

EBYTE 32 LoRa - STM32

Referencia para el uso con Arduino y ESP32 (Primeras pruebas)

https://www.youtube.com/watch?v=MHJXhq-yjCU&list=PLVTu9htVCVMQr2JeT-GUF3vCHM_EkuiDb5&index=2

Uso de I2C para la visualización de los datos (No es necesario, se puede usar dos periféricos USART).

Interfaz de configuración rápida para el módulo E32.

Para esto es necesario el CP2102 o FT232, convertidor de USB a TTL UART.



Configuración y modos de funcionamiento:

<https://www.youtube.com/watch?v=dSyxrRqToHE&t=1328s>

Librería utilizada EBYTE (Arduino)

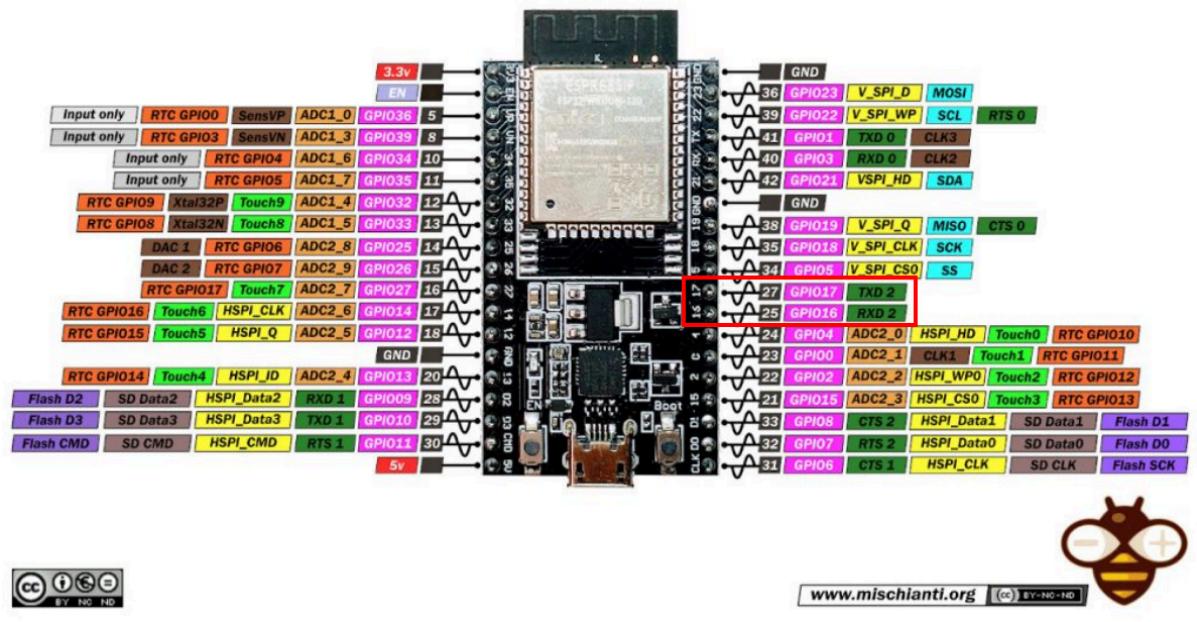
[GitHub - KrisKasprzak/EBYTE: Libraries to program and use UART-based EBYTE wireless data transceivers](https://github.com/KrisKasprzak/EBYTE-Libraries)

Instalación librerías externas Arduino

https://www.youtube.com/watch?v=64lql_u84c8

ESP32 DevKitC V4

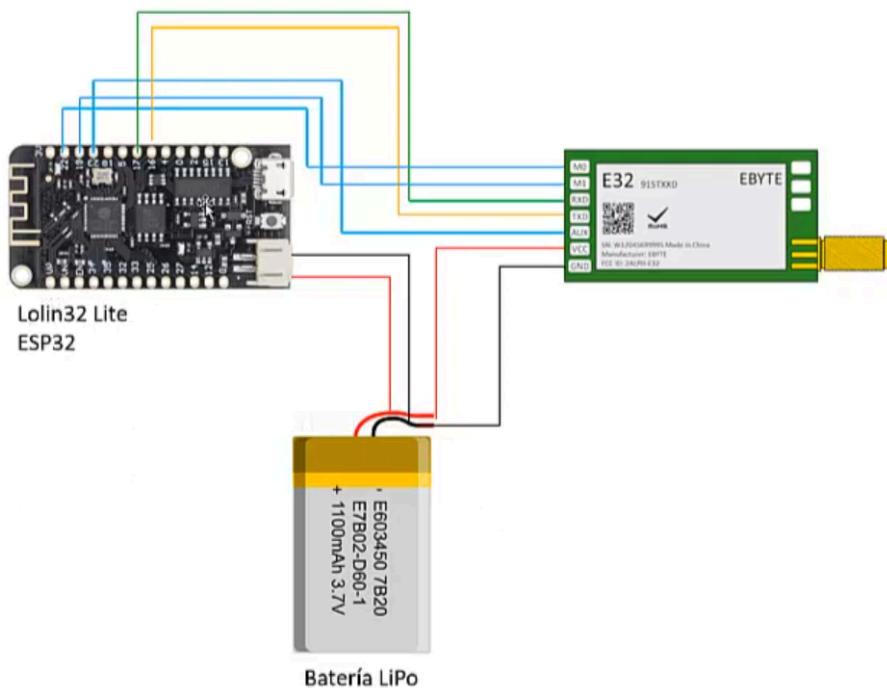
PINOUT

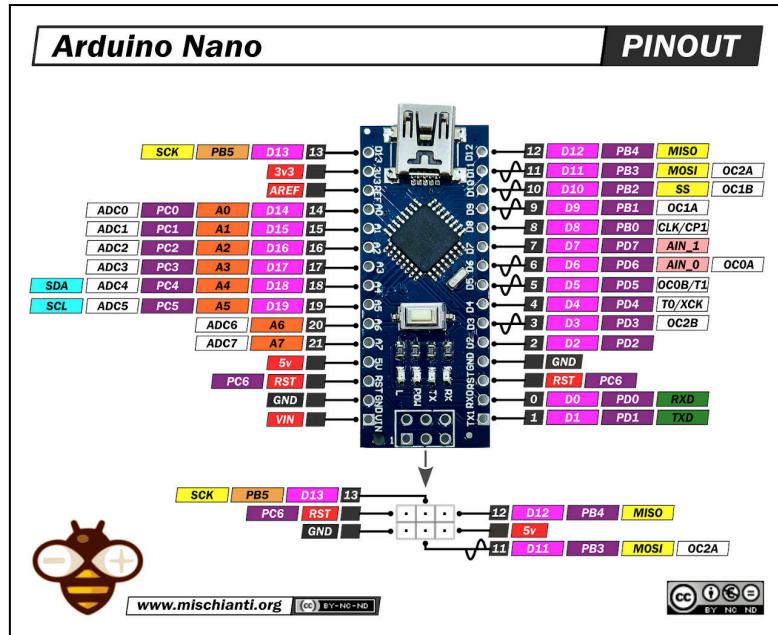


TX ARDUINO NANO (COM5, CH340)

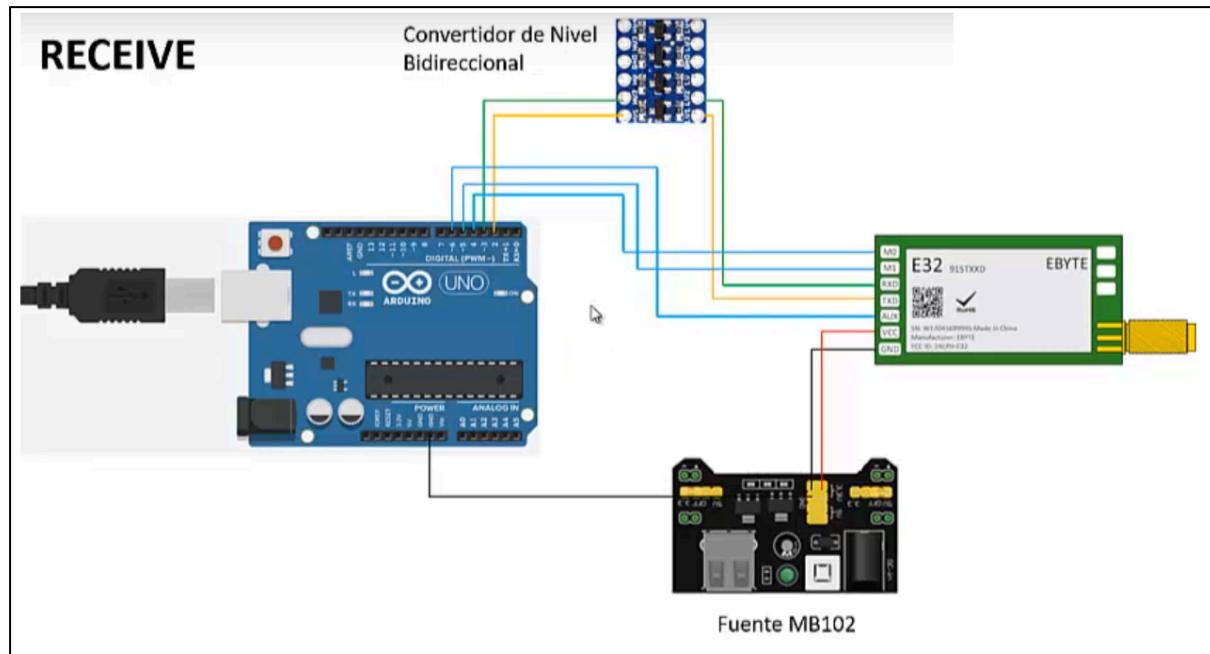
TX ESP32 (COM4, Silicon Labs)

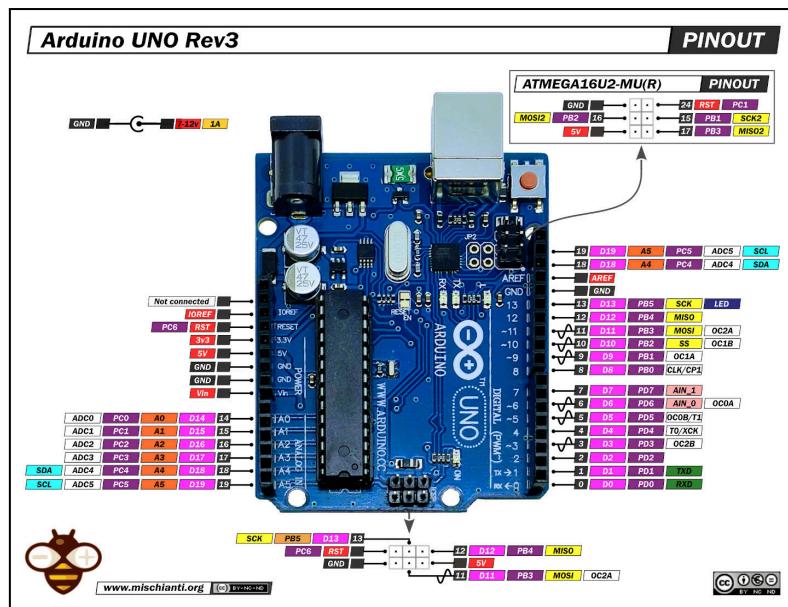
SEND



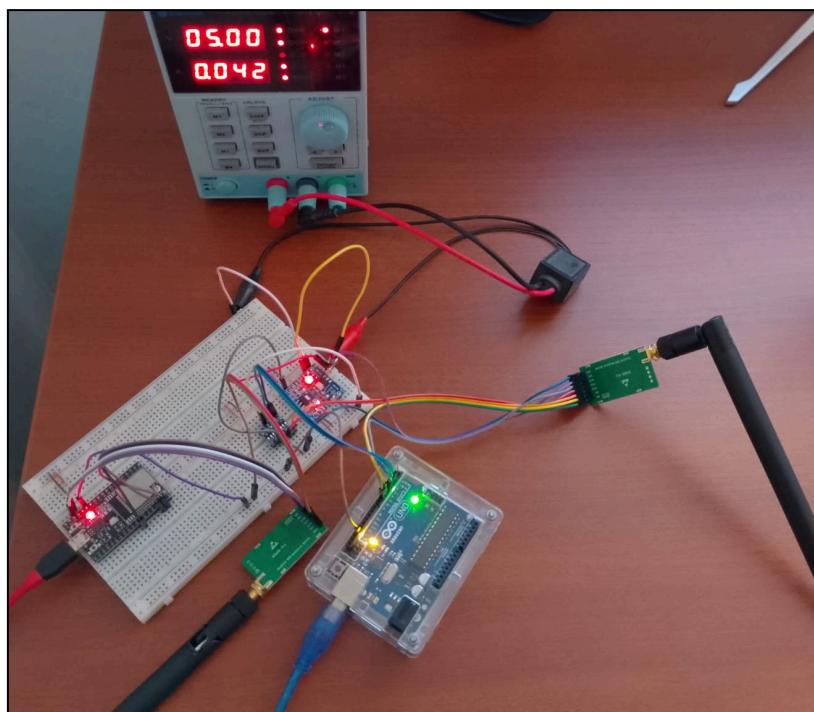


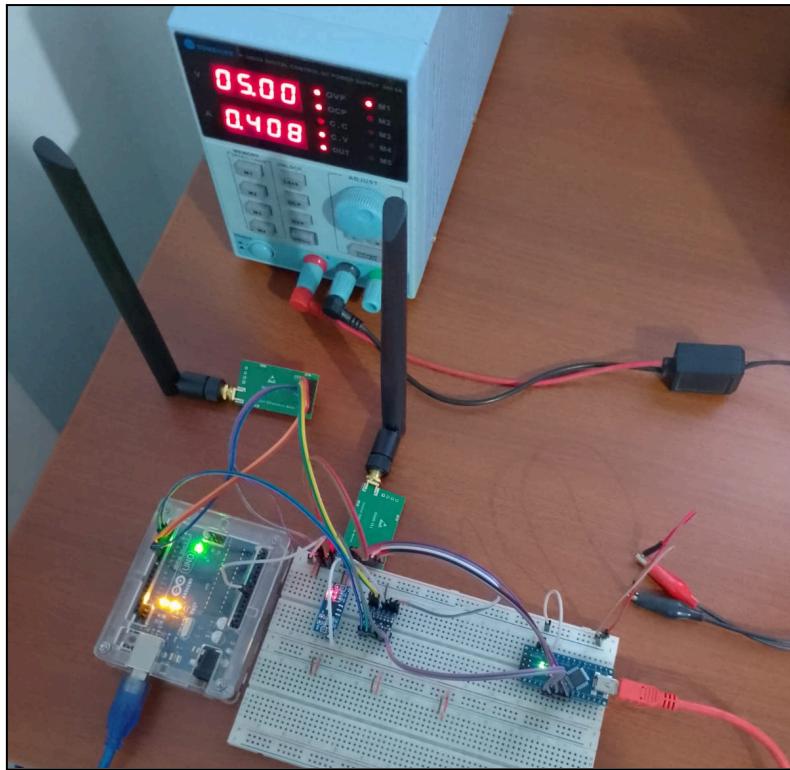
RX Arduino UNO (COM3)





Implementación





Corriente consumida por los dos módulos: 42 mA aprox (38mA de la última prueba con Arduinos). Para el envío de datos 0.356 A a 0.4 A aproximadamente.

Resultados

TX

```
Starting Sender
-----
Model no.: 32
Version : 10
Features : 1E

Mode (HEX/DEC/BIN): C0/192/11000000
AddH (HEX/DEC/BIN): 0/0/0
AddL (HEX/DEC/BIN): 0/0/0
Sped (HEX/DEC/BIN): 1A/26/11010
Chan (HEX/DEC/BIN): 17/23/10111
Optn (HEX/DEC/BIN): 44/68/1000100
Addr (HEX/DEC/BIN): 0/0/0
```

RX

```
Starting Reader
-----
Model no.: 32
Version : 10
Features : 1E

Mode (HEX/DEC/BIN): C0/192/11000000
AddH (HEX/DEC/BIN): 0/0/0
AddL (HEX/DEC/BIN): 0/0/0
Sped (HEX/DEC/BIN): 1A/26/11010
Chan (HEX/DEC/BIN): 17/23/10111
Optn (HEX/DEC/BIN): 44/68/1000100
Addr (HEX/DEC/BIN): 0/0/0
```

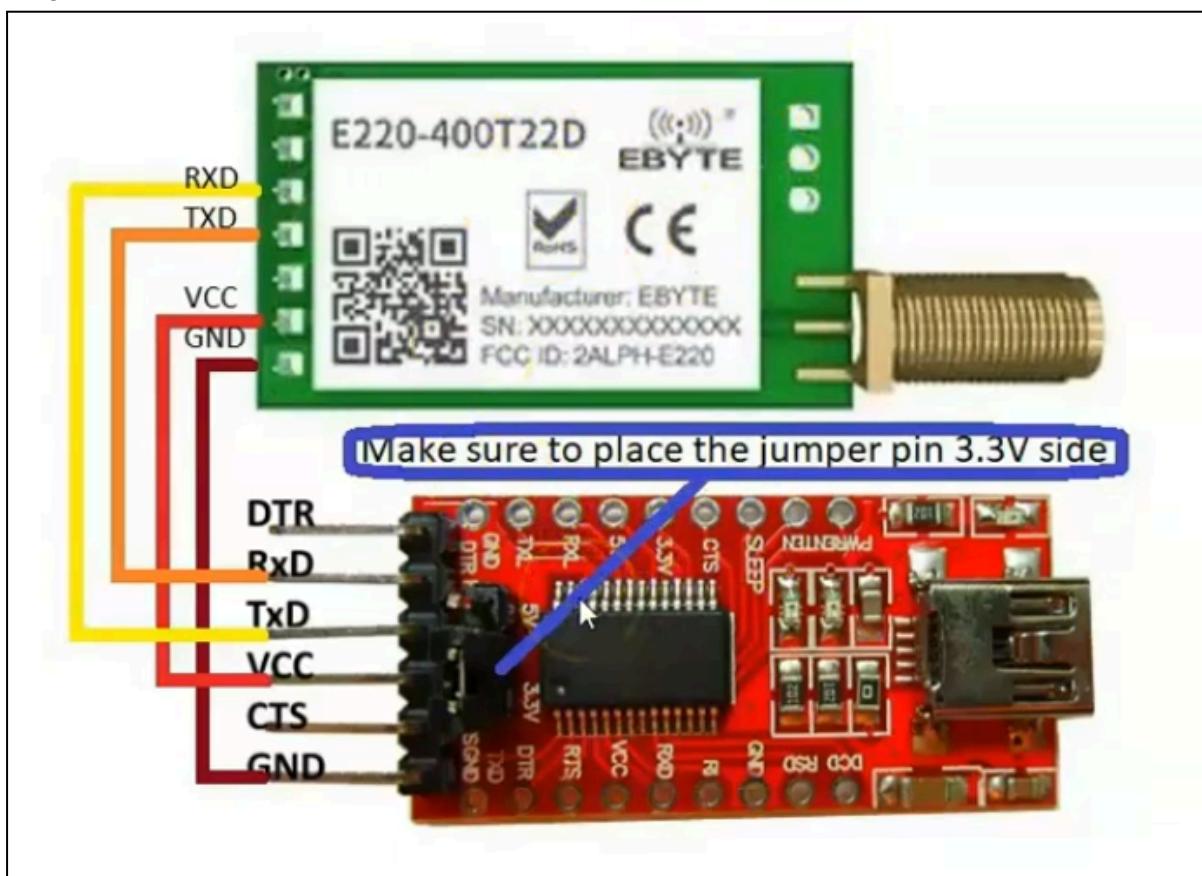
```

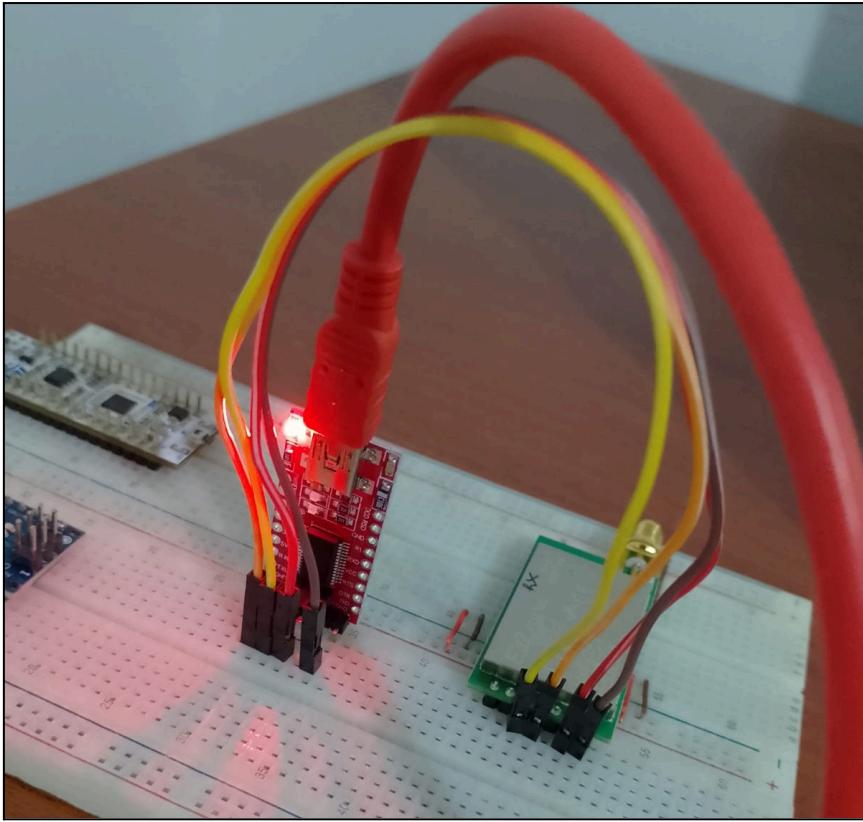
Send Arduino 1.8.19
Archivo Editar Programa Herramientas Ayuda
Send
// you can print all parameters and is good for debugging
// if your units will not communicate, print the parameters
// for both sender and receiver and make sure air rates, channel
// and address is the same
COMS
OptionFEC (HEX/DEC/BIN) : 1/1/1
OptionPower (HEX/DEC/BIN) : 0/0/0
-----
Sending: 1
Sending: 2
Sending: 3
Sending: 4
Sending: 5
Sending: 6
Sending: 7
Sending: 8
Sending: 9
Sending: 10
Sending: 11
Sending: 12
Count: 4
Searching: 5
Count: 5
Searching: 6
Count: 6
Searching: 7
Count: 7
Searching: 8
Count: 8
Searching: 9
Count: 9
Searching: 10
Count: 10
Searching: 11
Count: 11
Searching: 12
Count: 12
Autoscroll Mostrar marca temporal Nueva linea 9600 baudio Limpiar salida
// let the user know something was sent
Serial.print("Sending: "); Serial.println(MyData.Count);
delay(1000);
}

```

Configuración de parámetros del módulo LoRa

Diagrama de conexión





1. Conectamos solo el módulo FT232. Los Drivers se actualizaron automáticamente.
2. Realizamos la conexión anterior.
3. Conectamos nuevamente ahora con el módulo EBYTE.
4. Ejecutamos el programa de EBYTE.
5. Seleccionamos el puerto COM del dispositivo. Luego Open Port y Get Params.

Configuración para el módulo de TX



Configuración para el módulo de RX



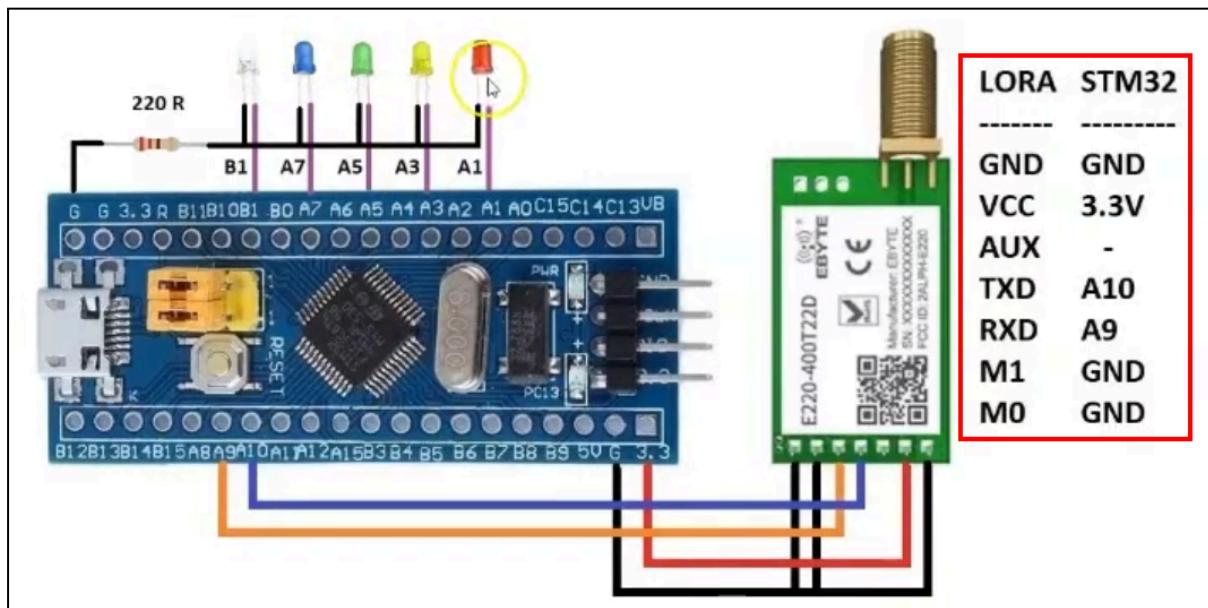
OBSERVACIONES

Menor tasa de datos **AirRate** mayor distancia de alcance. Mayor rendimiento ante interferencias y mayor tiempo de transmisión.

WOR Time es el tiempo de escucha después de inicializar el módulo.

El valor de **Channel** configura la frecuencia de transmisión (410 MHz + Channel*1 MHz).

Diagrama de referencia para la conexión del EBYTE con un NUCLEO STM32



Modo normal de operación

De los módulos EBYTE:

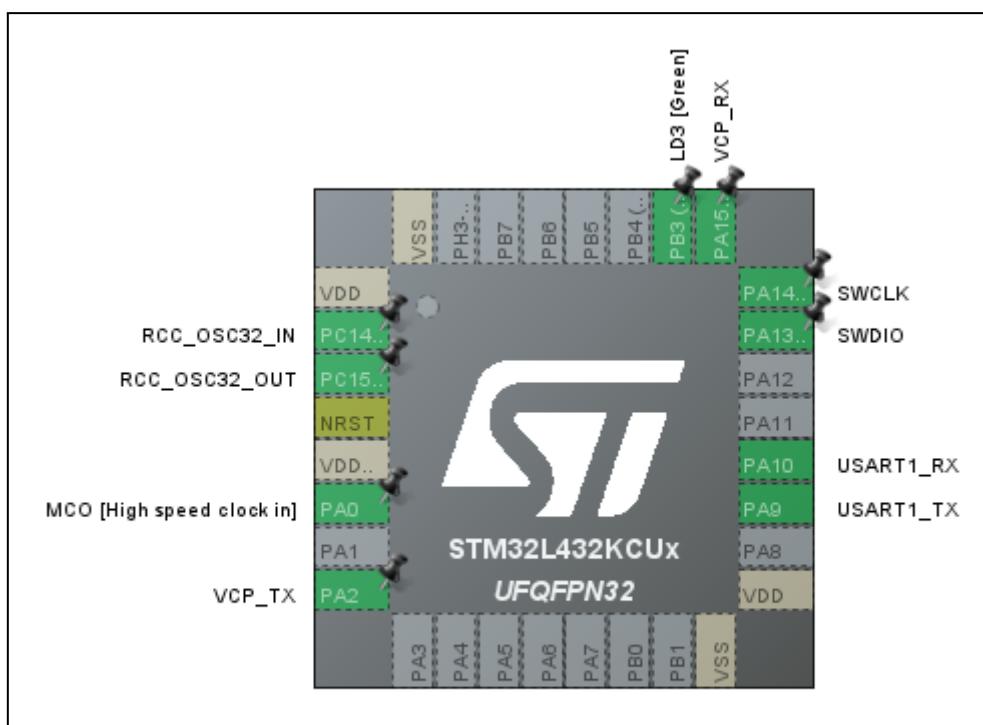
M0 GND

M1 GND

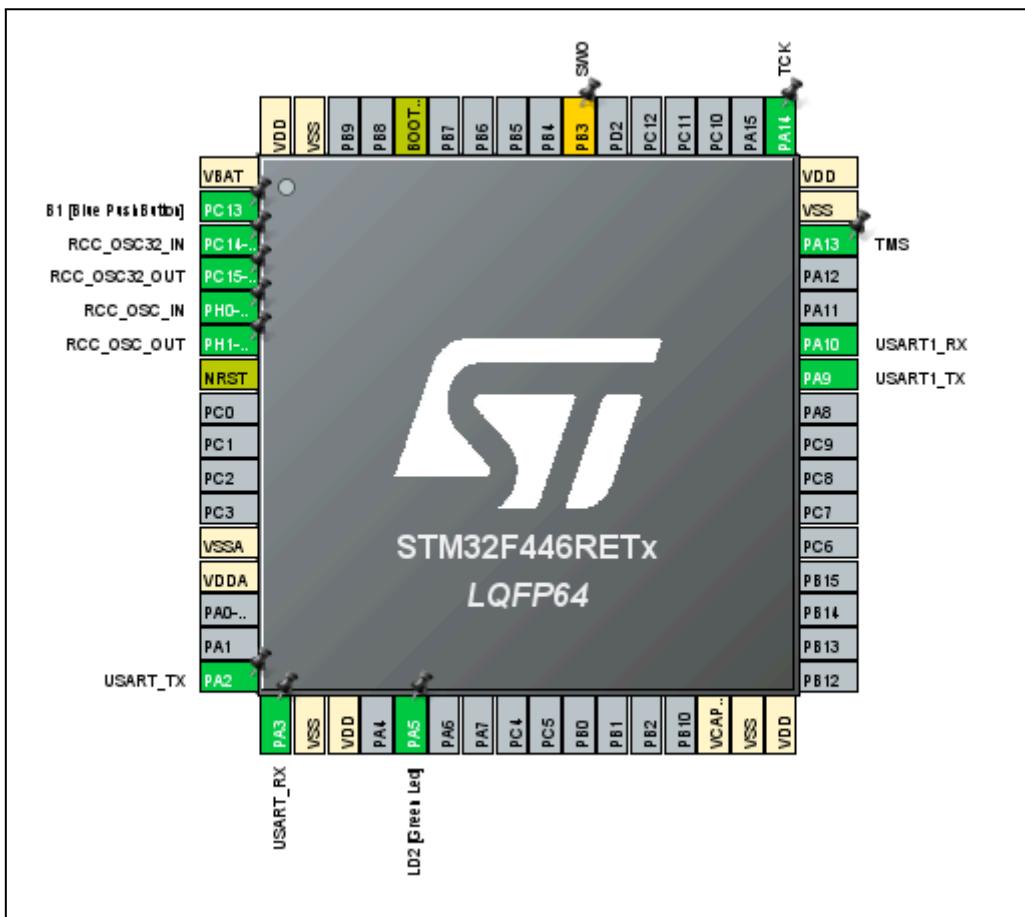
TX L432KC (COM7)

USART 1 TX PA9 D1 (Revisión de esquemático)

USART 1 RX PA10 D0 (Revisión de esquemático)



RX F446RE (COM8)
USART 1 TX PA9 D8
USART 1 RX PA10 D2



Documentación para el F446RE

<https://www.st.com/en/evaluation-tools/nucleo-f446re.html#cad-resources>

Algoritmo TX

```
/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1)
{
    if(count > 5){
        count = 0;
    }
    /* Review ASCII Code */
    charToTransmit[0] = count + 48;
    HAL_UART_Transmit(&huart1, charToTransmit, 1, 100);
    count++;

    HAL_Delay(500);

/* USER CODE END WHILE */
```

Algoritmo RX

```
/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1)
{
    HAL_UART_Receive(&huart1, receiveData, 1, 100);
    uint8_t c = receiveData[0];
    printf("Receive:\r\n");
    printf("%c", c);
    printf("\n");

    HAL_Delay(500);

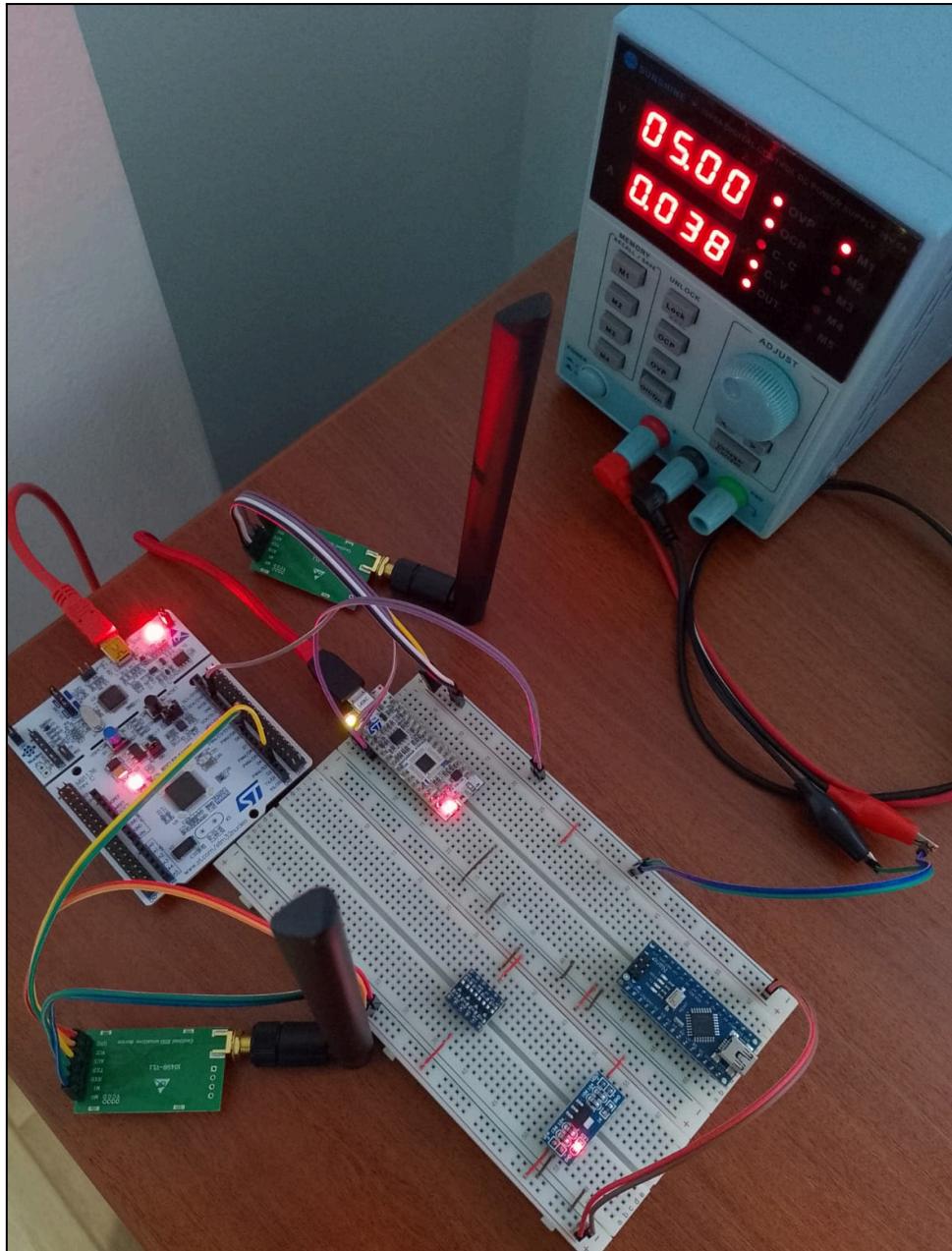
/* USER CODE END WHILE */
```

Resultados



```
Console X Progress Memory
COM8_EBYTE (CONNECTED)
3
Receive:
4
Receive:
5
Receive:
0
Receive:
1
Receive:
2
Receive:
3
Receive:
4
Receive:
5
Receive:
0
Receive:
1
```

Implementación



Videos de referencia

<https://www.youtube.com/watch?v=MHJXhq-yjCU&t=968s>

https://www.youtube.com/watch?v=lEfloatO4Pc4&list=PL5e8EBIOu5_PK2j2kD2Wt24SHHP9hosk&index=5

Verificación de envío de datos más grandes (tipo de dato *float*)

Volver a implementar el circuito de TX y RX con los módulos NUCLEO anteriores. Realizar pruebas cambiando el tipo de dato por uno más grande.

Creamos varias variables que simularán los datos obtenidos por los sensores cuando se realice la implementación final.

Para ambos NUCLEO de TX y RX es necesario las librerías:

```
#include <stdio.h>
#include <string.h>
```

Algoritmo TX

```
/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1)
{
    float temp = 12.4;
    float hum = 13.5;
    float press = 14.6;
    float hei = 15.7;
    float y = 16.8;
    float p = 17.9;
    float r = 19.0;

    sprintf((char*)charToTransmit, "T:%.1f H:%.1f P:%.1f H:%.1f Y:%.1f P:%.1f R:%.1f\r\n", temp, hum, press, hei, y, p, r);
    /* Review ASCII Code */
    HAL_UART_Transmit(&huart1, charToTransmit, strlen((char*)charToTransmit), 100);

    HAL_Delay(1000);

    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
}
```

El delay es necesario aumentarlo debido a que ahora estamos enviando mucho más datos a través del módulo LoRa. **Dependerá de la cantidad de datos.**

Algoritmo RX

```
/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1)
{
    rxLength = 0;
    while (1)
    {
        uint8_t c;
        if (HAL_UART_Receive(&huart1, &c, 1, 100) == HAL_OK)
        {
            if (c == '\n' || c == '\r' || rxLength >= 100 - 1) {
                break;
            }
            receiveData[rxLength++] = c;
        }
    }

    receiveData[rxLength] = '\0';

    printf("%s\r\n", receiveData);
    HAL_Delay(500);

    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
}
```

El delay en el caso de recepción no es tan necesario. Se agregó con la finalidad de obtener una mayor seguridad en la obtención de los datos.

Resultado

Valores recibidos

Verificación de los USART en la tarjeta H755ZI usados en el programa realizado

Se está usando el USART 3 para la comunicación con la PC e imprimir los datos almacenados en la SD Card.

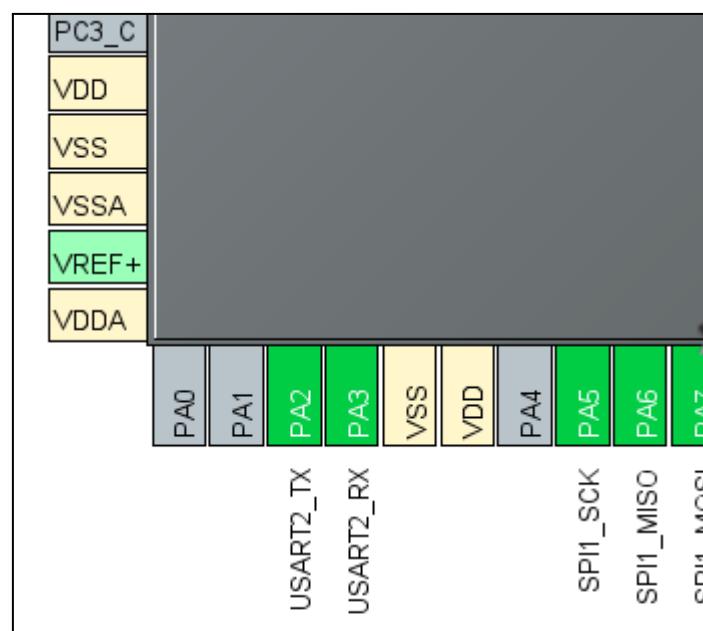
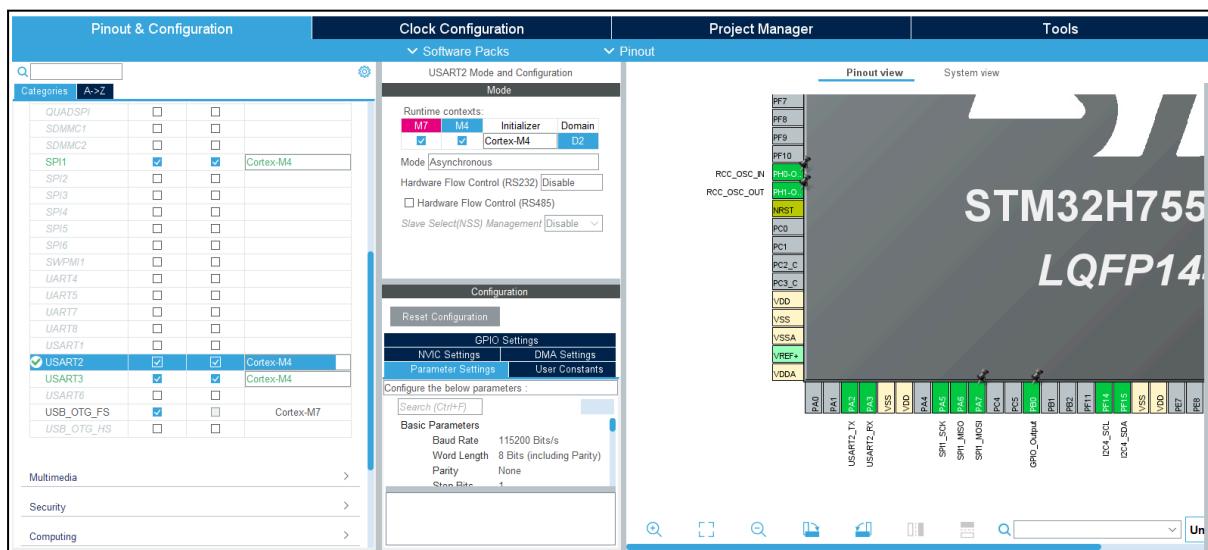
Usaremos el USART 2 para la comunicación con el módulo EBYTE. Primero veremos si a partir de las pruebas este periférico funciona bien.

Pines definidos

USART2_TX PA2

USART2_RX PA3

Definimos el modo Asíncrono y la configuración por defecto (**115200 Bits/s como baudrate**), aparte de otras configuraciones.



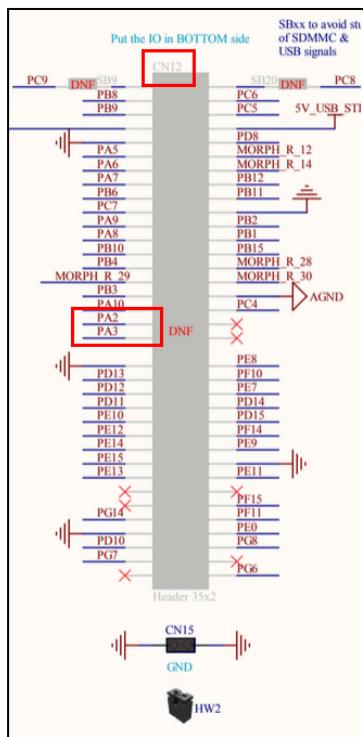
Verificaremos si no afecta en algo los programas realizados con los otros NUCLEO realizando una prueba individual de prueba para el módulo EBYTE.

Verificación de pines definidos para el USART2

Los pines definidos por el CubelDE son los siguientes:

USART2_TX PA2

USART2_RX PA3

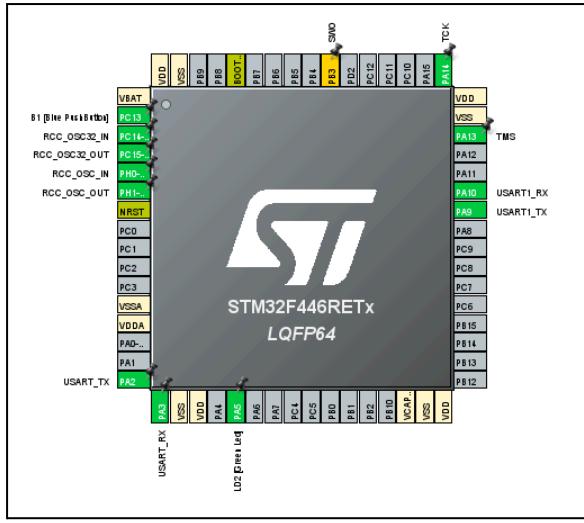


Desarrollo y pruebas con los NUCLEO H755ZI y F446RE

TX H755ZI (COM9)

USART2 TX PA2

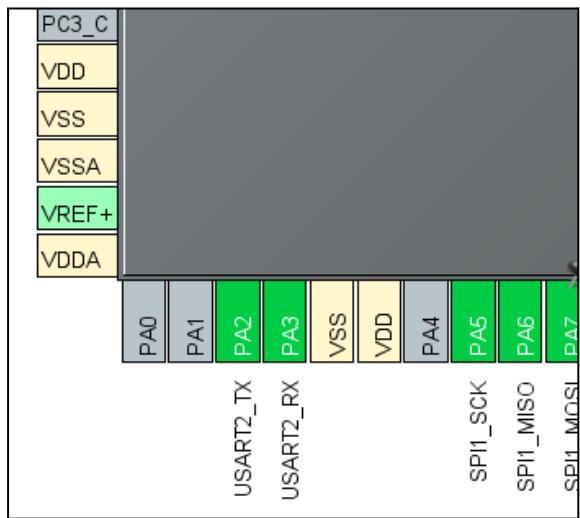
USART2 RX PA3



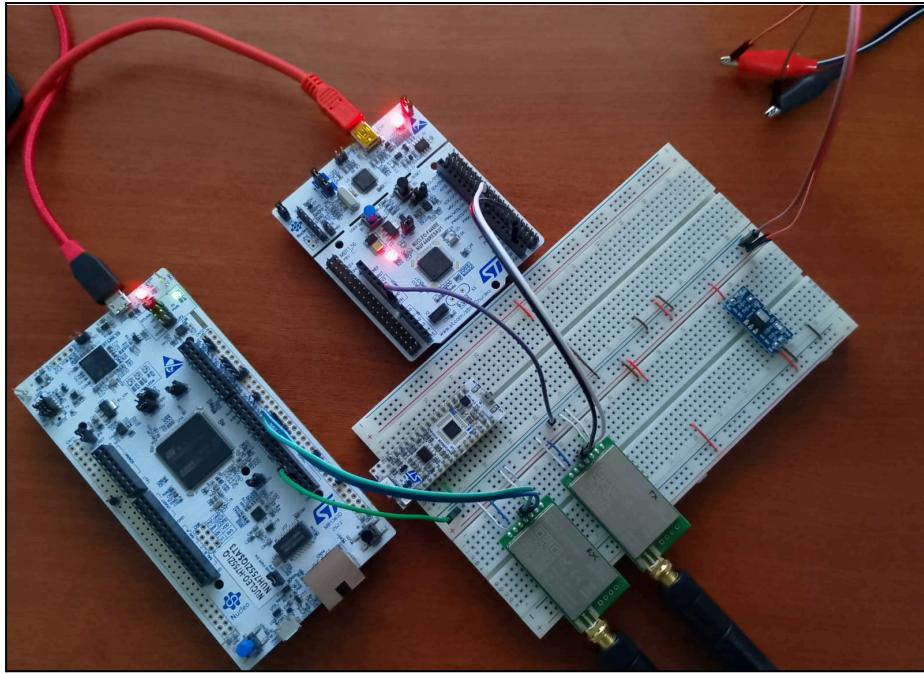
RX F446RE (COM8)

USART 1 TX PA9 D8

USART 1 RX PA10 D2



Implementación



Algoritmo TX

```

/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1)
{
    float temp = 12.4;
    float hum = 13.5;
    float press = 14.6;
    float hei = 15.7;
    float y = 16.8;
    float p = 17.9;
    float r = 19.0;

    sprintf((char*)charToTransmit, "T:%.1f H:%.1f P:%.1f H:%.1f Y:%.1f P:%.1f R:%.1f\r\n", temp, hum, press, hei, y, p, r);
    /* Review ASCII Code */
    HAL_UART_Transmit(&huart2, charToTransmit, strlen((char*)charToTransmit), 100);

    HAL_Delay(1500);
}

```

Algoritmo RX

```

/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1)
{
    HAL_UART_Receive(&huart1, receiveData, 1, 100);
    uint8_t c = receiveData[0];
    printf("Receive:\r\n");
    printf("%c", c);
    printf("\n");

    HAL_Delay(500);

    /* USER CODE END WHILE */
}

```

Resultados

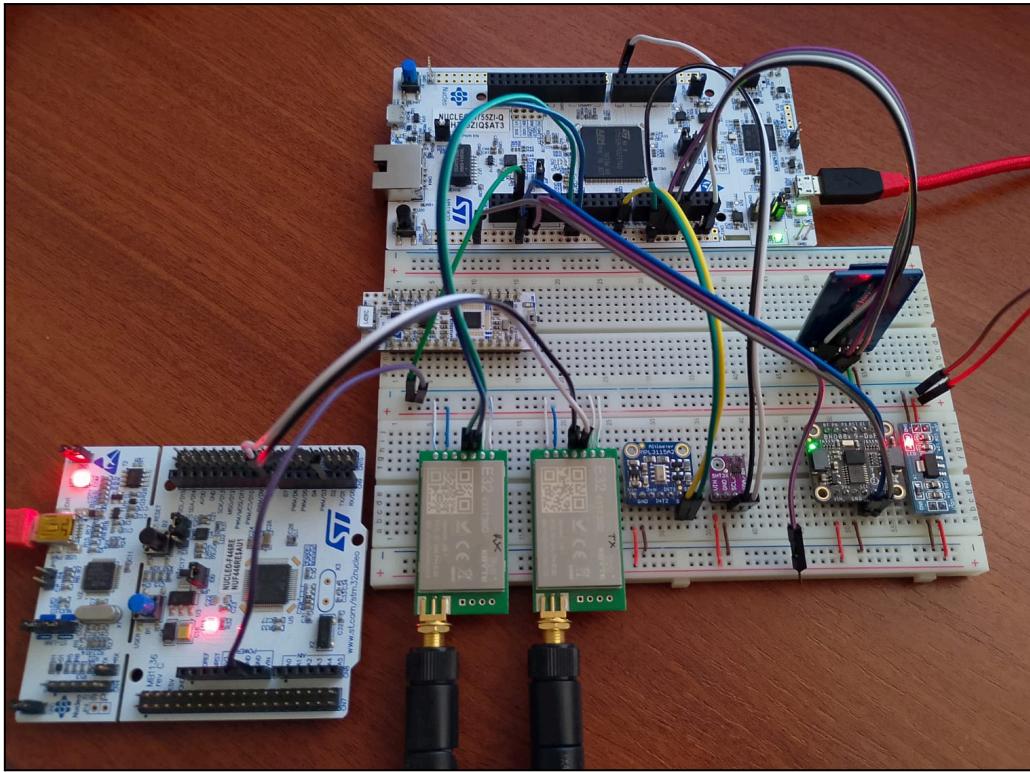
```
T:12.4 H:13.5 P:14.6 H:15.7 Y:16.8 P:17.9 R:19.0
```

Prueba final completo: todos los módulos, H755ZI (TX), F446RE (RE)

Se realiza la implementación con todos los módulos (ya hecha anteriormente) ahora junto a las tarjetas EBYTE de TX y RX.

Para la comunicación con la tarjeta en TX utilizamos el USART2 de acuerdo con las pruebas realizadas.

Implementación



Algoritmo TX

```
// BNO085
double *data = get_Orientation();

/* Conditional for delay */
if (counter > max_counter)
{
    /* Work with stdio.h */
    sprintf(array, "T %.lf, H %.lf, P %.lf, T %.lf; Y %.lf, P %.lf, R %.lf\r\n",
           temperature_sht31, humidity_sht31,
           pressure_mpl3115a2, temperature_mpl3115a2,
           data[0], data[1], data[2]);

    printf("Data\r\n");
    printf(array);

    // EBYTE Module
    HAL_UART_Transmit(&huart2, array, strlen(array), 100);
    HAL_Delay(1000);
    printf("Sent\r\n");

    /* Write data on SD Card*/
    if (counterSD < max_counterSD)
    {
```

Algoritmo RX

```

/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1)
{
    rxLength = 0;
    while (1)
    {
        uint8_t c;
        if (HAL_UART_Receive(&huart1, &c, 1, 100) == HAL_OK)
        {
            if (c == '\n' || c == '\r' || rxLength >= 100 - 1) {
                break;
            }
            receiveData[rxLength++] = c;
        }
    }

    receiveData[rxLength] = '\0';

    printf("%s\r\n", receiveData);
    printf("Received\r\n");
    HAL_Delay(500);

/* USER CODE END WHILE */

```

Procedimiento

1. Recordar primero alimentar luego arrancar los programas.
2. Como primer paso cargamos el programa de recepción en la tarjeta.
3. Desconectamos el NUCLEO F446RE (RX) y conectamos el H755ZI (TX).
4. Cargamos el programa de transmisión en el H755ZI y ejecutamos.
5. Con la tarjeta de TX conectada podemos conectar nuevamente la tarjeta de RX.

Resultados

TX

```
Problems Tasks Console X Properties
COM9_9600 (CONNECTED)
Data
T 21.5, H 66.4, P 97262.0, T 21.0; Y -133.7, P -76.4, R -152.9
Sent
Data
T 21.4, H 66.3, P 97266.0, T 21.0; Y -133.7, P -76.4, R -152.9
Sent
Data
T 21.5, H 66.4, P 97265.0, T 21.0; Y -133.7, P -76.4, R -152.9
Sent
Data
T 21.5, H 66.3, P 97267.0, T 21.1; Y -133.7, P -76.4, R -152.9
Sent
Data
T 21.5, H 66.4, P 97270.0, T 21.1; Y -133.7, P -76.4, R -152.9
Sent
Data
T 21.5, H 66.3, P 97266.0, T 21.1; Y -133.7, P -76.4, R -152.9
Sent
Data
T 21.5, H 66.3, P 97274.0, T 21.1; Y -133.7, P -76.4, R -152.9
```

RX

```
Problems Tasks Console X Properties
COM8_9600 (CONNECTED)
T 21.5, H 66.3, P 97258.0, T 21.0; Y -133.7, P -76.4, R -152.9
Received

Received
T 21.4, H 66.3, P 97259.0, T 21.0; Y -133.7, P -76.4, R -152.9
Received

Received
T 21.5, H 66.3, P 97260.0, T 21.0; Y -133.7, P -76.4, R -152.9
Received

Received
T 21.5, H 66.3, P 97263.0, T 21.0; Y -133.7, P -76.4, R -152.9
Received

Received
T 21.5, H 66.3, P 97262.0, T 21.0; Y -133.7, P -76.4, R -152.9
Received

Received
```

The screenshot shows a text editor window with the following details:

- File menu: Open, Save, Minimize, Maximize, Close.
- Title bar: dataOBC.txt, 31 GB Volume /media/david/407A-0F56
- Text content:

```
33 T 21.6, H 66.1, P 97222.0, T 21.1; Y 0.0, P 0.0, R 0.0
34
35 T 21.6, H 66.4, P 97226.0, T 21.1; Y 0.0, P 0.0, R 0.0
36
37 T 21.6, H 66.5, P 97224.0, T 21.1; Y 0.0, P 0.0, R 0.0
38
39 T 21.6, H 66.5, P 97230.0, T 21.1; Y 0.0, P 0.0, R 0.0
40
41 T 21.6, H 65.9, P 97230.0, T 21.1; Y 179.6, P 7.7, R -57.0
42
43 T 21.6, H 65.9, P 97234.0, T 21.1; Y 179.6, P 7.7, R -57.0
44
45 T 21.6, H 65.9, P 97227.0, T 21.1; Y 179.6, P 7.7, R -57.0
46
47 T 21.6, H 65.9, P 97229.0, T 21.1; Y 179.6, P 7.7, R -57.0
48
49 T 21.6, H 65.8, P 97228.0, T 21.1; Y 179.6, P 7.7, R -57.0
50
51 T 21.6, H 65.8, P 97232.0, T 21.1; Y 179.6, P 7.7, R -57.0
52
53 T 21.7, H 65.9, P 97229.0, T 21.1; Y 179.6, P 7.7, R -57.0
54
55 T 21.7, H 66.0, P 97230.0, T 21.1; Y 179.6, P 7.7, R -57.0
56
57 T 21.6, H 65.9, P 97227.0, T 21.1; Y 179.6, P 7.7, R -57.0
58
59 T 21.7, H 66.1, P 97234.0, T 21.1; Y 179.6, P 7.7, R -57.0
60
61 T 21.7, H 66.0, P 97233.0, T 21.1; Y 179.6, P 7.7, R -57.0
62
63 T 21.6, H 66.1, P 97228.0, T 21.1; Y 179.6, P 7.7, R -57.0
64
65 T 21.7, H 65.9, P 97228.0, T 21.1; Y 179.6, P 7.7, R -57.0
66
67 T 21.7, H 65.9, P 97233.0, T 21.1; Y 179.6, P 7.7, R -57.0
68
69 T 21.7, H 65.9, P 97233.0, T 21.1; Y 179.6, P 7.7, R -57.0
70
71 T 21.7, H 66.0, P 97231.0, T 21.1; Y 179.6, P 7.7, R -57.0
72
73 T 21.7, H 65.9, P 97230.0, T 21.1; Y 179.6, P 7.7, R -57.0
74
75 T 21.7, H 65.9, P 97232.0, T 21.1; Y 179.6, P 7.7, R -57.0
76
77 T 21.7, H 66.0, P 97228.0, T 21.1; Y 179.6, P 7.7, R -57.0
78
79 T 21.7, H 66.0, P 97232.0, T 21.1; Y 179.6, P 7.7, R -57.0
80
```
- Bottom status bar: Plain Text, Tab Width: 8, Ln 1, Col 1, INS.

Corriente consumida

La corriente consumida mínima es de 61 mA.

La corriente consumida máxima es de aproximadamente 307 mA.

No se está tomando en cuenta la alimentación de las tarjetas NUCLEO (alimentadas a través de la PC).

Considerar 7.4 V, 6600 mAh

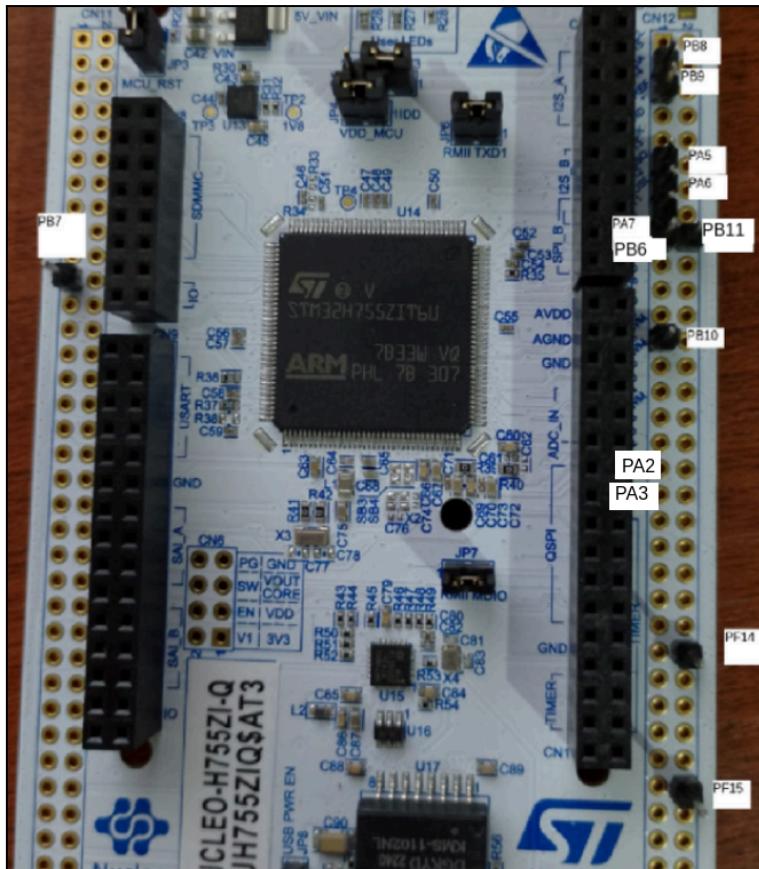
OBSERVACIONES

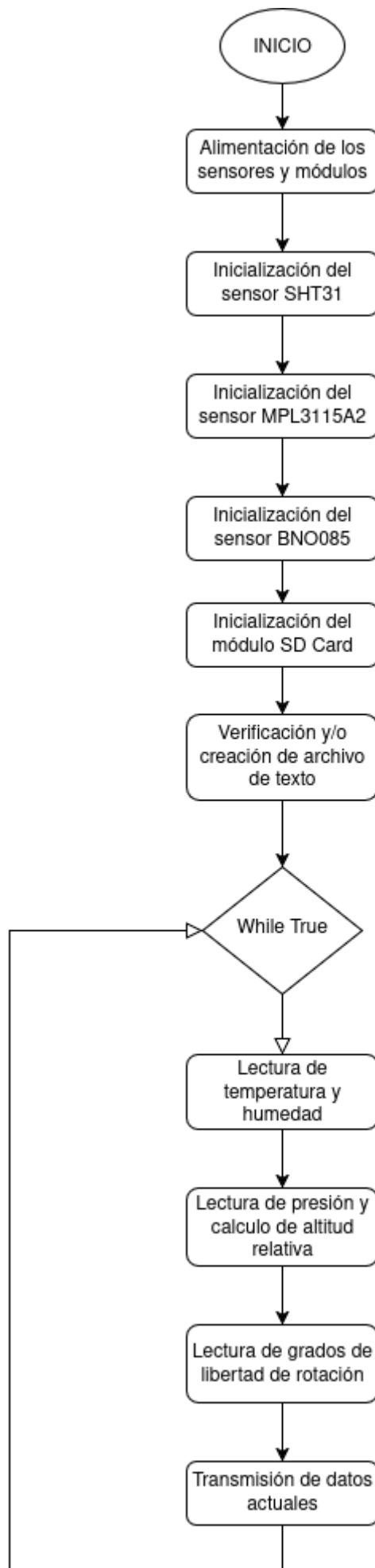
El Baudrate en ambos módulos para la comunicación debe ser de 9600 Bits/s (aplicar para comunicación con las tarjetas EBYTE. De la misma forma en la comunicación con la PC para la visualización de los datos).

En el algoritmos de TX fue necesario un aumento de *Delay* para la recepción adecuada de los datos.

Para el módulo SD Card y el BNO085 a veces es necesario reiniciar la alimentación para observar correctamente la adquisición de datos.

Pinout final





El sistema On Board Computer (OBC) fue desarrollado a partir del microcontrolador NUCLEO H755ZI, que cuenta con un procesador de doble núcleo. Para la adquisición de datos se emplearon los sensores SHT31, MPL3115A2 y BNO085. Asimismo, se integró un módulo de tarjeta SD para el almacenamiento de las variables medidas y un módulo de comunicación LoRa, específicamente la tarjeta EBYTE E32, destinada a la transmisión de la información registrada.

El sistema tiene como función principal la medición de parámetros como temperatura, humedad y presión, así como el cálculo de la altitud relativa del cohete. Además, permite determinar los ángulos de rotación del mismo. Los datos obtenidos son almacenados y posteriormente transmitidos hacia una estación terrena para su procesamiento.

Las pruebas realizadas incluyeron la programación y verificación individual de cada sensor y módulo, abarcando desde el testeo de las condiciones de funcionamiento de los dispositivos hasta el almacenamiento de datos en la tarjeta SD y su transmisión posterior. Las pruebas finales consistieron en la simulación de un enlace de comunicación, en la cual el NUCLEO H755ZI se utiliza como transmisor encargado de gestionar los sensores, mientras que la estación receptora fue implementada con un NUCLEO F446RE, configurado para recibir los datos de manera continua.

Los resultados obtenidos fueron satisfactorios, verificando tanto la correcta transmisión de los datos como su almacenamiento previo en la tarjeta SD. Dichos resultados brindan la confianza necesaria para avanzar hacia el diseño de un sistema OBC más complejo, incorporando mejoras en el prototipo utilizado con miras a su implementación final.

The On Board Computer (OBC) system was developed using the NUCLEO H755ZI microcontroller, which features a dual-core processor. For data acquisition, the sensors SHT31, MPL3115A2, and BNO085 were employed. In addition, an SD card module was integrated for storing the measured variables, along with a LoRa communication module, specifically the EBYTE E32 board, designated for transmitting the recorded information.

The main functions of the system include measuring parameters such as temperature, humidity, and pressure, as well as calculating the relative altitude of the rocket. Furthermore, the system enables the determination of its rotation angles. The collected data are stored and subsequently transmitted to a ground station for further processing.

The tests conducted involved the programming and individual verification of each sensor and module, ranging from evaluating device operating conditions to storing data on the SD card and transmitting them afterward. The final tests consisted of simulating a communication link, in which the NUCLEO H755ZI was used as the transmitter managing the sensors, while the receiving station was implemented with a NUCLEO F446RE, configured to continuously receive the data.

The results obtained were satisfactory, confirming both the correct transmission of the data and their prior storage on the SD card. These outcomes provide the necessary confidence to advance toward the design of a more complex OBC system, incorporating improvements to the prototype for its final implementation.