



NODO TTN MAD V2.2

TTN MAD Community

juanfelixmateos@gmail.com

AGENDA

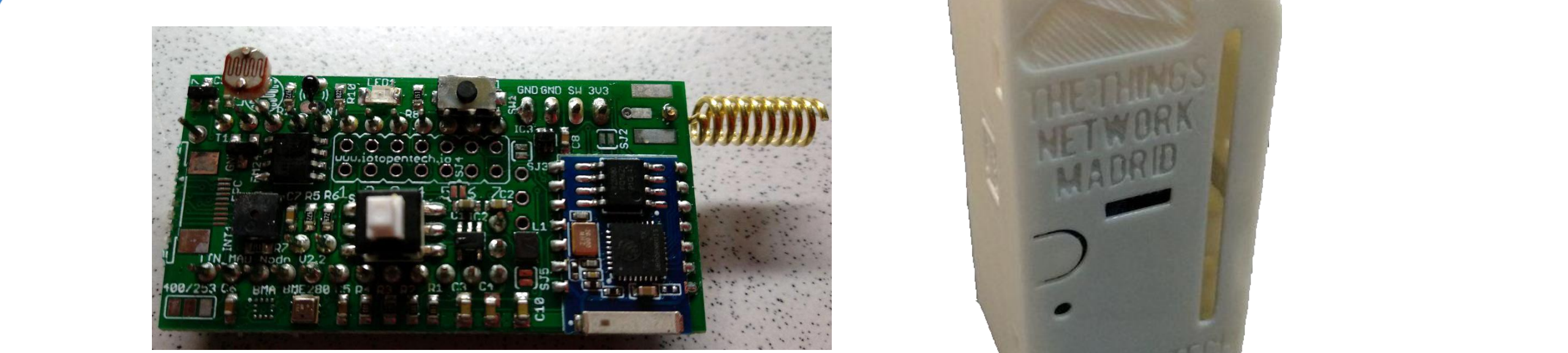
- 1 ¿Qué es el nodo TTN MAD v2.2?
- 2 La consola de TTN
- 3 Programa en Arduino para el nodo TTN MAD v2.2
- 4 Configuraciones del nodo
- 5 Montaje y programación de la configuración básica
- 6 Integraciones

¿Qué es el nodo TTN MAD v2.2?

Es un nodo LoRaWAN diseñado con los siguientes requerimientos:

- Fácil de montar
- Económico (monetaria y energéticamente)
 - Ampliable
- Muy flexible.

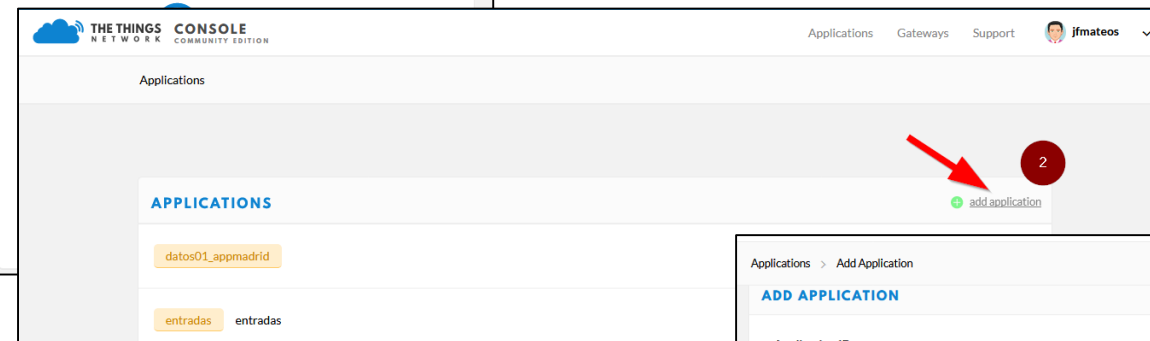
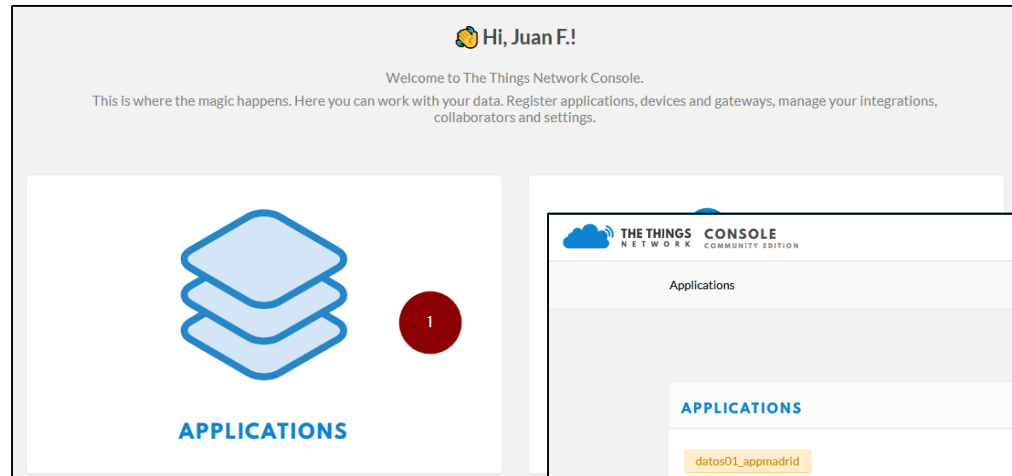
Nodo TTN MAD v2.2



La consola de The Things Network

Vamos a crear una aplicación nueva y, dentro de ella, un dispositivo que se una a la red por ABP.

Crear una aplicación



Applications > Add Application

ADD APPLICATION

Application ID
The unique identifier of your application on the network

jfmateos_hackmad

3

Description
A human readable description of your new app

Eg. My sensor network application

Application EUI
An application EUI will be issued for The Things Network block for convenience, you can add your own in the application settings page.

EUI issued by The Things Network

Handler registration
Select the handler you want to register this application to

ttn-handler-eu

4

Cancel Add application

Añadir un dispositivo

The screenshot shows the 'Applications' page for 'jfmateos_hackmad'. The 'DEVICES' section at the bottom shows '0 registered devices' and a 'register device' button, which is highlighted with a red circle and arrow labeled '1'. A modal titled 'REGISTER DEVICE' is open, showing the following fields:

- Device ID**: This is the unique identifier for this device in this app. The device ID will be immutable. The value 'jfmateos_hackmad_01' is entered, highlighted with a red circle and arrow labeled '2'.
- Device EUI**: This is the unique identifier for this device on the network. You can change the EUI later. The value 'this field will be generated' is shown, highlighted with a red circle and arrow labeled '3'.
- App Key**: The App Key will be used to secure the communication between you device and the network. The value 'this field will be generated' is shown.
- App EUI**: The value '70 B3 D5 7E D0 02 40 BF' is shown.

At the bottom of the modal, there are 'Cancel' and 'Register' buttons. The 'Register' button is highlighted with a red circle and arrow labeled '4'.

Configurar el dispositivo para activarse por ABP

The image shows two overlapping screenshots of the The Things Network (TTN) web interface, illustrating the steps to configure a device for activation by ABP (Activation by Parameters).

Left Screenshot (Device Overview):

- Navigation path: Applications > jfmateos_hackmad > Devices > jfmateos_hackmad_01
- Buttons: Overview, Data, **Settings** (indicated by a red arrow and a red circle with the number 1).
- Section: **DEVICE OVERVIEW**
- Fields: Application ID (jfmateos_hackmad), Device ID (jfmateos_hackmad_01), Activation Method (OTAA), Device EUI, Application EUI (70 B3 D5 7E D0 02 40 BF), and App Key.

Right Screenshot (Settings):

- Navigation path: Applications > jfmateos_hackmad > Devices > jfmateos_hackmad_01 > Settings
- Section: **Activation Method**
- Buttons: OTAA, **ABP** (indicated by a red arrow and a red circle with the number 2).
- Field: Device Address (The device address will be assigned by the network server).
- Field: Network Session Key (Network Session Key will be generated).
- Field: App Session Key (App Session Key will be generated).
- Field: Frame Counter Width (15, indicated by a red arrow and a red circle with the number 3).
- Field: ☐ **Frame Counter Checks** (Disabling frame counter checks drastically reduces security and should only be used for development purposes).
- Buttons: Delete Device, Cancel, **Save** (indicated by a red arrow and a red circle with the number 4).

Activar el formato de carga de pago Cayenne LPP

Applications > jfmateos_hackmad > Payload Formats


Overview Devices **Payload Formats** Integrations Data Settings

PAYLOAD FORMATS

Payload Format
The payload format sent by your devices

Cayenne LPP

Cancel no changes to save



Programa en Arduino para el nodo TTN MAD v2.2

Vamos a crear un programa que envíe un frame con la tensión de la batería cada 30 segundos.

Librerías de Arduino

Instalar las siguientes librerías en el IDE de Arduino:

- MCCI LoRaWAN LMIC Library by IBM, Matthis Kooijman, Terry Moore, ChaeHee Won, Frank Rose. Versión 2.3.2.
- Low-Power by Rocket Scream Electronics. Versión 1.6.0.
- CayenneLPP by Electronics Cats. Versión 1.0.1

Configurar la librería MCCI LoRaWAN LMIC Library

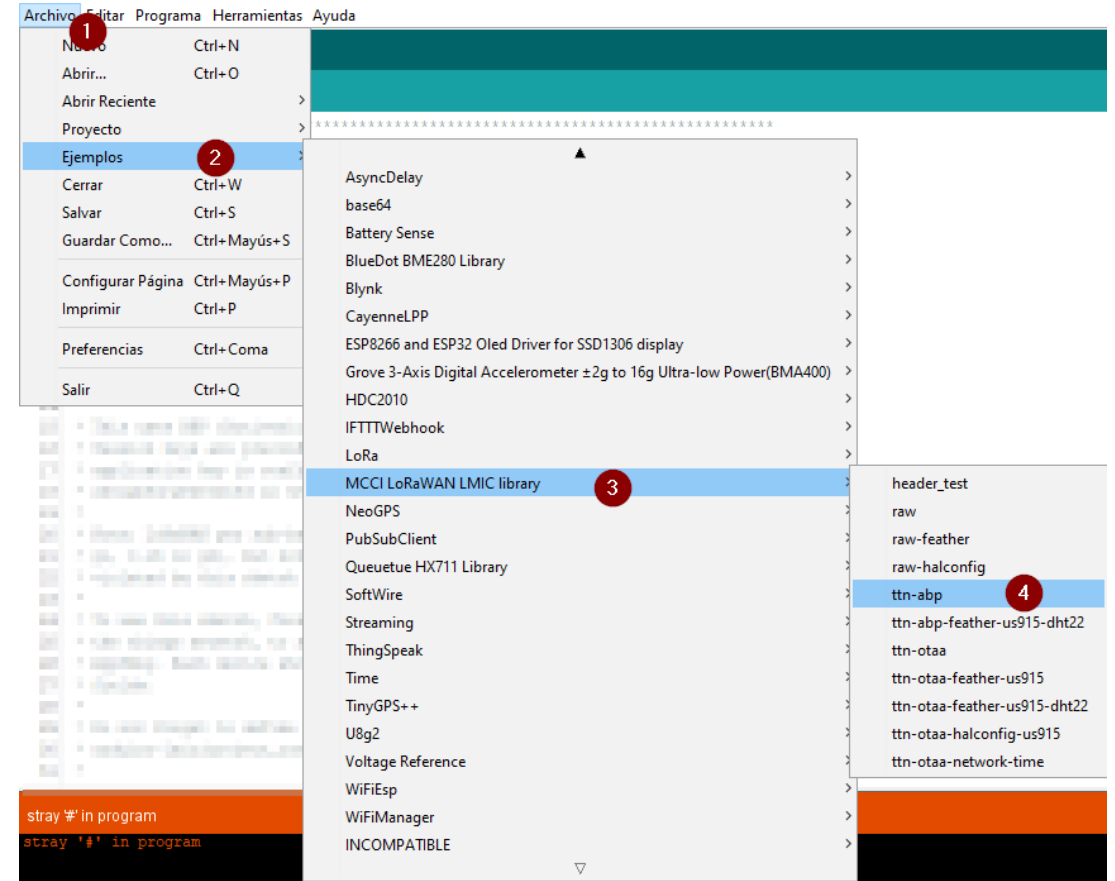
Para reducir al máximo los requisitos de memoria de la librería vamos a eliminar las funcionalidades que no necesitamos modificando el siguiente archivo:

MCCI_LoRaWAN_LMIC_library\project_config\lmic_project_config.h

```
// project-specific definitions
#define CFG_eu868 1
// #define CFG_us915 1
// #define CFG_au921 1 // #define CFG_as923 1
// #define LMIC_COUNTRY_CODE
LMIC_COUNTRY_CODE_JP /* for as923-JP */
// #define CFG_in866 1
#define CFG_sx1276_radio 1
// #define LMIC_USE_INTERRUPTS
#define DISABLE_PING
#define DISABLE_BEACONS
```

Punto de partida

- Partimos del ejemplo ttn-abp de la librería MCCI LoRaWAN LMIC Library



Indicar las credenciales del nodo

- Copiamos el NwkSkey (msb), AppSkey (msb) y DevAddr (expresar en hexadecimal anteponiendo 0x) de la consola de TTN en los lugares correspondientes del código.

The screenshot displays the Arduino IDE interface. On the left, the 'DEVICE OVERVIEW' panel shows the following fields:

- Application ID: jfmateos_hackmad
- Device ID: jfmateos_hackmad_01
- Activation Method: ABP
- Device EUI: 00 95 A7 08 96 89 D5 0D
- Application EUI: 70 B3 D5 7E D0 02 40 BF
- Device Address: 26 01 11 38
- Network Session Key: 0x65, 0xBB, 0xE5, 0xF5
- App Session Key: 0xFB, 0x78, 0x52, 0x47

On the right, the code file 'ttn_mad_v2_2_basico' is open, showing the following relevant lines:

```
56 static const PROGMEM ul_t NWSKEY[16] = { 0x65, 0xBB, 0xE5, 0xF5, ...  
58 // LoRaWAN AppSKey, application session key  
59 static const ul_t PROGMEM APPSKEY[16] = { 0xFB, 0x78, 0x52, 0x47, ...  
61 // LoRaWAN end-device address (DevAddr)  
62 // See http://thethingsnetwork.org/wiki/AddressSpace  
63 // The library converts the address to network byte order  
64 static const ul_t DEVADDR = 0x26011138 ; // <-- Change  
65  
66 // These callbacks are only used in over-the-air activation  
67 // left empty here (you cannot leave them out completely)
```

Red arrows indicate the mapping from the overview fields to the code: from 'Network Session Key' to line 56, from 'App Session Key' to line 59, and from 'Device Address' to line 64.

At the bottom, the 'Compilado' (Compiled) status shows: 'El Sketch usa 278284 bytes (55%) del espacio de almacenamiento' and 'Las variables Globales usan 28240 bytes (34%) de la memoria'.

Modificar los pines que controlan el RFM95W

- Configurar la sección
//Pin mapping como se
indica

```
// Pin mapping
const lmic_pinmap lmic_pins = {
    .nss = 10,
    .rxtx = LMIC_UNUSED_PIN,
    .rst = 9,
    .dio = {2, 7, LMIC_UNUSED_PIN},
};
```

- Cada sensor/actuador tiene asociado un tipo de dato

The diagram illustrates the conversion of a hex value from a JSON payload to a temperature reading. It features a light blue background with a white box containing a JSON snippet. A red circle highlights the value "00C8" in the "temperature_1" field. A blue callout box points to this value, containing the text: "Temperatura con resolución 0.1°C con signo y MSB 0x00C8=200=20.0C". Another blue callout box points to the "temperature_1" field in the JSON, containing the text: "Sensor de temperatura". A third blue callout box points to the "00C8" value, containing the text: "Canal 1".

Payload

```
{
  "analog_in_4": 3,
  "barometric_press": 1013,
  "digital_out_6": 1,
  "presence_5": 0,
  "relative_humidity_3": 44,
  "temperature_1": 20
}
```

Canal 1

Sensor de temperatura

Temperatura con resolución 0.1°C con signo y MSB
 $0x00C8 = 200 = 20.0C$

Incluir las librerías lowpower y cayenne

```
37 #include <lmic.h>
38 #include <hal/hal.h>
39 #include <SPI.h>
40 #include "LowPower.h"
41 #include <CayenneLPP.h>
42 CayenneLPP lpp(4);
43 //
44 // ...
```

Insertar la función readVCC

```
long readVcc() {  
  
    ADMUX = _BV(REFS0) | _BV(MUX3) | _BV(MUX2) |  
    _BV(MUX1);  
  
    delay(2);  
  
    ADCSRA |= _BV(ADSC);  
  
    while (bit_is_set(ADCSRA, ADSC));  
  
    long result = ADCL;  
  
    result |= ADCH << 8;  
  
    result = 1126400L / result; // Back-calculate AVcc in mV  
  
    return result;  
  
}
```

```
86     .rxtx = LMIC_UNUSED_PIN,  
87     .rst = 9,  
88     .dio = {2, 7, LMIC_UNUSED_PIN},  
89 };  
90 long readVcc() {  
91     ADMUX = _BV(REFS0) | _BV(MUX3) | _BV(MUX2) | _BV(MUX1);  
92     delay(2);  
93     ADCSRA |= _BV(ADSC);  
94     while (bit_is_set(ADCSRA, ADSC));  
95  
96     long result = ADCL;  
97     result |= ADCH << 8;  
98     result = 1126400L / result; // Back-calculate AVcc in mV  
99     return result;  
100 }  
101 void onEvent (ev_t ev) {  
102     Serial.print(os_getTime());  
103     Serial.print(": ");
```

Modificar el código de la función do_send

```
181 }
182
183 void do_send(osjob_t* j){
184     // Check if there is not a current TX/RX job running
185     if (LMIC.opmode & OP_TXRXPEND) {
186         Serial.println(F("OP_TXRXPEND, not sending"));
187     } else {
188         // Prepare upstream data transmission at the next possible time.
189         lpp.reset();
190         lpp.addAnalogInput(1, readVcc() / 1000.F);
191         //LMIC_setTxData2(1, mydata, sizeof(mydata)-1, 0);
192         LMIC_setTxData2(1, lpp.getBuffer(), lpp.getSize(), 0);
193         Serial.println(F("Packet queued"));
194     }
195     // Next TX is scheduled after TX COMPLETE event.
196 }
197
198 void setup() {
```

Modificar el código del estado EV_TXCOMPLETE

```
137 case EV_TXCOMPLETE:
138     Serial.println(F("EV_TXCOMPLETE (includes waiting for RX windows)"));
139     if (LMIC.txrxFlags & TXRX_ACK)
140         Serial.println(F("Received ack"));
141     if (LMIC.dataLen) {
142         Serial.println(F("Received "));
143         Serial.println(LMIC.dataLen);
144         Serial.println(F(" bytes of payload"));
145     }
146     // Schedule next transmission
147     //os_setTimedCallback(&sendjob, os_getTime()+sec2osticks(TX_INTERVAL), do_send);
148     Serial.println("Me voy a echar una siestecita");
149     delay(1000); // Para que tenrminen de imprimirse los mensajes en el terminal
150     for (byte contador = 0; contador < 3; contador++) {
151         LowPower.powerDown(SLEEP_8S, ADC_OFF, BOD_OFF);
152     }
153     os_setCallback (&sendjob, do_send);
154     break;
155 case EV_LOST_TSYNC:
156     // ...
```

¿Configuraciones del nodo?

Configuración Básica

El nodo carece de sensores, pero puede utilizarse para enviar paquetes o mapear coberturas con ttnmapper.

Configuraciones opcionales

Extensión Hall

Consiste en un sensor de efecto hall que sirve para detectar la apertura de puertas o ventanas.

Extensión auxiliar I2C

Permite utilizar sensores basados en la tecnología I2C, como el BME280.

Extensión auxiliar booster

Permite regular la tensión de funcionamiento a 3.3V desde cualquier fuente de al menos 1.2V (por ejemplo, una sola batería AA).

Extensión auxiliar puerto de ampliación

Consiste en un conector FPC que expone varios de los pines del nodo para que puedan añadirse otros sensores/actuadores (GPS, HDC2080...)

Configuraciones opcionales

Extensión WiFi

Permite configurar el nodo a través de una página web y, potencialmente, podría utilizarse en tareas de geolocalización basadas en la ubicación de los puntos de acceso WiFi.

Extensión LDR

Consiste en un sensor de luminosidad básico.

Extensión NTC

Consiste en un sensor de temperatura básico.

Extensión LED

Consiste en un LED que podemos controlar desde Internet.

Extensión Buzzer

Consiste en un buzzer de frecuencia fija que podemos controlar desde Internet.

Configuraciones opcionales

Extensión memoria ferromagnética

Permite almacenar datos de forma permanente, aunque el nodo pierda la alimentación eléctrica.

Extensión BME280

Consiste en un sensor de temperatura, presión y humedad.

Extensión BMA253/BMA400

Consiste en un acelerómetro de 3 ejes, con el que podemos detectar la orientación del nodo, impactos, contar pasos...

Extensión PIR

El PIR es un sensor de presencia.

Extensión GPS

Permite añadir un GPS + Magnetómetro

Montaje de la configuración básica

PCB

Arduino Pro mini 3v3 8MHz

RFM95W + antena helicoidal

Header 4 pines con tornillos

Pulsado On/Off

Portabaterías doble AAA

SEGURIDAD Y SOLDADURA



Riesgos

- **Químicos**

- Plomo nunca más
- Rosin (Colofonía) ¡Ojo con asmáticos!

- **Térmicos**

- Quemarse (poca importancia salvo que afecte a los ojos)

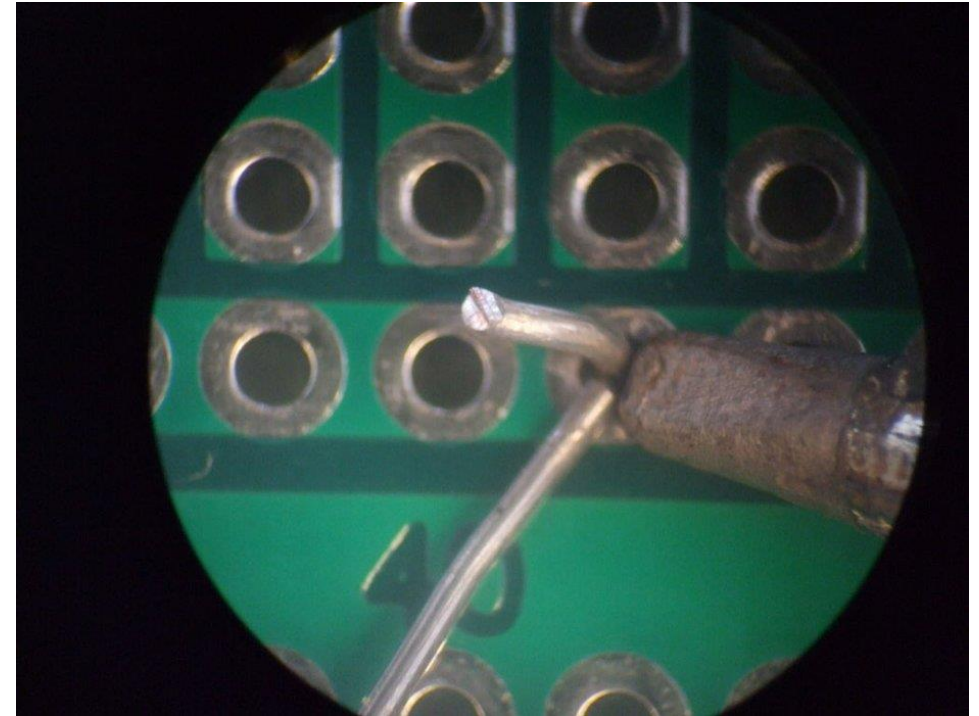
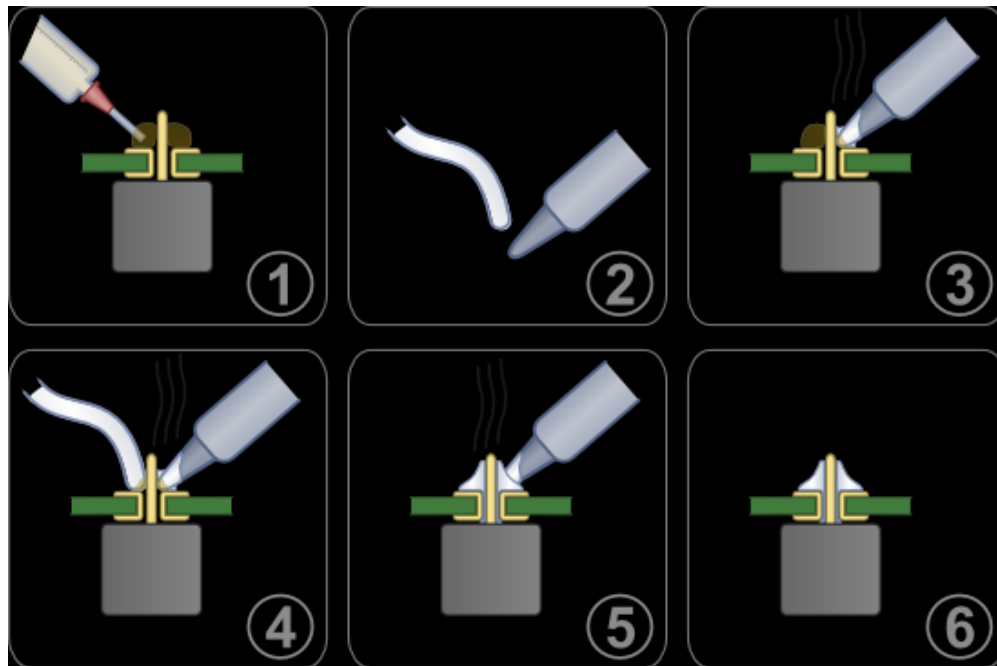
- **Eléctricos**

- Quemar el cable del soldador → Salta el magnetotérmico y susto
- No usar pulseras anti-estáticas → Estorban

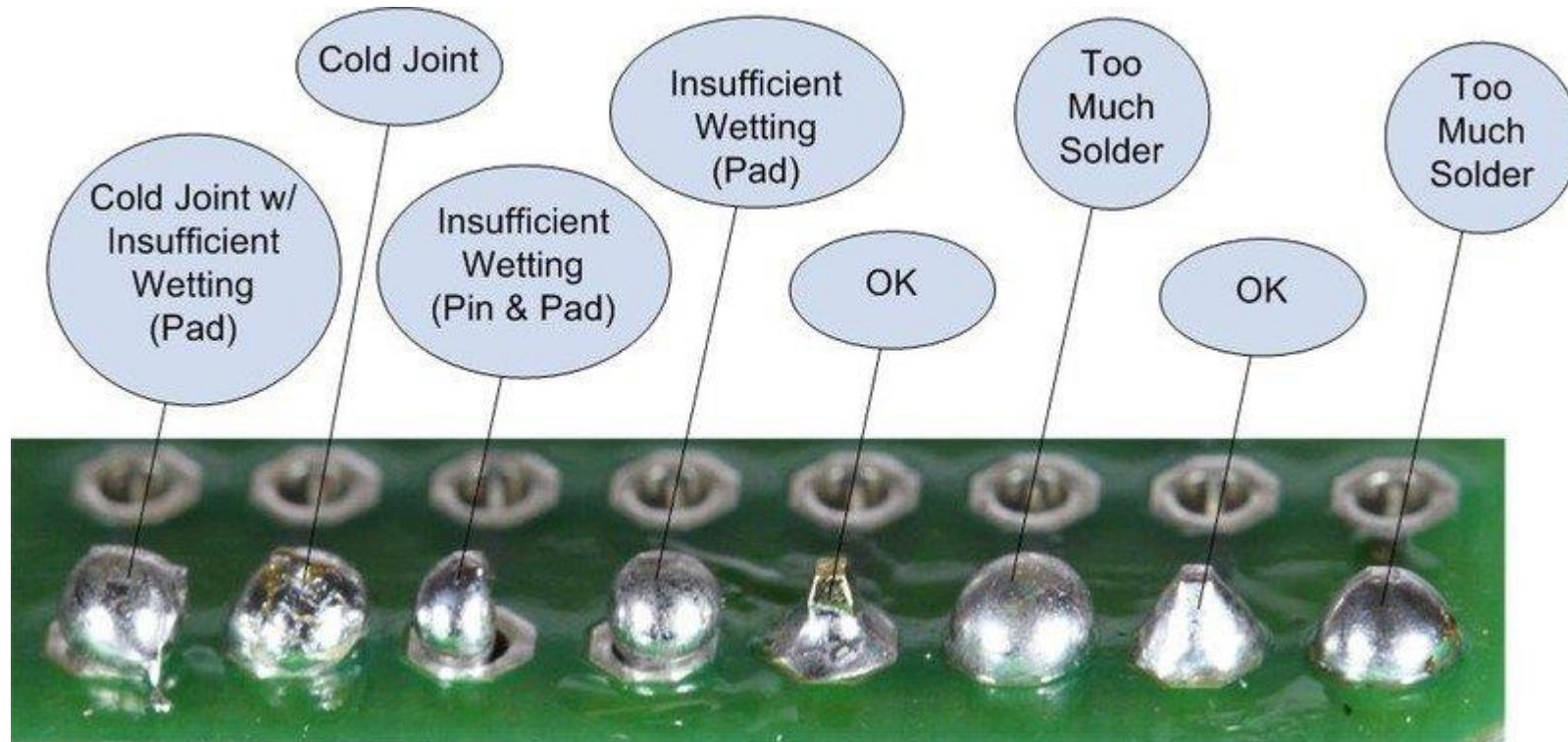
- **Mecánicos**

- Punta del soldador → Pinchar al compañero

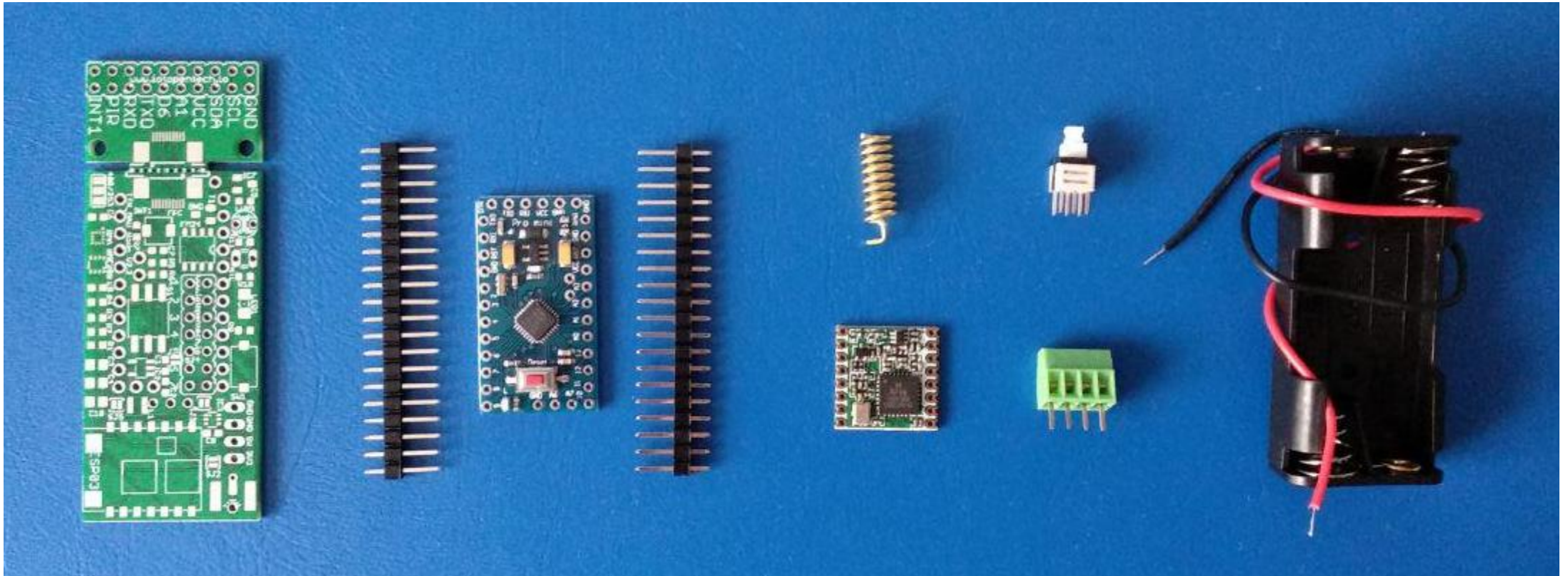
¿CÓMO SOLDAR?



SOLDADURA CORRECTA



Componentes de la configuración básica

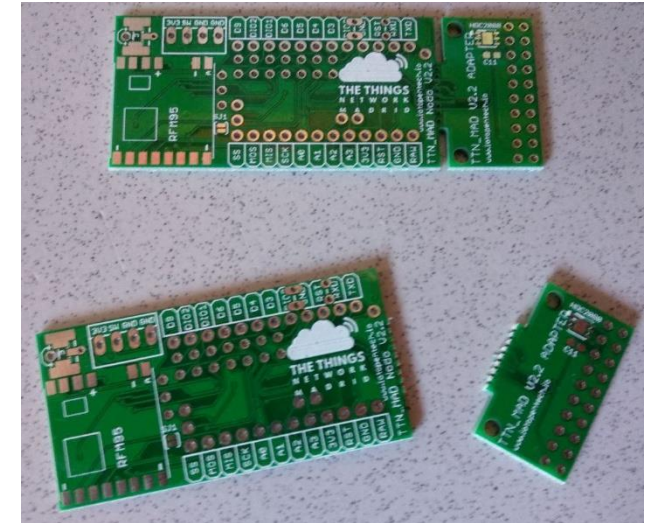


1. Decidir si separar o no la zona auxiliar del PCB

La zona auxiliar está pensada para facilitar el prototipado de otras ampliaciones, como la de GPS, bien sin separar del PCB principal o bien separándola (la conexión se realiza en este caso mediante un cable plano y conectores FPC).

Si no se va a utilizar conviene separarla ahora; hacerlo después de soldar los componentes podría dañar alguna soldadura.

Para separarla, se recomienda repasar cuidadosamente la línea de mouse bites con un cutter y flexionar.



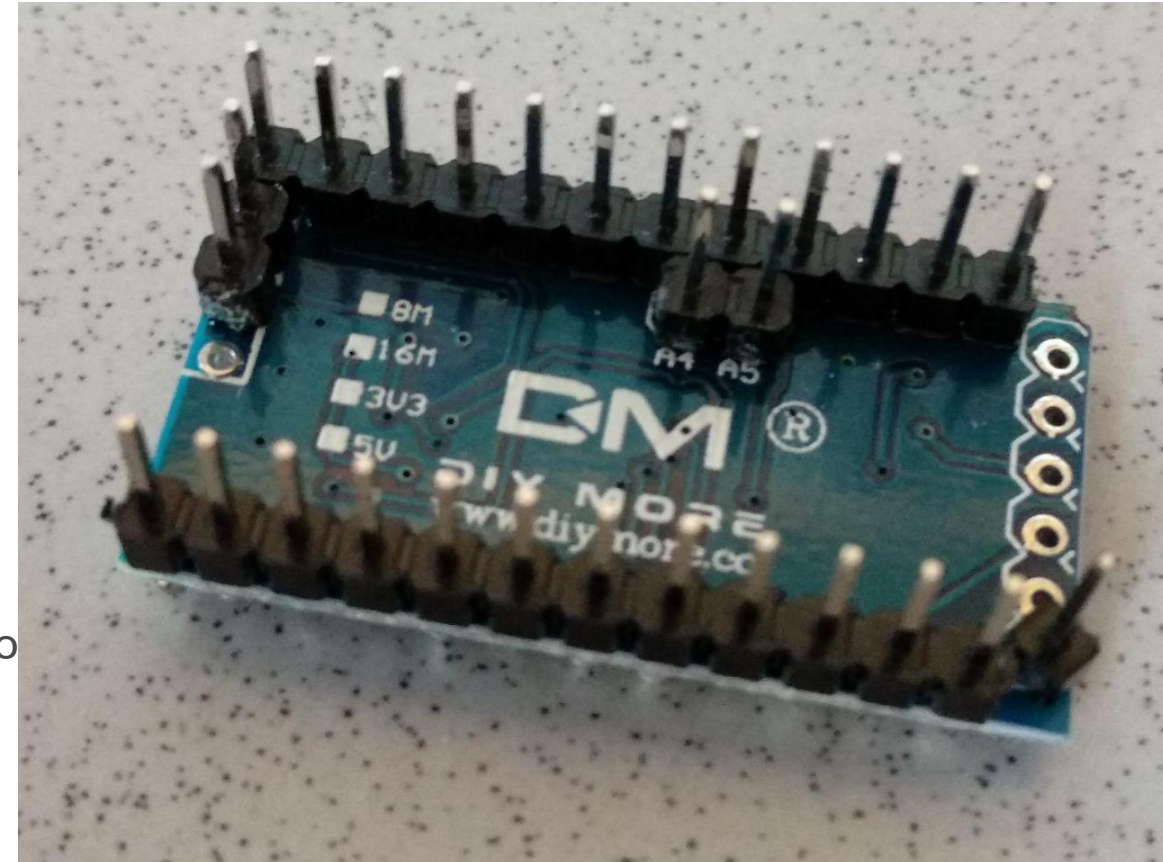
2. Solar los pines del Arduino Pro Mini

Deben soldarse:

- Las 2 tiras de pines de los lados largos del Arduino
- Los pines A4 y A5
- Los pines A6 y A7
- El pin DTR.

Obsérvese que las tiras de pines se introducen por la sección más corta de los pines y desde la cara posterior del Arduino Pro Mini.

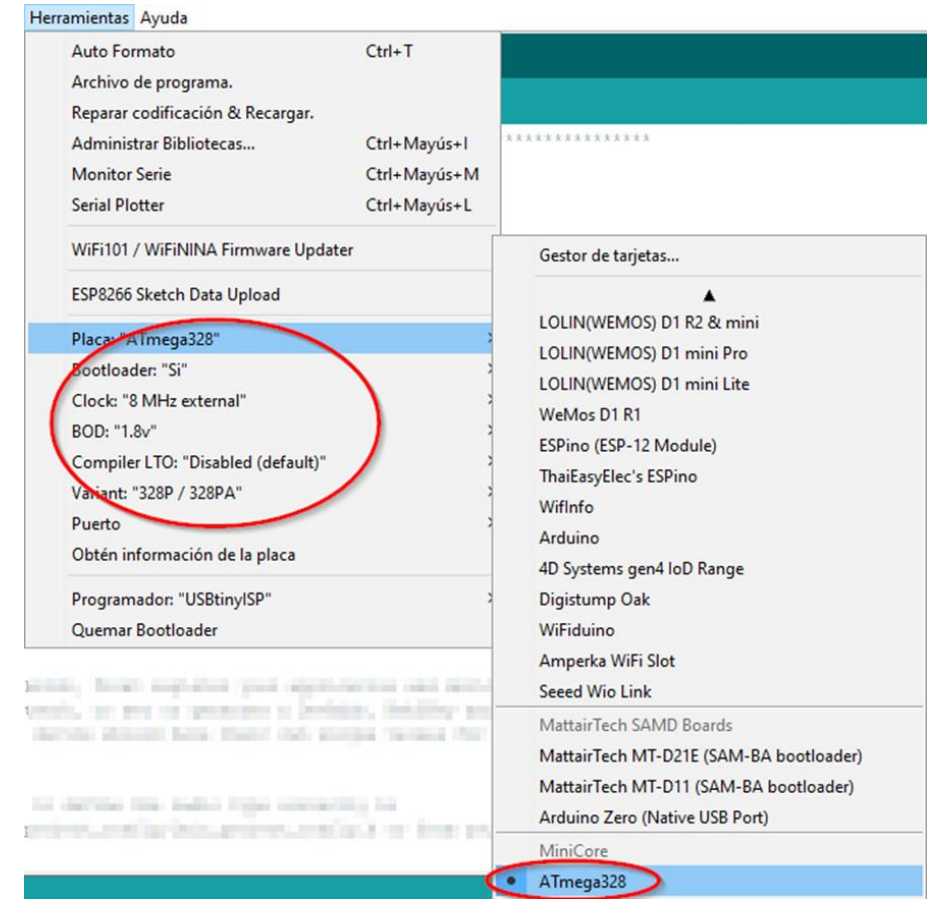
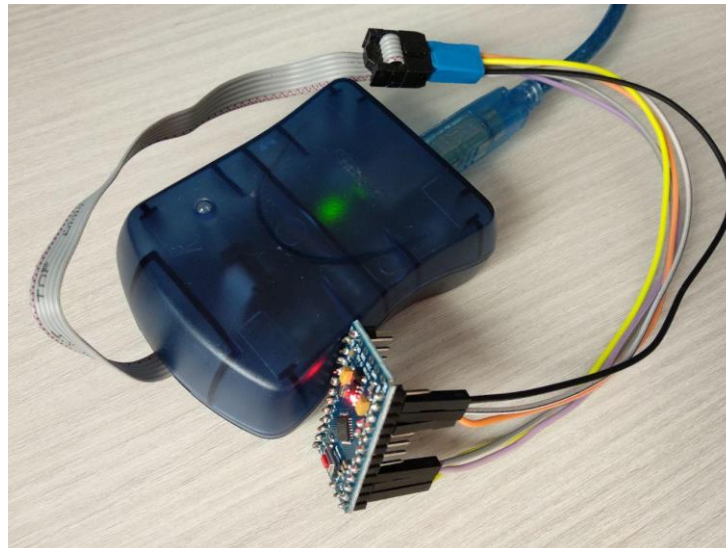
Se recomienda plantear el Arduino sobre el PCB del nodo para que los pines no queden inclinados.



3. Sustituir el bootloader por el de MiniCore (opcional)

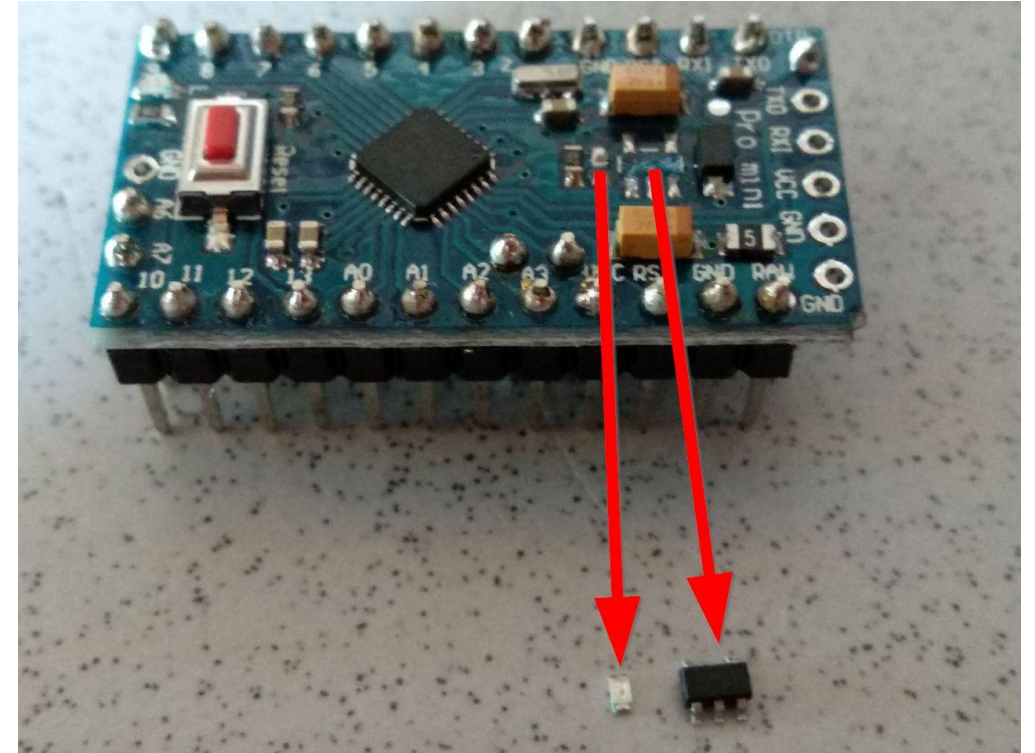
Sustituir el bootloader del Arduino Pro Mini por la versión de MiniCore (<https://github.com/MCUdude/MiniCore>) usando un programador (por ejemplo, el USBtinyISP). Para conectar el programador al Arduino se usarán los siguientes pines:

- VCC
- GND
- RST
- 11: MOSI
- 12: MISO
- 13: SCK



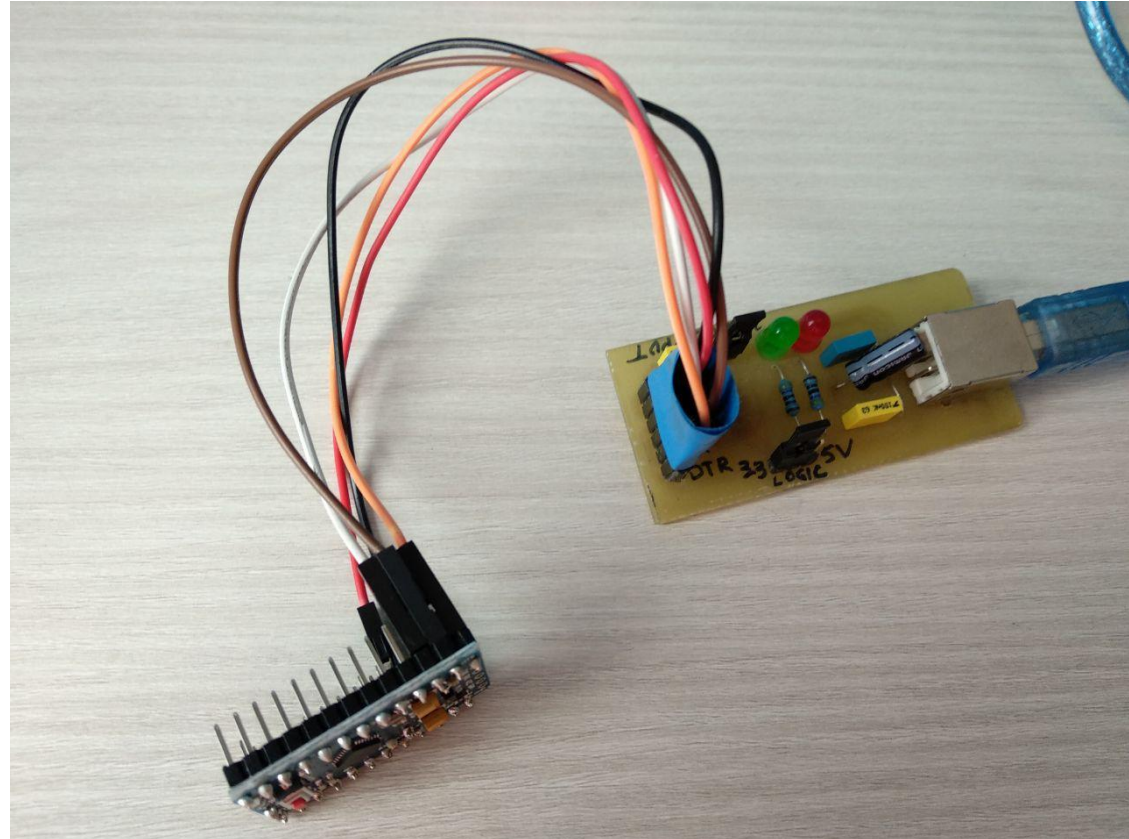
4. Eliminar el conversor y el LED (opcional)

Eliminar el conversor de tensión y el LED de alimentación del Arduino Pro Mini; la eliminación de estos 2 componentes persigue reducir el consumo al mínimo.



5. Programar el Arduino Pro Mini

Cargar el programa en el Arduino Pro Mini
usando un conversor USB → Serial



6. Soldar el RFM95W y la antena al PCB

Prestar atención a la orientación del RFM95W.

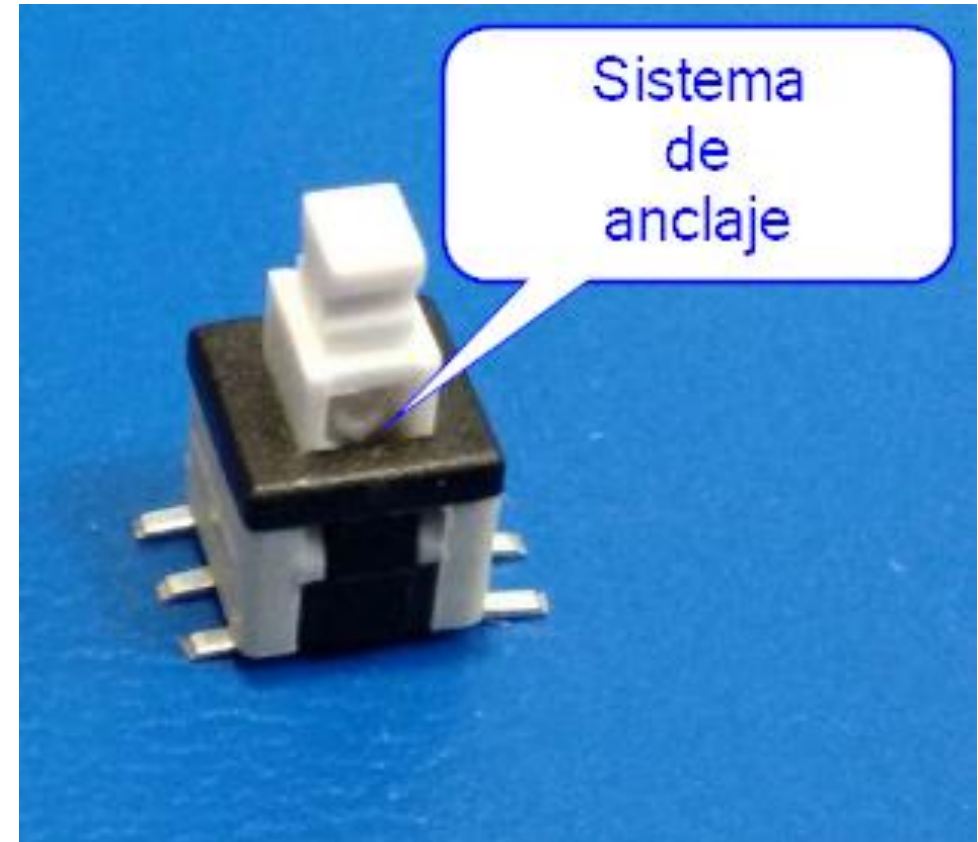
Se recomienda fijar "provisionalmente" uno de los pines castelados mientras se sujeta el módulo con un dedo para que no se mueva, y luego soldar el resto de pines.



7. Preparación del pulsador On/Off

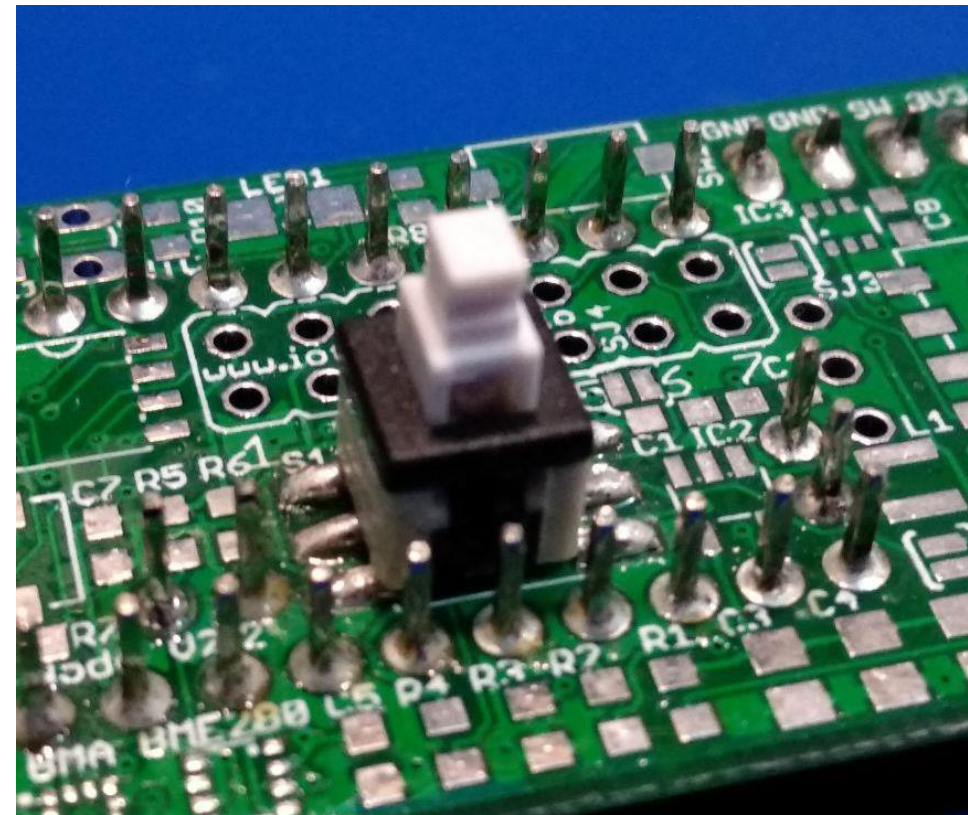
Doblar las patillas del pulsador On/Off y recortarlas para que queden con una longitud de unos 2 mm.

Obsevar que en uno de los lados del vástago del pulsador es visible el sistema de anclaje.



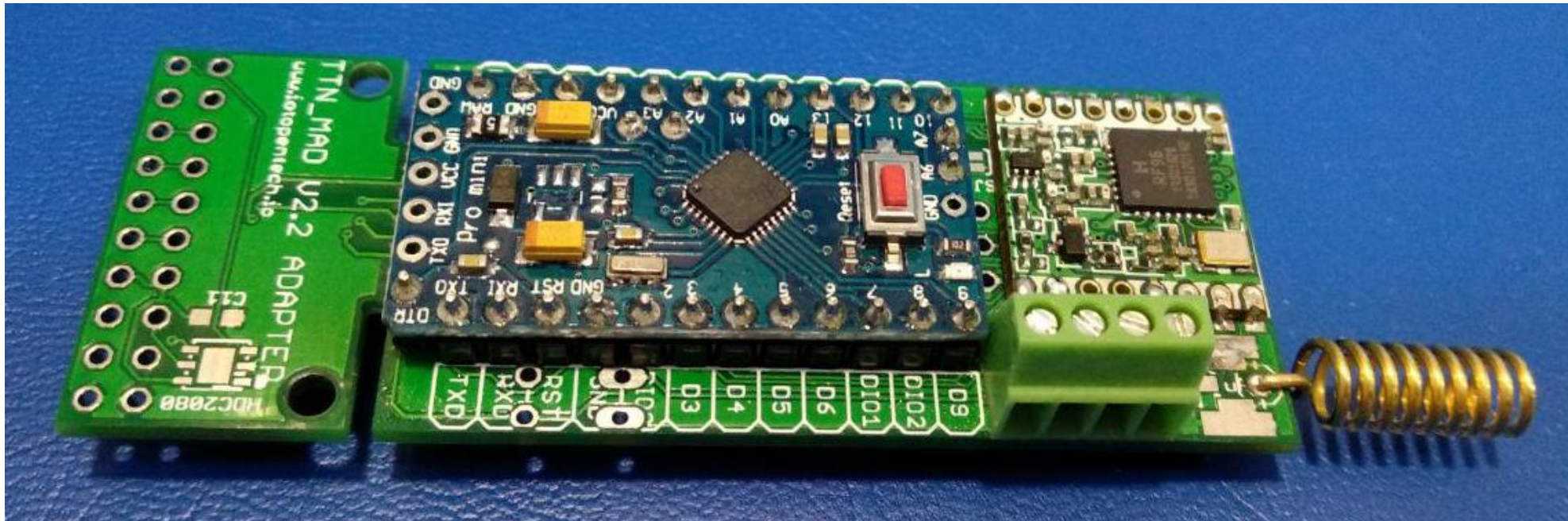
8. Soldar pulsador On/Off

Soldar el pulsador dejando el sistema de anclaje orientado hacia el borde más próximo del PCB.

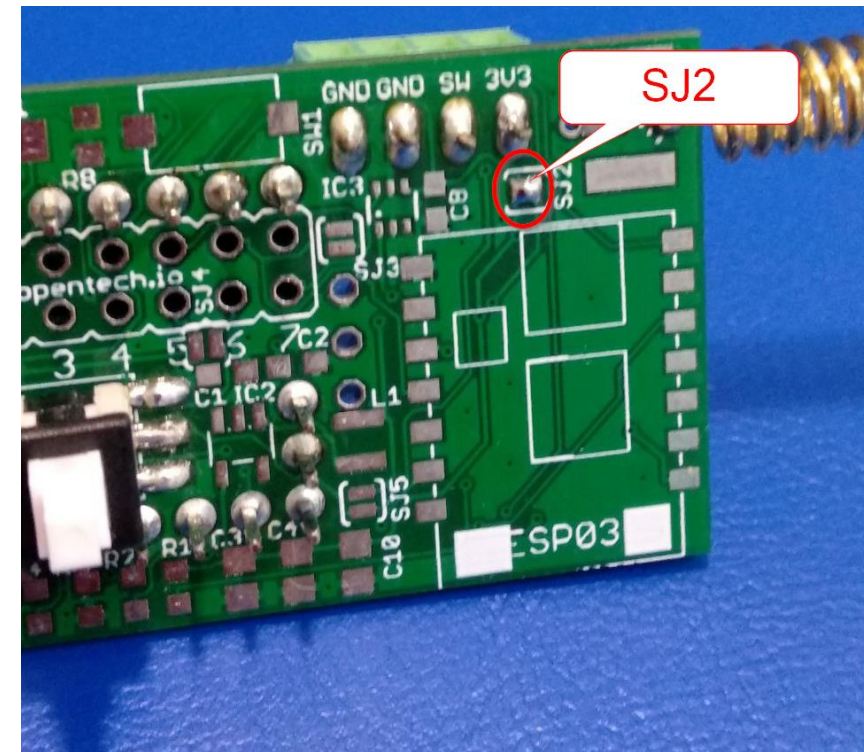
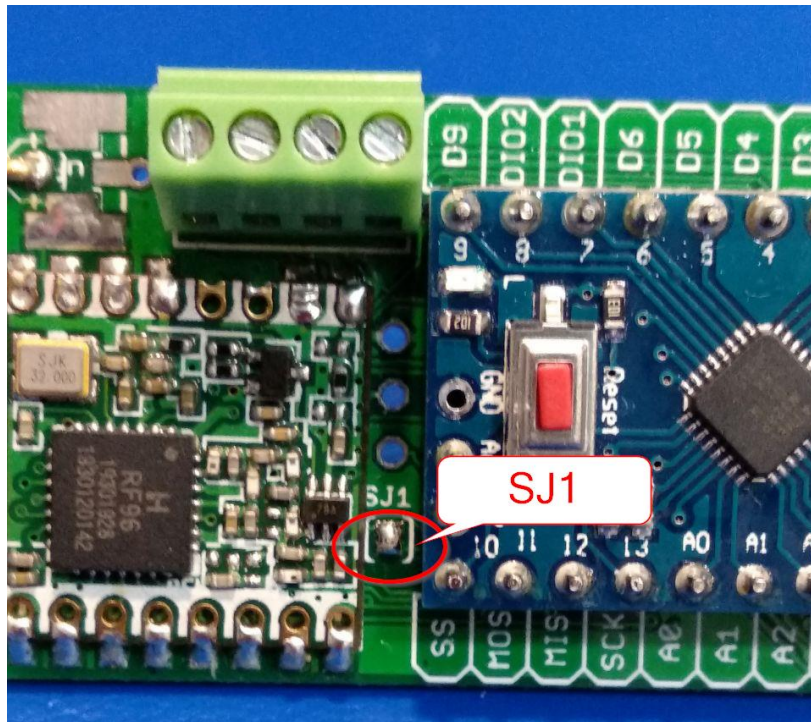


9. Soldar el Arduino y el header de 4 pines

Prestar atención a la orientación del header de 4 pines.

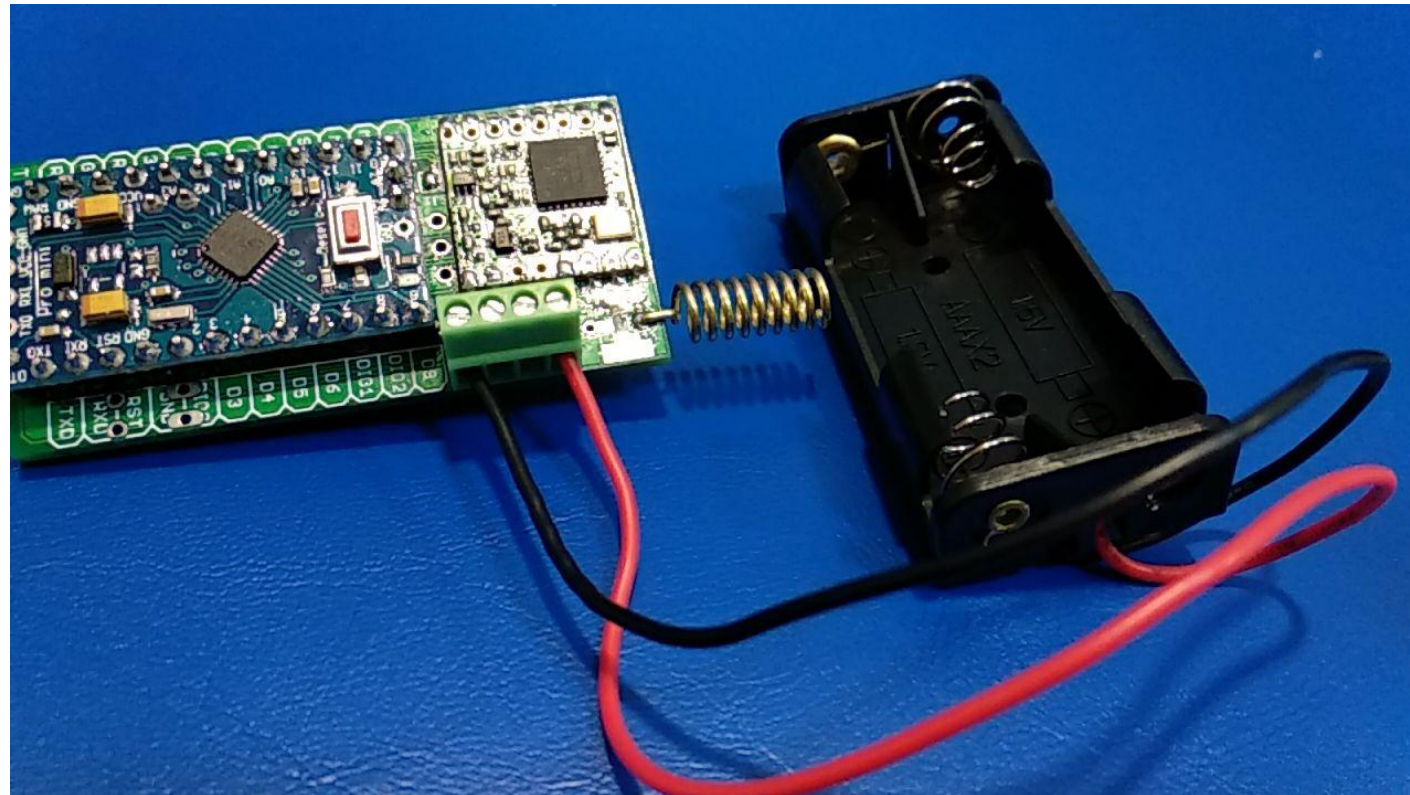


10. Cerrar con estaño los jumpers SJ1 y SJ2



11. Fijar con los tornillos los cables del portabaterías

Prestar atención a la polaridad.

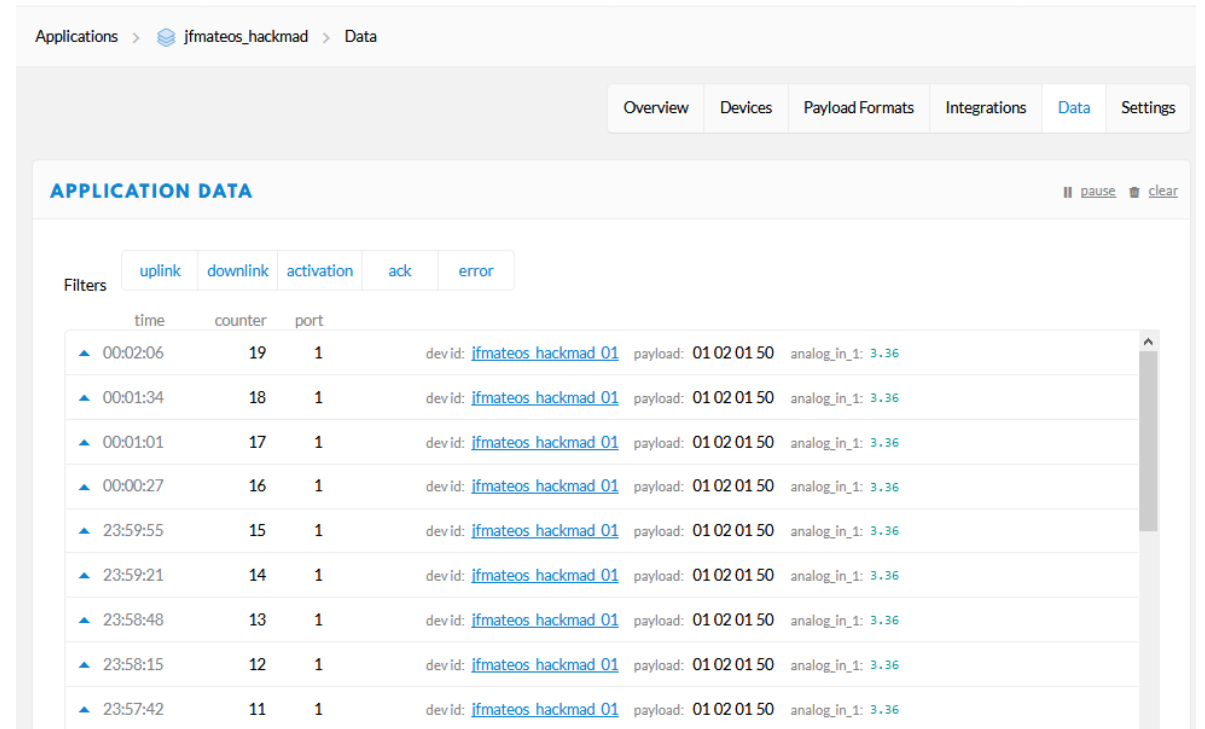


12. Puesta en funcionamiento

Insertar las baterías.

Pulsar el botón de encendido.

Los frames empezarán a aparecer en la consola de The Things Network.



The screenshot displays the 'Data' tab for the application 'ifmateos_hackmad'. The interface includes a navigation bar with tabs for Overview, Devices, Payload Formats, Integrations, Data (selected), and Settings. Below the navigation bar, the 'APPLICATION DATA' section is visible, featuring a 'Filters' bar with options for uplink, downlink, activation, ack, and error. The main data table lists frames with columns for time, counter, port, dev id, payload, and analog_in_1. The data shows a sequence of frames with increasing counters and decreasing times, indicating a reverse chronological order.

time	counter	port	dev id	payload	analog_in_1
00:02:06	19	1	ifmateos_hackmad_01	01 02 01 50	3.36
00:01:34	18	1	ifmateos_hackmad_01	01 02 01 50	3.36
00:01:01	17	1	ifmateos_hackmad_01	01 02 01 50	3.36
00:00:27	16	1	ifmateos_hackmad_01	01 02 01 50	3.36
23:59:55	15	1	ifmateos_hackmad_01	01 02 01 50	3.36
23:59:21	14	1	ifmateos_hackmad_01	01 02 01 50	3.36
23:58:48	13	1	ifmateos_hackmad_01	01 02 01 50	3.36
23:58:15	12	1	ifmateos_hackmad_01	01 02 01 50	3.36
23:57:42	11	1	ifmateos_hackmad_01	01 02 01 50	3.36

Integraciones

MyDevices

ttnmapper

IFTTT

NodeRED



YOU ARE THE NETWORK
LET'S BUILD THIS THING TOGETHER!

Juan Félix Mateos

juanfelixmateos@gmail.com

www.thethingsnetwork.org