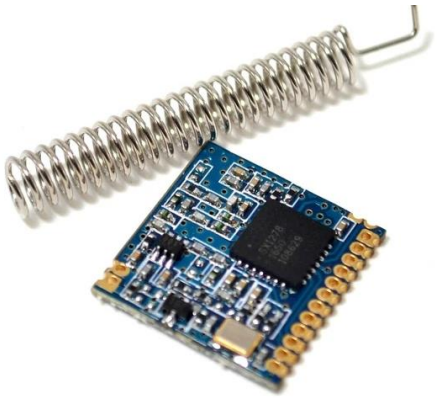
Teoría de las Comunicaciones

Interface con LoRaWAN



Grupo: “Electrones de Laplace”

Integrantes:

- Agüero, Nicolás - 39733915
- Cacchiarelli, Gastón - 39857357
- Campos, Pedro - 277753
- Fantoni, Omar - 41125052
- Ferrero, Alejandro - 40054394
- Romero, Matías - 39968473

Profesores:

- Ing. E. Danizio
- Ing. F. Patrito
- Ing. J. Galleguillo

Fecha:

- 1 de noviembre del 2018.

Agradecimientos:

A los profesores de la cátedra de "Teoría de las Comunicaciones" Ingenieros P. E. Danizio, F. Patrito, J. Galleguillo, no sólo por los conocimientos brindados y por darnos esta oportunidad sino también por confiar en nosotros y en el potencial de todos sus alumnos.

A los demás grupos y compañeros por aclarar dudas y trabajar en conjunto para lograr este objetivo.

A la empresa Adaptio, Fabio Grigorjev y compañía, por su charla introductoria y orientativa en el tema, y su disposición para ayudarnos.

A la Facultad de Ciencias Exactas, Físicas y Naturales por proveernos del espacio de trabajo como así también las herramientas necesarias.

Índice

Objetivo

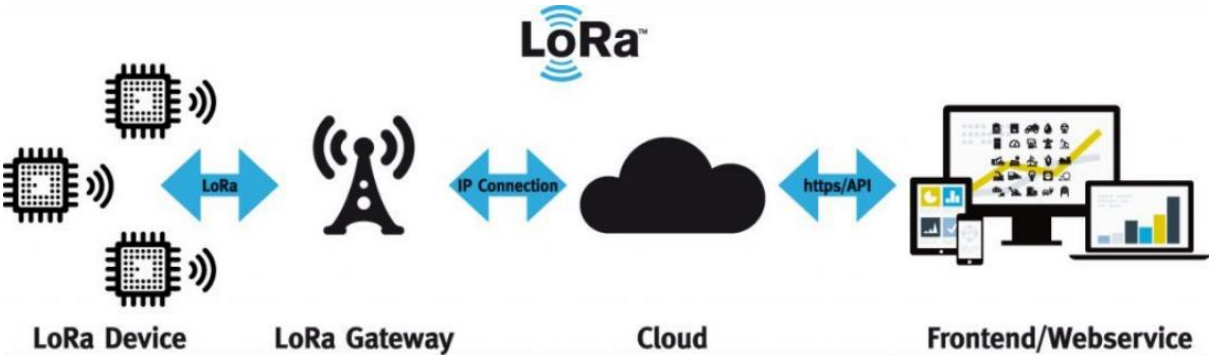
Se deberá desarrollar un enlace punto multipunto mediante una interface de radio LoRa. En uno de los extremos se colectará información de temperatura con una tasa de transmisión de 30 seg. y se mostrará en una página WEB alojada en un concentrador. A modo de establecer un orden de mérito de los grupos se evaluará el tiempo de entrega y el consumo de energía del mismo.

Marco Teórico

Para iniciar este trabajo, se realizará una breve introducción al concepto: *LoRa,* del cual se basa este proyecto:
LoRa (en inglés, *Low Power Wide Area Network*) es un tipo de modulación en radiofrecuencia patentado por Semtech y tiene como principales ventajas: una alta tolerancia a las interferencias, una alta sensibilidad para recibir datos (-168dB), está basada en modulación *Chirp*, posee bajo consumo (lo cual una batería puede alargar su vida hasta los 10 años), un largo alcance (hasta 20 km) y una baja transferencia de datos (hasta 255 bytes).
Las bandas de frecuencia asignadas para operación sin licencia son: 915Mhz América, 868MHz Europa, 433MHz Asia.
Con esta tecnología nace LoRaWAN, éste es un protocolo de red que usa la tecnología LoRa para transmitir datos y administrar dispositivos LoRa, se compone de dos partes principalmente: **gateways y nodos**, los primeros son los encargados de recibir y enviar información a los nodos y los segundos, son los dispositivos finales que envían y reciben información hacia el gateway.
Las principales características de LoRaWAN son:

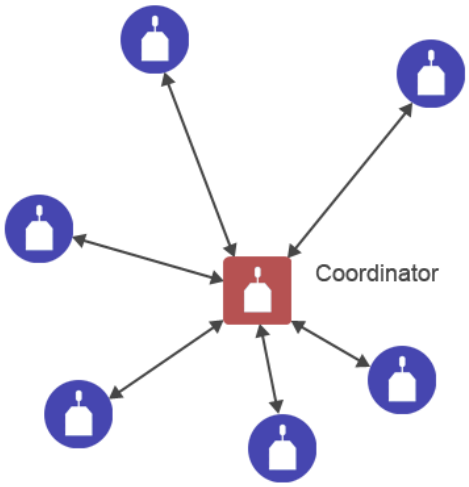
- Topología estrella.
- Alcance de 10 a 15km en línea de vista.
- Encriptación AES 128.
- Soporte para 3 clases de nodos.
- Administración de dispositivos.
- Redes públicas y privadas.
- Bajo consumo y largo alcance.
- Baja transferencia de datos (hasta 242 bytes).

En la siguiente figura se puede observar claramente cómo se compone una red LoRaWAN clásica, en la que una serie de dispositivos finales (nodos), se conectan a un Gateway y este envía toda la información captada a un servidor, que por medio de una API entrega los datos a una aplicación final para el usuario.



Otra conexión posible mediante LoRaWAN, es la conexión punto-a-punto (P2P), la principal característica de este modo es que no se requiere un dispositivo intermediario que administre la comunicación (Gateway), los dispositivos pueden enviar entre ellos información directamente, esto es perfecto para comunicaciones sencillas y simples.

La otra manera de conectar es en tipo estrella, donde hay un nodo que se encarga de coordinar la red, esta conexión posee una desventaja, ya que está limitada a 255 redes de 255 nodos, y a que el nodo coordinador solo puede escuchar un nodo a la vez.



En protocolo LoRaWAN existen tres tipos de clases de nodo:
Clase A: La más soportada en casi todos los dispositivos, este tipo de clase ofrece el mayor ahorro de energía debido a que solo entra en modo escucha (llamado ventana RX) después de enviar un datos hacia el gateway, por eso es ideal para dispositivos que usan una batería.
Clase B: Este tipo de dispositivos tiene las ventanas de recepción con base a tiempos predeterminados con el gateway, este tipo de nodos puede usar una batería o una fuente externa dependiendo de los tiempos asignados de escucha.
Clase C: Este tipo de clase ofrece el menor ahorro de energía debido a que siempre está en modo escucha y solo cuando es necesario en modo transmitir, la recomendación es usarlo en dispositivos que cuentan con una fuente externa de alimentación.

LoRawan utiliza el estándar de comunicaciones SPI, el cuál lo describimos a continuación.

Estándar de comunicaciones SPI:

El Bus SPI (del inglés Serial Peripheral Interface) es un estándar de comunicaciones, usado principalmente para la transferencia de información entre circuitos integrados en equipos electrónicos. El bus de interfaz de periféricos serie o bus SPI es un estándar para controlar casi cualquier dispositivo electrónico digital que acepte un flujo de bits serie regulado por un reloj (comunicación sincrónica).

La sincronización y la transmisión de datos se realiza por medio de 4 señales:

- **SCLK** *(Clock)*: Es el pulso que marca la sincronización. Con cada pulso de este reloj, se lee o se envía un bit. También llamado TAKT (en alemán).
- **MOSI** *(Master Output Slave Input)*: Salida de datos del Master y entrada de datos al Esclavo. También llamada SIMO.
- **MISO** *(Master Input Slave Output)*: Salida de datos del Esclavo y entrada al Master. También conocida por SOMI.
- **SS/Select**: Para seleccionar un Esclavo, o para que el Master le diga al Esclavo que se active. También llamada SSTE.

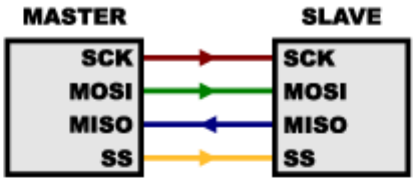
La Cadena de bits es enviada de manera síncrona con los pulsos del reloj, es decir con cada pulso, el Master envía un bit. Para que empiece la transmisión el Master baja la señal SSTE ó SS/Select a cero, con esto el Esclavo se activa y empieza la transmisión, con un pulso de reloj al mismo tiempo que el primer bit es leído. Nótese que los pulsos de reloj pueden estar programados de manera que la transmisión del bit se realice en 4 modos diferentes, a esto se llama polaridad y fase de la transmisión:

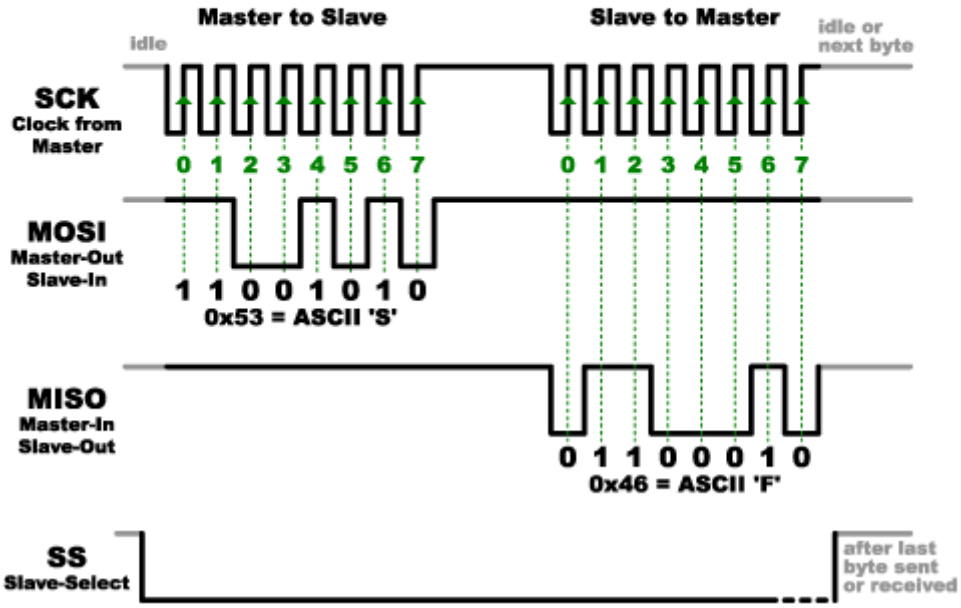
- 1. Con el flanco de subida sin retraso.
- 2. Con el flanco de subida con retraso.
- 3. Con el flanco de bajada sin retraso.
- 4. Con el flanco de bajada con retraso.

El funcionamiento para un envío de un Master es el siguiente:

- Se habilita el chip al que hay que enviar la información mediante el CS (Opcional).
- Se carga en el buffer de salida el byte a enviar.
- La línea de Clock empieza a generar la señal cuadrada donde normalmente por cada flanco de bajada se pone un bit en MOSI.
- El receptor normalmente en cada flanco de subida captura el bit de la linea MISO y lo incorpora en el buffer.

Se repite el proceso 8 veces y se ha transmitido un byte. Si se ha terminado de transmitir se vuelve a poner la linea CS en reposo. Hay que tener en cuenta que a la vez que el Master está enviando un dato también lo recibe así que si el Slave ha depositado algún byte en el buffer de salida, este también será enviado y recibido por el Master, comunicación full-duplex.





Ventajas

- Comunicación Full Duplex (es decir permite transmitir y recibir información al mismo tiempo)
- Mayor velocidad de transmisión que con I²C o SMBus
- Protocolo flexible en que se puede tener un control absoluto sobre los bits transmitidos
    - No está limitado a la transferencia de bloques de 8 bits
    - Elección del tamaño de la trama de bits, de su significado y propósito
- Su implementación en hardware es extremadamente simple
    - Consume menos energía que I²C o que SMBus debido que posee menos circuitos (incluyendo las resistencias *pull-up*) y estos son más simples
    - No es necesario arbitraje o mecanismo de respuesta ante fallos
    - Los dispositivos *clientes* usan el reloj que envía el *servidor*, no necesitan por tanto su propio reloj
    - No es obligatorio implementar un transceptor (emisor y receptor), un dispositivo conectado puede configurarse para que solo envíe, sólo reciba o ambas cosas a la vez
- Usa mucho menos terminales en cada chip/conector que una interfaz paralelo equivalente.
- Como mucho una única señal específica para cada *cliente* (señal SS), las demás señales pueden ser compartidas.

Desventajas

- Consume más pines de cada chip que I²C, incluso en la variante de 3 hilos
- El direccionamiento se hace mediante líneas específicas (señalización fuera de banda) a diferencia de lo que ocurre en I²C que se selecciona cada chip mediante una dirección de 7 bits que se envía por las mismas líneas del bus
- No hay control de flujo por hardware
- No hay señal de asentimiento. El *servidor* podría estar enviando información sin que estuviese conectado ningún *cliente* y no se daría cuenta de nada
- No permite fácilmente tener varios *servidores* conectados al bus
- Sólo funciona en las distancias cortas a diferencia de, por ejemplo, RS-232, RS-485, o Bus CAN.

Indicador de la intensidad de la señal recibida:

El indicador de la fuerza de la señal recibida (RSSI por las siglas del inglés *Received Signal Strength Indicator*), es una escala de referencia (en relación a 1 mW) para medir el nivel de potencia de las señales recibidas por un dispositivo en las redes inalámbricas.

La escala tiene al valor 0 (cero) como centro; representa *0 RSSI*, o *0 dBm*. Aunque teóricamente puede darse el caso de medirse valores positivos, generalmente la escala se expresa dentro de valores negativos; cuanto más negativo, mayor pérdida de señal.

El RSSI indica intensidad recibida, no calidad de señal, ya que ésta última se determina contrastando la intensidad de la señal respecto de la relación señal/ruido.

En esta escala, un nivel de 0 dBm es igual a 1 mW (mili vatio).

Interpretación de los valores

En una escala de 0 a -80 RSSI:

- 0: señal ideal, difícil de lograr en la práctica.
- -40 a -60: señal idónea con tasas de transferencia estables.
- -60: enlace bueno; ajustando TX y basic rates se puede lograr una conexión estable al 80%.
- -70: enlace normal -bajo; es una señal medianamente buena, aunque se pueden sufrir problemas con lluvia y viento.
- -80: es la señal mínima aceptable para establecer la conexión; puede ocurrir caídas, que se traducen en corte de comunicación (pérdida de llamada, perdida de datos, mensajes (sms) corruptos (ilegibles).

Tabla de equivalencias aproximada para averiguar el nivel de cobertura en función de los [dBm] en aire recibidos:

- Más de -71 [dBm] = Excelente
- Entre -87 y -71 [dB] = Muy buena
- Entre -97 y -87 [dBm] = Buena
- Entre -111 y -97 [dBm] = Baja cobertura
- A partir de -113 [dBm] = Sin cobertura

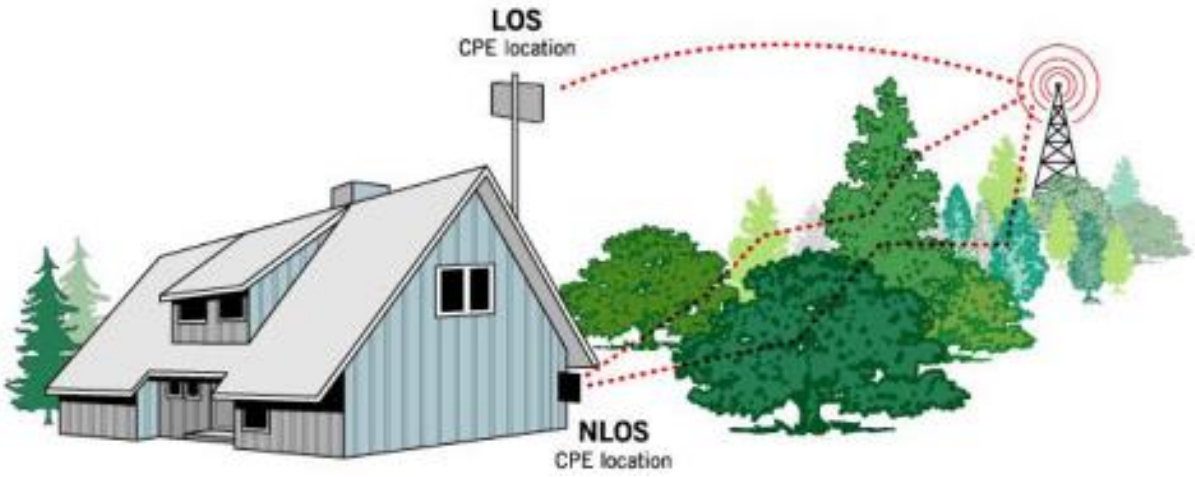
SNR

La relación señal/ruido o S/R (en inglés *signal-to-noise ratio*, abreviado SNR o S/N) se define como la proporción existente entre la potencia de la señal que se transmite y la potencia del ruido que la corrompe. Este margen es medido en decibelios.
A continuación, presentamos algunos valores de relación señal ruido con su respectiva conclusión.
6dB o menos es relativamente bajo y se podría experimentar falta de sincronismo o problemas intermitentes de sincronismo.
7dB-10dB es aceptable pero no admite demasiadas variaciones en las condiciones.
11dB-20dB es bueno con pocos o ningún problema de sincronismo.
20dB-28dB es excelente.
29dB o por encima es extraordinario.

Pérdidas en obstáculos

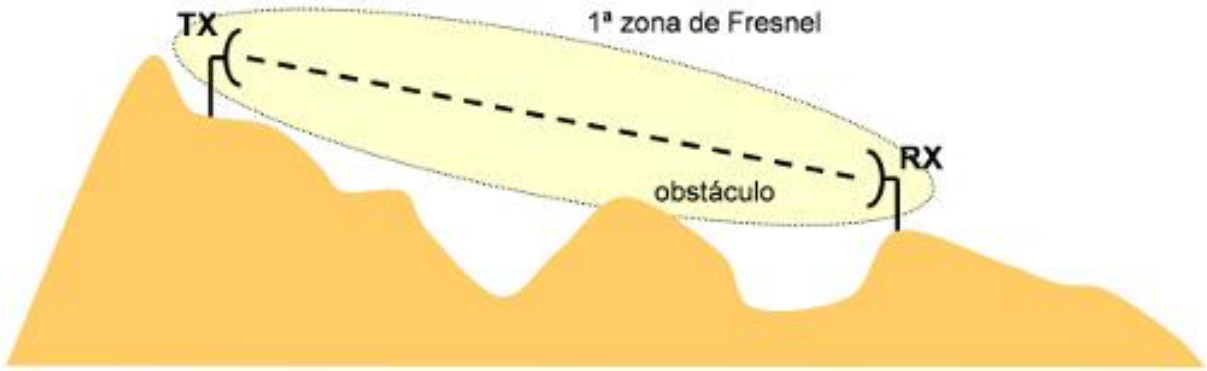
Para modelar las pérdidas que se producen por la **obstrucción del enlace radioeléctrico**

(*Non Line Of Sight*, NLOS) se utiliza el concepto de las llamadas **zonas de Fresnel**.



Las zonas de Fresnel son unos elipsoides concéntricos que rodean al rayo directo de un enlace radioeléctrico y que quedan definidos a partir de las posiciones de las antenas transmisora y receptora. Tienen la propiedad de que una onda que partiendo de la antena transmisora, se reflejara sobre la superficie del elipsoide y después incidiera sobre la antena receptora, habría recorrido una distancia superior a la recorrida por el rayo directo en múltiplos de media longitud de onda. Es decir, la onda reflejada se recibiría con un retardo respecto al rayo directo equivalente a un desfase múltiplo de 180º. Precisamente este valor del múltiplo determina el n-ésimo elipsoide de Fresnel.

De este modo, la primera zona de Fresnel (n = 1) se caracteriza por el volumen interior al elipsoide con diferencia de distancias igual a una semilongitud de onda o diferencia de fases de 180º. Luego posibles reflexiones cerca del borde de la primera zona de Fresnel pueden causar atenuación, ya que la onda reflejada llegaría a la antena receptora en oposición de fase. Por lo tanto, durante la fase de planificación del radioenlace debe asegurarse que la primera zona de Fresnel se encuentre libre de obstáculos, bien aumentando la altura de los mástiles de las antenas o bien situándolos en otra posición del edificio. Evidentemente, una obstrucción completa de la zona de Fresnel produciría pérdidas todavía mayores.



Perfil de un radioenlace.

En la figura 1 se representa el perfil de un radioenlace en el cual se ha añadido el contorno de la primera zona de Fresnel para detectar posibles obstáculos. La distancia entre Tx y Rx depende de la longitud del radioenlace y de la frecuencia utilizada, y suele igualarse al radio máximo de la primera zona de Fresnel (en mitad del radioenlace). El radio de la primera zona de Fresnel, R_1, en un punto cualquiera de un radioenlace puede calcularse a partir de la siguiente expresión:

$$r1 = 8{,}657 * \sqrt{\frac{D}{f}}$$

donde

- *r1* = radio en metros (m).
- *D* = distancia en kilómetros (km)
- *f* = frecuencia de la transmisión en gigahercios (GHz).

A mayor frecuencia, las zonas de Fresnel son cada vez más estrechas.

Se observa que la atenuación desaparece cuando el despejamiento es igual al 60% del radio de la primera zona de Fresnel, criterio que suele utilizarse en la práctica para el diseño del radioenlace.

Códigos del proyecto

Emisor:

```
//Definimos las librerias que necesitamos utilizar:
#include <SPI.h>
#include <LoRa.h>
#include <OneWire.h>
#include <LowPower.h>
#include <DallasTemperature.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#define OLED_RESET 4
Adafruit_SSD1306 display(OLED_RESET);
#define LOGO16_GLCD_HEIGHT 16
#define LOGO16_GLCD_WIDTH  16
#if (SSD1306_LCDHEIGHT != 32)
#error("Height incorrect, please fix Adafruit_SSD1306.h!");
#endif
#define ONE_WIRE_BUS A3     //Definimos la entrada del dato del sensor de temp
//Creamos una conexión oneWIRE para poder comunicarnos con el sensor de temp con
este protocolo
OneWire oneWire(ONE_WIRE_BUS);
//Le decimos a nuestro sensor Dallas que nos vamos a comunicar usando el protocolo
oneWire
DallasTemperature sensors(&oneWire);
float Temp=0;      //Definimos la variable Temperatura y le asignamos en principio
un valor de cero
//int contador=1;
void setup() {
  Serial.begin(9600);   //Arrancamos el puerto serie para poder leer los datos
enviados
  while (!Serial);
  Serial.println("Electrones de Laplace");
  if (!LoRa.begin(433E6)) {                    //Se inicializa la librería  con la
frecuencia de transmición de 433MHz
    Serial.println("Error de comunicacion");   //en el caso de de que no se pueda
inicializar la transmición
    while (1);
  }
  //iniciamos la libreria del sensor();
  sensors.begin();
  //inicializamos el display
  display.begin(SSD1306_SWITCHCAPVCC, 0x3C);
  delay(100);
  display.clearDisplay();
  LoRa.setTxPower(20);              //  2-20  defecto 17
  LoRa.setSpreadingFactor(7);       // 6-12 defecto 7   (igual recep)
  LoRa.setSignalBandwidth(125E3);   //  defecto 125E3    (igual recep)
    //BW 7.8E3 10.4E3 15.6E3 20.8E3 31.25E3 41.7E3 62.5E3 125E3 250E3
  LoRa.setCodingRate4(5);           //5-8 defeto 5
  LoRa.setPreambleLength(8);        //6-65535 defecto 8
  LoRa.setSyncWord(0x12);           // defecto 0x12
    Temp = sensorTemp();
}
void loop() {
 //establece el arduino en LowPower por 28 segundos
  for (int i = 0 ;  i  <  3 ; i++)
            LowPower.powerDown(SLEEP_8S, ADC_OFF, BOD_OFF);
  LowPower.powerDown(SLEEP_4S, ADC_OFF, BOD_OFF)
  Temp = sensorTemp();                          // almacenamos la temperatura en
Temp
  //visualización monitor serial local
  Serial.print("Temperatura Local:  ");
  Serial.println(Temp);
  // envio de datos
```

```
  //se inicializa la secuencia de envio
  LoRa.beginPacket();
  LoRa.print("{\"1234\": ");
  LoRa.print("t:");
  LoRa.print(Temp,1);
  LoRa.print("}");
  //LoRa.print("c:");
  //LoRa.print(contador);
  LoRa.endPacket();                //Se finaliza la secuencia de envio
  LoRa.sleep();                    //SleepMode
  wDisplay();                      //llamaos la función que escribe en el display
 // delay(3000);
 contador++;
}
//Función para obtención de la Temperatura
float sensorTemp (){

   sensors.requestTemperatures();             //Solicitamos la temperatura al
Sensor
   float temp = sensors.getTempCByIndex(0);   //Obtenemos la temperatura
   return temp;                               //Devolvemos el valor de la
temperatura
}
 //la siguiente funcion es la usada para escribir el display local
void wDisplay(){
  display.setTextSize(1);                   // establece el tamaño del texto del
prier rengolon
  display.setTextColor(WHITE);              //color texto
  display.setCursor(0,0);                  //ubica el cursor en la esquina sup. izq.
  display.println("Electrones de Laplace");//texto preimer renglon + salto de linea
  display.println("Temp: ");               //texto segundo renglon + salto de linea
  display.setTextSize(2);                   //esablece tamaño de texto
  display.print("   ");                     //
  display.print(Temp,2);                    // dato de temperatura con 2 decimales
  display.print(" C");
  display.display();
  delay(10);
  display.clearDisplay();                   //limpiamos el display

}
```

Receptor de prueba:

```
#include <SPI.h>
#include <LoRa.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27, 2, 1, 0, 4, 5, 6, 7, 3, POSITIVE);
String lectura1,lectura2;
void setup() {
  Serial.begin(9600);
  while (!Serial);
  Serial.println("Electrones de Laplace");
  if (!LoRa.begin(433E6)) {
    Serial.println("Error LoRa!");
    while (1);
  }
  lcd.begin(16,2);
}
void loop() {
  int packetSize = LoRa.parsePacket();
  if (packetSize) {
    Serial.print("Dato recibido: '");
  lectura1="";
   lcd.setCursor(0,0);
   while (LoRa.available()) {
      char i =(char)LoRa.read();
      Serial.print(i);
      lectura1+=i
    }
    lcd.clear();
lcd.print(lectura1);
lcd.setCursor(0,1);
lcd.print(lectura2);
lectura2=lectura1;0
    Serial.print("' con RSSI ");
    Serial.println(LoRa.packetRssi());
    Serial.print("' con SNR ");
    Serial.println(LoRa.packetSnr());
  }
}
```

Librerías utilizadas en el código del Proyecto

Debido a la facilitación que otorgan las librerías en proyectos de programación y a fines de no gastar tiempo en asuntos que ya se trabajaron por investigadores y desarrolladores en el pasado, decidimos emplear el siguiente conjunto de librerías optimizando así nuestro proyecto:

- **Librería <LoRa.h>**
  - **Asignación de pines:** ésta librería, asigna por defecto:
    PIN 10 – NSS
    PIN 9 – RESET
    PIN 2 – DIO0 (este pin sirve para hacer interrupciones, en este caso no lo utilizamos)

  Podemos cambiar la asignación de estos pines utilizando la función **LoRa.setPins(nuevoPinSS, nuevoPinRESET, nuevoPinDIO0);**
  En nuestro caso decidimos usar los pines por defecto ya que no presentaba ninguna ventaja cambiar dicha asignación.
  Por otro lado, debido a la distribución de hardware de nuestro arduino nano, los pines correspondientes a la comunicación SPI son:
    PIN 12 – MISO
    PIN 11 – MOSI
    PIN 13 - SLCK
  - **Asignación de la frecuencia de comunicación SPI entre el arduino y la LoRa:**
    Por defecto la comunicación se realiza a 10MHz, ésta comunicación puede ser reemplazada usando la función **LoRa.setSPIFrequency(frecuencia);**
  - **Inicialización de la librería especificando la frecuencia de comunicación:**
    LoRa.Begin(frecuencia); en nuestro caso empleamos una frecuencia de 433MHz, debido a que es la frecuencia a la que funciona nuestra LoRa.
  - **Envío de datos:** Para el envío de datos se crean paquetes de datos, de la siguiente manera:
    LoRa.beginPacket(); inicia la secuencia de envío de un paquete de datos.
    LoRa.print(datos); escribe los datos del paquete a enviar, cada paquete puede contener hasta 255 bytes.
    LoRa.endPacket(); finaliza la secuencia de envío de un paquete.
  - **Control de Potencia:**
    LoRa.setTxPower(valor); ésta función permite establecer la potencia de transmisión de datos, el valor puede variar entre 2 y 20, siendo el valor por defecto igual a 17.
  - **Factor de Propagación:**
    LoRa.setSpreadingFactor(valor); permite variar el ensanchamiento "spread" de la onda transmitida. A éste valor lo dejamos por defecto debido a que tiene que coincidir con el del receptor. Ambos valores son igual a 7 en nuestro caso.
  - **Ancho de Banda de señal:**

LoRa.setSignalBandwidth(valor); el ancho de banda también debe coincidir con el del receptor para la comunicación sea compatible. Por defecto está en 125KHz.

- **Preámbulo:**
  LoRa.setPreambleLength(valor); establece el tamaño de la palabra de sincronización para iniciar la comunicación. Debe ser menor o igual que la del receptor.
- **Sincronizar palabra:**
  LoRa.setSyncWord(palabra); establece la palabra código de sincronización.
- **Verificación de errores(CRC):** éstas funciones se utilizan para crear una palabra al final del paquete de datos para verificar que el paquete fue recibido correctamente.
  LoRa.enableCrc();
  Lora.desableCrc()

- **Librería <SPI.h>**
  Ésta librería permite realizar la comunicación con dispositivos SPI (Serial Peripheral Interface), usando el arduino como dispositivo patrón.
- **Librerías <OneWire.h>, < DallasTemperature.h>**
  Éstas librerías se utilizan para el intercambio de datos con el sensor de temperatura. La librería Dallas Temperature es la que nos provee las funciones para tomar los datos del sensor y procesarlo, por su parte la librería OneWire provee el protocolo de comunicación entre el sensor "Dallas" y el arduino. Las funciones empleadas son:
  - **requestTemperatures(); :** Ésta función se encarga de preparar al sensor para medir la temperatura, es decir lo habilita a tomar la temperatura ahorrando consumo de energía.
  - **getTempCByIndex(0) :** Con ésta función se toma la medida de temperatura dada por el sensor, al dato obtenido se lo procesa y lo convierte a un valor de temperatura en °C. Ésta función es útil además cuando se tienen varios sensores ya que con el índice se puede seleccionar el sensor con el cuál nos vamos a comunicar. En nuestro caso tenemos un único sensor y por ellos elegimos el índice "0".
- **Librería<LowPower.h>**
  De esta librería utilizamos la función **LowPower.powerDown()** la cual permite que el Arduino entre en modo Sleep, dejando el dispositivo en un estado de mínimo consumo energético.
  Hay tres maneras de despertar un Arduino del modo Sleep:
  - Interrupciones hardware.
  - Timers.
  - UART o puerto serie.

  En nuestro caso, decidimos optar una interrupción por timer, utilizando el Watch Dog Timer, este es una interrupción periódica que se genera cuando un contador llega a 0.
  El WDT acepta diferentes periodos de tiempo, nosotros utilizaremos SLEEP_4S Y SPLEEP_8S, en dicha tabla el máximo valor posible es 8 segundos, pero nosotros utilizamos tres bucles de 8 segundos y uno de 4 segundos para llegar al valor de 28

segundos, sumándole los 2 segundos aproximados que requieren los comandos de lectura, impresión y transmisión, y con esto llegamos a los 30 segundos, que son los requeridos en la consigna de nuestro trabajo práctico.
Las otras dos líneas que acompañan al modo Sleep son: ADC_OFF y BOD_OFF, donde:
ADC_OFF: Apaga los convertidores Analógicos/digitales,
BOD_OFF: Apaga el circuito de Brown Out Detection, que es un circuito para detector niveles peligrosamente bajos de tensión, no necesitamos esta parte ya que el móduloregulador de tensión ya posee dicha función.

- **Librería <LiquidCrystal_I2C.h>** (utilizado en el receptor de prueba)

  Esta librería fue de mucha utilidad para el manejo de del display 16x2(16 caracteres, 2 filas) a través del protocolo I2C el cual permite la reducción de los 16 pines disponibles en el display a 4 pines, los cuales son: Vcc (alimentación) conectada a 5v, GND(masa), SDA(serial Data) conectada al pin analógico A4 y SCL(serial Clock) conectado al pin analógico A5.

  - **LiquidCrystal_I2C lcd(0x27, 2, 1, 0, 4, 5, 6, 7, 3, POSITIVE);**

  Función constructor, crea un objeto de la clase LiquidCrystal_I2C, con dirección, columnas y filas indicadas. Aquí se configuran los pines asignados a la pantalla del PCF8574.

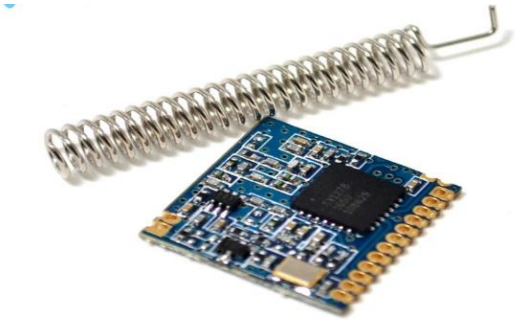
  - **lcd.begin(16,2)** ;

  Inicializa el LCD para 16x2

  - **init();** Inicializa el módulo adaptador LCD a I2C, esta función internamente configura e inicializa el I2C y el LCD.
  - **clear();** Borra la pantalla LCD y posiciona el cursor en la esquina superior izquierda (posición (0,0)).
  - **setCursor(col, row);** Posiciona el cursor del LCD en la posición indicada por col y row(x,y); es decir, establecer la ubicación en la que se mostrará posteriormente texto escrito para la pantalla LCD.
  - **print()/write();** Escribe un texto o mensaje en el LCD, su uso es similar a un Serial.print.
  - **backlight();** Enciende la Luz del Fondo del LCD.
  - **noBacklight();** Apaga la Luz del Fondo del LCD.

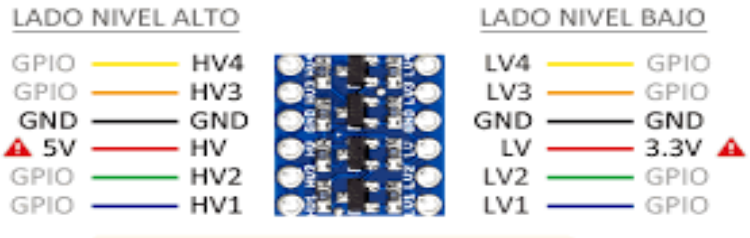
Hardware

Dicho y explicado el funcionamiento de las librerías, protocolos y entre otras cosas necesarias para el correcto funcionamiento, se pasará a explicar el hardware a utilizado.
Como es claro, se utilizó un módulo LoRa. Éste es el XL 1278-SMT en donde en su interior se programa el SX1278.

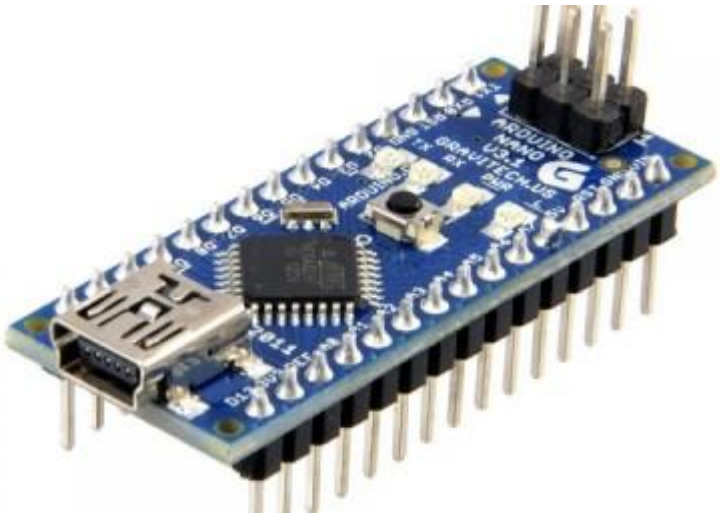


Este tiene un precio actual de $350(AR); su frecuencia de transmisión es de 433 MHz. Esta frecuencia no está contemplada dentro del rango de frecuencias para este fin en Argentina.

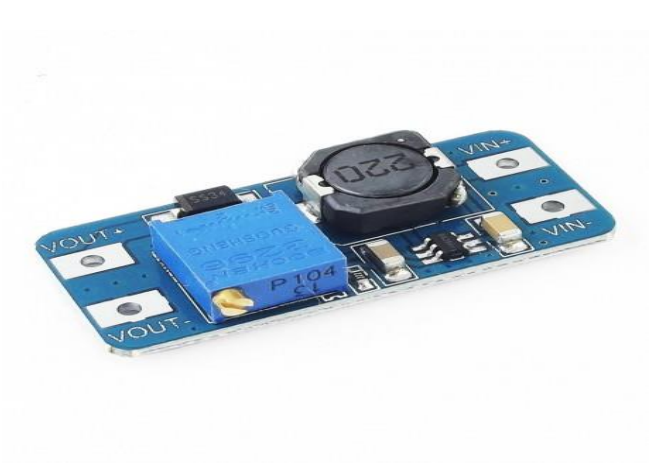
Otro módulo utilizado es un conversor bidireccional de 5v-3v3. Este es muy útil debido a que la tensión de trabajo del módulo LoRa es 3v3. Que sea bidireccional es importante porque el puerto SPI del arduino envía y recibe datos de 5v.

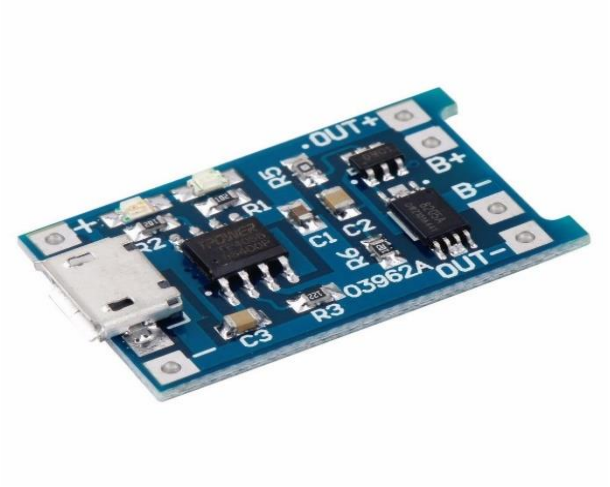


Para este trabajo decidimos utilizar dos arduinos NANO, debido a que los disponíamos y a que su consumo es menor que el UNO y MEGA.

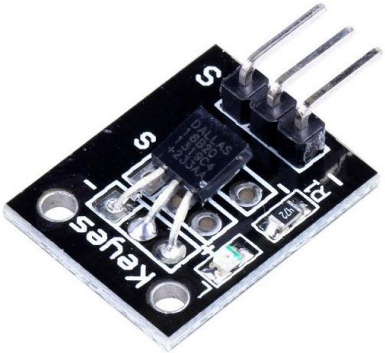


Luego, propusimos alimentar el emisor con una batería de 9v, pero, analizando que éstas no son recargables y además que tendríamos una pérdida de energía ya que el arduino posee un regulador (AMS1117-5.0) a 5V, disipando la tensión extra que se le sumistra . Por esto decidimos alimentar la placa con una batería pequeña de 3,7[V] nominales. Esto trajo como consecuencia agregar un cargador de baterías y un elevador de tension (UP CONVERTER). Ambos módulos también los disponíamos y a un costo muy accesible. Cabe aclarar que el elevador de tensión es necesario debido a que debemos alimentar el arduino con una tensión minima de 6,5V (por el puerto Vin).

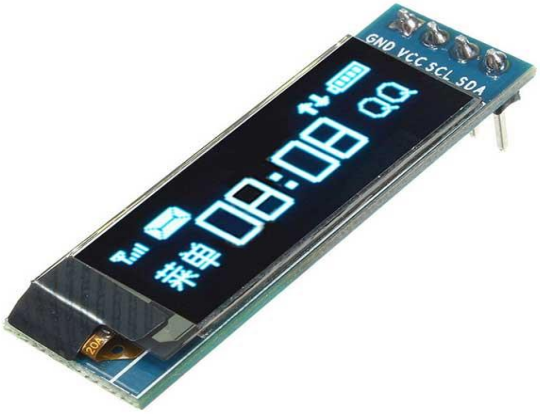




Luego como sensor de temperatura, utilizamos otro módulo llamado "DALLAS 18B20 1713C4". Teniendo como ventaja, que éste sólo consume cuando se le solicita la temperatura, lo que nos permite ahorrar energía.

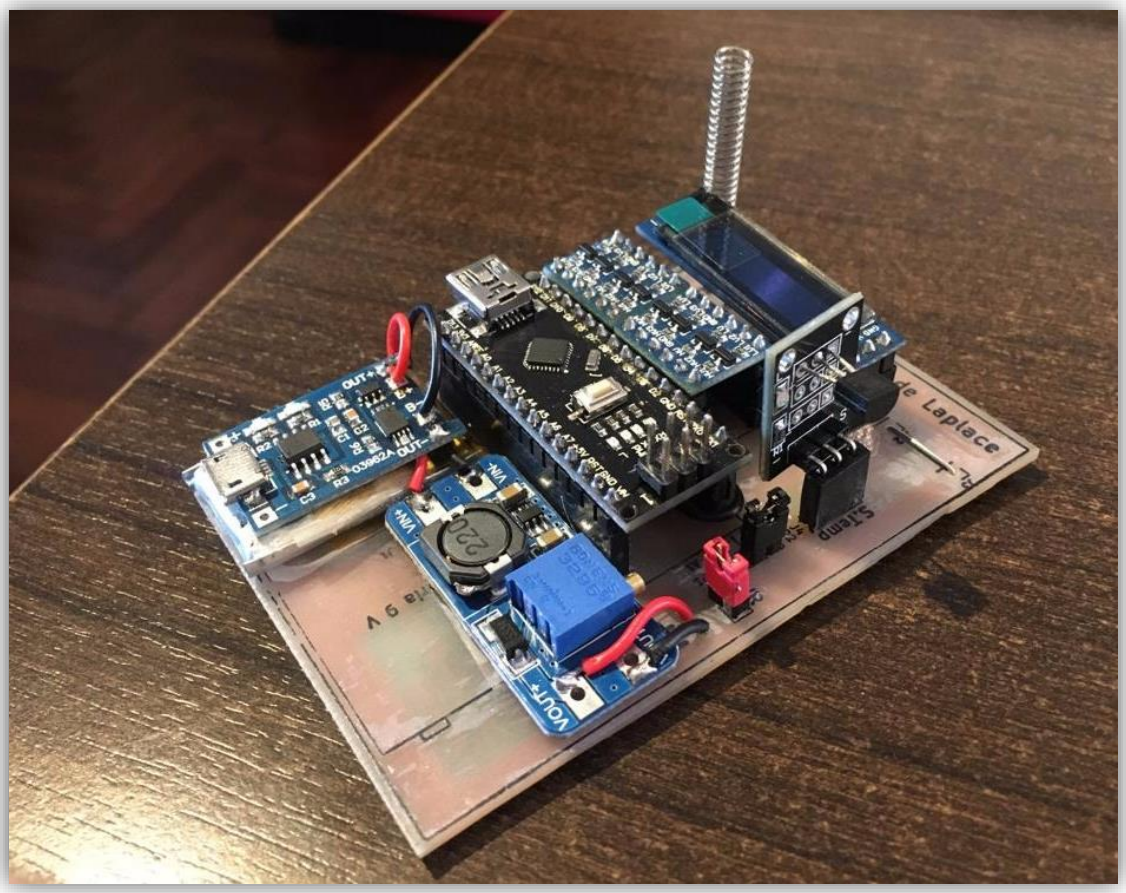


También se utilizó un pequeño display (módulo DIY SSD1306) con el fin de facilitar la lectura local de temperatura, el cual trabaja con el protocolo I2C. Éste display es de tecnología OLED la cuál es de muy bajo consumo.



Por último, en el receptor pusimos un display I2C para poder hacer pruebas y mediciones sin necesidad de disponer de una computadora.

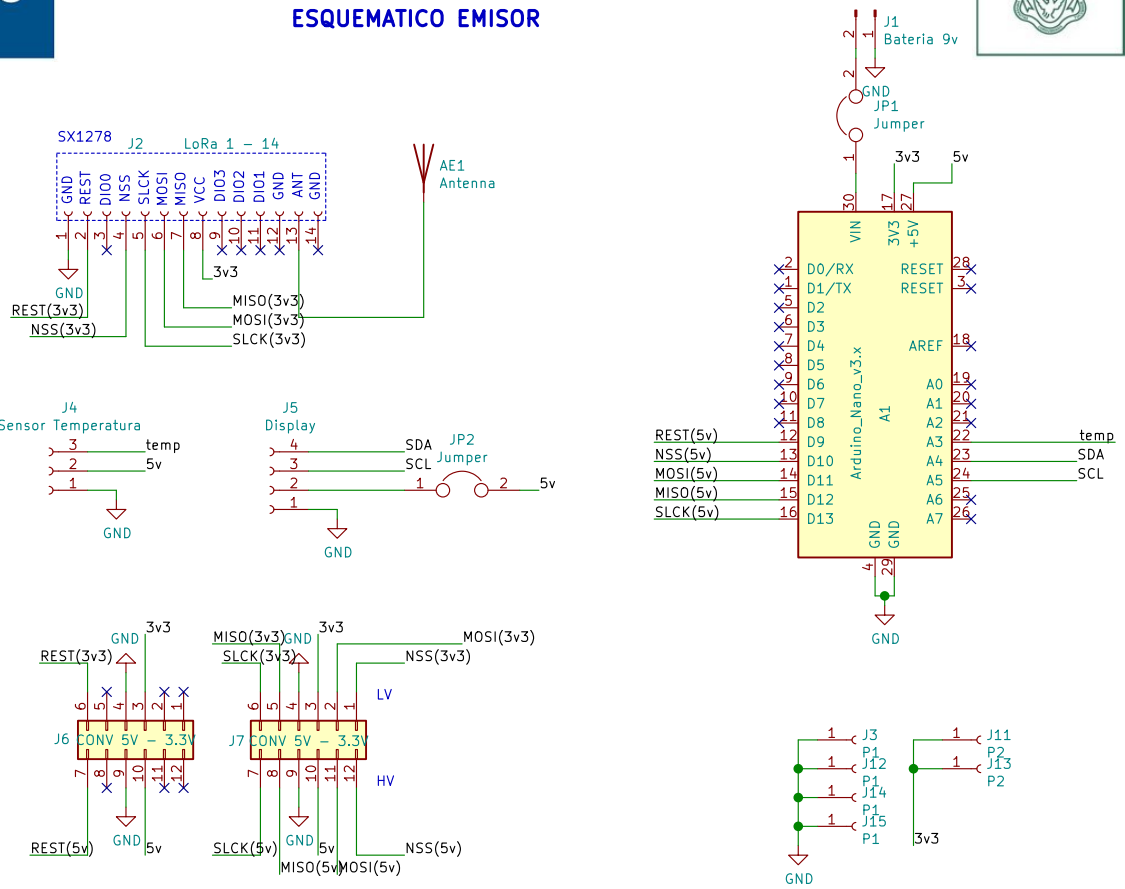
A continuación, presentamos el esquemático del trabajo realizado y el diseño de la placa PCB correspondiente a nuestro circuito.



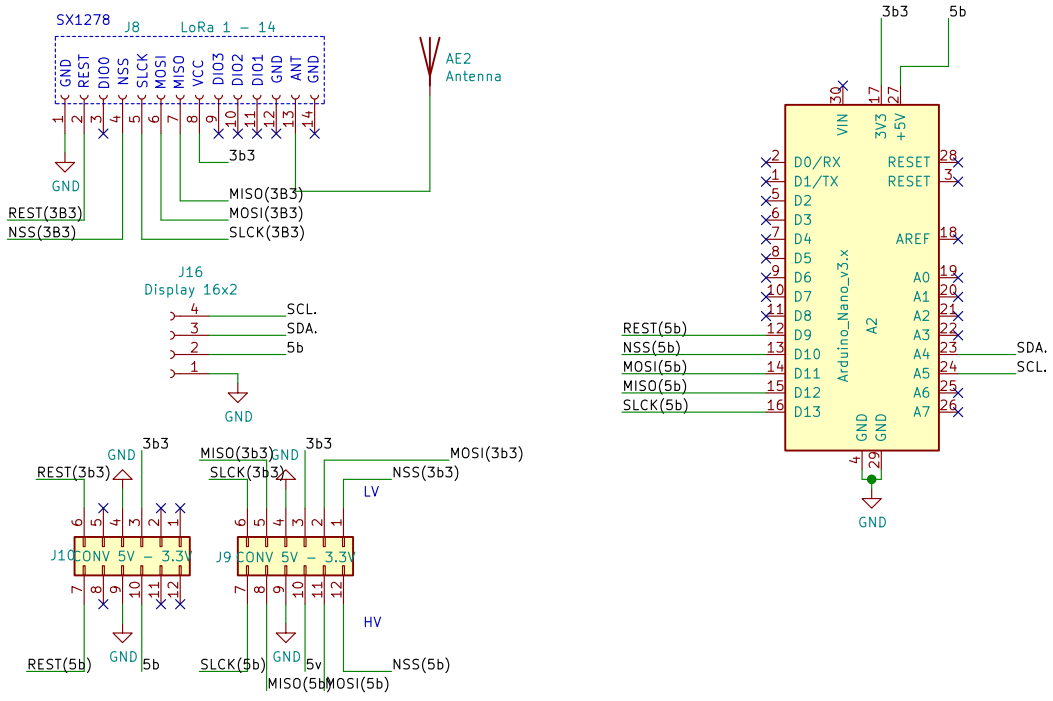
UNC

FCEFyN

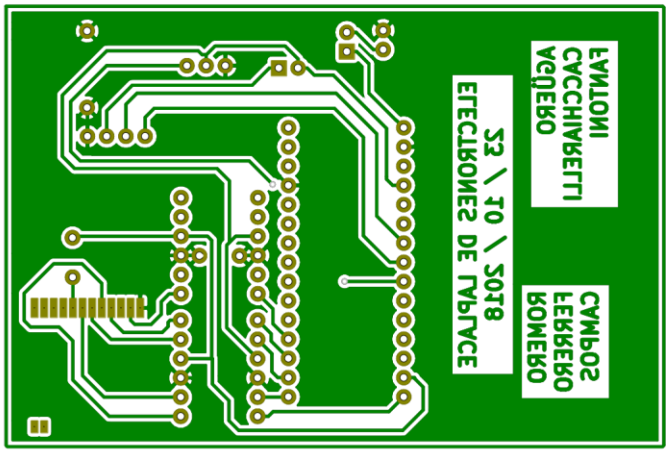
ESQUEMATICO EMISOR

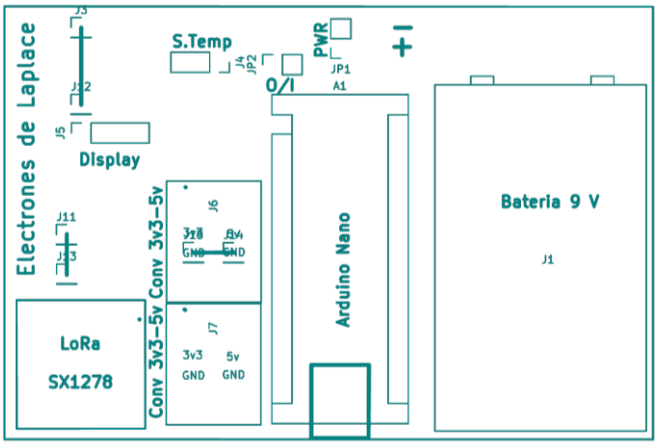


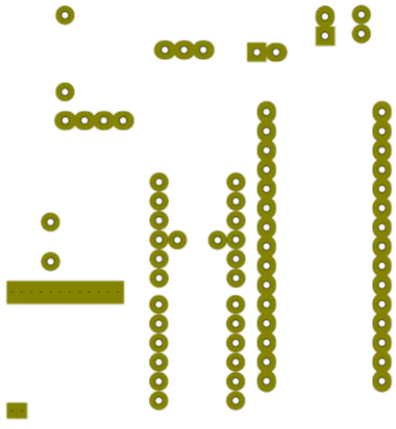
ESQUEMATICO RECEPTOR



LoRa SX1278		
Sheet: / File: loraemisor.sch		
Title: Trabajo Practico Teoria de las Comunicaciones		
Size: A4	Date: 2018-10-20	**Rev:**
KiCad E.D.A. kicad (5.0.0)		Id: 1/1



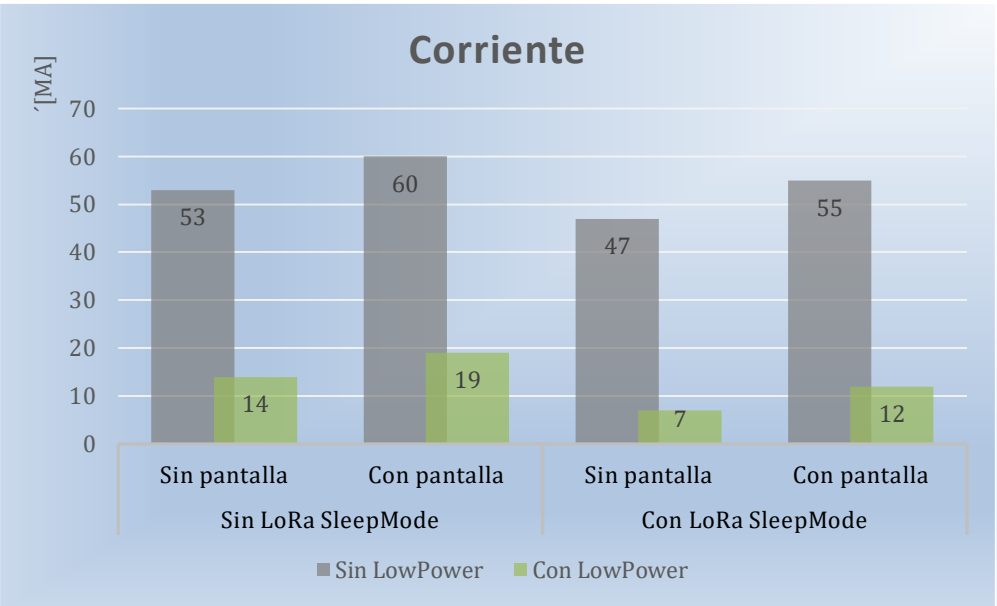




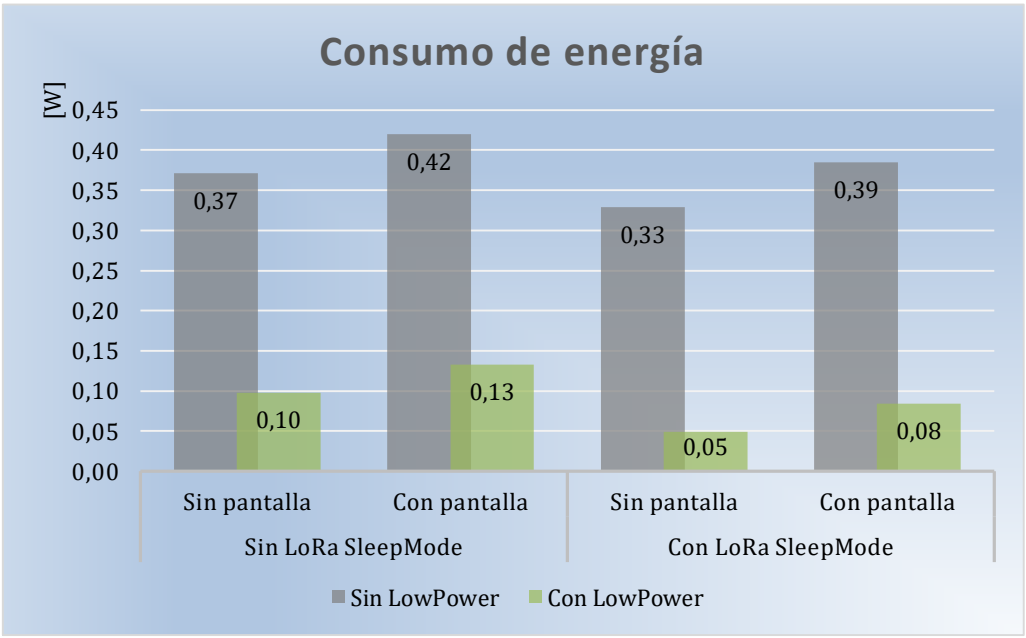
LoRa SX1278		
Sheet: File: loraemisor.kicad_pcb		
Title: Trabajo Practico Teoria de las Comunicaciones		
Size: A4	Date: 2018-10-26	**Rev:**
KiCad E.D.A. kicad (5.0.0)		Id: 1/1

Análisis de Potencia

	Sin LoRa SleepMode		Con LoRa SleepMode	
	Sin pantalla	**Con pantalla**	**Sin pantalla**	**Con pantalla**
Sin LowPower[mA]	53	60	47	55
Con LowPower[mA]	14	19	7	12



	Sin LoRa SleepMode		Con LoRa SleepMode	
	Sin pantalla	**Con pantalla**	**Sin pantalla**	**Con pantalla**
Sin LowPower[W]	0,37	0,42	0,33	0,39
Con LowPower[W]	0,10	0,13	0,05	0,08



Resultados experimentales del análisis de Potencia y alcance de la comunicación

Posición:	Plaza Vélez Sardfield		80m							
Potencia[mW]	2		5		10		15		20	
	RSSI[dbM]	SNR[dB]	RSSI[dbM]	SNR[dB]	RSSI[dbM]	SNR[dB]	RSSI[dbM]	SNR[dB]	RSSI[dbM]	SNR[dB]
Medición 1	-102	5,25	-100	2,50	-89	7,50	-78	8,75	-82	8,50
Medición 2	-98	5,00	-92	7,25	-89	8,25	-78	8,50	-81	9,25
Medición 3	-96	6,25	-90	8,50	-84	8,50	-79	8,75	-83	9,00
Medición 4	-99	3,75	-93	8,00	-82	9,25	-94	6,50	-90	8,00
Medición 5	-97	6,00	-91	7,75	-81	8,75	-88	8,75	-85	9,00
Media	**-98**	**5,25**	**-93**	**6,80**	**-85**	**8,45**	**-83**	**8,25**	**-84**	**8,75**

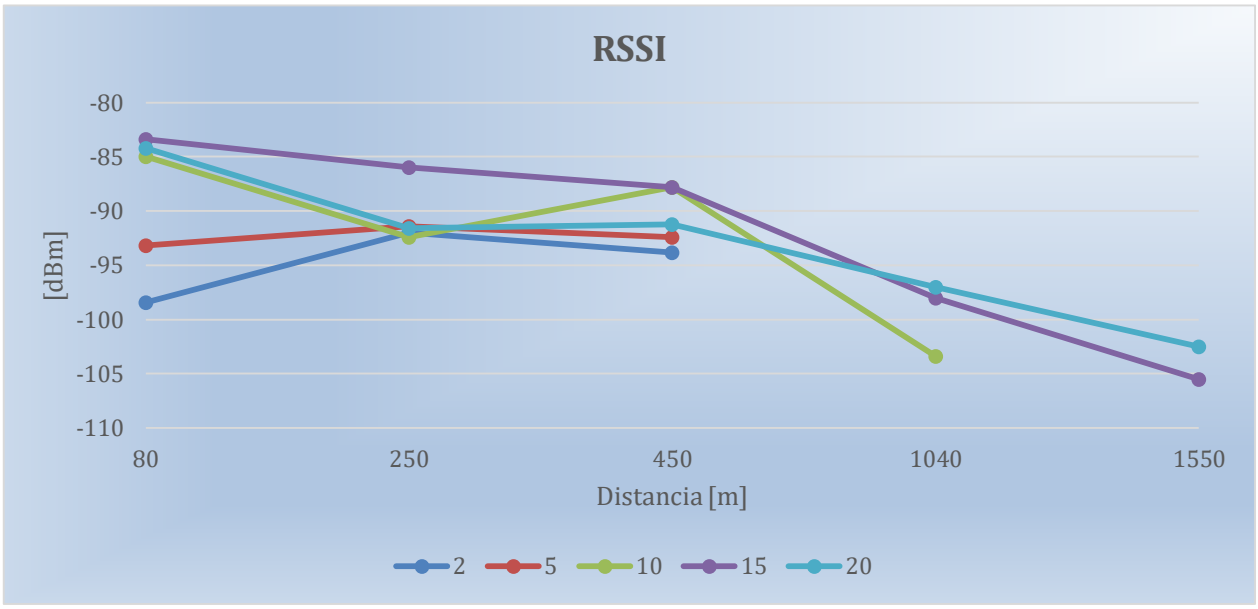
Posición:	Calle Obispo Trejo		250m							
Potencia[mW]	2		5		10		15		20	
	RSSI[dbM]	SNR[dB]	RSSI[dbM]	SNR[dB]	RSSI[dbM]	SNR[dB]	RSSI[dbM]	SNR[dB]	RSSI[dbM]	SNR[dB]
Medición 1	-88	8,25	-98	2,75	-96	4,00	-85	8,50	-92	6,50
Medición 2	-89	8,25	-98	3,50	-90	8,00	-85	8,75	-91	8,50
Medición 3	-90	8,75	-87	8,00	-89	8,25	-99	2,25	-90	8,50
Medición 4	-94	5,25	-89	8,50	-93	8,25	-80	10,00	-97	9,25
Medición 5	-99	2,25	-85	7,25	-94	6,50	-81	8,75	-88	6,50
Media	**-92**	**6,55**	**-91**	**6,00**	**-92**	**7,00**	**-86**	**7,65**	**-92**	**7,85**

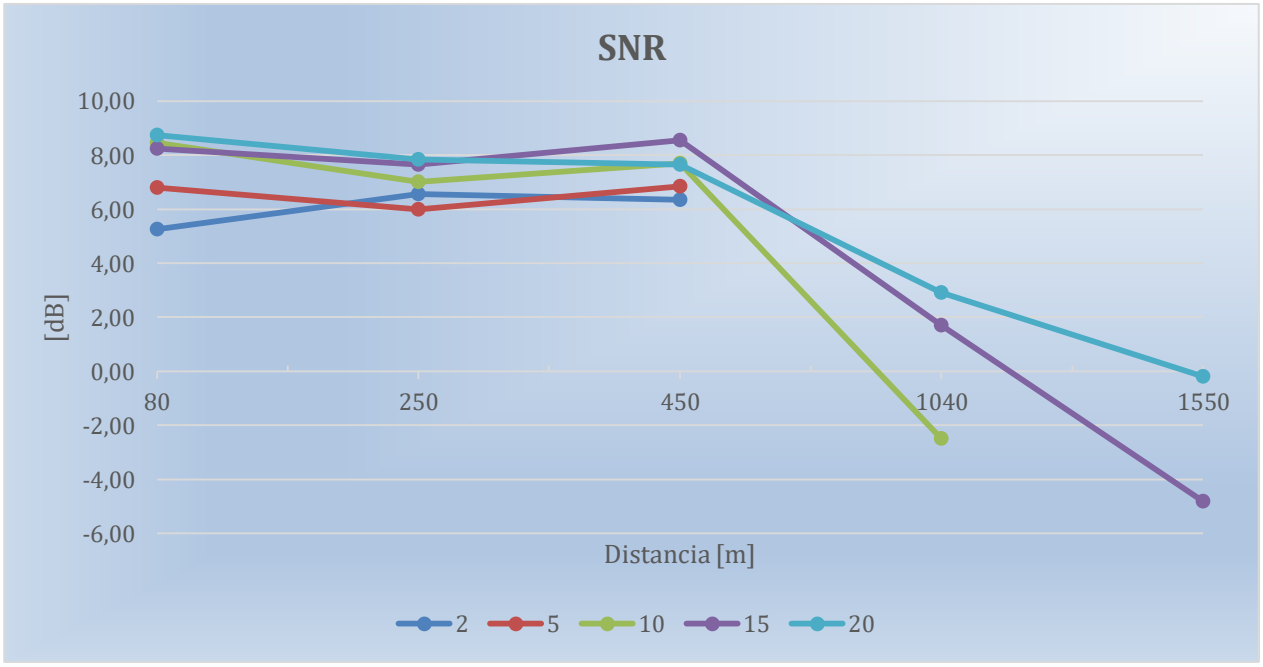
Posición:	Paseo del Buen Pastor		450m							
Potencia[mW]	2		5		10		15		20	
	RSSI[dbM]	SNR[dB]	RSSI[dbM]	SNR[dB]	RSSI[dbM]	SNR[dB]	RSSI[dbM]	SNR[dB]	RSSI[dbM]	SNR[dB]
Medición 1	-90	7,75	-92	7,75	-88	7,25	-94	8,00	-94	7,50
Medición 2	-92	6,50	-92	6,50	-89	7,50	-89	8,00	-91	7,75
Medición 3	-91	7,50	-95	7,00	-96	6,00	-86	8,75	-86	8,75
Medición 4	-96	7,25	-92	6,00	-83	8,50	-85	9,00	-90	7,50
Medición 5	-100	2,70	-91	7,00	-83	9,25	-85	9,00	-95	6,75
Media	**-94**	**6,34**	**-92**	**6,85**	**-88**	**7,70**	**-88**	**8,55**	**-91**	**7,65**

Posición:	Plaza España		1,04 Km							
Potencia[mW]	2		5		10		15		20	
	RSSI[dbM]	SNR[dB]	RSSI[dbM]	SNR[dB]	RSSI[dbM]	SNR[dB]	RSSI[dbM]	SNR[dB]	RSSI[dbM]	SNR[dB]
Medición 1	-	-	-	-	-103	-3,75	-98	4,00	-100	-1,50
Medición 2	-		-	-	-107	-7,25	-95	4,00	-94	5,00
Medición 3	-	-	-	-	-101	1,25	-103	-4,00	-102	-2,25
Medición 4	-	-	-	-	-106	-6,25	-101	-0,50	-96	5,75
Medición 5	-	-	-	-	-100	3,50	-93	5,00	-93	7,50
Media	**-**	**-**	**-**	**-**	**-103**	**-2,50**	**-98**	**1,70**	**-97**	**2,90**

Posición:	Parque Sarmiento		1,55 Km							
Potencia[mW]	2		5		10		15		20	
	RSSI[dbM]	SNR[dB]	RSSI[dbM]	SNR[dB]	RSSI[dbM]	SNR[dB]	RSSI[dbM]	SNR[dB]	RSSI[dbM]	SNR[dB]
Medición 1	-	-	-	-	-	-	-106	-4,00	-103	-8,50
Medición 2	-	-	-	-	-	-	-103	-7,50	-101	-4,50
Medición 3	-	-	-	-	-	-	-106	-4,25	-102	7,25
Medición 4	-	-	-	-	-	-	-107	-3,50	-104	5,00
Medición 5	-	-	-	-	-	-	-106	-4,81	-103	-0,19
Media	**-**	**-**	**-**	**-**	**-**	**-**	**-106**	**-4,81**	**-103**	**-0,19**

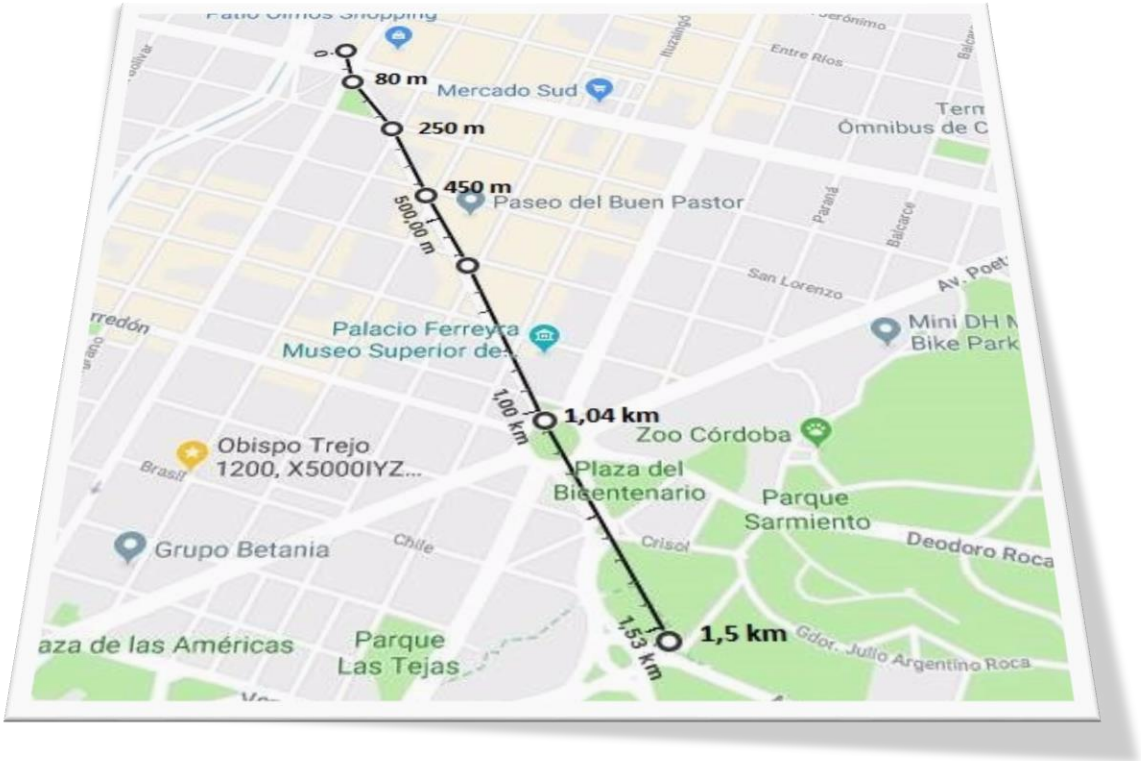
Gráficos de intensidad de señal y relación señal ruido





Las correspondientes mediciones de alcance del radioenlace, fueron realizadas a lo largo de la calle H. Yrigoyen y con el dispositivo emisor a una altura elevada (aprox. 40m), para minimizar posibles obstáculos.

Previamente se escogieron cinco puntos para realizar la comunicación:



En cada punto se realizaron un total de veinticinco mediciones, cinco por cada valor de potencia de transmisión, las cuáles propusimos 2, 5, 10, 15, 20.
Dicha experiencia superó nuestras expectativas, ya que en nuestras primeras transmisiones en la facultad alcanzaban un máximo de 100m.
Luego de haber realizado las mediciones entre los diferentes puntos, pudimos determinar, como era de esperarse, que la intensidad de la señal disminuye a medida que la distancia entre los dispositivos aumenta. Notamos también que cuando el valor de SNR era menor que -10, la información recibida presentaba distorsión (recibíamos caracteres no deseados).
Como se puede apreciar en el gráfico, cuando trabajábamos con máxima potencia, la relación señal ruido se mantenía más estable al aumentar las distancias.

Conclusión:

El presente Trabajo Práctico fue de suma utilidad para llevar el desarrollo de la materia a un ámbito práctico, fortalecer el trabajo en equipo y realizar un proyecto de mucha utilidad y de una posible extensión del mismo en un futuro.
Los resultados obtenidos en relación al enlace de radiofrecuencia realizado fueron mayores a los esperados, resultando el más sorprendente la capacidad de alcance cercana a los 2 km.
Dicho proyecto, el cual surgió como una forma alternativa de evaluación, fue de total aceptación por el grupo que se vio reflejado en las constantes reuniones, donde se fueron introduciendo mejoras continuas al dispositivo.
Cabe destacar la importancia de este trabajo como la apertura hacia el nuevo mundo de Internet of Things (IoT) o Internet de las cosas, de gran repercusión en la actualidad y en un futuro no muy lejano, de donde se obtuvo bastante información de la Web, de todo tipo y con las más sorprendentes variantes.

Bibliografía:

- DISPOSITIVO LoRa IoT DE COMUNICACIÓN A LARGO ALCANCE Y BAJO CONSUMO ENERGÉTICO. Youtube: https://www.youtube.com/watch?v=7Rud8FemIl0
- RODRÍGUEZ MUNCA, José Daniel (2016). *Dispositivo LoRa de comunicación a largo alcance y bajo consumo energético para aplicaciones del ámbito del desarrollo*. Tutor académico: Sierra Castañer, Manuel. Tutor profesional: Gutiérrez Martín, Álvaro. Trabajo fin de Master. Madrid, España.
- PierAisa #346: Test Moduli LoRa Ra-02 SX1278 con Arduino. Youtube: https://youtu.be/v58KixUgIsQ
- RAMOS, Francisco. *Pérdidas en obstáculos.* http://www.radioenlaces.es/articulos/perdidas-en-obstaculos/
- WIKIPEDIA. *Serial Peripheral Interface.*https://es.wikipedia.org/wiki/Serial_Peripheral_Interface
- WIKIPEDIA. *LoRaWAN*. https://es.wikipedia.org/wiki/LoRaWAN
- WIKIPEDIA. *Indicador de fuerza de la señal recibida.* https://es.wikipedia.org/wiki/Indicador_de_fuerza_de_la_se%C3%B1al_recibida
- FERNÁNDEZ PÉREZ, M. Aprendiendo Arduino. Bus SPI. https://aprendiendoarduino.wordpress.com/2016/11/13/bus-spi/
- COMUNIDAD ELECTRÓNICOS. https://www.comunidadelectronicos.com/
- MEDIUM. *Haciendo IoT con LoRa: Capítulo 1.- ¿Qué es LoRa y LoRaWAN?.* https://medium.com/beelan/haciendo-iot-con-lora-cap%C3%ADtulo-1-qu%C3%A9-es-lora-y-lorawan-8c08d44208e8
- MEDIUM. *Haciendo IoT con LoRa: Capitulo 2.- Tipos y Clases de Nodos.* https://medium.com/beelan/haciendo-iot-con-lora-capitulo-2-tipos-y-clases-de-nodos-3856aba0e5be
- GITHUB. LoRa API. https://github.com/sandeepmistry/arduino-LoRa/blob/master/API.md
- thethingsnetwork.org
- Libro: "Teoría de las Comunicaciones" de P. E. Danizio.