //Balioak IFTTT plataformara bidaltzeko prestatu
    float value1 = t;
    float value2 = temperatureC;
    float value3 = tdsValue;

    //Serie monitorean sentsoreetatik jasotako balioak erakutsi
    Serial.print("Temperatura (Airea): ");
    Serial.print(value1);
    Serial.print("°C Tenperatura (Ura): ");
    Serial.print(value2);
    Serial.print("°C Eroankortasuna: ");
    Serial.print(value3);
    Serial.println("ppm");

    delay(500);

    //Sentsoreek jasotako balioak IFTTT plataformara bidali
    Serial.print("Connecting to: ");
    Serial.print(ssid);
    WiFi.begin(ssid, password);

```

```

int timeout = 100 * 4;
while(WiFi.status() != WL_CONNECTED && (timeout-- > 0)) {
    delay(250);
    Serial.print(".");
}
Serial.println("");

if(WiFi.status() != WL_CONNECTED) {
    Serial.println("Failed to connect, going back to sleep");
}

Serial.print("WiFi connected in: ");
Serial.print(millis());
Serial.print(", IP address: ");
Serial.println(WiFi.localIP());
Serial.print("Connecting to ");
Serial.print(server);

WiFiClient client;
int retries = 10;
while(!client.connect(server, 80) && (retries-- > 0)) {
    Serial.print(".");
}
Serial.println();
if(!client.connected()) {
    Serial.println("Failed to connect...");
}

Serial.print("Request resource: ");
Serial.println(resource);

String jsonObject = String("{\"value1\":\"" + value1 + "\",\"value2\":\"" + value2 +
 "\",\"value3\":\"" + value3 + "\"}");

client.println(String("POST ") + resource + " HTTP/1.1");
client.println(String("Host: ") + server);
client.println("Connection: close\r\nContent-Type: application/json");
client.print("Content-Length: ");
client.println(jsonObject.length());
client.println();
client.println(jsonObject);

int timeout2 = 5 * 10;
while(!client.available() && (timeout2-- > 0)){
    delay(100);
}
if(!client.available()) {
    Serial.println("No response...");
}

```

```
while(client.available()){
    Serial.write(client.read());
}

Serial.println("\nclosing connection");
client.stop();

esp_sleep_enable_timer_wakeup(TIME_TO_SLEEP * uS_TO_S_FACTOR);
Serial.println("Going to sleep now");
esp_deep_sleep_start();

}

void loop() {
}
```