

Introducción a la programación del nodo TTNMAD EASY

Juan Félix Mateos

Mayo 2024

juanfelixmateos@gmail.com

Opciones para programar el RAK3172

- **RUI3**: La guardería. Fácil pero poco flexible. Se pierden algunas funcionalidades (puertos I2C, pines analógicos...), pero admite LoRaWAN clase A, B y C.
- **STM32DUINO**: El instituto. Fácil pero incompleto (LoRaWAN sin Low power automático, ni clases B ni C).
- **RTOS** (Mbed OS, Zephyr...): La universidad. Experimental o abandonado.
- **STM32CubeIDE**: El máster. Prepárate para lidiar con bugs.

RAK11720:

[Online Help](#)

[More Info](#)

RAKwireless RUI STM32 Boards by RAKwireless versión 4.0.1 **INSTALLED**

Tarjetas incluidas en éste paquete

RAK3172-E, RAK3272-SiP, RAK3272LP-SiP.

[Online Help](#)

[More Info](#)

Seleccione versión ▾

Instalar

Actualizar

Eliminar

RAKwireless RUI nRF Boards by RAKwireless

Tarjetas incluidas en éste paquete

RAK4631, RAK3401, RAK5010

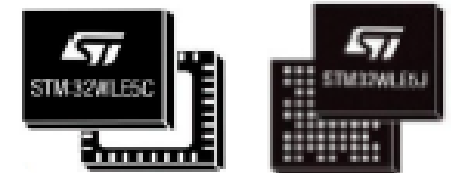
Configurar el entorno de Arduino

- Documentación
 - <https://docs.rakwireless.com/Product-Categories/WisDuo/RAK3172-Module/Quickstart/#rak3172-as-a-stand-alone-device-using-rui3>
- Instalar Arduino IDE desde esta dirección:
 - <https://www.arduino.cc/en/software>
- Agregar el soporte para el microcontrolador RAK3172
 - https://raw.githubusercontent.com/RAKWireless/RAKwireless-Arduino-BSP-Index/main/package_rakwireless.com_rui_index.json
 - Herramientas>Placa>Gestor de tarjetas → RAK Wireless RUI STM32

STM32WLE

KEY FEATURES

- Arm® Cortex®-M4 & DSP up to 48 MHz
- Up to 256 KB Flash and 64 KB SRAM
- **Sub-GHz Radio**
 - Multi-modulation: LoRa, (G)FSK, (G)MSK, BPSK
 - 2 embedded power amplifiers:
 - 1 output up to +15 dBm
 - 1 output up to +22 dBm
 - LoRa RX sensitivity: -148 dBm (SF12, BW=10.4kHz)
 - RX: 4.82mA and TX: 15mA (at 10dBm) / 87mA (at 20dBm) [3.3V]
- **Ultra-Low Power consumption**
 - < 71µA/MHz Active mode (3V - RF OFF)
 - 1 µA Stop2 mode with RAM retention
 - 390 nA Standby mode with RTC
 - 31 nA Shutdown mode
- **Peripherals**
 - 3xI²C, 2xUSART, 1xLP-UART, 2xSPI
 - 7x timers + 2x ULP Comparators
- 1.8 to 3.6V voltage range (DC/DC, LDO)
- -40 to up to +105°C temperature range

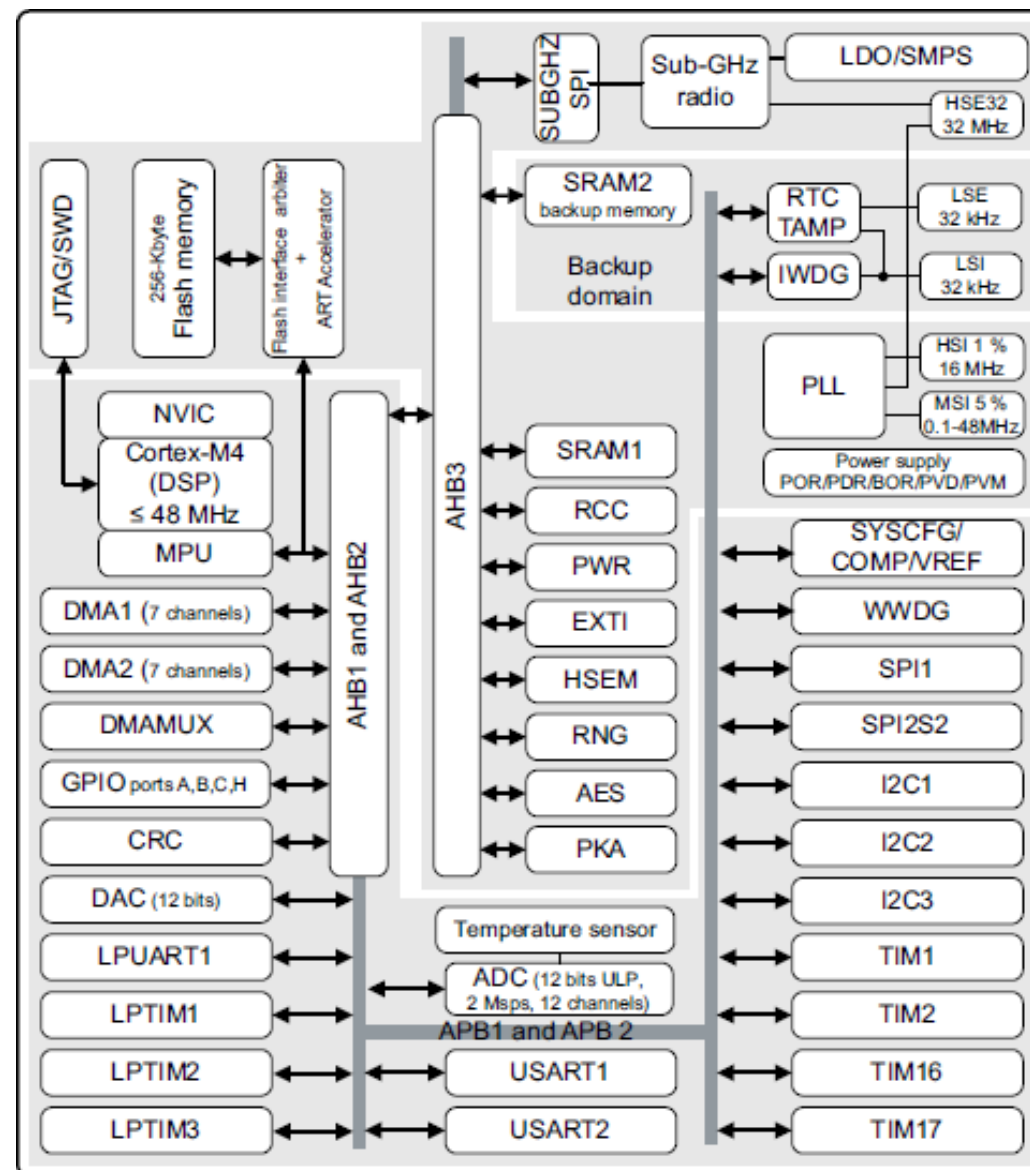


-> Packages: QFN48, BGA73

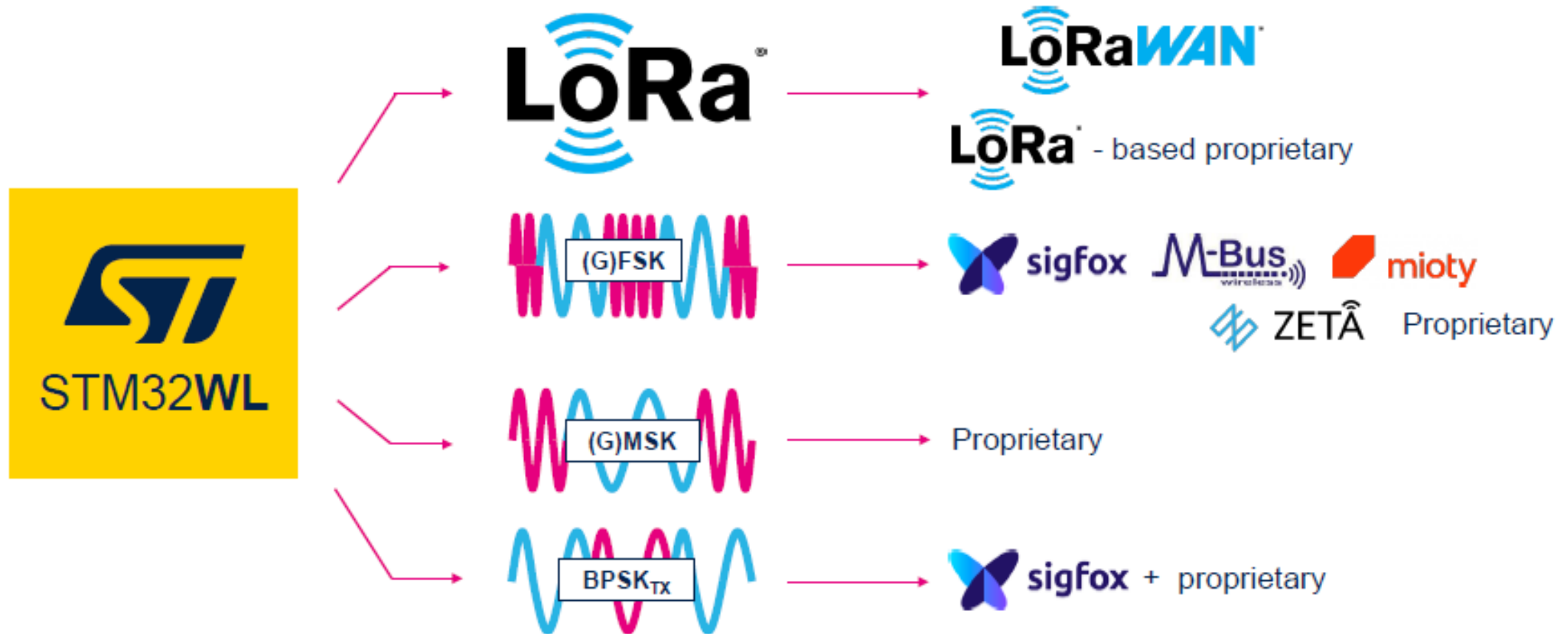


STM32WLE5CCU6

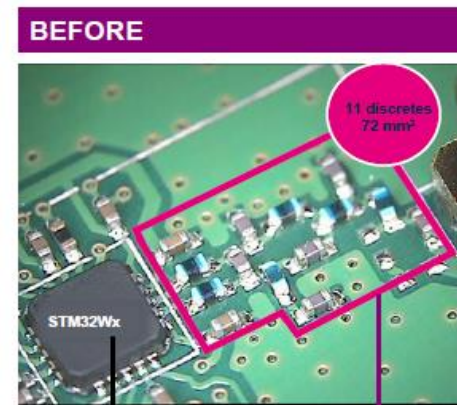
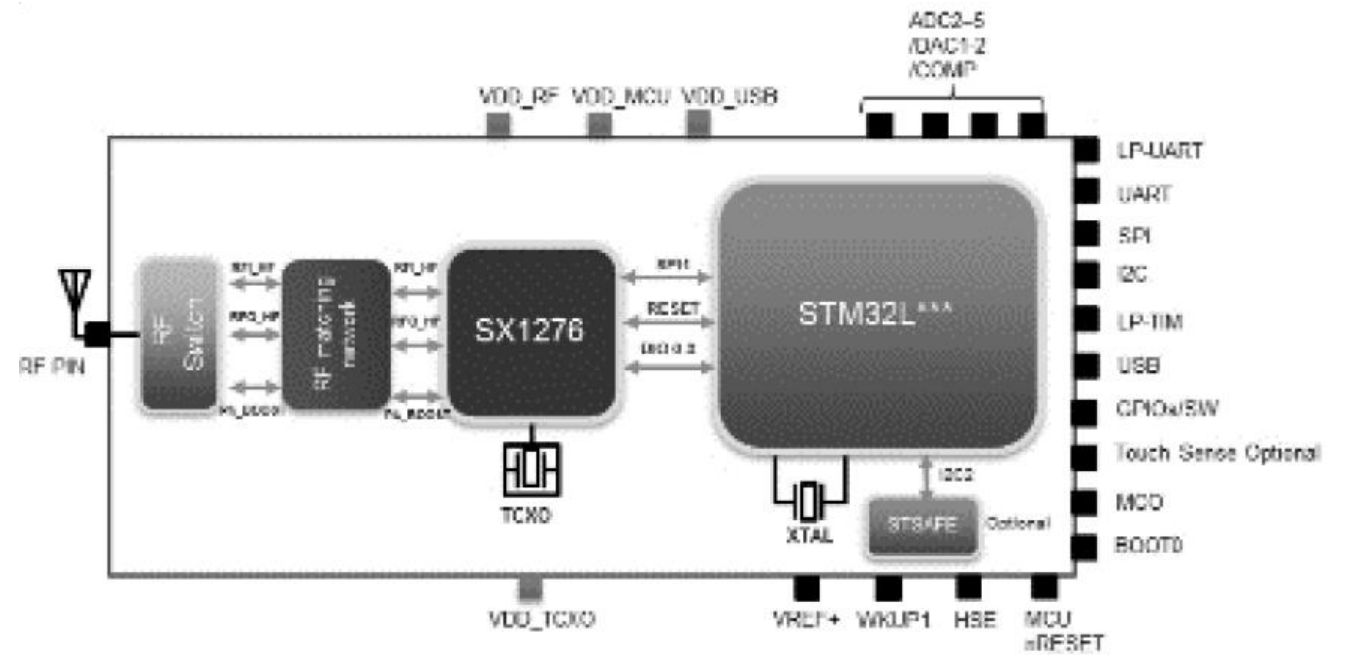
Cortex M4 + SX1262



LoRa + Sigfox

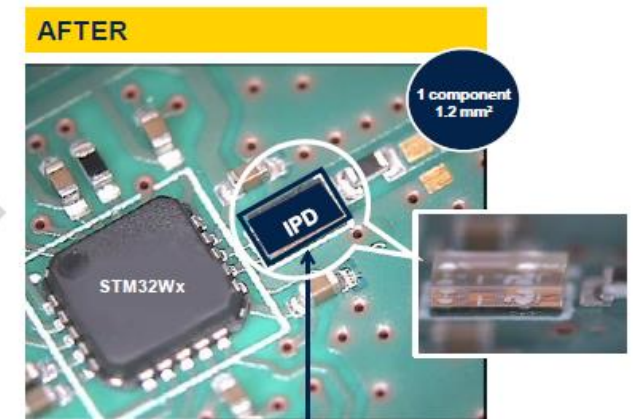


Configuración aproximada



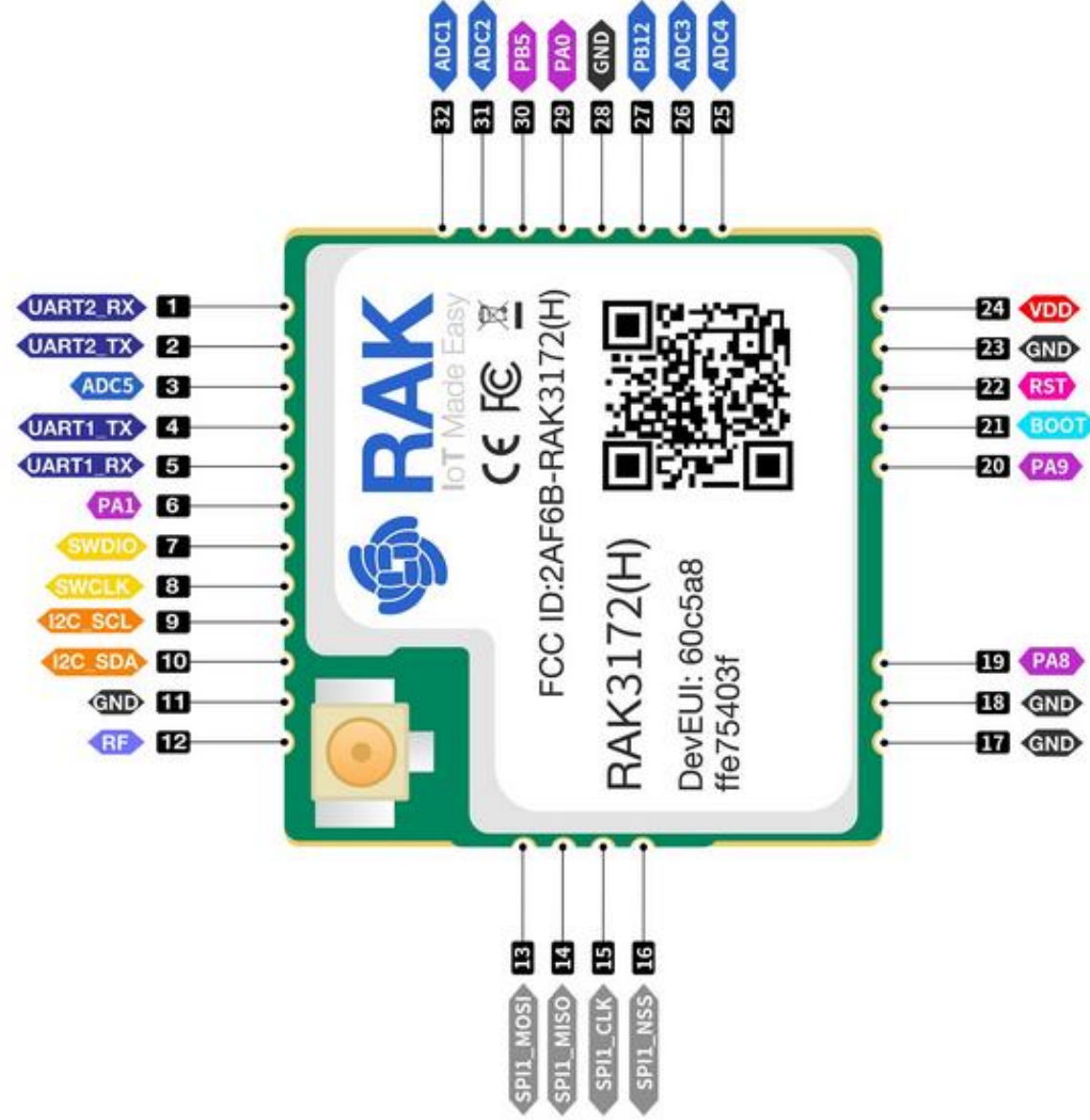
Wireless MCU

Discrete balun & matching network



RF IPD balun & matching network

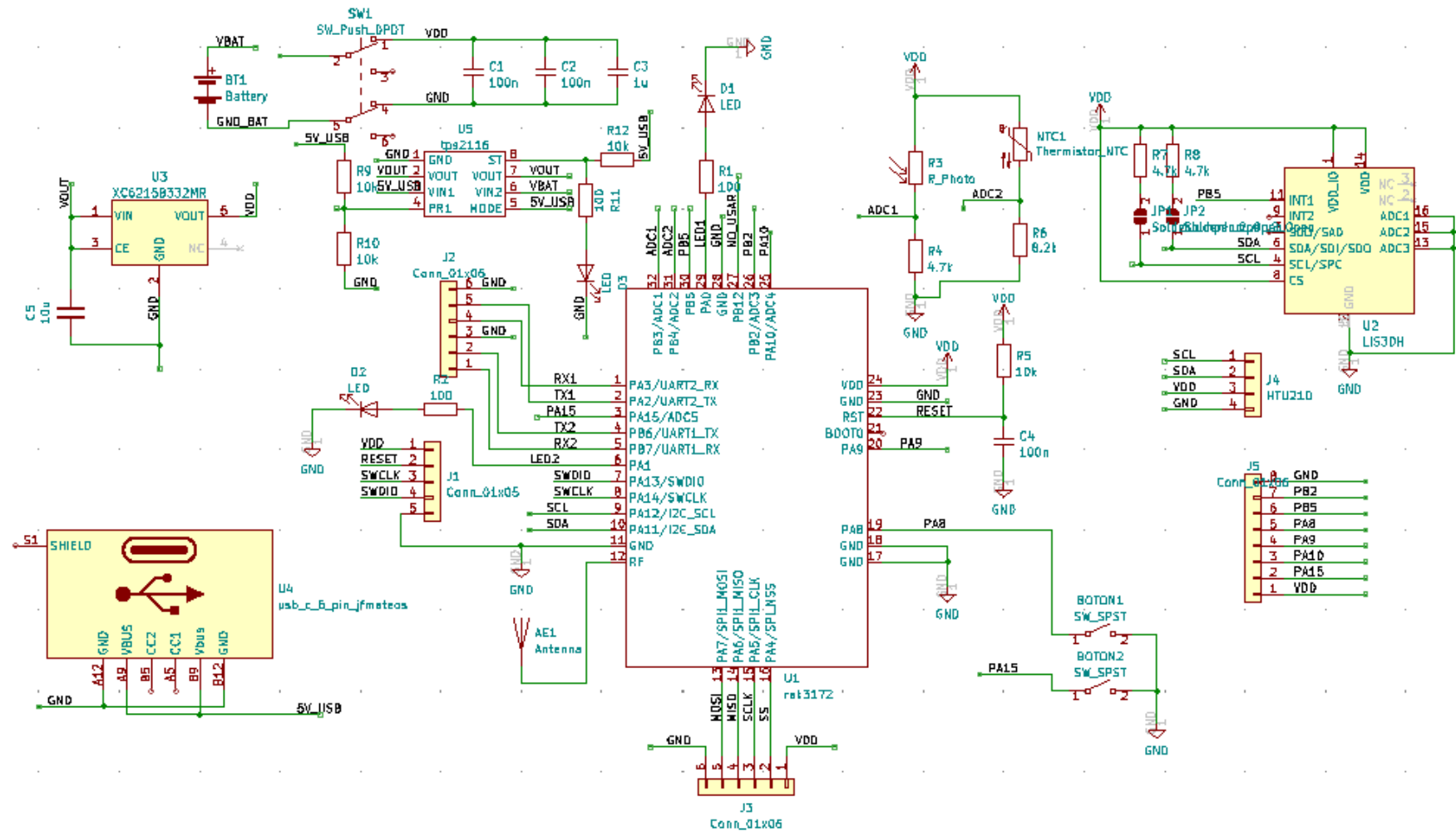
RAK3172



RAK3172: No usar estos pines

Pin name	Pin Usage
PA2	UART2_TX
PA3	UART2_RX
PA13	SWDIO
PA14	SWCLK
PB8	RAK3172 Internal
PB12	Internal 10k pull-up resistor for RAK3172 high frequency variant (8xx - 9xx Mhz) or pull-down resistor for RAK3172 low frequency variant (4xx Mhz)

Nuestra placa



RUI3: USARTs

- El RAK3172 lo programaremos por los pines TX1 y RX1 de nuestra placa, que son los pines PA2 y PA3, denominados UART2 en la hoja de datos del STM32WLE5.
- Tiene un modo de comandos AT
 - <https://docs.rakwireless.com/Product-Categories/WisDuo/RAK3172-Module/AT-Command-Manual/>
 - Instalar Realterm
 - Configurar la velocidad a 115200
 - Enviar el comando AT? para obtener un listado de todos los comandos disponibles
 - Enviar el comando AT+BOOTVER=? para ver la versión

UART

There are two UART peripherals available on the RAK3172 module. There are also different [Serial Operating Modes](#) possible in RUI3, namely [Binary Mode](#) and [Custom Mode](#) .

Serial Port	Serial Instance Assignment	Default Mode
UART1 (pins 4, 5)	Serial1	Custom Mode
UART2 (pins 1, 2)	Serial	AT Command

Conexión al ordenador



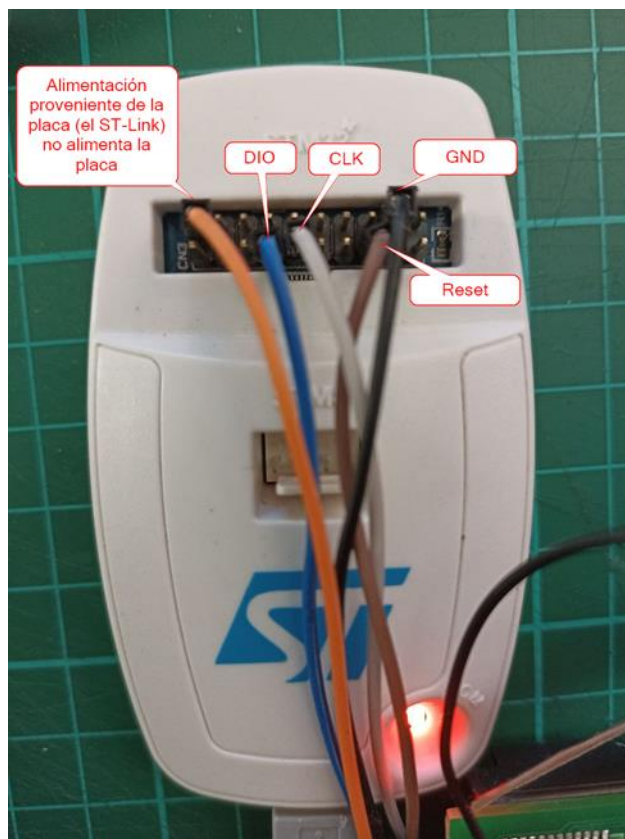
Algunos puertos USB de ordenadores no pueden suministrar la corriente que requiere el RAK3172, por lo que puede ser necesario alimentar la placa a través del puerto USB tipo C desde un cargador de móvil (por ejemplo). En este caso, el cable 3v3 del adaptador Serial-USB no se conectará a la placa (pero sí el GND).

Recuperar/Actualizar el Bootloader 1/4

- Por diversas circunstancias puede ser necesario recuperar el bootloader, para lo que necesitaremos un programador ST-LINK
- **Bootloader de RAK:** Descargar el **archivo HEX** de la versión apropiada del bootloader (hay versiones para RAK3172 sin TCXO, y con TCXO que usan el sufijo -T):
 - <https://docs.rakwireless.com/Product-Categories/WisDuo/RAK3172-Module/Datasheet/#firmware>
 - La versión con TCXO tiene la marca (TI) junto al código QR de la pegatina, mientras que la que no tiene TCXO utiliza simplemente la marca (I)
- Descargar STM32CubeProgrammer:
 - <https://www.st.com/en/development-tools/stm32cubeprog.html#get-software>

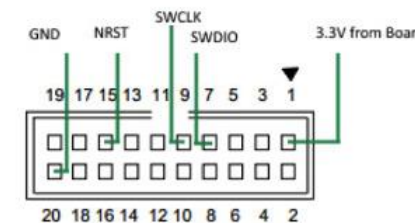
Recuperar/Actualizar el Bootloader 2/4

Conexionado del ST-LINK original



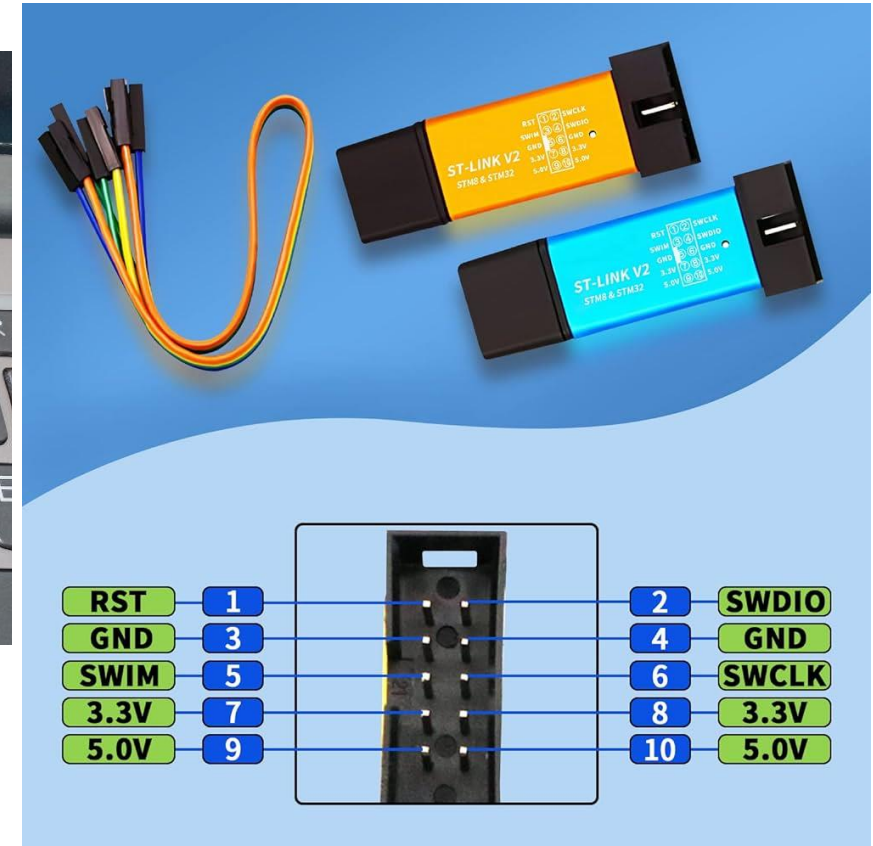
Para evitar dañar el STM32WL es importante que la antenna esté conectada.

La placa debe tener su propia alimentación (no la obtiene del ST-LINK).



Recuperar/Actualizar el Bootloader 3/4

Conexionado del ST-LINK clónico



Recuperar/Actualizar el Bootloader 4/4

Actualizar el firmware

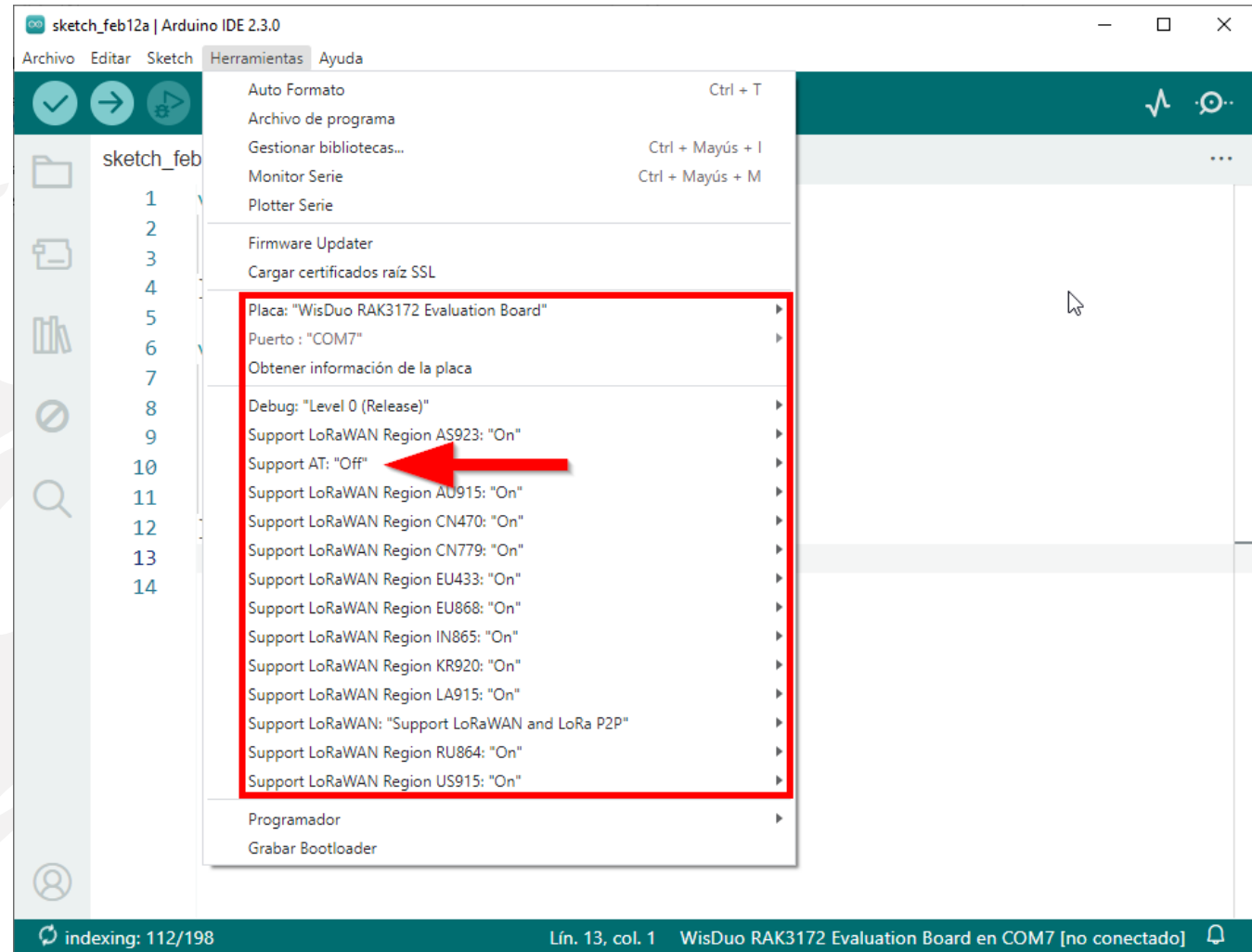
The screenshot displays the STM32CubeProgrammer software interface, specifically the 'Erasing & Programming' tab. The interface is divided into several sections:

- Left Panel:** Contains icons for various functions: Erase, Program, Verify, Run, CPU, SWV, REG, and a question mark.
- Main Area:**
 - File path:** Set to 'C:\Users\juanfe\Downloads\RAK3172-E_latest' (labeled 6).
 - Start addr...:** A text input field (labeled 5).
 - Options:**
 - ☐ Skip flash erase before programming
 - ☒ Verify programming
 - ☒ Run after programming
 - Buttons:** 'Start Programm...' (labeled 7) and 'Start automatic mode'.
 - Automatic Mode:**
 - ☐ Full chip erase
 - ☒ Download file
 - ☐ Option bytes commands: '-ob'
 - Table:** A table for erasing memory sectors with columns 'Select', 'Index', 'Start Address', and 'Size'. It lists 9 sectors (Index 0 to 8) with a size of 2K each.
 - Log:** A window at the bottom showing the progress of the download operation, including timestamps and status messages like 'Download verified successfully' and 'Application is running, Please Hold on...'.
- Right Panel:**
 - ST-LINK configuration:** Shows the current configuration for the ST-LINK, including Serial number, Port, Frequency, Mode, Access port, Reset mode, and Shared status. The 'Firmware version V2J38S7' is highlighted with a red box (labeled 1).
 - Buttons:** 'Disconnect' (labeled 4) and 'Firmware upgrade'.
 - Device information:** A section at the bottom right showing details about the target device, including STM32WLxx, MCU, Device ID, Revision ID, Flash size, CPU, and Bootloader Version.

A red callout bubble with the text 'Pueder ser necesario actualizar el firmware' points to the highlighted firmware version.

Blink LED

```
void setup() {  
    // put your setup code here, to run once:  
    pinMode(PA0,OUTPUT);  
}  
  
void loop() {  
    // put your main code here, to run repeatedly:  
    digitalWrite(PA0,HIGH);  
    delay(1000);  
    digitalWrite(PA0,LOW);  
    delay(1000);  
}
```



LED y BOTON

```
void setup() {  
    pinMode(PA0,OUTPUT);  
    pinMode(PA8,INPUT_PULLUP);  
}  
void loop() {  
    digitalWrite(PA0,!digitalRead(PA8));  
}
```

ALTERNAR LED CON BOTON: Variables

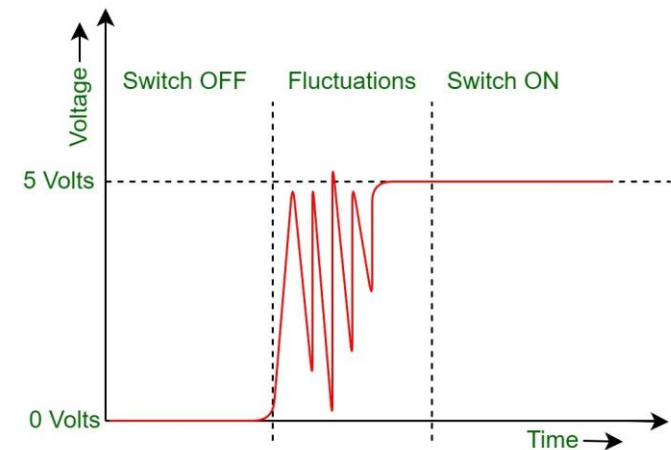
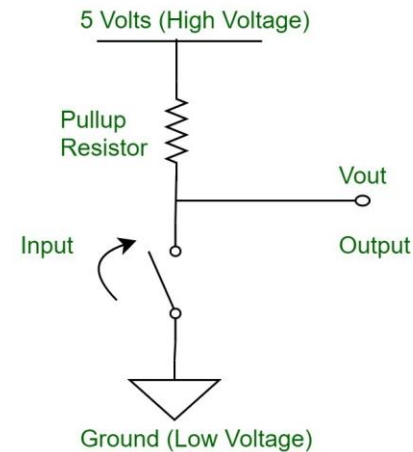
```
bool led_encendido=false;
void setup() {
  pinMode(PA0,OUTPUT);
  pinMode(PA8,INPUT_PULLUP);
}
void loop() {
  if(digitalRead(PA8)==LOW){
    led_encendido=!led_encendido;
    delay(100);
  }
  digitalWrite(PA0,led_encendido);
}
```

Data Types

array
bool
boolean
byte
char
double
float
int
long
short
size_t
string
String()
unsigned char
unsigned int
unsigned long
void
word

ALTERNAR LED CON BOTON: Debounce básico

```
bool led_encendido = false;
void setup() {
  pinMode(PA0, OUTPUT);
  pinMode(PA8, INPUT_PULLUP);
}
void loop() {
  if (digitalRead(PA8) == LOW) {
    led_encendido = !led_encendido;
    digitalWrite(PA0, led_encendido);
    while (digitalRead(PA8) == LOW) {
      delay(100);
    }
  }
}
```



PWM: analogWrite()

```
byte duty_cycle=0;
int incremento=1;
void setup() {
}
void loop() {
  if(duty_cycle==255){
    incremento=-5;
  }else if(duty_cycle==0){
    incremento=5;
  }
  duty_cycle+=incremento;
  analogWrite(PA0,duty_cycle);
  delay(10);
}
```

El duty cycle tenemos que expresarlo entre 0 y 255

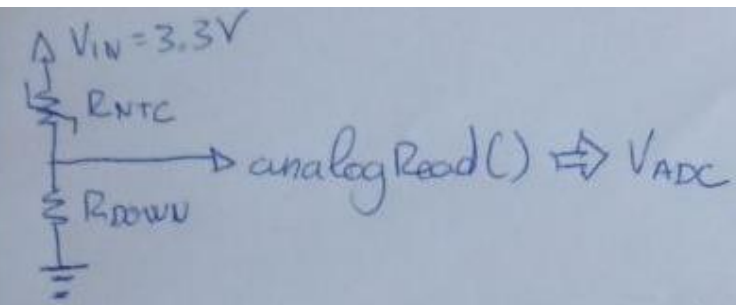
LDR y SERIAL

```
void setup() {  
  Serial.begin(115200);  
}  
void loop() {  
  Serial.println(analogRead(PB3));  
}
```

Documentación de la API RUI3 para Arduino:

- <https://docs.rakwireless.com/RUI3/Arduino-API>

NTC y SERIAL (1/2)



$$V_{ADC} = V_{IN} \frac{R_{DOWN}}{R_{DOWN} + R_{NTC}} \Rightarrow R_{NTC} = \left(\frac{V_{IN}}{V_{ADC}} - 1 \right) R_{DOWN}$$

$$\frac{V}{V_{IN}} \xrightarrow{\text{ADC}} 4095 \Rightarrow V_{ADC} = \frac{V_{IN} \cdot \text{analogRead}}{4095}$$

$$V_{ADC} \rightarrow \text{analogRead}()$$

$$R_{NTC} = \left(\frac{4095}{\text{analogRead}()} - 1 \right) \cdot R_{DOWN}$$

$$\frac{1}{T} = \frac{1}{T_{25^{\circ}\text{C}}} + \frac{1}{\beta} \ln \left(\frac{R_{NTC}}{R_{25^{\circ}\text{C}}} \right)$$

$$T[K] = \frac{1}{\frac{1}{273'15 + 25} + \frac{1}{3450} \ln \left[\frac{\left(\frac{4095}{\text{analogRead}()} - 1 \right) \cdot 820}{10000} \right]}$$

$$T[^{\circ}\text{C}] = T[K] - 273'15$$

NTC y SERIAL (2/2)

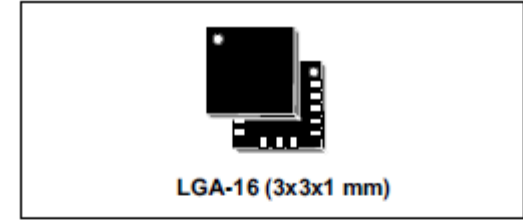
```
void setup() {  
  Serial.begin(115200);  
  analogReadResolution(12);  
}  
void loop() {  
  const float BETA = 3950; // Coeficiente beta de la NTC; su R25 es 10k  
  float temperatura = (1. / ((1./((25+273.15))+((1./3950.)*log(((4095./analogRead(PB4))-  
1)*8200./10000.))))) - 273.15;  
  Serial.print("Temperatura= ");  
  Serial.println(temperatura);  
  delay(1000);  
}
```

La resolución predeterminada del ADC del RAK3172 en RUI3 es 10bits, pero se puede cambiar con `analogReadResolution(12)`

Acelerómetro LIS3DH

```
#include <SparkFunLIS3DH.h>
#include <Wire.h>
LIS3DH acelerometro( I2C_MODE, 0x19 );
void setup() {
  acelerometro.settings.accelSampleRate = 200;
  //Hz. Can be:
  0,1,10,25,50,100,200,400,1600,5000 Hz
  acelerometro.settings.accelRange = 2;    //Max
  G force readable. Can be: 2, 4, 8, 16
  Serial.begin(115200);
  while (!Serial);
  if ( acelerometro.begin() != 0 ) {
    Serial.println("Error iniciando el acelerómetro
    en 0x19.");
  } else {
    Serial.println("Acelerómetro detectado en
    0x19.");
  }
}
```

```
void loop() {
  Serial.print(acelerometro.readFloatAccelX(), 4);
  Serial.print(",");
  Serial.print(acelerometro.readFloatAccelY(), 4);
  Serial.print(",");
  Serial.println(acelerometro.readFloatAccelZ(), 4);
  delay(10);
}
```

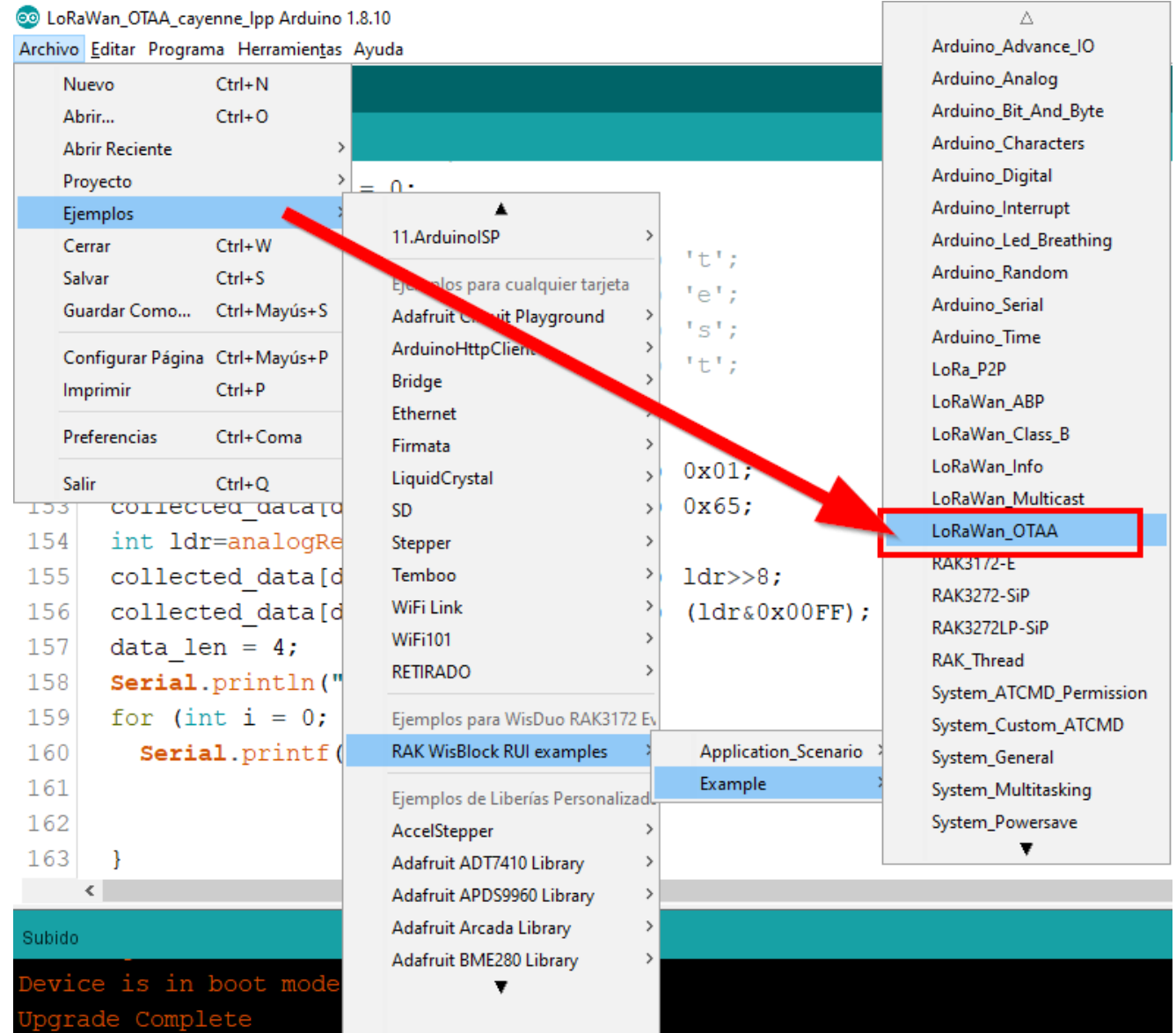


Features

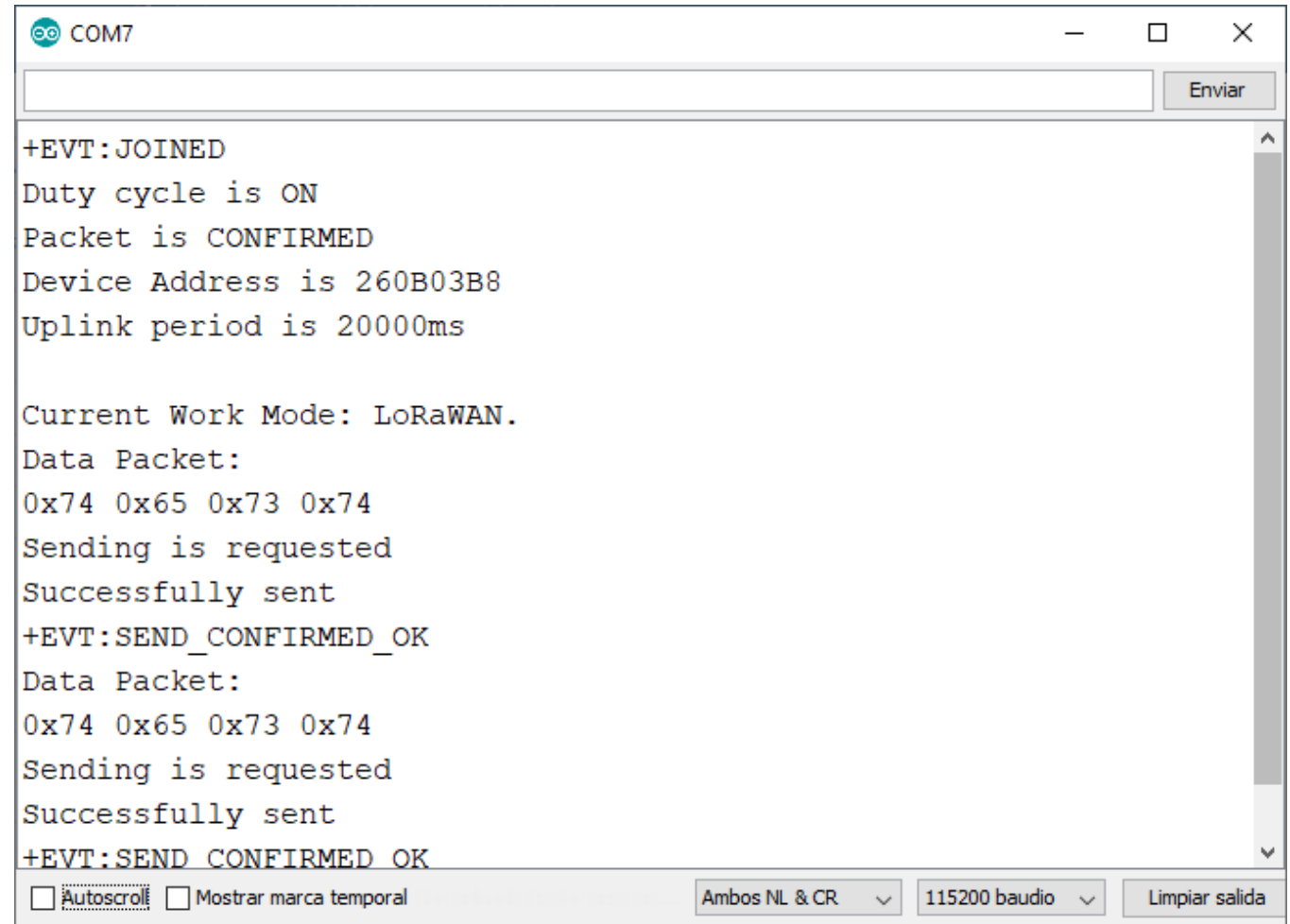
- Wide supply voltage, 1.71 V to 3.6 V
- Independent IO supply (1.8 V) and supply voltage compatible
- Ultra-low-power mode consumption down to 2 μ A
- $\pm 2g/\pm 4g/\pm 8g/\pm 16g$ dynamically selectable full scale
- I²C/SPI digital output interface
- 16-bit data output
- 2 independent programmable interrupt generators for free-fall and motion detection
- 6D/4D orientation detection
- Free-fall detection
- Motion detection
- Embedded temperature sensor
- Embedded self-test
- Embedded 32 levels of 16-bit data output FIFO
- 10000 g high shock survivability
- ECOPACK[®], RoHS and "Green" compliant

LoRaWAN

OTAA 1/2



LoRaWAN OTAA 2/2



The screenshot shows a serial terminal window titled "COM7" with a standard Windows-style title bar (minimize, maximize, close buttons). Below the title bar is a text input field and an "Enviar" button. The main area of the window displays the following log output in a monospaced font:

```
+EVT:JOINED
Duty cycle is ON
Packet is CONFIRMED
Device Address is 260B03B8
Uplink period is 20000ms

Current Work Mode: LoRaWAN.
Data Packet:
0x74 0x65 0x73 0x74
Sending is requested
Successfully sent
+EVT:SEND_CONFIRMED_OK
Data Packet:
0x74 0x65 0x73 0x74
Sending is requested
Successfully sent
+EVT:SEND_CONFIRMED_OK
```

At the bottom of the window, there is a status bar containing several controls: a checkbox for "Autoscroll", a checkbox for "Mostrar marca temporal", a dropdown menu set to "Ambos NL & CR", a dropdown menu set to "115200 baudio", and a button labeled "Limpiar salida".

Probar el envío de un downlink

The screenshot displays the 'Messaging' tab in a LoRaWAN management interface. On the left, the 'Downlink' section is active, showing a 'Schedule downlink' form. The 'Insert Mode' is set to 'Replace downlink queue', 'FPort' is 1, 'Payload type' is 'Bytes', and the 'Payload' is '01 02 03'. A red arrow points from the payload field to the terminal window. The terminal window, titled 'COM7', shows a sequence of events: 'Sending is requested', 'Successfully sent', '+EVT:SEND_CONFIRMED_OK', and 'Data Packet: 0x74 0x65 0x73 0x74'. This sequence is repeated three times. The final event, '+EVT:RX 1:-52:14:UNICAST:1:010203', is followed by 'Something received!', which is highlighted with a red box. The terminal also shows '123' at the bottom. At the bottom of the interface, a green status bar indicates 'Downlink scheduled'.

Overview Live data **Messaging** Location Payload formatters Claiming General settings

Uplink **Downlink**

Schedule downlink

Insert Mode

- ☒ Replace downlink queue
- ☐ Push to downlink queue (append)

FPort*

1

Payload type

- ☒ Bytes ☐ JSON

Payload

01 02 03

The desired payload bytes of

☐ Confirmed downlink

Schedule downlink

COM7

Enviar


Sending is requested
Successfully sent
+EVT:SEND_CONFIRMED_OK
Data Packet:
0x74 0x65 0x73 0x74
Sending is requested
Successfully sent
+EVT:SEND_CONFIRMED_OK
Data Packet:
0x74 0x65 0x73 0x74
Sending is requested
Successfully sent
+EVT:SEND_CONFIRMED_OK
+EVT:RX 1:-52:14:UNICAST:1:010203
Something received!
123

☒ Autoscroll ☐ Mostrar marca temporal Ambos NL & CR 115200 baudio Limpiar salida

Downlink scheduled

LoRaWAN + Cayenne LPP

```
143  /** Payload of Uplink */
144  uint8_t data_len = 0;
145  /*
146  collected_data[data_len++] = (uint8_t) 't';
147  collected_data[data_len++] = (uint8_t) 'e';
148  collected_data[data_len++] = (uint8_t) 's';
149  collected_data[data_len++] = (uint8_t) 't';
150  */
151
152  collected_data[data_len++] = (uint8_t) 0x01;
153  collected_data[data_len++] = (uint8_t) 0x65;
154  int ldr=analogRead(PB3);
155  Serial.print("Luminosidad= ");
156  Serial.println(ldr);
157  collected_data[data_len++] = (uint8_t) (ldr>>8);
158  collected_data[data_len++] = (uint8_t) (ldr&0x00FF);
159  data_len = 4;
160  Serial.println("Data Packet:");
161  for (int i = 0; i < data_len; i++) {
162      Serial.printf("0x%02X ", (uint8_t)collected_data[i]);
163  }
164  Serial.println("");
```



LoRaWAN: Envío por interrupción

- Ponemos al final del setup esta instrucción
 - `api.system.sleep.setup(RUI_WAKE_UP_FALLING_EDGE,PA8);`
- Y en el loop lo dormimos sin timeout
 - `api.system.sleep.all();`

```
180 void loop()
181 {
182     static uint64_t last = 0;
183     static uint64_t elapsed;
184
185     if ((elapsed = millis() - last) > OTAA_PERIOD) {
186         uplink_routine();
187
188         last = millis();
189     }
190     Serial.printf("Try sleep %ums..", OTAA_PERIOD);
191     api.system.sleep.all();
192     Serial.println("Wakeup..");
193 }
```

LoRaWAN: Cuestiones adicionales

- Clase C
 - `api.lorawan.deviceClass.set(RA_K_LORA_CLASS_C)`
- Data Rate en ABP
 - `api.lorawan.adr.set(false)`
 - `api.lorawan.dr.set(0 a 5)`
//Recordar que DR0 es SF12

Required SNR	
DR/SF	SNR (dB)
DR0/SF12	-20
DR1/SF11	-17.5
DR2/SF10	-15
DR3/SF09	-12.5
DR4/SF08	-10
DR5/SF07	-7.5

Programación con STM32Duino

<https://github.com/stm32duino>

Cuestiones generales 1/2

- Para programar desde Arduino utilizando el puerto serie tendremos que utilizar el bootloader de fábrica del sistema (no el de RUI3). Para activar este bootloader tendremos que mantener el pin BOOT0 en alto mientras reseteamos el RAK3172. Desafortunadamente, el nodo TTNMAD EASY no tiene un botón para realizar esta operación, pero puede instalarse uno cómo se muestra en la imagen.
- Utilizando este método se borrará el bootloader de RAK RUI3, por lo que necesitaríamos un ST-LINK si posteriormente quisiéramos restituirlo.
- El gestor de placas STM32Duino se instala mediante la URL:
 - https://github.com/stm32duino/BoardManagerFiles/raw/main/package_stmicroelectronics_index.json



Manteniendo este botón pulsado mientras pulsamos el botón Reset conseguiremos el el uC arranque con el bootloader de fábrica.

Cuestiones generales 2/2

- Es necesario utilizar las últimas versiones del gestor de placas STM32Duino y de la librería STM32LoRaWAN:
 - STM32 MCU Based Boards: Versión 2.7.1
 - STM32duinoLoRaWAN: Versión 0.2.0
- Si tenemos problemas al utilizar LoRaWAN, podemos probar a cambiar el valor que se muestra en la imagen.

```
126 // Ali
127
128 #ifndef DEBUG_SUBGHZSPI_MOSI
129 #define DEBUG_SUBGHZSPI_MOSI PA7_ALT1
130 #endif
131 #ifndef DEBUG_SUBGHZSPI_MISO
132 #define DEBUG_SUBGHZSPI_MISO PA6_ALT1
133 #endif
134 #ifndef DEBUG_SUBGHZSPI_SCLK
135 #define DEBUG_SUBGHZSPI_SCLK PA5_ALT1
136 #endif
137 #ifndef DEBUG_SUBGHZSPI_SS
138 #define DEBUG_SUBGHZSPI_SS PA4_ALT1
139 #endif
140
141 // Extra HAL modules
142 #if !defined(HAL_DAC_MODULE_DISABLED)
143 #define HAL_DAC_MODULE_ENABLED
144 #endif
145
146 // LoRaWAN definitions
147 #define LORAWAN_BOARD_HAS_TCXO 0U
148 #define LORAWAN_BOARD_HAS_DCDC 1U
149 #define LORAWAN_TX_CONFIG RBI_CONF_RFO_HP
150
151 #define LORAWAN_RFSWITCH_PINS PB8, PC13
152 #define LORAWAN_RFSWITCH_PIN_COUNT 2
153 #define LORAWAN_RFSWITCH_OFF_VALUES LOW, LOW
154 #define LORAWAN_RFSWITCH_RX_VALUES HIGH, LOW
155 #define LORAWAN_RFSWITCH_RFO_LP_VALUES LOW, HIGH
156 #define LORAWAN_RFSWITCH_RFO_HP_VALUES LOW, HIGH
157
158 /*-----
159 * Arduino objects - C++ only
160 *-----*/
161
162 #ifndef __cplusplus
163 // These serial port names are intended to allow libraries and architecture-neutral
164 // sketches to automatically default to the correct port name for a particular type
```

Configuración de Arduino IDE

