LoRa proiektua

Egileak

Jon.S eta Ander.B

1. orrialdea

Ziklo bukaerako proiektua: aurkibidea

1 Erronkaren planteamendua

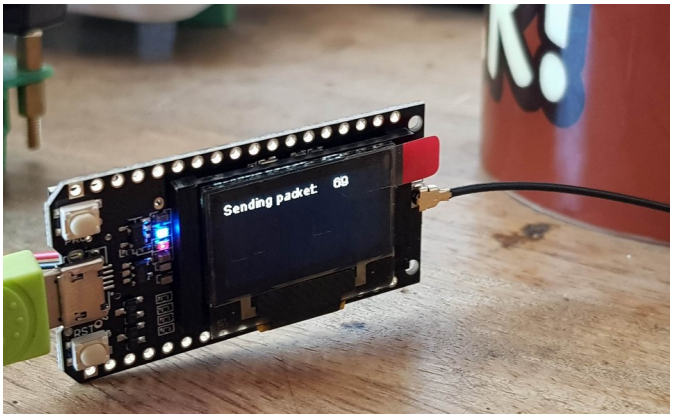
Erronka honetan bi sentsore erabiliko ditugu, DHT-11 hezetasun eta tenperatura sentsorea eta CJMU-811 edo CCS-811 aireko CO_2 kantitatea eta TVOC (bolatil konposatu organikoak) neurtze duena.

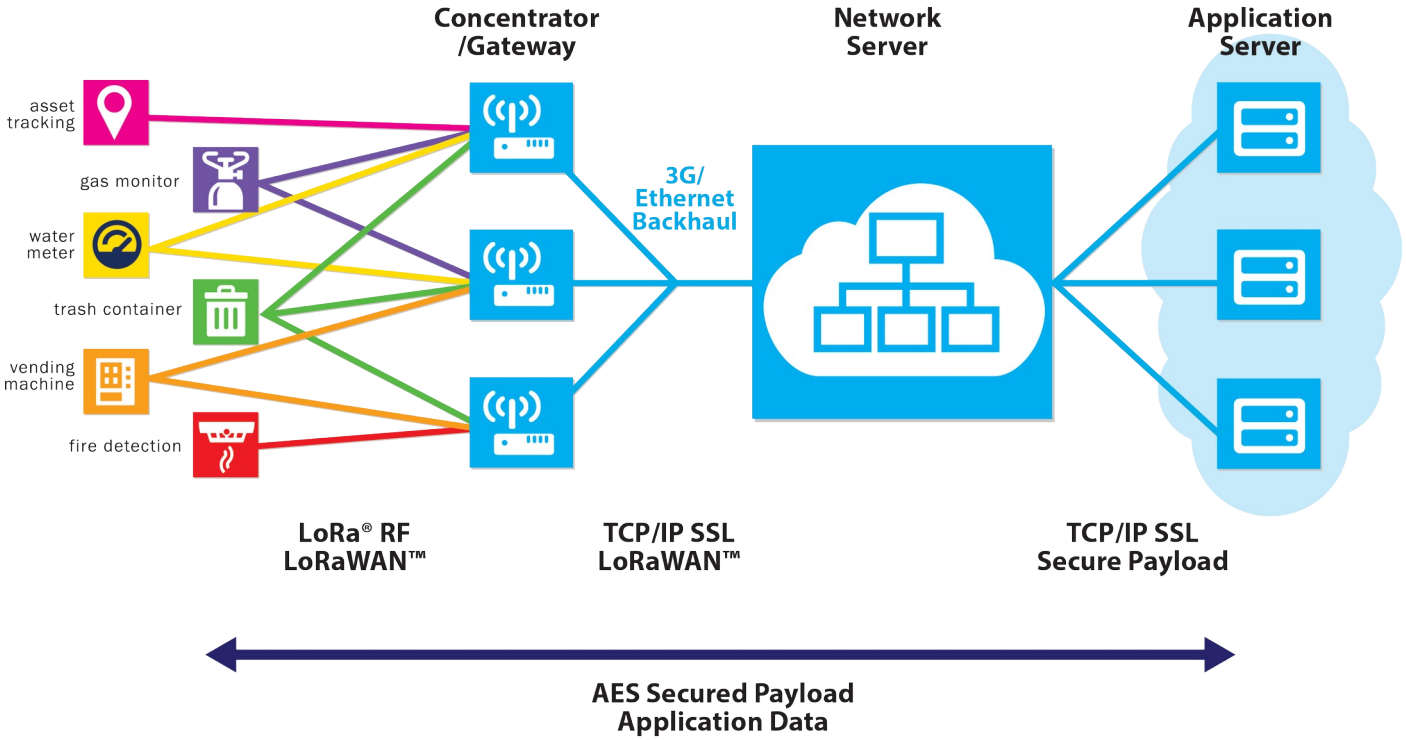
ESP32 TTGO kontrolagailuarekin lan egingo dugu, eta LORA komunikazioa erabiliko dugu, gure kontrolagailua, konfiguratu dugun GateWay-arekin komunikatzeko eta balioak partekatu ahal izateko.

Ondoren GateWay-a konfiguratu dugu TTN (The things Network)ean sortu dugun kontu batean sentsoreetako balioak bertan bistaratzeko.

Eta azkenik Raspberry bat erabiliz dashboard edo aginte panel bat sortzeko eta hor partekatzeko gure ditugun balioak edo informazioa.

Gure helburua elektronika eraikinaren geletako CO_2 kantiatea, tenperatura eta hezetasunaren kontrola izatea da, eta edozein lekutan egonda, balio hauek ikus ahal izatea.





2 Hausnarketak

2.1 HLK-PM03 AC DC

HLK-PM03 AC DC bihurgailua elikadura-iturri, botere-iturri edo iturri kommutatu deitzen den gailu elektroniko bat da. Elektronikan, korronte alternoa irteera batean edo gehiagotan korronte zuzen bihurtzen duen tresna da.

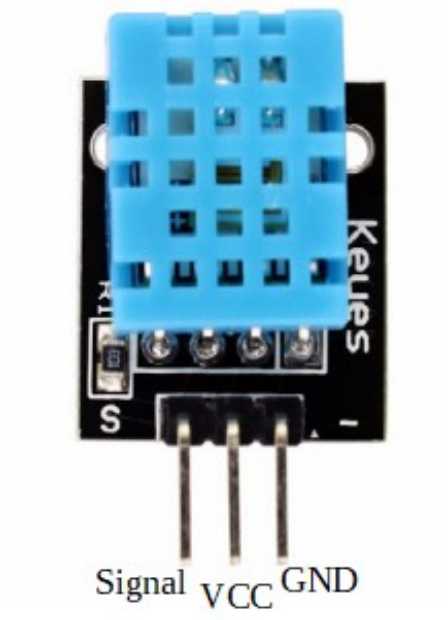
HLK-PM03 AC DC)

2.1.1 DHT11

DHT-11 erabiliko ditugun sentsoreetako bat izango da, sentsore hau tenperatura eta hezetasuna neurtzuko balio du, gure kasuan erabiliko duguna 3 pin ditu, eskumako irudian ikusten denez.

Tenperatura ºCtan neurtzen du eta hezetasuna %tan.

Kostu txikiko eta erabiltzeko erraza den sentsorea da.Hezetasun kapazitarioaren sentsorea eta termistore bat integratzen ditu inguruko airea neurtzeko, eta datuak seinale digital baten bidez erakusten ditu signal pinean (ez du irteera analogikorik)



DHT11 aukeratu dugu sentsore honekin tenperatura eta hezetazuna jakin dezakegu eta aurretik probak egin dugu eta montatzerako orduan erraza izan da.

CJMCU-811

CCS811 airearen kalitatearen sentsorea oso potentzia txikiko gas digitalaren sentsore detektatzaile bat da, oxido metalikoko gas-sentsore bat duena konposatu organiko lurrunkorren gama zabal bat identifikatzeko, airearen kalitatea monitorizatzeko ADC bat eta I2C interfaze bat ematen dituen mikrokontrolagailu batekin.

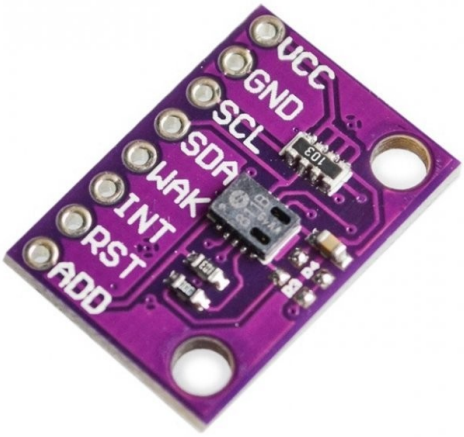
CCS811 argi-sentsorea gas digitalaren sentsorerik onena da airearen kalitatea zehazki monitorizatzeko, Arduino, STM32, Raspberry Pi eta abar erabiliz. Ccs811 gy-811 breakout board sentsorearen modulu hau tamaina txiki, gama zabal eta bereizmen handikoa da.

Sentsore hau gadget pertsonaletako (smartwatchak eta telefonoak) barneko airearen kalitatearen jarraipena egiteko diseinatuta dago. Erraz iristeko, CJMCU-811 modulua nahiago da I2C gailu estandar gisa erabili ahal izateko.

Sentsore hau aukeratu dugu erraza izan delako montatzerakoan. Bere abantaila onetarikoa da ez duela energia asko kontsumitzen eta bebai aukeratu dugu eskuragarri izan dugulako.

ABANTAILAK
Sentsorea eragiteko moduak kudeatzen ditu eta KOLak detektatzen diren bitartean neurketak egitea
ECO2-aren maila edo KOLen indikazioa ematen du Ostatuaren esku-hartzea
Hardware eta softwarearen integrazioa errazten du
Bateriaren iraupena zabaltzen du aplikazio eramangarrietan
Faktorea modu txikian diseinatzeko egokia
Aurreztu% 60rainoko tartea zirkuitu inprimatuaren plakan

Bolumen eta fidagarritasun handirako diseinatua
(>5 urteko balio-bizitza)



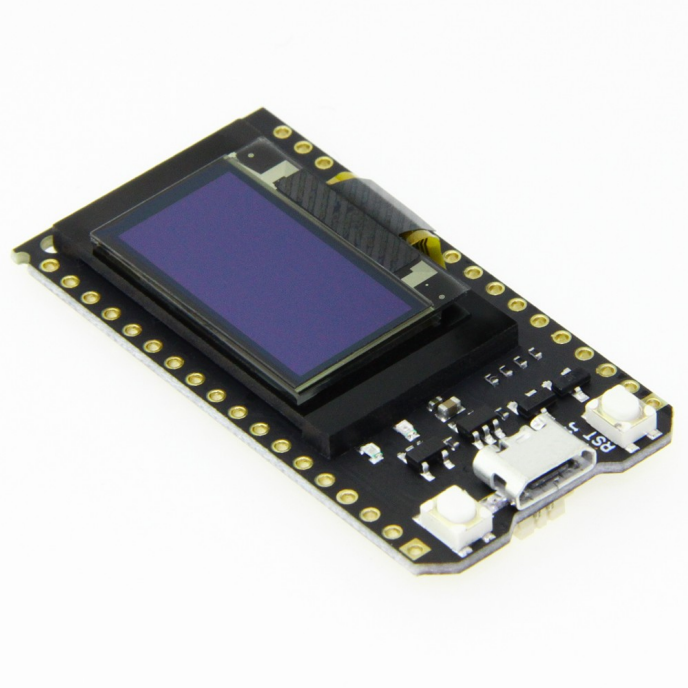
2.1.4 ESP32 TTGO LORA

ESP-32 RF modulu generiko indartsua da, WiFi eta Bluetooth funtzioekin. WiFi eta Bluetooth konbinazio dualak modulu hau aplikazio askotara bideratzea ahalbidetzen du. Modulu honi erantsita, bi nukleoko PUZ bat gehitu da, 80 MHz eta 24 MHz artean konfigura daitekeen erloju-osziladore batekin batera kontrola daitekeena. ESP-32ak ere kontsumo baxuko modu bat dauka, efizientzia distiratsu baterako modu anitzekin.

Ezaugarriak eta abantailak:

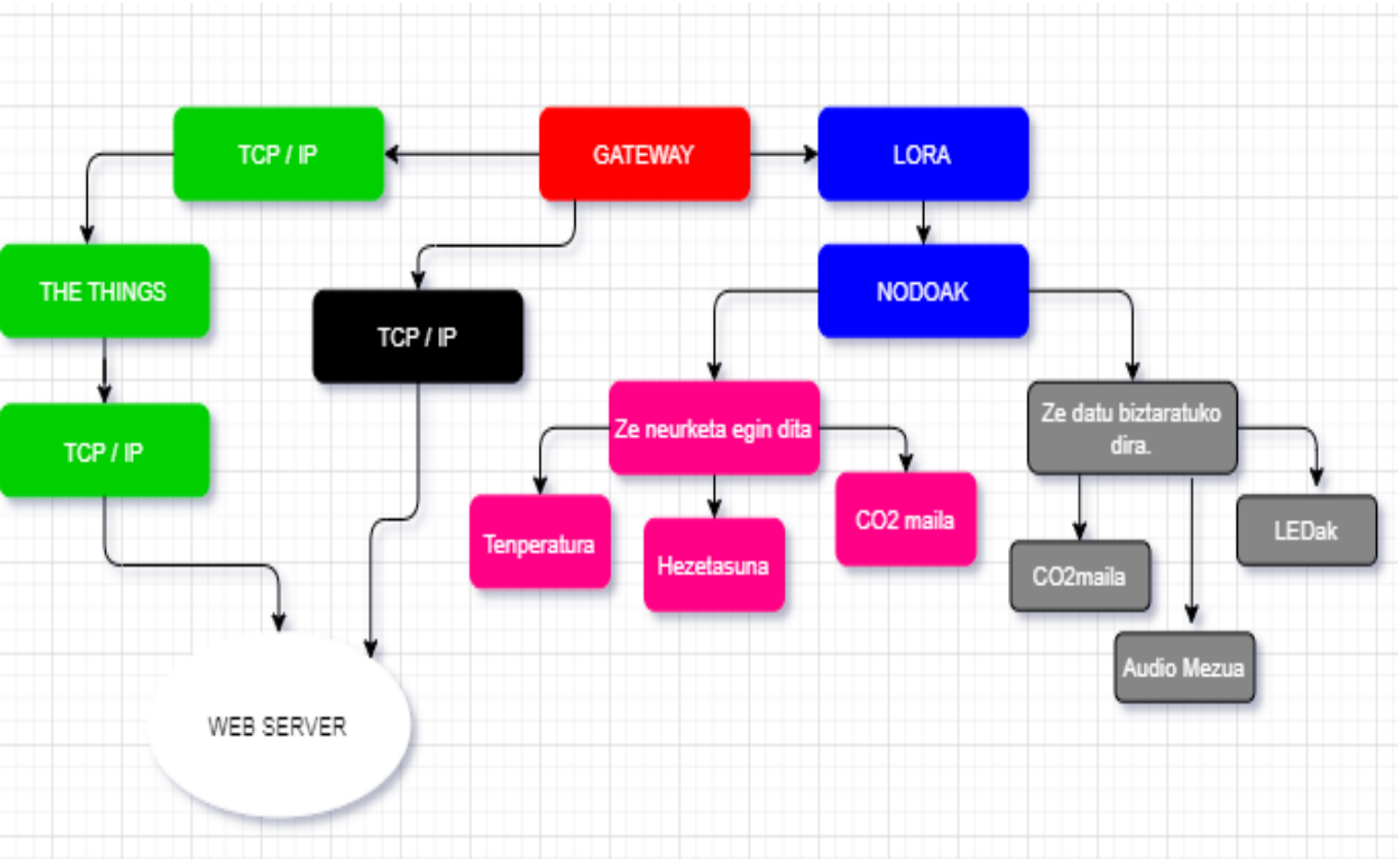
• 2,4 GHz-ko tarteak distantzia eta irismen handia ematen du.
• PUZ kontrolagarria eta erloju-osziladorea
• Funtzio anitzeko txipa; WiFi eta Bluetooth
• Bateragarria honako hauekin: IPv4, IPv6, SSL, TCP/UDP/http/FTP/MQTT
• 150 Mbps-era arteko datu-tartea du.
• WiFi segurtasuna: WPA, WPA2, WPA2-Enterprise-WPS
• Funtzionamendu-tentsioa: 2,7/3,6V kargaren arabera
• -40 ° C - +85 ° C funtzionamendu-tenperaturak
• Periferiko integratuak: ADC eta DAC funtzioak, ukimen-sentsorea, host SD/SDIO/MMC kontrolatzailea, I2C eta I2S interfazea, SPI interfazea, infragorrien kontrolatzailea eta hardware-azeleragailua

ESP32 TTGO aukeratu dugu eskuragarri izan dugulako eta ere gure proiektuan lora erabili behar dugunez gai da mezuak distantzia oso luzetara bidaltzeko.

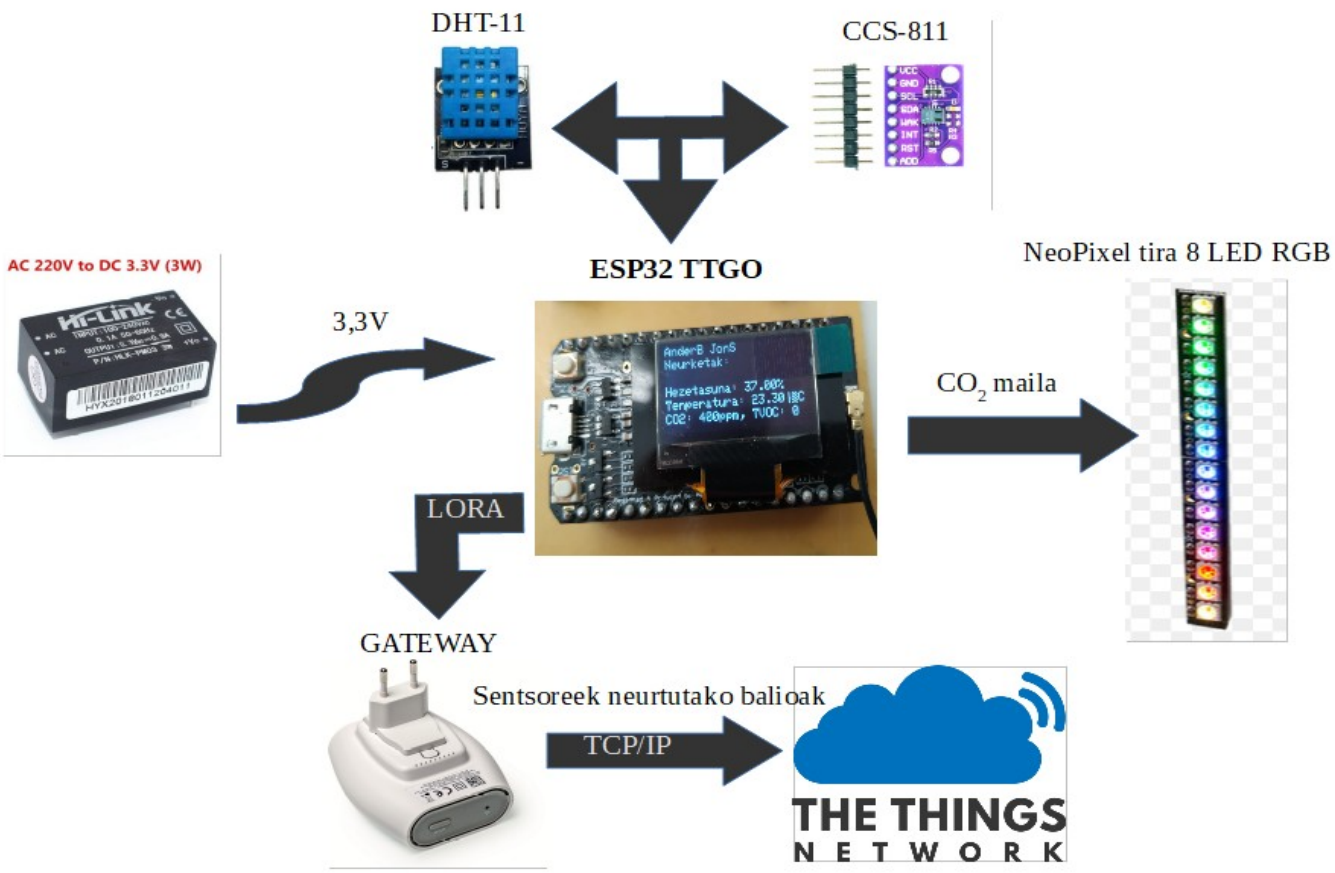


3 Bloke-diagramak

3.1 Softwarearen hasierako bloke-diagrama



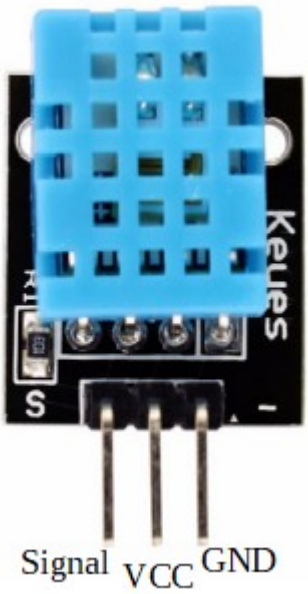
3.2 Hardwarearen hasierako bloke-diagrama



4 Osagaien analisia

4.1 DHT-11

4.1.1 Sarrera



DHT-11 erabiliko ditugun sentsoreetako bat izango da, sentsore hau tenperatura eta hezetasuna neurtzuko balio du, gure kasuan erabiliko duguna 3 pin ditu, eskumako irudian ikusten denez.

Tenperatura ºCtan neurtzen du eta hezetasuna %tan.

Kostu txikiko eta erabiltzeko erraza den sentsorea da.Hezetasun kapazitarioaren sentsorea eta termistore bat integratzen ditu inguruko airea neurtzeko, eta datuak seinale digital baten bidez erakusten ditu signal pinean (ez du irteera analogikorik).

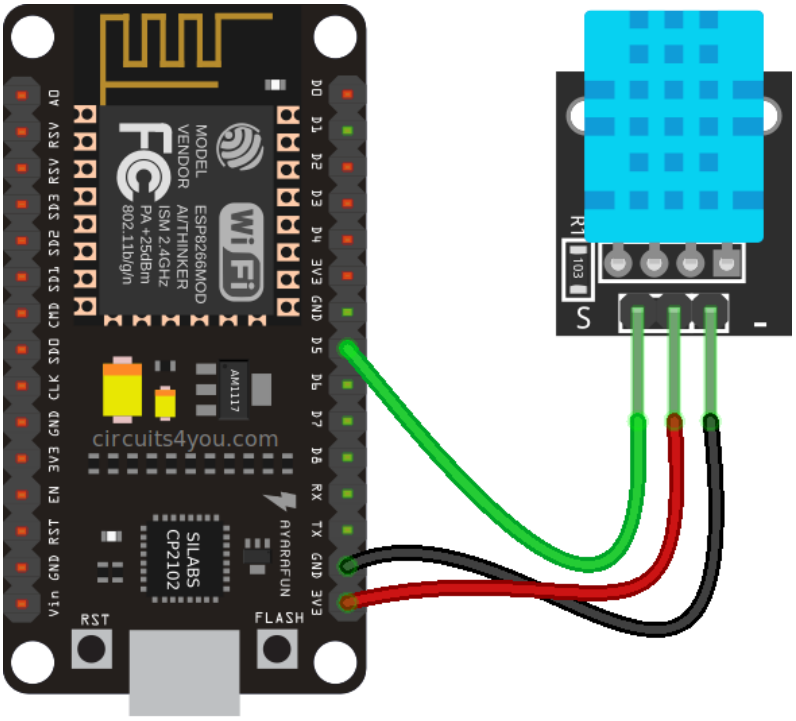
Datasheet Link

https://components101.com/sites/default/files/component_datasheet/DHT11-Temperature-Sensor.pdf

Ezaugarriak:

- Kontsumoa = 2,5mA
- Irteera seinalea = Digitala
- Tenperatura maila = 0ºC – 50ºC
- Hezetasun maila = 20 % - 90 %

4.1.2 Eskema elektronikoak



4.1.3 Sentsorearen edo shieldaren pinout-a

Berezitasunik balego adierazi hemen (pull-up erresistentziak, tentsio balioen mugak...)

4.1.4 Oinarrizko programa

```
#include "DHT.h"

// Uncomment whatever type you're using!
#define DHTTYPE DHT11 // DHT 11
//#define DHTTYPE DHT22 // DHT 22 (AM2302), AM2321
//#define DHTTYPE DHT21 // DHT 21 (AM2301)

// Connect pin 1 (on the left) of the sensor to +5V
// NOTE: If using a board with 3.3V logic like an Arduino Due connect pin 1
// to 3.3V instead of 5V!
// Connect pin 2 of the sensor to whatever your DHTPIN is
// Connect pin 4 (on the right) of the sensor to GROUND
// Connect a 10K resistor from pin 2 (data) to pin 1 (power) of the sensor

const int DHTPin = 5; // what digital pin we're connected to

DHT dht(DHTPin, DHTTYPE);

void setup() {
 Serial.begin(9600);
 Serial.println("DHTxx test!");

 dht.begin();
}

void loop() {
 // Wait a few seconds between measurements.
 delay(2000);

 // Reading temperature or humidity takes about 250 milliseconds!
 float h = dht.readHumidity();
 float t = dht.readTemperature();

 if (isnan(h) || isnan(t)) {
 Serial.println("Failed to read from DHT sensor!");
 return;
 }


 Serial.print("Humidity: ");
 Serial.print(h);
 Serial.print(" %\t");
 Serial.print("Temperature: ");
 Serial.print(t);
 Serial.println(" *C ");

}
```

4.1.5 Oinarrizko programa funtzioak bihurtuta

Nagusia:

```
/*-----( LIBURUTEGIAK )----*/
//CO2
#include "Adafruit_CCS811.h"
Adafruit_CCS811 ccs;
//DHT11
#include "DHT.h"

/*-----( KONSTANTEAK )-----*/
#define DHTTYPE DHT11 // DHT 11
const int DHTPin = 5; // what digital pin we're connected to


/*-----( OBJETUAK )-----*/

//DHT-11
DHT dht(DHTPin, DHTTYPE);

/*-----( ALDAGAIAK )----*/
//Programa osoan zehar erabiliko diren aldagai orokorrak

//Komunikazio serierako aldagaiak


/*-----( FUNTZIOAK )----*/
float dhtSentsoreaLoop(void);
void CO2SentsoreaLoop();
void C02SentsoreaSetup();
void dhtSentsoreaSetup();


void setup() {
 // put your setup code here, to run once:
Serial.begin(9600);
dhtSentsoreaSetup();
C02SentsoreaSetup();

}

void loop() {

 dhtSentsoreaLoop();

 CO2SentsoreaLoop();

}
 // put your main code here, to run repeatedly:
```

CO2:

```
/* Funtzioa: CO2 sentsorea
 * Zeregina: Funtzio honek C02 sentsoreak neurtutako balioak irakurtzeko
 * Erabilitako liburutegiak: https://github.com/adafruit/Adafruit_CCS811
 * Bueltatzen duena: C02 eta TVOC neurketa
 * Sartutako parametroak: ezer
 *  */

void C02SentsoreaSetup()
```

```
{
 Serial.println("CCS811 test");

if(!ccs.begin()){
 Serial.println("Failed to start sensor! Please check your wiring.");
 while(1);
}
while(!ccs.available());

}

void CO2SentsoreaLoop() {
 if(ccs.available()){
 if(!ccs.readData()){
 Serial.print("CO2: ");
 Serial.print(ccs.geteCO2());
 Serial.print("ppm, TVOC: ");
 Serial.println(ccs.getTVOC());
 }
 else{
 Serial.println("ERROR!");
 while(1);
 }
 }
 delay(500);
}
```

DHT-11:

```
/* Funtzioa: DHT-11 sentsorea

 * Zeregina: Funtzio honek DHT-11 sentsoreak neurtutako balioak irakurtzeko
 * Erabilitako liburutegiak: https://github.com/adafruit/DHT-sensor-library
 * Bueltatzen duena: Tenperatura eta hezetasun neurketa
 * Sartutako parametroak: ezer
 * */
 void dhtSentsoreaSetup()
 {
 Serial.println("DHTxx test!");

 dht.begin();
 }

float dhtSentsoreaLoop(void) {

 // Wait a few seconds between measurements.
 delay(2000);

 // Reading temperature or humidity takes about 250 milliseconds!
 float h = dht.readHumidity();
 float t = dht.readTemperature();

 if (isnan(h) || isnan(t)) {
 Serial.println("Failed to read from DHT sensor!");
 return;
 }
 Serial.print("Humidity: ");
 Serial.print(h);
 Serial.print(" %\t");
 Serial.print("Temperature: ");
 Serial.print(t);
 Serial.println(" *C ");
```

```
}
```

4.1.6 Material zerrenda

Nahi duzuen formatuan, sistema hau gauzatzeko material zerrenda. Gutxienez, azpiko taulan agertzen diren atalak agertu behar dira:

Izena	Kopurua	Oharrak edota ezaugarri bereziak
DHT-11	1	3 pin
ESP32 kontrolagailua	1	Nodo

4.1.7 Gorabeherak

Ez ditugu izan gorabeherarik atal honetan

4.2 CJMCU-811

4.2.1 Sarrera

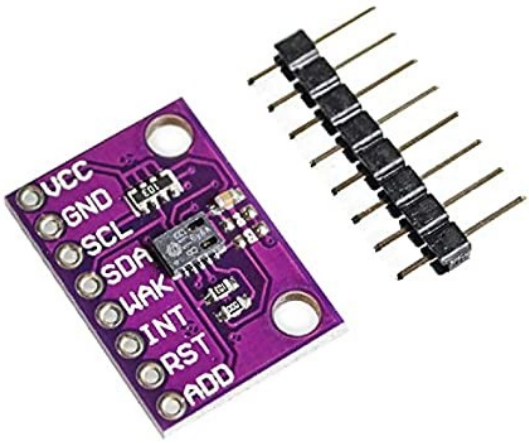
https://innovatorsguru.com/ccs811-arduino-code/

https://innovatorsguru.com/wp-content/uploads/2020/01/CCS811_Datasheet-DS000459.pdf

CCS811 airearen kalitatearen sentsorea oso potentzia txikiko gas digitalaren sentsore detektatzaile bat da, oxido metalikoko gas-sentsore bat duena konposatu organiko lurrunkorren gama zabal bat identifikatzeko, airearen kalitatea monitorizatzeko ADC bat eta I2C interfaze bat ematen dituen mikrokontrolagailu batekin.

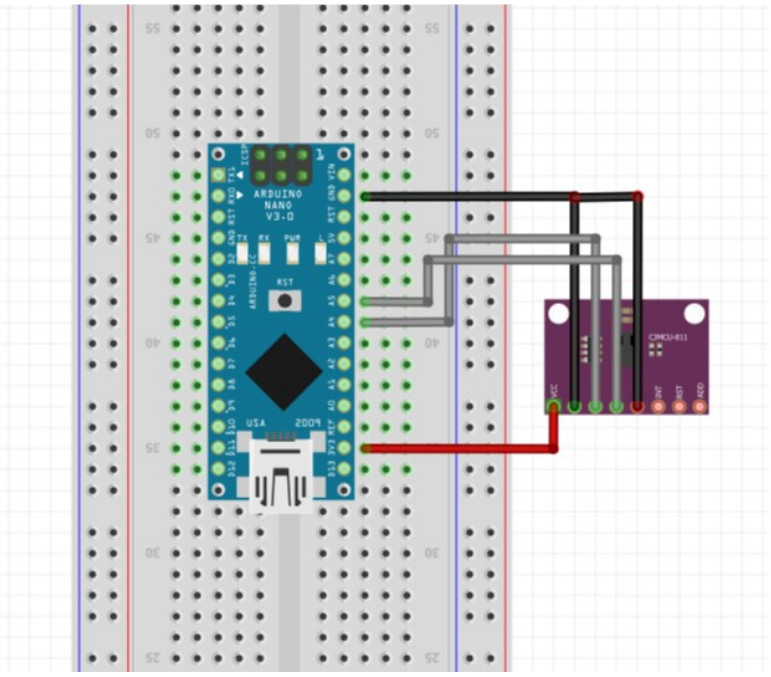
CCS811 argi-sentsorea gas digitalaren sentsorerik onena da airearen kalitatea zehazki monitorizatzeko, Arduino, STM32, Raspberry Pi eta abar erabiliz. Ccs811 gy-811 breakout board sentsorearen modulu hau tamaina txiki, gama zabal eta bereizmen handikoa da.

Sentsore hau gadget pertsonaletako (smartwatchak eta telefonoak) barneko airearen kalitatearen jarraipena egiteko diseinatuta dago. Erraz iristeko, CJMCU-811 modulua nahiago da I2C gailu estandar gisa erabili ahal izateko.

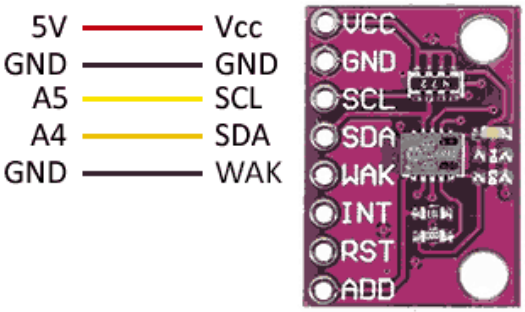


Pin Number	Pin Name	Description
1	VCC	Power supply for the module can typically 3.3V is used
2	GND	Ground of the module, connected to ground of the circuit
3	SCL	Serial Clock Line, used to provide clock pulse for I2C communication
4	SDA	Serial Data Address, used to transfer the data through I2C communication
5	WAK	Wake (active low)
6	INT	Interrupt (active low)
7	RST	Reset (active low)
8	ADD	Single address select bit to allow alternate address to be selected

4.2.2 Eskema elektronikoak



4.2.3 Sentsorearen edo shieldaren pinout-a



Specifications	Discriptions
Total Volatile Organic Compound (TVOC) sensing from	0 to 32,768 parts per billion
eCO_2 sensing from	400 parts per million to 29,206 ppm
Module size:	15mm*21mm
Working voltage	1.8V ~ 3.3V DC
Supply Current	30ma
Storage Temperature	-40 ~ 125C
	60mw

Datasheet:https://cdn-learn.adafruit.com/downloads/pdf/adafruit-ccs811-air-quality-sensor.pdf

4.2.4 Oinarrizko programa

```
#include "Adafruit_CCS811.h"

Adafruit_CCS811 ccs;

void setup() {
  Serial.begin(9600);
```

```
  Serial.println("CCS811 test");

  if(!ccs.begin()){
    Serial.println("Failed to start sensor! Please check your wiring.");
    while(1);
  }

  // Wait for the sensor to be ready
  while(!ccs.available());
}

void loop() {
  if(ccs.available()){
    if(!ccs.readData()){
      Serial.print("CO2: ");
      Serial.print(ccs.geteCO2());
      Serial.print("ppm, TVOC: ");
      Serial.println(ccs.getTVOC());
    }
    else{
      Serial.println("ERROR!");
      while(1);
    }
  }
  delay(500);
}
```

4.2.5 Oinarrizko programa funtzioak bihurtuta

Nagusia:

```
/*-----( LIBURUTEGIAK )----*/
//CO2
#include "Adafruit_CCS811.h"
Adafruit_CCS811 ccs;
//DHT11
#include "DHT.h"

/*-----( KONSTANTEAK )-----*/
#define DHTTYPE DHT11 // DHT 11
const int DHTPin = 5; // what digital pin we're connected to


/*-----( OBJETUAK )-----*/

//DHT-11
DHT dht(DHTPin, DHTTYPE);

/*-----( ALDAGAIAK )----*/
//Programa osoan zehar erabiliko diren aldagai orokorrak

//Komunikazio serierako aldagaiak


/*-----( FUNTZIOAK )----*/
float dhtSentsoreaLoop(void);
void CO2SentsoreaLoop();
void C02SentsoreaSetup();
void dhtSentsoreaSetup();
```

```
void setup() {
 // put your setup code here, to run once:
Serial.begin(9600);
dhtSentsoreaSetup();
C02SentsoreaSetup();

}

void loop() {

 dhtSentsoreaLoop();

 CO2SentsoreaLoop();

}
 // put your main code here, to run repeatedly:
```

CO2:

```
/* Funtzioa: CO2 sentsorea
 * Zeregina: Funtzio honek C02 sentsoreak neurtutako balioak irakurtzeko
 * Erabilitako liburutegiak: https://github.com/adafruit/Adafruit_CCS811
 * Bueltatzen duena: C02 eta TVOC neurketa
 * Sartutako parametroak: ezer
 *  */

void C02SentsoreaSetup()
{
 Serial.println("CCS811 test");

if(!ccs.begin()){
 Serial.println("Failed to start sensor! Please check your wiring.");
 while(1);
}
while(!ccs.available());

}

void CO2SentsoreaLoop() {
 if(ccs.available()){
 if(!ccs.readData()){
 Serial.print("CO2: ");
 Serial.print(ccs.geteCO2());
 Serial.print("ppm, TVOC: ");
 Serial.println(ccs.getTVOC());
 }
 else{
 Serial.println("ERROR!");
 while(1);
 }
 }
 delay(500);
}
```

DHT-11:

```
/* Funtzioa: DHT-11 sentsorea

 * Zeregina: Funtzio honek DHT-11 sentsoreak neurtutako balioak irakurtzeko
 * Erabilitako liburutegiak: https://github.com/adafruit/DHT-sensor-library
 * Bueltatzen duena: Tenperatura eta hezetasun neurketa
```

```
 * Sartutako parametroak: ezer
 * */
 void dhtSentsoreaSetup()
 {
 Serial.println("DHTxx test!");

 dht.begin();
 }

float dhtSentsoreaLoop(void) {

 // Wait a few seconds between measurements.
 delay(2000);

 // Reading temperature or humidity takes about 250 milliseconds!
 float h = dht.readHumidity();
 float t = dht.readTemperature();

 if (isnan(h) || isnan(t)) {
 Serial.println("Failed to read from DHT sensor!");
 return;
 }
 Serial.print("Humidity: ");
 Serial.print(h);
 Serial.print(" %\t");
 Serial.print("Temperature: ");
 Serial.print(t);
 Serial.println(" *C ");

}
```

4.2.6 Material zerrenda

Nahi duzuen formatuan, sistema hau gauzatzeko material zerrenda. Gutxienez, azpiko taulan agertzen diren atalak agertu behar dira:

Izena	Kopurua	Oharrak edota ezaugarri bereziak
4.3 CJMCU-811 edo CCS-811	1	3,3V-ra konektatu behar da
Arduino nano	1	

4.3.1 Gorabeherak

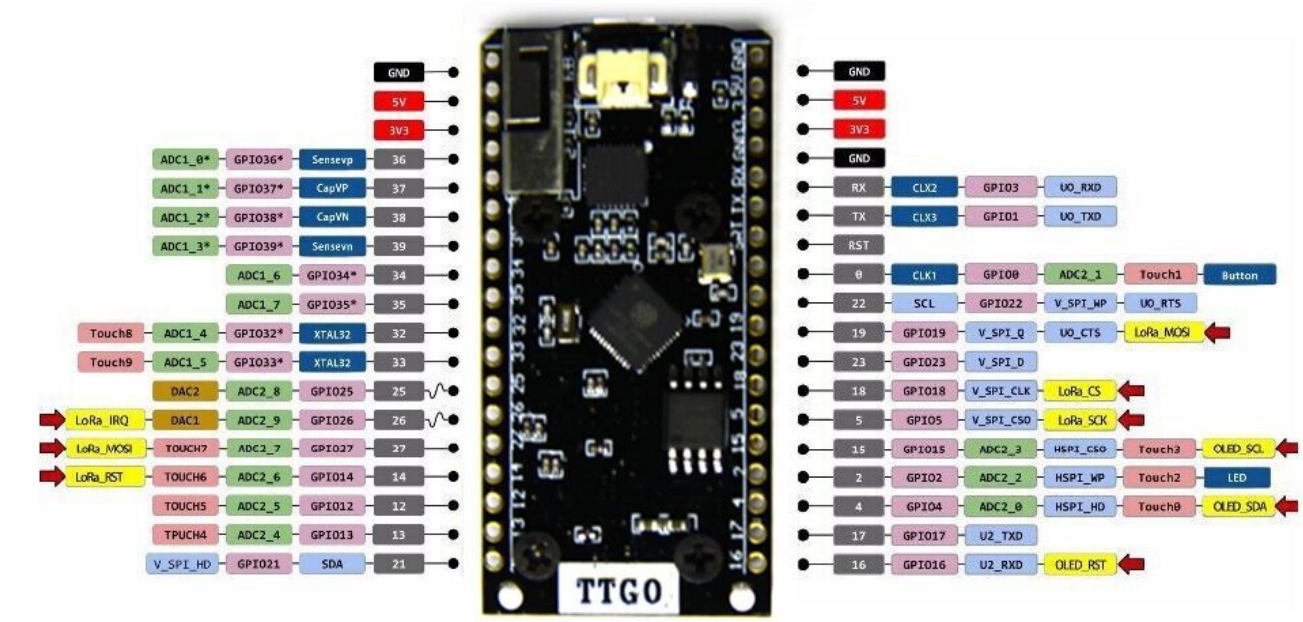
Hasieran errore bat ematen zigun kableatua txarto zegoelako. Errore hori konpontzeko SDA eta SCL haien artean aldatu dugu kablea eta funtzionatu digu.

4.4 TTGO Lora ESP32

4.4.1 Sarrera

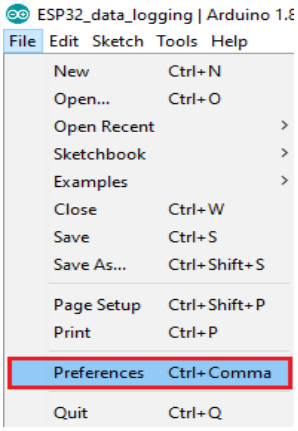
ESP32 OLEDek ESP32 modulua eta plaka berean OLED pantaila konbinatzen ditu, wifi eta bluetooth 4.0 konektibitatea eskainiz. Gainera, mikroUSB konektorea dauka elikatzeko eta JST konektorea plaka bera elikatzeko. Arduinoko IDEtik zuzenean programatu daiteke. Plakak LoRa modulu bat ere badu, datuak distantzia handietara bi noranzkotan transmititzea ahalbidetzen duena, The Things Network sarera konektatzeko aukerarekin

4.4.2 Pinout

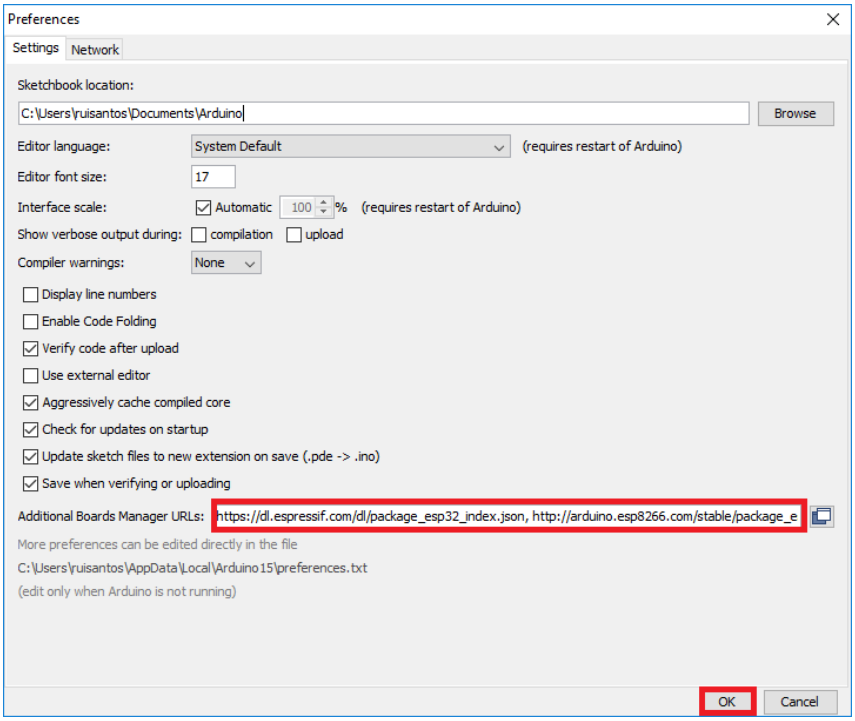


4.4.3 ESP32 plaka arduinon zelan instalatu

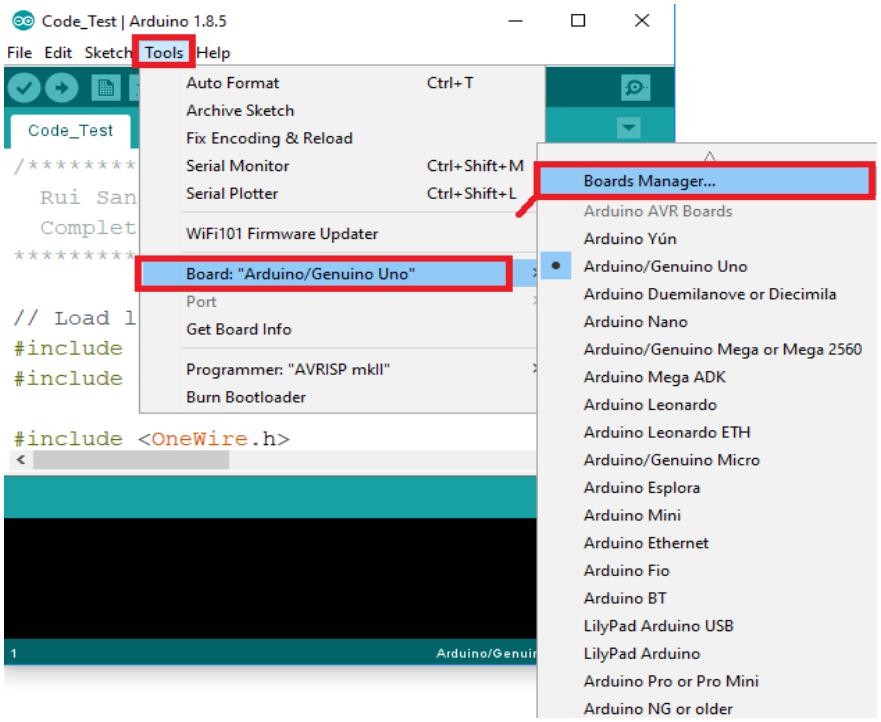
1. Arduinoko zure IDEan,joan Archivo > Preferencias



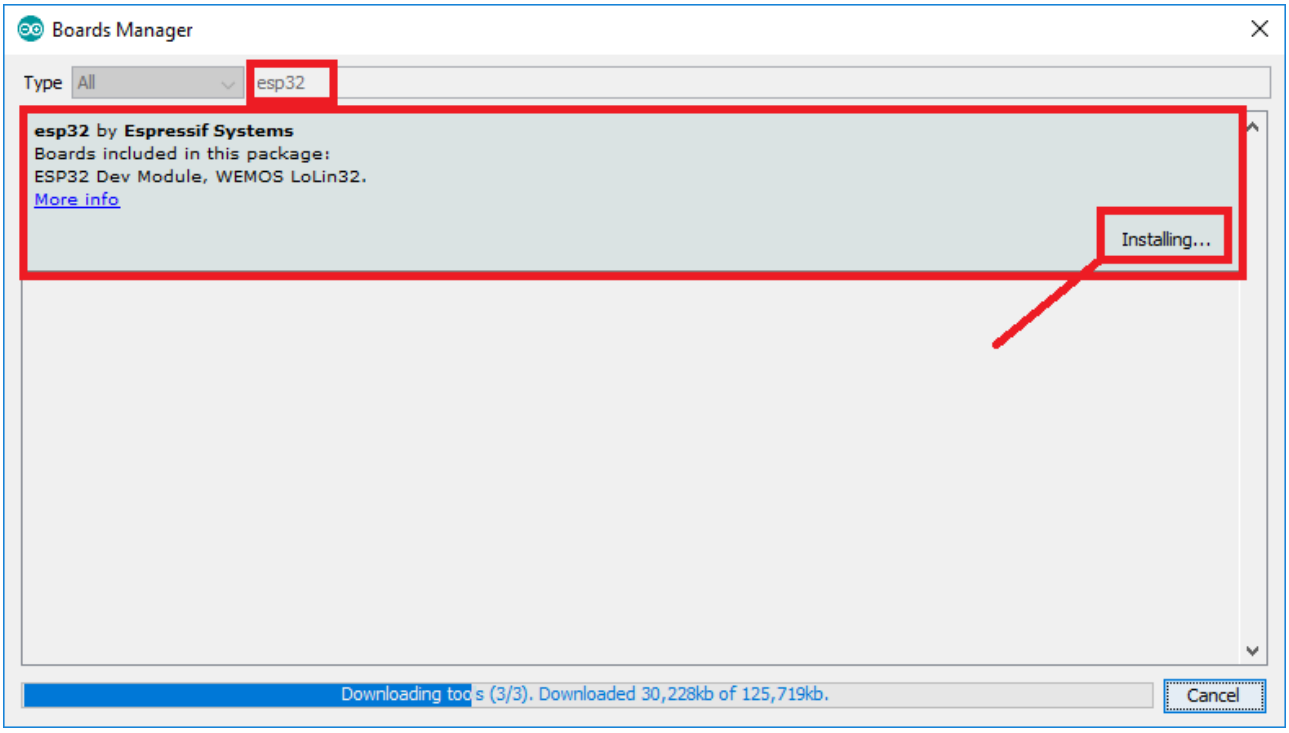
2. Sartu https://dl.espressif.com/dl/package_esp32_index.json "taula-administratzailearen URL gehigarriak" eremuan, hurrengo irudian agertzen den bezala. Gero, egin klik "onartu" botoian:



3. Ondoren “Herramientas” atalean, plaka jartzen duen eremuan, “Gestor de tarjetas”-ensartu.



4. Eta hor zaudenean ESP32 bilatu eta instalatu barik badago instalatu.10 minututan prest egongo da



5. Instalatu eta ondoren, plaka berri hori instalatuta egongo lirake.

Bertan utziko dizuet nik jarraitutako pusuak plaka hau instalatzeko :

https://randomnerdtutorials.com/installing-the-esp32-board-in-arduino-ide-windows-instructions/

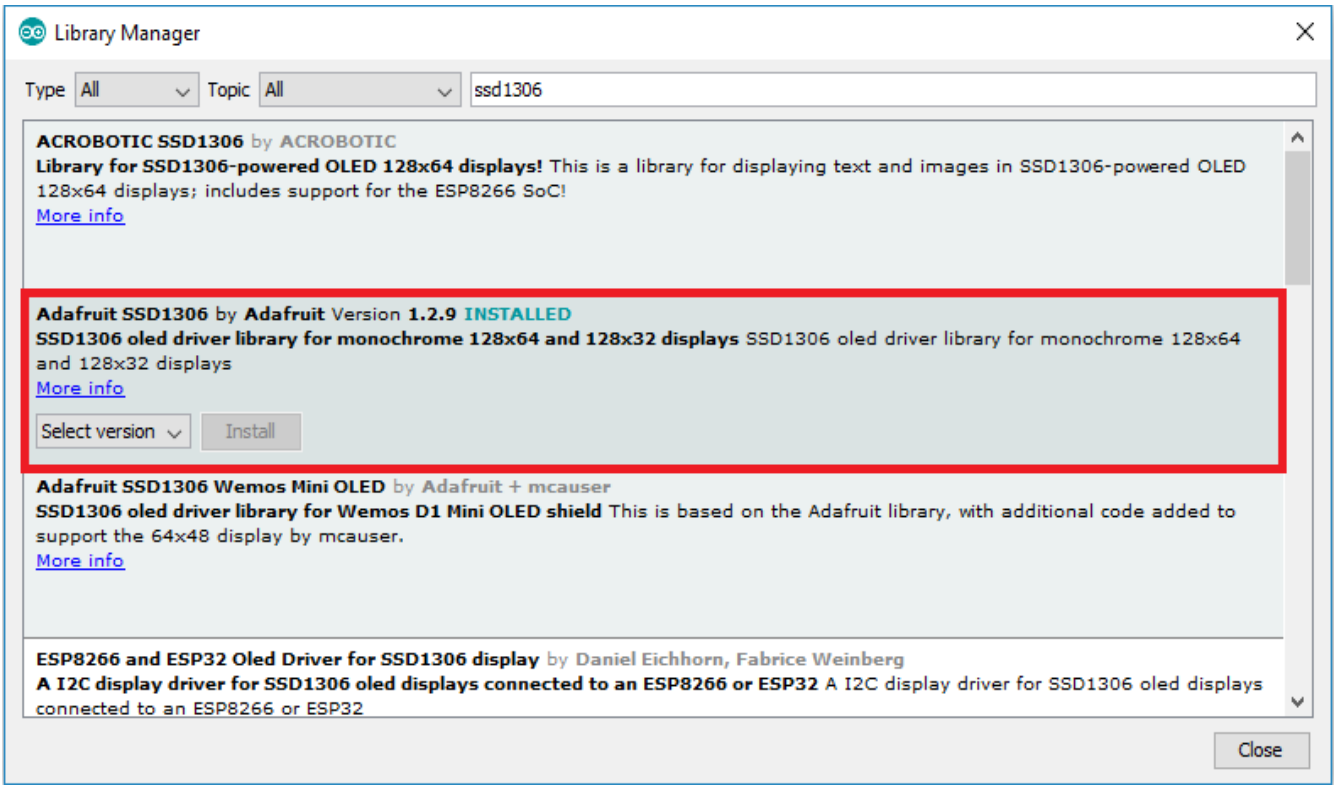
4.4.4 OLED liburutegiak instalatzea

Liburutegi batzuk daude eskuragarri OLED pantaila ESP32 sistemarekin kontrolatzeko. Tutorial honetan Adafruit-eko bi liburutegi erabiliko ditugu: Adafruit_SSD1306 liburutegia eta Adafruit_GFX liburutegia.

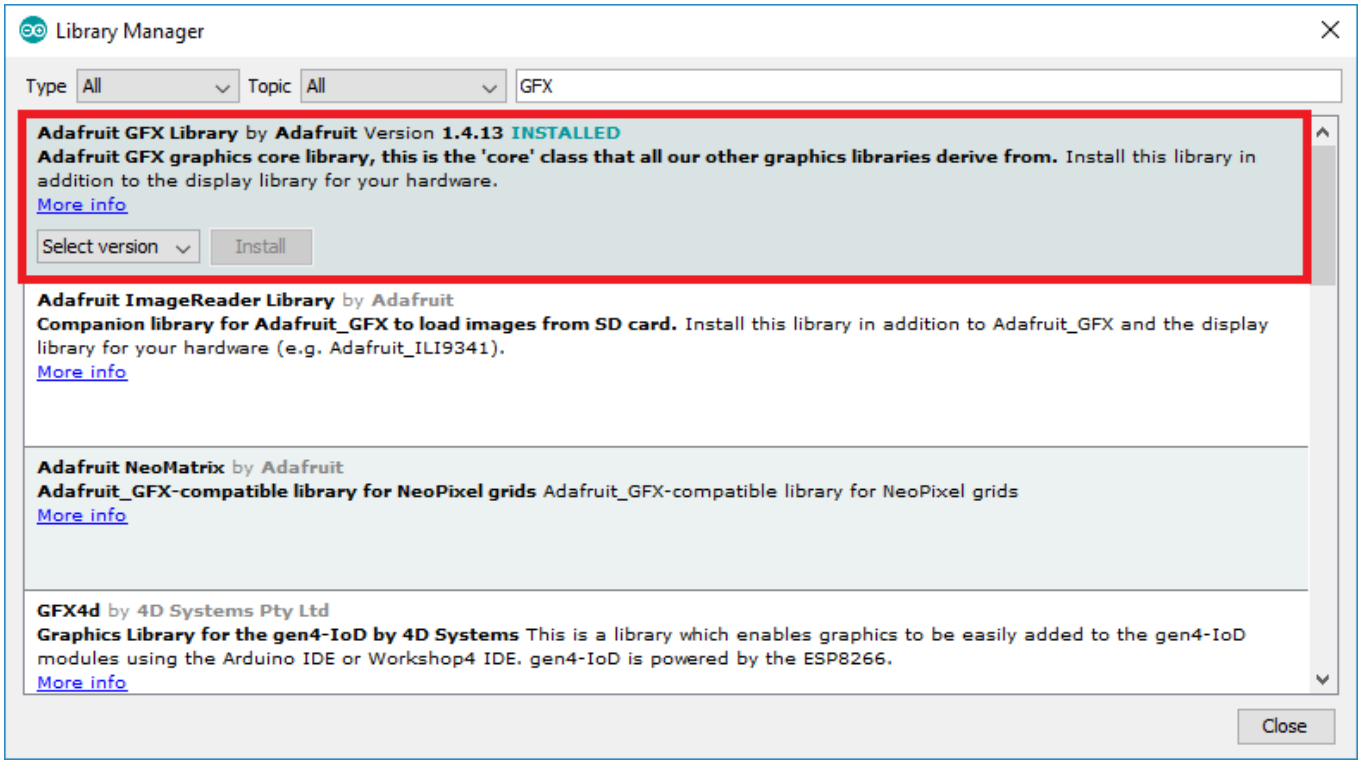
Jarraitu hurrengo urratsak liburutegi horiek instalatzeko.

1. Ireki ezazu Arduinoren IDE eta joan Sketch > Incluir biblioteca > Administrar bibliotecas. Liburutegiko administratzailea ireki beharko litzateke.

2. Idatzi "SSD1306" bilaketa-koadroan eta instalatu Adafruit-eko SSD1306 liburutegia.



3. Adafruit-eko SSD1306 liburutegia instalatu ondoren, idatzi "GFX" bilaketa-koadroan eta instalatu liburutegia.



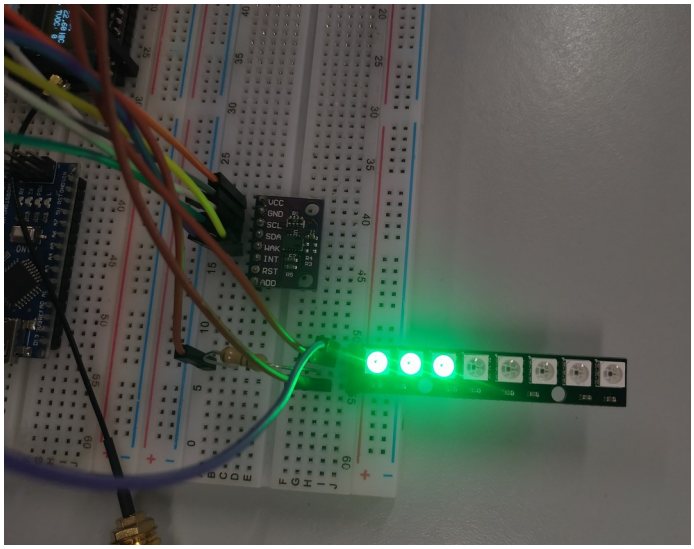
4.5 Led NeoPixel

4.5.1 Sarrera

NeoPixelak 5050 motako Led diodoak dira, WS2812 kontrolagailu integratu batekin. Hainbat neurri eta formatan daude, eta dokumentazio zabala dute, tutoretzak eta Arduinorekin erabiltzeko prest dauden liburu-dendak. Oso erraza da minutu gutxitan martxan jartzea, pin bat besterik ez baitute behar eta kolore distiratsuak eskaintzen baitituzte!

Ezaugarriak:

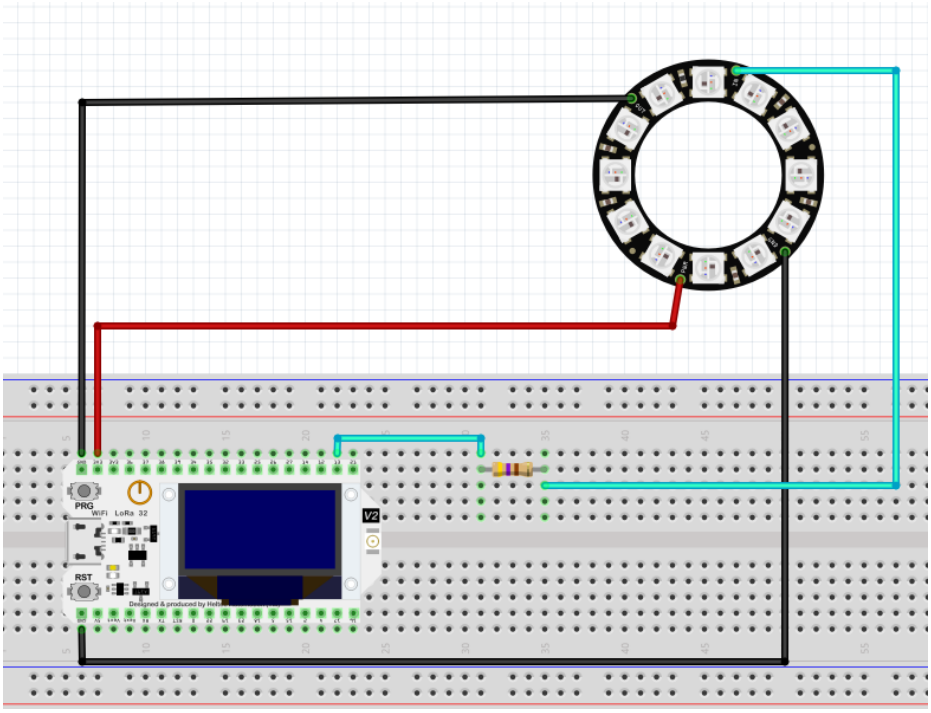
Forma: Laukizuzena
Pixel kopurua: 8
Neurriak: 51x10x3mm
Pisua: 2.57 gramo
Elikadura: 5V-3,3V
Kontsumoa: 18 mA pixel bakoitzeko



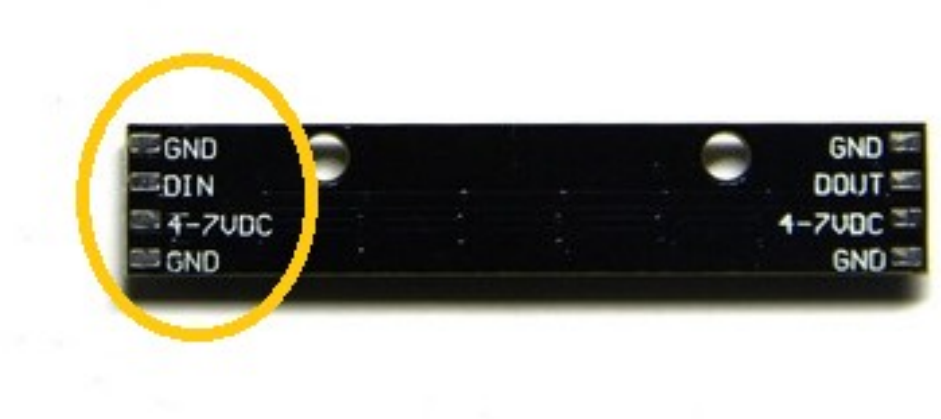
Datasheet:

https://cdn-shop.adafruit.com/datasheets/WS2812.pdf

4.5.2 Eskema elektrikoa



4.5.3 Sentsorearen edo shieldaren pinout-a



4.5.4 Oinarrizko programa

```
// NeoPixel Ring simple sketch (c) 2013 Shae Erisson
// Released under the GPLv3 license to match the rest of the
// Adafruit NeoPixel library

#include <Adafruit_NeoPixel.h>
#ifdef __AVR__
 #include <avr/power.h> // Required for 16 MHz Adafruit Trinket
#endif

// Which pin on the Arduino is connected to the NeoPixels?
#define PIN        6 // On Trinket or Gemma, suggest changing this to 1
```

```
// How many NeoPixels are attached to the Arduino?
#define NUMPIXELS 16 // Popular NeoPixel ring size

// When setting up the NeoPixel library, we tell it how many pixels,
// and which pin to use to send signals. Note that for older NeoPixel
// strips you might need to change the third parameter -- see the
// strandtest example for more information on possible values.
Adafruit_NeoPixel pixels(NUMPIXELS, PIN, NEO_GRB + NEO_KHZ800);

#define DELAYVAL 500 // Time (in milliseconds) to pause between pixels

void setup() {
  // These lines are specifically to support the Adafruit Trinket 5V 16 MHz.
  // Any other board, you can remove this part (but no harm leaving it):
#if defined(__AVR_ATtiny85__) && (F_CPU == 16000000)
  clock_prescale_set(clock_div_1);
#endif
  // END of Trinket-specific code.

  pixels.begin(); // INITIALIZE NeoPixel strip object (REQUIRED)
}

void loop() {
  pixels.clear(); // Set all pixel colors to 'off'

  // The first NeoPixel in a strand is #0, second is 1, all the way up
  // to the count of pixels minus one.
  for(int i=0; i<NUMPIXELS; i++) { // For each pixel...

    // pixels.Color() takes RGB values, from 0,0,0 up to 255,255,255
    // Here we're using a moderately bright green color:
    pixels.setPixelColor(i, pixels.Color(0, 150, 0));

    pixels.show();   // Send the updated pixel colors to the hardware.

    delay(DELAYVAL); // Pause before next pass through loop
  }
 }
```

4.5.5 Material zerrenda

Izena	Kopurua	Oharrak edota ezaugarri bereziak
Led Neopixel	1	
TTGO Lora	1	

4.5.6 Gorabeherak

Ez dugu izan.

4.5.7 Estekak

https://learn.adafruit.com/adafruit-neopixel-uberguide/

https://tienda.bricogeek.com/led-neopixel/660-barra-neopixel-8-x-ws2812.html?gclid=Cj0KCQiA4feBBhC9ARIsABp_nbVA6xkaQWV2733y1gMET1h4rGiRIGw8Mn2Gw73FIECJkmrDtVslwYkaAuBmEALw_wcB

4.6 Elikatze Iturria (HLK-PM03 AC DC)

4.6.1 Sarrera

HLK-PM03 AC DC bihurgailua elikadura-iturri, botere-iturri edo iturri kommutatu deitzen den gailu elektroniko bat da. Elektronikan, korronte alternoa irteera batean edo gehiagotan korronte zuzen bihurtzen duen tresna da.

HLK-PM03 AC DC bihurgailua espazio trinkoetan inplementatu daiteke, edo proiektuak ahalik eta txikiena izan behar badu, iturri hori adierazitakoa da.

Energia elektrikoa VCA izatetik VCD izatera aldatzean, hainbat gailu elektroniko elikatu ditzake, hala nola CD motorra, led-ak, txartelak edo Arduino moduluak, wifi modulua, sentsoreak, eragingailuak, anplifikadoreak, zirkuitu integratuak, etab.

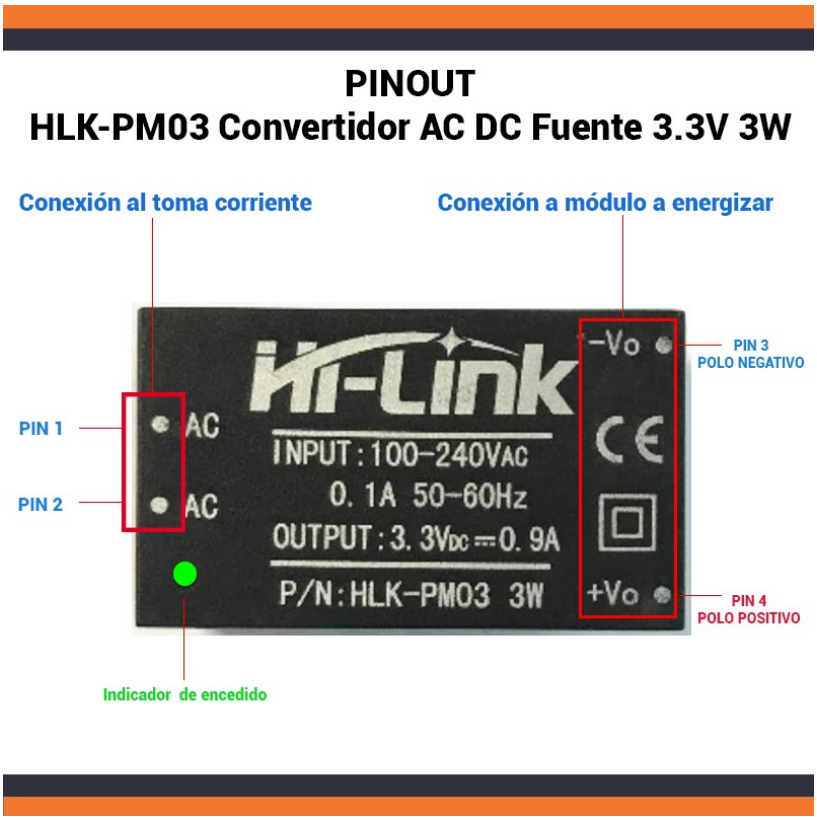
Ezaugarriak:

Tipo: **HLK-PM03** Convertidor **AC DC**.

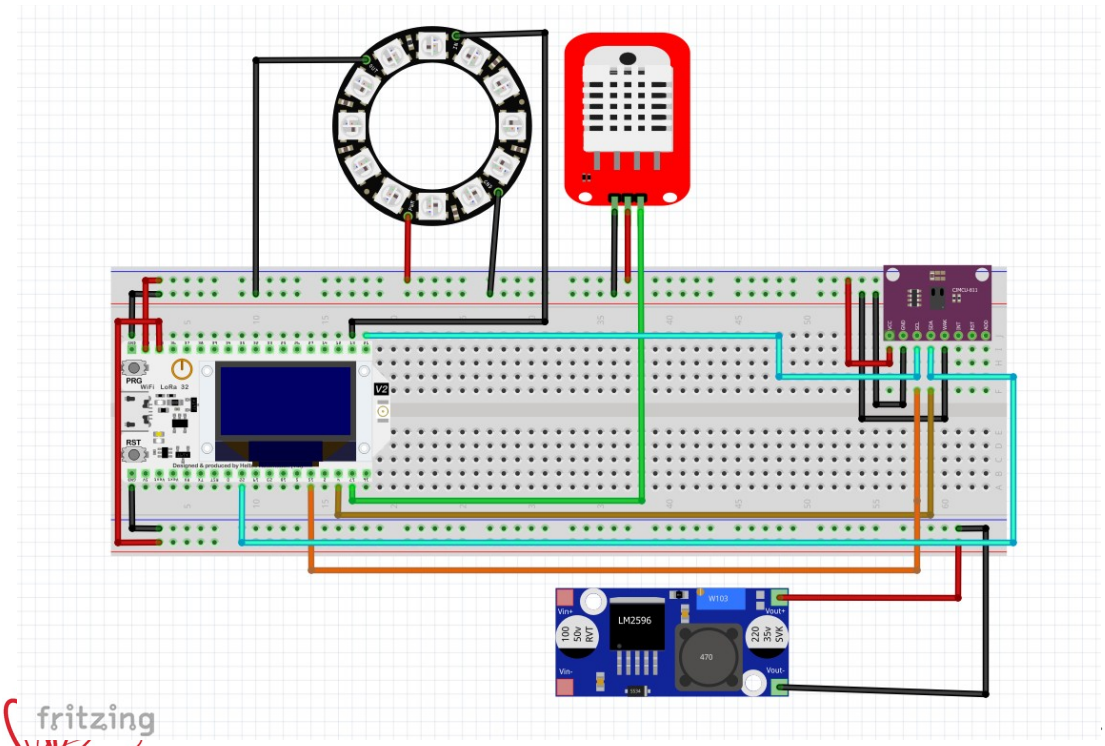
- Voltaje de entrada: AC100-240V 50/60Hz.
- Voltaje de salida: 3.3 VCD a 909mA.
- Potencia Máxima de salida: 3W.
- Eficiencia en voltaje de salida: 70%

- Bajo rizado y bajo nivel de ruido.
- Salida de protección contra sobrecarga y cortocircuito.
- Pines: 4.

4.6.2 PinOut

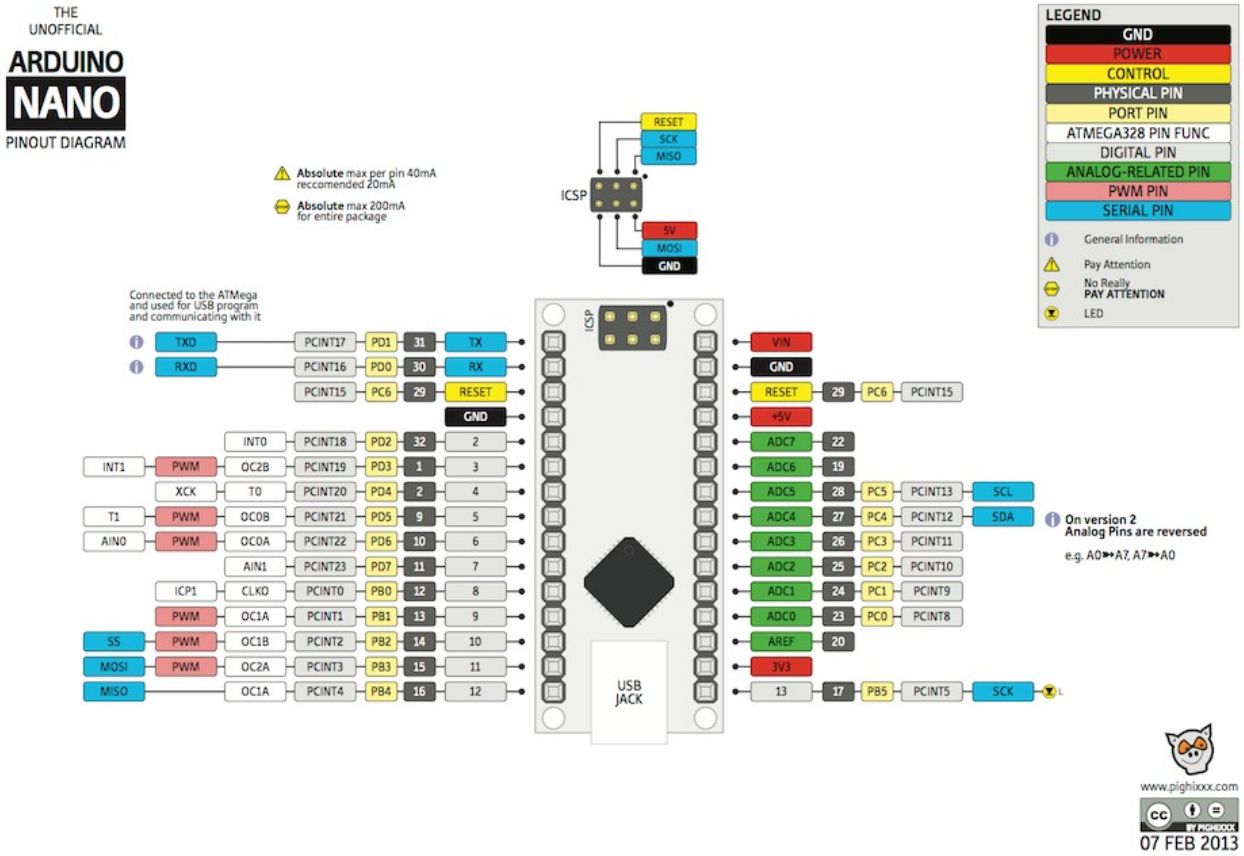


4.6.3 Eskema elektrikoa



5 ESP32 TTGO pinen esleipena

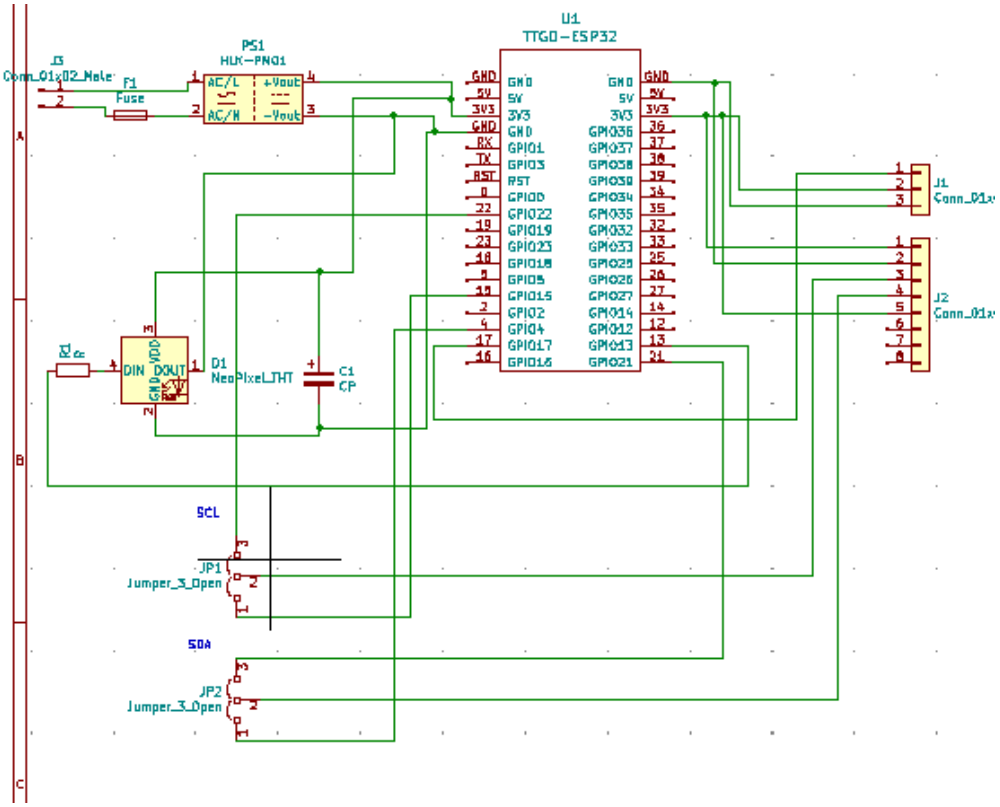
PLAKAKO ELIKADURA					KANPOKO ELIKADURA		ERABILERA
VIN	5V	3V3	GND	RESET	V	mA	
					3,3	20	TTGO Lora
		X				2,5	DHT11 (Tenperatura eta Hezetasuna)
		X				10	CJMCU-811(CO2_Sentsorea)
		X				480	LED NEOPIXEL
					220 V	900	HLK-PM03 AC-DC 220 V a 3,3V



DIGITALA	KOMUNIKAZIOAK	GAINERAKOAK	ERABILERA
GPI0/GPO0			
GPI1/GPO1	UART TxD		Ordenagailuarekin komunikatzeko (Serial) Datuak jaso
GPI2/GPO2			
GPI3/GPO3	UART RxD		Ordenagailuarekin komunikatzeko (Serial) Datuak bidali
GPI4/GPO4	SDA		OLED_SDA--CJMCU-811(CO2_Sentsorea)
GPI5/GPO5	SPI CSO		LORA_SCK
GPI12/GPO12			
GPI13/GPO13			LED NEOPIXEL
GPI14/GPO14			LORA_RST
GPI15/GPO15	SCL		OLED_SCL--CJMCU-811(CO2_Sentsorea)
GPI16/GPO16	U2 RXD		OLED_RST
GPI17/GPO17	U2 TXD		DHT11 (Tenperatura eta Hezetasuna)
GPI18/GPO18	SPI CLK		LORA_CS
GPI19/GPO19			LORA_MISO
GPI21/GPO21	SDA(jumper)		CJMCU-811(CO2_Sentsorea)
GPI22/GPO22	SCL(jumper)		CJMCU-811(CO2_Sentsorea)
GPI23/GPO23			
GPI24/GPO24			
GPI25/GPO25			
GPI26/GPO26			LORA_IRQ
GPI27/GPO27			LORA_MOSI
GPI32/GPO32			
GPI3/GPO33			
GPI34/GPO34			
GPI35/GPO35			

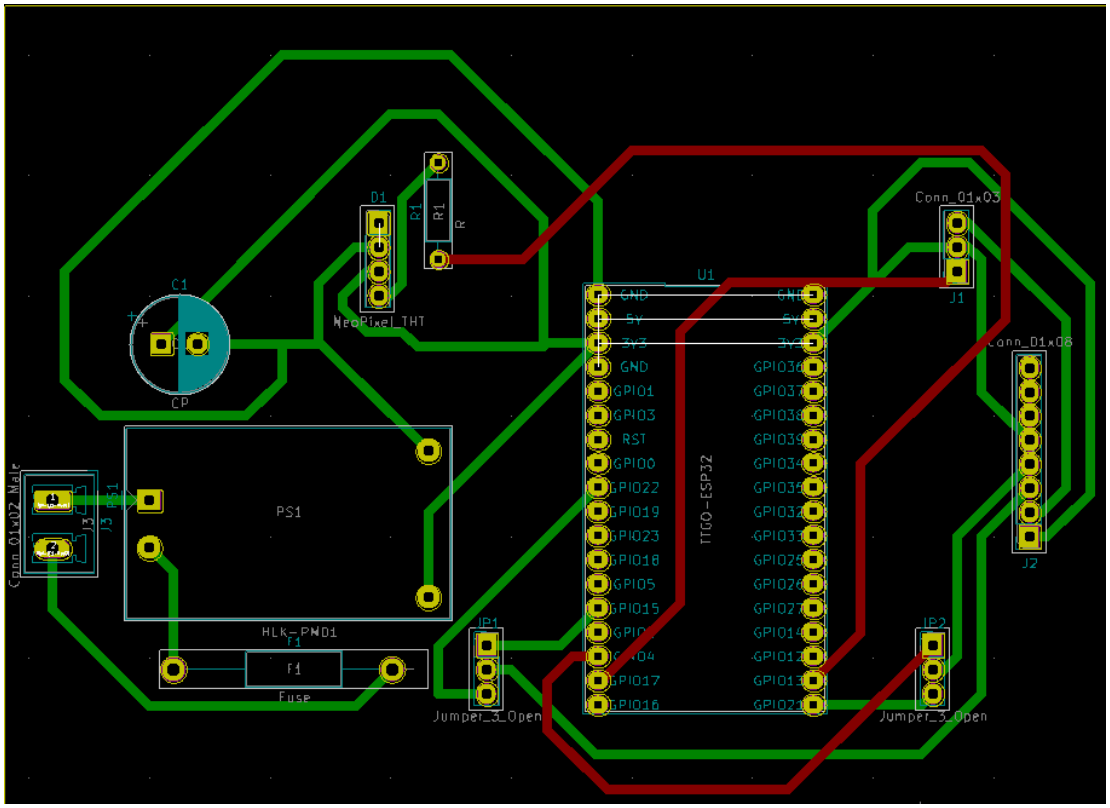
6 Eskema elektrikoak

Kicad eskema



PCB eskema

(Zubiak eginez akatsak zuzendu ditugu)

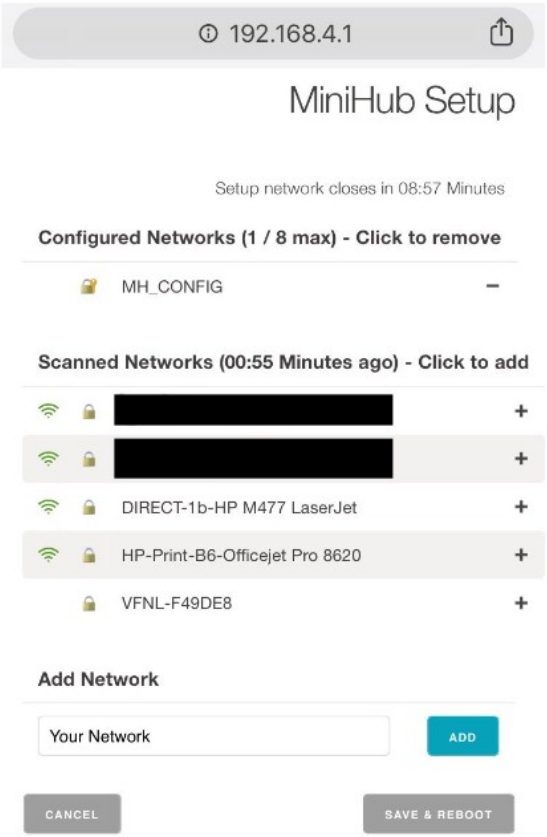


7 GateWay konfigurazioa

GateWay bat LoRa transmisio modulua duen gailua da eta informazioa Internet eta berarekin komunikatzen diren nodoen artean birbidaltzen du eta alderantziz.

GateWay-a konfiguratzeko tutorial edo manual bat aurkitu dugu interneten.
Hainbat pasu segi ditugu, honako hauek:

- Lehenengo pausua, gure GateWay-a elikatu ondoren Reset botoia 5 segunduz pultsatzea da, goiko aldean duen LEDa azkar parpadeatzen(berde-gorri) jarri arte
- Ondoren Setup botoia 10 segunduz pultsauta mantendu LEDa gorriz parpadeatzen jarri harte.
- Hau lortu eta gero GateWay-a orain WiFi AP bat erakusten du, SSID MINIHUB-xxxxxx dena, non xxxxxx GateWay-aren IDA da, 6 digito.
- GateWay-aren pasahitza gailuaren atzeko aldean ikus dezakegu.
- Azkenik, 192.168.4.1 sarrera sartu eta hor Wifia aukeratu dugu



- Konfigurazioa zuzena bada, Gateway-a berdez kliskatuko du segundo batzuetan, sare horretara konektatzen den bitartean.
- Konfigurazioa zuzena bada, pasabideak BERDE < ->GORRIAN kliskatuko du segundo batzuetan, CUPS azken puntura konektatzen den bitartean, eta LNS trafikoko azken puntura konektatzeko beharrezkoa den informazioa lortzen du.

- Konfigurazioa zuzena izan bada, Led berde finkoan egongo da, eta horrek esan nahi du GateWay-a konektatuta dagoela.

8 Azken programa osoaren atal desberdinak azalpenekin

PROGRAMA NAGUSIA:

```
#include "Nagusia.h"

void setup() {
  // put your setup code here, to run once:
Serial.begin(115200);
dhtSentsoreaSetup();
OledSetup();
C02SentsoreaSetup();
TTNsetup();
NeoPixelsetup();
}

void loop() {

  dhtSentsoreaLoop();

  OledLoop();

  CO2SentsoreaLoop();

  TTNloop();

  NeoPixelLoop();

}
  // put your main code here, to run repeatedly:
```

BIDALI TTN:

```
void BidalketaLoop (void){

  float h = dht.readHumidity();
  float t = dht.readTemperature();

unsigned long balioa =((t*10000000)+(h*10000)+(ccs.geteCO2()));
//bidaliko dugun zenbakizko txurroaren simulazioa, balioa adibide bat da TTTHHCCCC
String payload = String(balioa);                                  //txurroa string
batera pasatu
static uint8_t payloadChar[10];                                   //karaktereen arraya
sortu
payload.toCharArray((char *)payloadChar, sizeof(payloadChar));    //stringa char array
bihurtu


// Prepare upstream data transmission at the next possible time. Bidalketa egiteko
lekua, payloadChar aldagaia bidaltzen du
LMIC_setTxData2(1, payloadChar, sizeof(payloadChar)-1, 0);
Serial.println(F("Packet queued"));
}
```

CO2:

```
/* Funtzioa: CO2 sentsorea
 * Zeregina: Funtzio honek C02 sentsoreak neurtutako balioak irakurtzeko
 * Erabilitako liburutegiak: https://github.com/adafruit/Adafruit_CCS811
 * Bueltatzen duena: C02 eta TVOC neurketa
 * Sartutako parametroak: ezer
 *  */

void C02SentsoreaSetup()
{
   Serial.println("CCS811 test");

  if(!ccs.begin()){
    Serial.println("Failed to start sensor! Please check your wiring.");
    while(1);
}
while(!ccs.available());

}

void CO2SentsoreaLoop() {
  if(ccs.available()){
    if(!ccs.readData()){
      Serial.print("CO2: ");
      Serial.print(ccs.geteCO2());
      Serial.print("ppm, TVOC: ");
      Serial.println(ccs.getTVOC());
    }
    else{
      Serial.println("ERROR!");
      while(1);
    }
  }
  delay(500);
}
```

DHT11:

```
/* Funtzioa: DHT-11 sentsorea
 * Zeregina: Funtzio honek DHT-11 sentsoreak neurtutako balioak irakurtzeko
 * Erabilitako liburutegiak: https://github.com/adafruit/DHT-sensor-library
 * Bueltatzen duena: Tenperatura eta hezetasun neurketa
 * Sartutako parametroak: ezer
 *  */
 void dhtSentsoreaSetup()
 {
  Serial.println("DHTxx test!");

  dht.begin();
 }

void dhtSentsoreaLoop(void) {

   // Wait a few seconds between measurements.
   delay(2000);

   // Reading temperature or humidity takes about 250 milliseconds!
   float h = dht.readHumidity();
   float t = dht.readTemperature();

   if (isnan(h) || isnan(t)) {
      Serial.println("Failed to read from DHT sensor!");
```

```
      return;
   }
   Serial.print("Humidity: ");
   Serial.print(h);
   Serial.print(" %\t");
   Serial.print("Temperature: ");
   Serial.print(t);
   Serial.println(" *C ");

}
```

LIBURUTEGIAK, KONSTANTEAK, OBJETUAK, FUNTZIOAK, ALDAGAIAK

Nagusia.h

```
/*-----( LIBURUTEGIAK )----*/
//CO2Sentsorea
#include "Adafruit_CCS811.h"
Adafruit_CCS811 ccs;

//DHT11
#include "DHT.h"

//TTN
#include <lmic.h>
#include <hal/hal.h>
#include <SPI.h>

//Libraries for LoRa
#include <SPI.h>
#include <LoRa.h>

//Libraries for OLED Display
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

//NeoPixel
#include <Adafruit_NeoPixel.h>
#ifdef __AVR__
 #include <avr/power.h> // Required for 16 MHz Adafruit Trinket
#endif
#define NUMPIXELS 8 // Popular NeoPixel ring size
#define PIN        13 // On Trinket or Gemma, suggest changing this to 1
Adafruit_NeoPixel pixels(NUMPIXELS, PIN, NEO_GRB + NEO_KHZ800);

/*-----( KONSTANTEAK )-----*/

//DHT-11
#define DHTTYPE DHT11   // DHT 11
const int DHTPin = 17;     // what digital pin we're connected to


//TTN
static const PROGMEM u1_t NWKSKEY[16] = {
0xCF,0x18,0xAA,0xE4,0xCA,0x60,0xFA,0x73,0x36,0x54,0x3B,0xD3,0x32,0x23,0x91,0x5A }
;//Network session key
static const u1_t PROGMEM APPSKEY[16] = {
0xD4,0x58,0x30,0x4E,0x51,0xD5,0x29,0x58,0x8B,0xD5,0x82,0x76,0xB2,0x80,0x0B,0xD7 };//APP
session key
static const u4_t DEVADDR = 0x26013458 ; // <-- Change this address for every node!
DEVICE ADDRESS
// These callbacks are only used in over-the-air activation, so they are
```

```
// left empty here (we cannot leave them out completely unless
// DISABLE_JOIN is set in config.h, otherwise the linker will complain).
void os_getArtEui (u1_t* buf) { }
void os_getDevEui (u1_t* buf) { }
void os_getDevKey (u1_t* buf) { }


// Schedule TX every this many seconds (might become longer due to duty
// cycle limitations).
const unsigned TX_INTERVAL = 60;

// Pin mapping
const lmic_pinmap lmic_pins = {
    .nss = 18,
    .rxtx = LMIC_UNUSED_PIN,
    .rst = 14,
    .dio = {26, 33, 32},
};

//OLED

//define the pins used by the LoRa transceiver module
#define SCK 5
#define MISO 19
#define MOSI 27
#define SS 18
#define RST 14
#define DIO0 26

//433E6 for Asia
//866E6 for Europe
//915E6 for North America
#define BAND 866E6

//OLED pins
#define OLED_SDA 4
#define OLED_SCL 15
#define OLED_RST 16
#define SCREEN_WIDTH 128 // OLED display width, in pixels
#define SCREEN_HEIGHT 64 // OLED display height, in pixels

int counter = 0;

Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RST);

/*-----( OBJETUAK )-----*/

//DHT-11
DHT dht(DHTPin, DHTTYPE);

/*-----( ALDAGAIAK )----*/
//Programa osoan zehar erabiliko diren aldagai orokorrak

static uint8_t mydata[] = "Hello, world!";
static osjob_t sendjob;

//Komunikazio serierako aldagaiak

/*-----( FUNTZIOAK )----*/
void dhtSentsoreaLoop(void);
void CO2SentsoreaLoop();
void C02SentsoreaSetup();
void dhtSentsoreaSetup();
void TTNsetup();
```

```
void TTNloop();
void OledSetup();
void OledLoop();
void NeoPixelsetup();
void NeoPixelLoop();
void BidalketaLoop ();
```

NEOPIXEL:

```
// NeoPixel Ring simple sketch (c) 2013 Shae Erisson
// Released under the GPLv3 license to match the rest of the
// Adafruit NeoPixel library



// When setting up the NeoPixel library, we tell it how many pixels,
// and which pin to use to send signals. Note that for older NeoPixel
// strips you might need to change the third parameter -- see the
// strandtest example for more information on possible value

void NeoPixelsetup() {
  // These lines are specifically to support the Adafruit Trinket 5V 16 MHz.
  // Any other board, you can remove this part (but no harm leaving it):
#if defined(__AVR_ATtiny85__) && (F_CPU == 16000000)
  clock_prescale_set(clock_div_1);
#endif
  // END of Trinket-specific code.

  pixels.begin(); // INITIALIZE NeoPixel strip object (REQUIRED)
}

void NeoPixelLoop() {
  pixels.clear(); // Set all pixel colors to 'off'

  // The first NeoPixel in a strand is #0, second is 1, all the way up
  // to the count of pixels minus one.
  //for(int i=0; i<NUMPIXELS; i++) { // For each pixel...

    // pixels.Color() takes RGB values, from 0,0,0 up to 255,255,255
    // Here we're using a moderately bright green color:
    if(ccs.geteCO2()>= 400 & ccs.geteCO2()< 500)
    {
    pixels.setPixelColor(0, pixels.Color(0, 50, 0));
    pixels.setPixelColor(1, pixels.Color(0, 50, 0));
    pixels.setPixelColor(2, pixels.Color(0, 50, 0));
    }
    else
    {
      if(ccs.geteCO2()>= 500 & ccs.geteCO2()< 700)
      {
    pixels.setPixelColor(0, pixels.Color(0, 50, 0));
    pixels.setPixelColor(1, pixels.Color(0, 50, 0));
    pixels.setPixelColor(2, pixels.Color(0, 50, 0));
    pixels.setPixelColor(3, pixels.Color(50, 50, 0));
    pixels.setPixelColor(4, pixels.Color(50, 50, 0));

      }
      else{
        if(ccs.geteCO2()>= 700 & ccs.geteCO2()< 1000)
        {
    pixels.setPixelColor(0, pixels.Color(0, 50, 0));
```

```
    pixels.setPixelColor(1, pixels.Color(0, 50, 0));
    pixels.setPixelColor(2, pixels.Color(0, 50, 0));
    pixels.setPixelColor(3, pixels.Color(50, 50, 0));
    pixels.setPixelColor(4, pixels.Color(50, 50, 0));
    pixels.setPixelColor(5, pixels.Color(100, 25, 0));
    pixels.setPixelColor(6, pixels.Color(100, 25, 0));
        }
    else
    {
    pixels.setPixelColor(0, pixels.Color(0, 50, 0));
    pixels.setPixelColor(1, pixels.Color(0, 50, 0));
    pixels.setPixelColor(2, pixels.Color(0, 50, 0));
    pixels.setPixelColor(3, pixels.Color(50, 50, 0));
    pixels.setPixelColor(4, pixels.Color(50, 50, 0));
    pixels.setPixelColor(5, pixels.Color(100, 25, 0));
    pixels.setPixelColor(6, pixels.Color(100, 25
    , 0));
    pixels.setPixelColor(7, pixels.Color(50, 0, 0));



      }
    }
    }

    pixels.show();   // Send the updated pixel colors to the hardware.

    // Pause before next pass through loop
  }
```

OLED:

```
/* Funtzioa: Oled
 * Zeregina: Oled pantailan balioak bistaraztea
 * Erabilitako liburutegiak:
 * Bueltatzen duena:
 * Sartutako parametroak: ezer
 *  */
void OledSetup() {

  //reset OLED display via software
  pinMode(OLED_RST, OUTPUT);
  digitalWrite(OLED_RST, LOW);
  delay(20);
  digitalWrite(OLED_RST, HIGH);

  //initialize OLED
  Wire.begin(OLED_SDA, OLED_SCL);
  if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3c, false, false)) { // Address 0x3C for
128x32
    Serial.println(F("SSD1306 allocation failed"));
    for(;;); // Don't proceed, loop forever
  }

  display.clearDisplay();
  display.setTextColor(WHITE);
  display.setTextSize(1);
  display.setCursor(0,0);
  display.print("LORA SENDER ");
  display.display();

  //initialize Serial Monitor
```

```
  Serial.begin(115200);

  Serial.println("LoRa Sender Test");

  //SPI LoRa pins
  SPI.begin(SCK, MISO, MOSI, SS);
  //setup LoRa transceiver module
  LoRa.setPins(SS, RST, DIO0);

  if (!LoRa.begin(BAND)) {
    Serial.println("Starting LoRa failed!");
    while (1);
  }
  Serial.println("LoRa Initializing OK!");
  display.setCursor(0,10);
  display.print("LoRa Initializing OK!");
  display.display();
  delay(2000);
//DHT11
      Serial.println(F("DHTxx test!"));

  dht.begin();

  //CCS811
      Serial.println("CCS811 test");

  if(!ccs.begin()){
    Serial.println("Failed to start sensor! Please check your wiring.");
    while(1);
  }

  // Wait for the sensor to be ready
  while(!ccs.available());
}

void OledLoop() {

    // Itxaron neurketa egin baino lehen
  delay(2000);

  float h = dht.readHumidity();
  // Irakurri tenperatura Celsius
  float t = dht.readTemperature();

  if(ccs.available()){
    if(!ccs.readData()){
      Serial.print("CO2: ");
      Serial.print(ccs.geteCO2());
      Serial.print("ppm, TVOC: ");
      Serial.println(ccs.getTVOC());
    }
    else{
      Serial.println("ERROR!");
      while(1);
    }
  }
  delay(500);

  display.clearDisplay();
  display.setCursor(0,0);
  display.println("AnderB JonS");
  display.setCursor(0,10);
  display.setTextSize(1);
  display.print("Neurketak:");
```

```
    display.setCursor(0,30);
  display.print("Hezetasuna: ");
  //display.setCursor(50,30);
  display.print(h);
  display.print("% ");
  display.setCursor(0,40);
  display.print("Tenperatura: ");
  display.print(t);
  display.print("°C ");
  display.setCursor(0,50);
  display.print("CO2: ");
  display.print(ccs.geteCO2());
  display.print("ppm, TVOC: ");
  display.print(ccs.getTVOC());

  display.display();

  // Egiaztatu irakurketaren batek huts egin duen eta goiz atera (berriro saiatzeko).
  if (isnan(h) || isnan(t)) {
    Serial.println(F("Ezin izan da DHT sentsoretik irakurri!"));
    return;
  }

  //Serial.print(F("Hezetasuna: "));
  //Serial.print(h);
  //Serial.print(F("%  Tenperatura: "));
  //Serial.print(t);
  //Serial.print(F("°C "));

}
```

TTN:

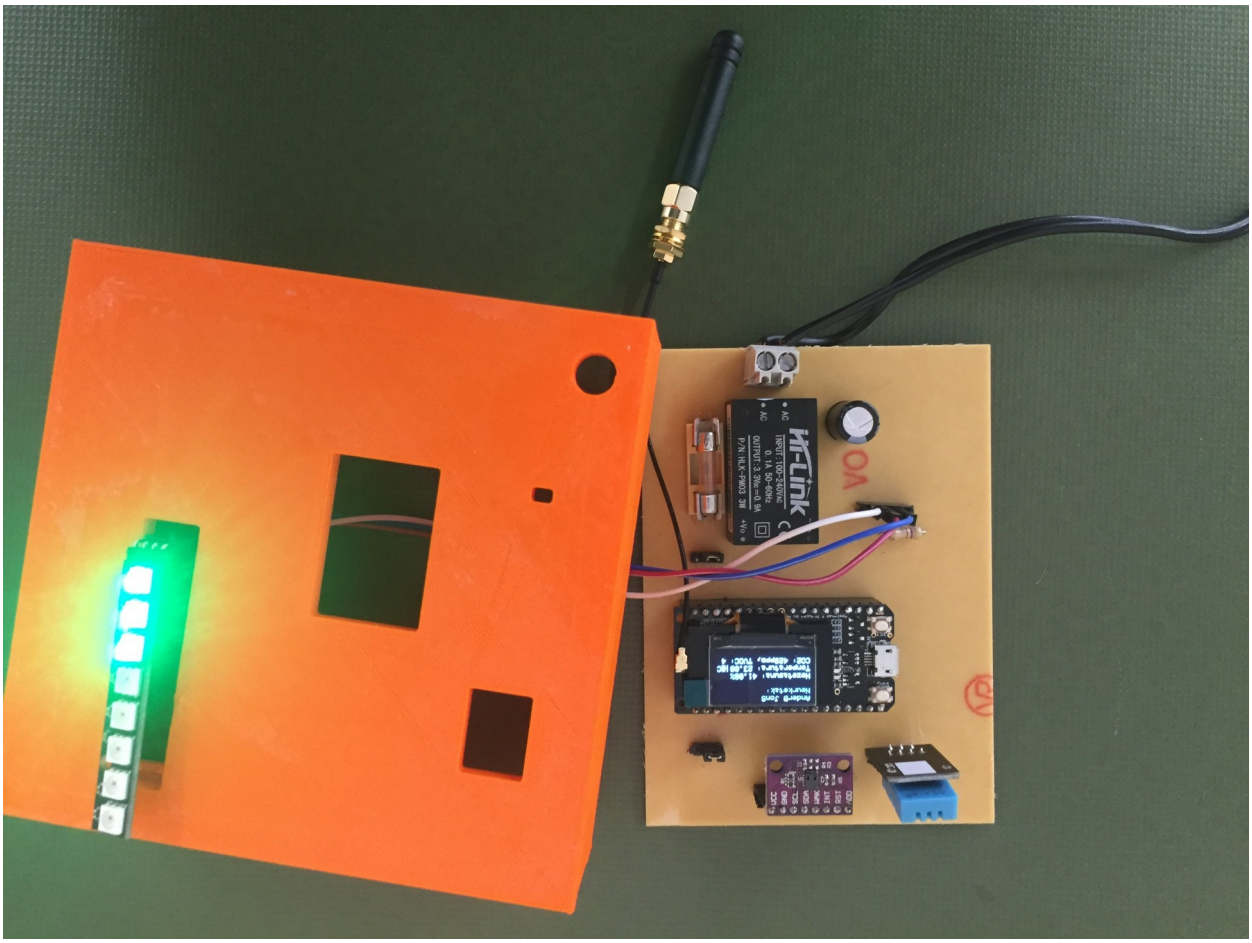
```
/* Funtzioa: TTN
 * Zeregina: Balioak TTNra bidaltzeko
 * Erabilitako liburutegiak:
 * Bueltatzen duena:
 * Sartutako parametroak: ezer
 *  */
void onEvent (ev_t ev) {
    Serial.print(os_getTime());
    Serial.print(": ");
    switch(ev) {
        case EV_SCAN_TIMEOUT:
            Serial.println(F("EV_SCAN_TIMEOUT"));
            break;
        case EV_BEACON_FOUND:
            Serial.println(F("EV_BEACON_FOUND"));
            break;
        case EV_BEACON_MISSED:
            Serial.println(F("EV_BEACON_MISSED"));
            break;
        case EV_BEACON_TRACKED:
            Serial.println(F("EV_BEACON_TRACKED"));
            break;
        case EV_JOINING:
            Serial.println(F("EV_JOINING"));
            break;
        case EV_JOINED:
            Serial.println(F("EV_JOINED"));
            break;
        case EV_RFU1:
```

```
            Serial.println(F("EV_RFU1"));
            break;
        case EV_JOIN_FAILED:
            Serial.println(F("EV_JOIN_FAILED"));
            break;
        case EV_REJOIN_FAILED:
            Serial.println(F("EV_REJOIN_FAILED"));
            break;
        case EV_TXCOMPLETE:
            Serial.println(F("EV_TXCOMPLETE (includes waiting for RX windows)"));
            if (LMIC.txrxFlags & TXRX_ACK)
              Serial.println(F("Received ack"));
            if (LMIC.dataLen) {
              Serial.println(F("Received "));
              Serial.println(LMIC.dataLen);
              Serial.println(F(" bytes of payload"));
            }
            // Schedule next transmission
            os_setTimedCallback(&sendjob, os_getTime()+sec2osticks(TX_INTERVAL),
do_send);
            break;
        case EV_LOST_TSYNC:
            Serial.println(F("EV_LOST_TSYNC"));
            break;
        case EV_RESET:
            Serial.println(F("EV_RESET"));
            break;
        case EV_RXCOMPLETE:
            // data received in ping slot
            Serial.println(F("EV_RXCOMPLETE"));
            break;
        case EV_LINK_DEAD:
            Serial.println(F("EV_LINK_DEAD"));
            break;
        case EV_LINK_ALIVE:
            Serial.println(F("EV_LINK_ALIVE"));
            break;
         default:
            Serial.println(F("Unknown event"));
            break;
    }
}

void do_send(osjob_t* j){
    // Check if there is not a current TX/RX job running
    if (LMIC.opmode & OP_TXRXPEND) {
        Serial.println(F("OP_TXRXPEND, not sending"));
    } else {
        // Prepare upstream data transmission at the next possible time.
       // LMIC_setTxData2(1, mydata, sizeof(mydata)-1, 0);

        BidalketaLoop();
        Serial.println(F("Packet queued"));
    }
    // Next TX is scheduled after TX_COMPLETE event.
}
void TTNsetup() {
    Serial.begin(115200);
    Serial.println(F("Starting"));

    #ifdef VCC_ENABLE
    // For Pinoccio Scout boards
    pinMode(VCC_ENABLE, OUTPUT);
    digitalWrite(VCC_ENABLE, HIGH);
```

```
    delay(1000);
    #endif

    // LMIC init
    os_init();
    // Reset the MAC state. Session and pending data transfers will be discarded.
    LMIC_reset();

    // Set static session parameters. Instead of dynamically establishing a session
    // by joining the network, precomputed session parameters are be provided.
    #ifdef PROGMEM
    // On AVR, these values are stored in flash and only copied to RAM
    // once. Copy them to a temporary buffer here, LMIC_setSession will
    // copy them into a buffer of its own again.
    uint8_t appskey[sizeof(APPSKEY)];
    uint8_t nwkskey[sizeof(NWKSKEY)];
    memcpy_P(appskey, APPSKEY, sizeof(APPSKEY));
    memcpy_P(nwkskey, NWKSKEY, sizeof(NWKSKEY));
    LMIC_setSession (0x1, DEVADDR, nwkskey, appskey);
    #else
    // If not running an AVR with PROGMEM, just use the arrays directly
    LMIC_setSession (0x1, DEVADDR, NWKSKEY, APPSKEY);
    #endif

    #if defined(CFG_eu868)
    // Set up the channels used by the Things Network, which corresponds
    // to the defaults of most gateways. Without this, only three base
    // channels from the LoRaWAN specification are used, which certainly
    // works, so it is good for debugging, but can overload those
    // frequencies, so be sure to configure the full frequency range of
    // your network here (unless your network autoconfigures them).
    // Setting up channels should happen after LMIC_setSession, as that
    // configures the minimal channel set.
    // NA-US channels 0-71 are configured automatically
    LMIC_setupChannel(0, 868100000, DR_RANGE_MAP(DR_SF12, DR_SF7),  BAND_CENTI);
// g-band
    LMIC_setupChannel(1, 868300000, DR_RANGE_MAP(DR_SF12, DR_SF7B), BAND_CENTI);
// g-band
    LMIC_setupChannel(2, 868500000, DR_RANGE_MAP(DR_SF12, DR_SF7),  BAND_CENTI);
// g-band
    LMIC_setupChannel(3, 867100000, DR_RANGE_MAP(DR_SF12, DR_SF7),  BAND_CENTI);
// g-band
    LMIC_setupChannel(4, 867300000, DR_RANGE_MAP(DR_SF12, DR_SF7),  BAND_CENTI);
// g-band
    LMIC_setupChannel(5, 867500000, DR_RANGE_MAP(DR_SF12, DR_SF7),  BAND_CENTI);
// g-band
    LMIC_setupChannel(6, 867700000, DR_RANGE_MAP(DR_SF12, DR_SF7),  BAND_CENTI);
// g-band
    LMIC_setupChannel(7, 867900000, DR_RANGE_MAP(DR_SF12, DR_SF7),  BAND_CENTI);
// g-band
    LMIC_setupChannel(8, 868800000, DR_RANGE_MAP(DR_FSK,  DR_FSK),  BAND_MILLI);
// g2-band
    // TTN defines an additional channel at 869.525Mhz using SF9 for class B
    // devices' ping slots. LMIC does not have an easy way to define set this
    // frequency and support for class B is spotty and untested, so this
    // frequency is not configured here.
    #elif defined(CFG_us915)
    // NA-US channels 0-71 are configured automatically
    // but only one group of 8 should (a subband) should be active
    // TTN recommends the second sub band, 1 in a zero based count.
    // https://github.com/TheThingsNetwork/gateway-conf/blob/master/US-global_conf.json
    LMIC_selectSubBand(1);
    #endif
```

```
    // Disable link check validation
    LMIC_setLinkCheckMode(0);

    // TTN uses SF9 for its RX2 window.
    LMIC.dn2Dr = DR_SF9;

    // Set data rate and transmit power for uplink (note: txpow seems to be ignored by
the library)
    LMIC_setDrTxpow(DR_SF7,14);

    // Start job
    do_send(&sendjob);
}

void TTNloop() {
    os_runloop_once();
}
```

9 Egindako sistemaren argazkiak



10 Gorabeherak

Izan dugun gorabehera plakan gnd-a ez dugula konektatu Sentsoreetara eta CJMCU-811 WAK pina 3,3V-RA konektatu dugu Gnd-n ordez.

11 Erronka hobetzeko proposamenak