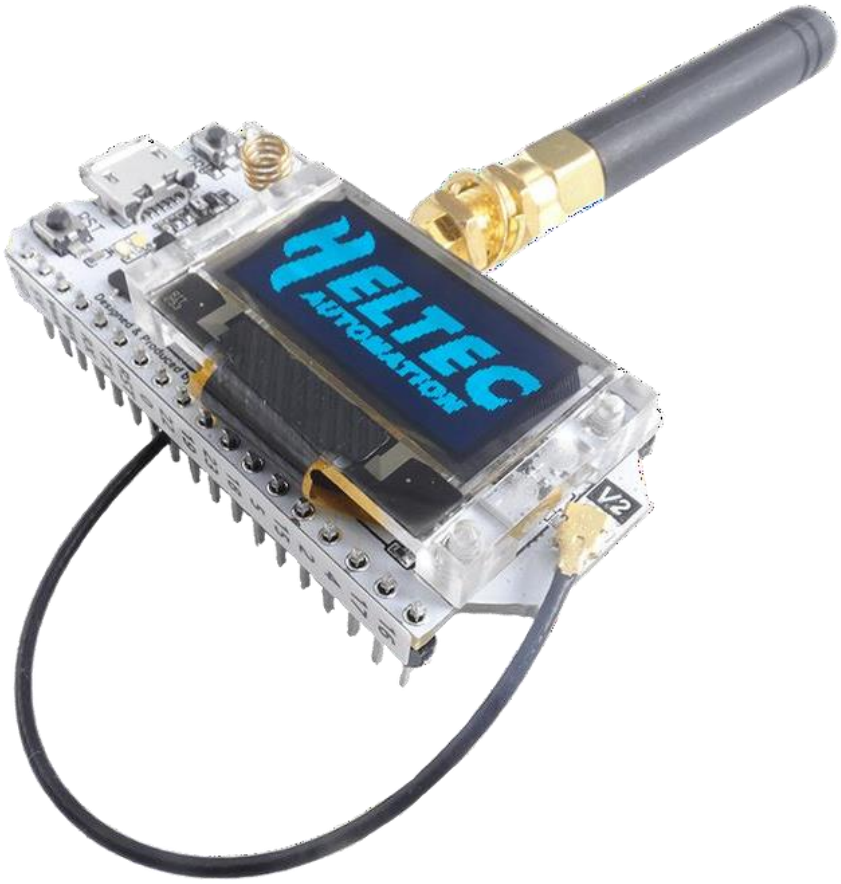
Telemetría



Alumnos: Matías Galliano, Ignacio Makara, Urbizu Thiago

Profesor: José Basgall y Hernán Varela

Índice

Introducción

Para empezar, **¿Qué es la telemetría?**
La telemetría es una tecnología que permite la medición remota de magnitudes físicas y el posterior envío de la información hacia el operador del sistema. El objetivo principal de un sistema de telemetría es permitir la detección de magnitudes (por ejemplo, la temperatura ambiente en la que funciona una máquina) sin estar físicamente en las inmediaciones del objeto a medir, y sin tener que estar sujeto a limitaciones del tiempo.

La tecnología LoRa o HTCC son inalámbricas (al igual que WiFi, Bluetooth, LTE, SigFox o Zigbee) que emplea un tipo de modulación en radiofrecuencia patentado por Semtech, una importante empresa fabricante de chips de radio. La tecnología de modulación se denomina Chirp Spread Spectrum (o CSS) y se emplea en comunicaciones militares y espaciales desde hace décadas.

Sus ventajas:

- Alta tolerancia a las interferencias
- Alta sensibilidad para recibir datos (-168dB)
- Basado en modulación “chirp“
- Bajo Consumo (hasta 10 años con una batería)
- Largo alcance 10 a 20 km
- Baja transferencia de datos (hasta 255 bytes)
- Conexión punto a punto
- Frecuencias de trabajo: 868 Mhz en Europa, 915 Mhz en América, y 433 Mhz en Asia

Son tecnologías ideales para conexiones a grandes distancias y para redes de IoT en las que se necesitan sensores que no dispongan de corriente eléctrica de red, teniendo grandes aplicaciones:

- Para Smart Cities (ciudades inteligentes)
- En lugares con poca cobertura (cómo explotaciones agrícolas o ganaderas en el campo)
- Para construir redes privadas de sensores y/o actuadores.

También estas tecnologías nos muestran, mediante una pantalla led, los datos del sensor para su visualización de la siguiente manera

- Gráficos en tiempo real de datos de sensores
- Cuadros de mando dinámicos
- Funciones de agregación Historial de datos ilimitado
- Acceso a datos a través de API, incluidos conectores para R-Script y Python
- Alarmas por correo electrónico
- Sitio de demostración en tiempo real

Elección del dispositivo

LoRa (HTCC-WB32LA-F)

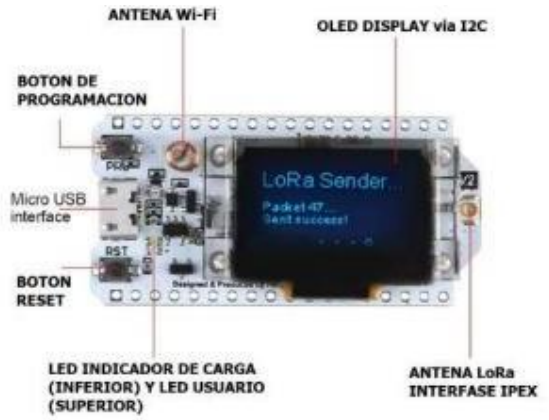
Decidimos elegir el **LoRa (**HTCC-WB32LA-F), debido a que permite una comunicación de largo alcance, lo que lo hace ideal para la transmisión de las distintas magnitudes que necesita nuestro vehículo.
Además, cuenta con una pantalla oled, al cual es lo suficientemente grande para poder leer los mensajes.

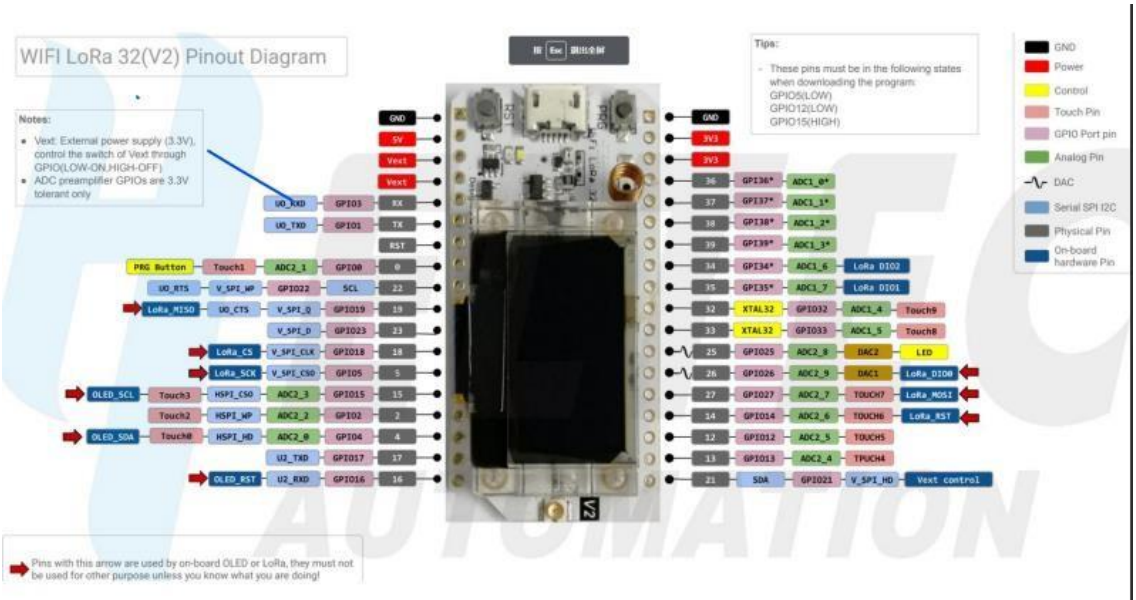
Concluimos que, en base a todos los modelos que tenemos a elección, este es uno de los candidatos porque no es, ni demasiado potente, ni demasiado débil.

- Microprocesador: ESP32 (dual-core 32-bit MCU + ULP core), con ESP32 chip SX1276/SX1278;
- Micro USB interfaz con un regulador de voltaje, protección ESD, con un escudo anti cortocircuitos y una protección de señales RF para prevenir interferencias

- Pantalla OLED de matriz de puntos integrada de 0,96 pulgadas y 128 * 64 puntos, que se puede usar para mostrar datos de sensores
- Soporte de Arduino Dev

Diagrama de pines LoRa 32





Parámetros	**Descripción**
Master Chip	ESP32 (240MHz Tensilica LX6 dual-core+1 ULP, 600 DMIPS)
LoRa Chipset	SX1276/SX1278
USB to Serial Chip	CP2102
Frecuencia	470~510 MHz, 863~923 MHz
Max TX Power	19dB ± 1d
Sensibilidad de recepción	-135 dBm
Wi-Fi	802.11 b/g/n (802.11n up to 150 Mbps)
Bluetooth	Bluetooth V4.2 BR/EDR
Hardware Resource	UART x 3; SPI x 2; I2C x 2; I2S x 1; 12-bits ADC input x 18; 8-bits DAC output x 2; GPIO x 22, GPI x 6
Memoria	8MB(64M-bits) SPI FLASH; 520KB internal SRAM
Interface	Micro USB x 1; LoRa Antenna interface(IPEX) x 1; 18 x 2.54 pin x 2
Bateria	3.7V Lithium（SH1.25 x 2 socket
Temperatura de operacion	-20 ~ 70 ℃
Dimensiones	51 x 25.5 x 10.6 mm
Low Power	Deep Sleep 800μA
Display Size	0.96-inch OLED

Power supply	Minimo	Tipico	Maximo
USB powered}（≥500mA）	4,7V	5V	6V
Lithium battery(≥250mA)	3.3V	3,7V	4,2V
5V pin(≥500mA)	4,7V	5V	6V
3V3 pin(≥150mA)	2,7V	3,3V	3,5V

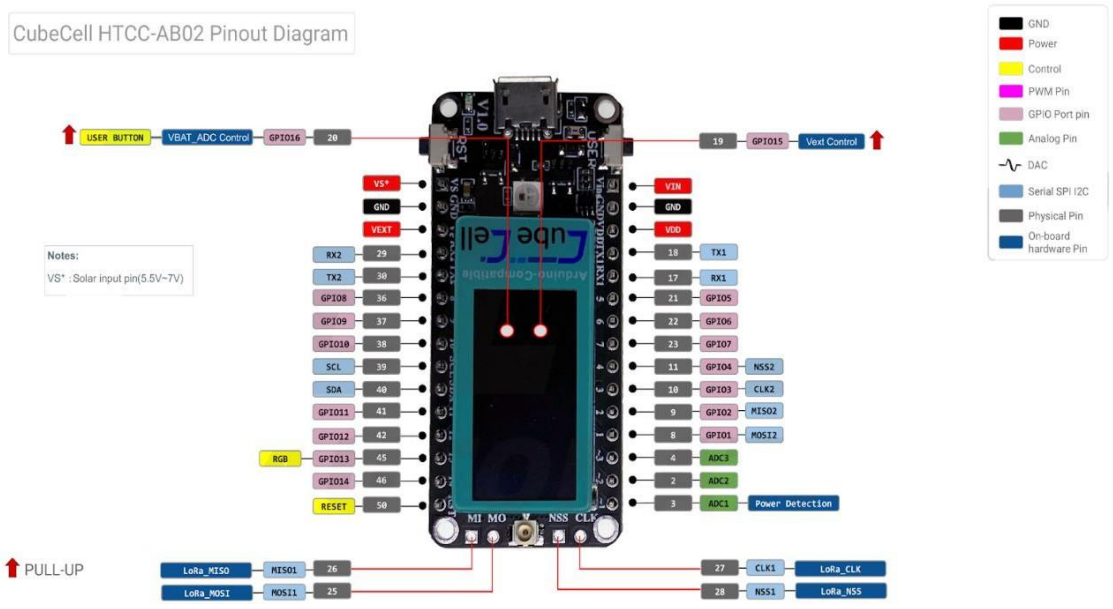
HTCC-AB02 (HTCC-AB02-F)

Elegimos este modelo porque a comparación del HTCC-AB02S este puede ser una variante especializada para seguridad o censado ambiental, y no es lo que estamos buscando, aparte puede ofrecer diferentes variantes de conectividad y no es lo que necesitamos.

Creemos que estas son las mejores opciones de acuerdo a lo que vamos a usar, ya que en el caso del módulo Wireless Stick, su pantalla es demasiado pequeña y no la correcta visualización de los mensajes.

El HTIT-Tracker, es demasiado potente y caro para un uso demasiado sencillo como el que le podemos llegar a dar en la telemetría.

Diagrama de pines HTCC-AB02



Parámetros	**Descripción**
Master Chip	ASR6502 (48 MHz ARM® Cortex® M0+ MCU
LoRa Chipset	SX1262
USB to Serial Chip	CP2102
Frecuencia	470~510 MHz, 863~928 MHz
Max TX Power	22 ± 1 dBm
Sensibilidad de recepción	-135 dBm
Energía Solar	5.5～7V solar panel
Low Power	Deep Sleep 3.5μA
Hardware Resource	UART x 2; SPI x 2; I2C x 2; SWD x 1; 12-bits ADC input x 3; 8– channel DMA engine; GPIO x 16
Memoria	128KB internal FLASH; 16KB internal SRAM
Interface	Micro USB x 1; LoRa Antenna interface(IPEX) x 1; 15 x 2.54 pin x 2+2 x 2.54 pin x 3
Batería	3.7V Lithium（SH1.25 x 2 socket）
Temperatura de operación	-20 ~ 70 ℃
Dimensiones	51.9 x 25 x 8 mm
Display Size	0.96-inch OLED

Power supply	Minimo	Tipico	Maximo
USB powered}（≥500mA）	4,7V	5V	6V
Lithium battery(≥250mA)	3,3V	3,7V	4,2V
5V pin(≥500mA)	4,7V	5V	6V
3V3 pin(≥150mA)	2,7V	3,3V	3,5V

Códigos

Código transmisor:

```
#include <LoRa.h>
#include <SPI.h>

//Libraries
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

//Debemos definir los pines
#define SCK 5
#define MISO 19
#define MOSI 27
#define SS 18
#define RST 14
#define DIO0 26

//Aca definimos frecuencia
#define BAND 915E6

//Definimos los pines necesarios para conectar con pantalla OLED
#define ANCHOPANTALLA 128 // El ancho de la pantalla en pixeles es de 128px
#define ALTOPANTALLA 64 // El ancho de la pantalla en pixeles es de 64px
#define OLED_SDA 4
#define OLED_SCL 15
#define OLED_RST 16

int  Contador = 0;//Haremos un contador de paquetes enviados

Adafruit_SSD1306 display(ANCHOPANTALLA, ALTOPANTALLA, &Wire,
OLED_RST);

void setup() {

  Serial.begin(115200);//inicia monitor serial

  pinMode(OLED_RST, OUTPUT);//reseteamos la pantalla OLED para comenzar
  digitalWrite(OLED_RST, LOW);
  delay(20);
  digitalWrite(OLED_RST, HIGH);

  Wire.begin(OLED_SDA, OLED_SCL); //inicia OLED
if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3c, false, false)) { // 0x3C representa
128x32
    Serial.println(F("Fallo iniciando SSD1306"));
```

```
    for(;;); // Si detecta el fallo anterior, detiene el código aca hasta que reinicie
  }

  display.clearDisplay();//Borramos pantalla
  display.setTextColor(WHITE);//Definimos texto color blanco
  display.setTextSize(1);//Tamao de fuente a 1 punto
  display.setCursor(0,0);//Comenzamos a graficar desde coordenadas 0,0
  display.print("TRANSMISOR LORA ");
  display.display();

  Serial.println("Prueba de envio LoRa");

  SPI.begin(SCK, MISO, MOSI, SS);  //Definimos pines SPI
  LoRa.setPins(SS, RST, DIO0); //Configuramos el LoRa para enviar

  if (!LoRa.begin(BAND)) {//Intenta transmitir en la banda elegida
    Serial.println("Error iniciando LoRa");//Si no puede transmitir, marca error
    while (1);
  }
  Serial.println("Inicio exitoso de LoRa!");//Mensaje de todo bien en puerto serial
  display.setCursor(0,10);
  display.print("Inicio exitoso de LoRa!");//Mensaje de todo bien en pantalla OLED
  display.display();
  delay(2000);//Esperamos un par de segundos
}

void loop() {

  Serial.print("Enviando paquete: ");//Muestra mensaje
  Serial.println( Contador);//Muestra la cuenta actual


  //Para mandar paquete al LoRa receptor
  LoRa.beginPacket();//Inicia protocolo
  LoRa.print( Contador);//Manda cuenta actual
  LoRa.endPacket();//Fin de paquete enviado

  display.clearDisplay();//Limpia pantalla
  display.setCursor(0,0);//Posicionamos en siguiente renglón
  display.setTextSize(1);//TamaÃ±o de fuente a 1 punto
  display.println("Transmisor LoRa");//
  display.setCursor(0,30);
  display.print("Paquete LoRa enviado.");//Mensaje de confirmación
  display.setCursor(0,50);
  display.print(" Contador:");//Mensaje
  display.setCursor(80,50);
  display.print( Contador);//La cuenta actual que se envía
  display.display();

  Contador++;

  delay(1500);//Esperamos 10 segundos entre cada envió
}
```

Código del receptor:

```
void setup() {
  //initialize Serial Monitor
  Serial.begin(115200);

  pinMode(OLED_RST, OUTPUT);//reseteamos la pantalla OLED para comenzar
  digitalWrite(OLED_RST, LOW);
  delay(20);
  digitalWrite(OLED_RST, HIGH);

  Wire.begin(OLED_SDA, OLED_SCL); //inicia OLED
if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3c, false, false)) { // 0x3C representa
128x32
    Serial.println(F("Fallo iniciando SSD1306"));
    for(;;); // Si detecta el fallo anterior, detiene el codigo aca hasta que se reinicie
  }

  display.clearDisplay();//Borramos pantalla
  display.setTextColor(WHITE);//Definimos texto color blanco
  display.setTextSize(1);//Tamaño de fuente a 1 punto
  display.setCursor(0,10);//Comenzamos a graficar desde coordenadas 0,0
  display.print("RECEPTOR LORA ");
  display.display();

  Serial.println("Prueba de recepcion LoRa");

  SPI.begin(SCK, MISO, MOSI, SS);  //Definimos pines SPI
  LoRa.setPins(SS, RST, DIO0); //Configuramos el LoRa para enviar

  if (!LoRa.begin(BAND)) {//Intenta transmitir en la banda elegida
    Serial.println("Error iniciando LoRa");//Si no puede transmitir, marca error
    while (1);
  }
  Serial.println("Inicio exitoso de LoRa!");//Mensaje de todo bien en puerto serial
  display.setCursor(0,30);
  display.print("Inicio exitoso de LoRa!");//Mensaje de todo bien en pantalla OLED
  display.display();
  delay(2000);//Esperamos un par de segundos
}

void loop() {

  int tamanoPaquete = LoRa.parsePacket();  //analizamos paquete
  if (tamanoPaquete) {//Si nos llega paquete de datos
  Serial.print("Paquete recibido ");//Muestra confirmación

    while (LoRa.available()) {//Leemos el paquete
      DatoLoRa = LoRa.readString();//Guardamos cadena en variable
      Serial.print(DatoLoRa);//Lo imprimimos en monitor serial
    }
```

```
    int rssi = LoRa.packetRssi();//Esto nos imprime la intensidad de señal recibida
    Serial.print(" con RSSI ");
    Serial.println(rssi);

   // Mostramos información captada
   display.clearDisplay();
   display.setCursor(0,0);
   display.print("Receptor LoRa");//Mensaje
   display.setCursor(0,20);
   display.print("Paquete recibido:");//Imprime datos recibidos
   display.setCursor(0,30);
   display.print(DatoLoRa);
   display.setCursor(0,40);
   display.print("RSSI:");//Imprime intensidad de señal
   display.setCursor(30,40);
   display.print(rssi);
   display.display();
  }
}

display.print("

Contador:");//Mensaje
  display.setCursor(80,50);
  display.print( Contador);//La cuenta actual que se envió
  display.display();

  Contador++;

  delay(1500);//Esperamos 10 segundos entre cada envió
}
```

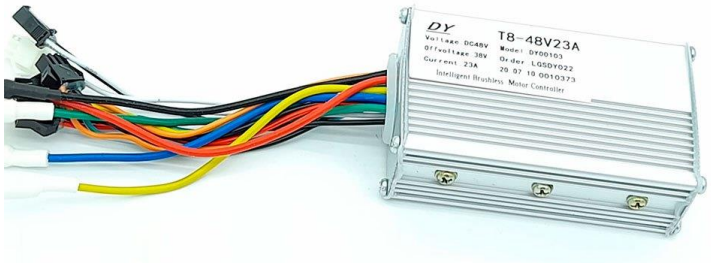
Esp32 Multithread:

```
#include <Arduino.h>
#include <freertos/FreeRTOS.h>
#include <freertos/task.h>

// Pines para los LEDs
const int ledPin1 = 2;
const int ledPin2 = 4;
const int ledPin3 = 5;

void task1(void *pvParameters)
  { while (true) {
  digitalWrite(ledPin1, HIGH);
  delay(10000);
  digitalWrite(ledPin1, LOW);
  delay(10000);
  }
}

void task2(void *pvParameters)
  { while (true) {
  digitalWrite(ledPin2, HIGH);
  delay(5000);
  digitalWrite(ledPin2, LOW);
  delay(5000);
  }
}

void task3(void *pvParameters)
  { while (true) {
  digitalWrite(ledPin3, HIGH);
  delay(3000);
  digitalWrite(ledPin3, LOW);
  delay(3000);
  }
}

void setup() {
  pinMode(ledPin1,
  OUTPUT);
  pinMode(ledPin2,
  OUTPUT);
  pinMode(ledPin3,
  OUTPUT);

  xTaskCreatePinnedToCore(task1, "Task1", 1000, NULL, 1, NULL, 0);
  xTaskCreatePinnedToCore(task2, "Task2", 1000, NULL, 1, NULL, 1);
  xTaskCreatePinnedToCore(task3, "Task3", 1000, NULL, 1, NULL, 1);
}

void loop() {
```

}

Magnitudes a medir

Antes que nada, el motor del eco-auto, es un Motor Brushless 500w con 500RPM 48v



Con una controladora específica para este motor que nos permitirá medir varias de las magnitudes que vamos a nombrar.
Las magnitudes que vamos a medir dentro del auto eléctrico serán las siguientes:

- Estado de las baterías: Para esto utilizaremos un "shunt" lo utilizaremos para medir la corriente eléctrica que fluye a través de un circuito específico. La pantalla integrada en el shunt puede mostrar información relevante sobre la corriente, como la corriente instantánea, la carga de la batería, la eficiencia del sistema y otros datos importantes.

- Velocidad: Por medio de la controladora, vamos a utilizar un cable “speedometer” que es una función que se puede implementar en la controladora del motor para medir y mostrar la velocidad del vehículo y "sale" una señal analógica, que nos muestra que tan rápido vamos

- Temperatura en el motor: mediante un sensor de temperatura, podemos saber si este esta demasiado caliente de lo normal.

- Aceleración: Mediante otro cable de la controladora medimos la aceleración

- Censado del volante: Mediremos también la posición del volante mediante alguna resistencia variable insertada en este, podemos saber hacia qué lado gira el conductor el volante.