

Construcción de una Estación Terrestre Satelital de Código Abierto*

Agustín Costarelli, Ramiro Linares, Rodrigo Pérez, Victor Silva, and Mía Torres López

Ingeniería en Mecatrónica. Universidad Nacional de Cuyo, Facultad de Ingeniería,
Centro Universitario, Mendoza, Argentina

1 Objetivo

Intentar realizar una estación de seguimiento de satélites de baja altura, lo que permitiría obtener algunos parámetros tales como: temperatura, aceleración, velocidad, posición, estado de carga de baterías; y también lograr una subida de datos a un satélite. Mediante la utilización de dos radios LoRa, y utilizando frecuencias muy altas, entre 433MHz y 815MHz, ya que son frecuencias con muy poco ruido atmosférico, y con poca potencia (equipos de miliWatts) y LORA se pueden lograr alcances de 200km. Otro objetivo que también se propuso, es el de aplicar herramientas de gestión de proyectos, para ganar experiencia en cómo planificarse y organizarse con presupuestos.

2 Etapas

El presente proyecto se dividió en tres etapas: la primera trata el envío y decodificación de señales en código morse generadas por nosotros mismos, la segunda cubre la decodificación de señales reales, la tercera comunicación mediante protocolo LoRa, y finalmente, configuración de una estación terrestre con TinyGS.

3 Materiales y Recursos

3.1 Arduino

Arduino es una placa electrónica de hardware libre que utiliza un microcontrolador reprogramable con una serie de pines que permiten establecer conexiones entre el controlador y los diferentes sensores. En un circuito se suele utilizar como fuente de alimentación y “puente” entre los diferentes componentes para lograr que interactúen entre ellos. En nuestro caso, el objetivo es conectar un módulo LoRa a cada Arduino, que funcione correctamente, y que ambos dispositivos puedan comunicarse uno con el otro.

* Proyecto realizado para la cátedra ”Técnicas y Herramientas Modernas”, dictada por el Dr. Ing. Ricardo R. Palma

3.2 LoRa

Las redes LPWAN (Low-Power Wide Area Network) son redes de bajo consumo y área extensa. LoRa es una de las tecnologías LPWAN y cuenta con grandes ventajas para proyectos de Internet de las Cosas o IoT (Internet of Things), con dispositivos alimentados por baterías.

LoRa es una tecnología inalámbrica al igual que WiFi, Bluetooth, LTE, SigFox o Zigbee. Utiliza un tipo de modulación en radiofrecuencia, como la AM o la FM o el PSK.

La gran ventaja de la misma es que puede lograr comunicaciones a largas distancias (típicamente kilómetros) y tiene gran solidez frente a las interferencias.

La tecnología LoRa fue originalmente desarrollada por Semtech, pero actualmente está administrada por la “LoRa Alliance”. De este modo, cualquier fabricante de hardware que desee trabajar con esta tecnología debe estar certificado por la alianza.

Entre las principales ventajas de LoRa se encuentran las siguientes:

- Alta tolerancia a las interferencias
- Alta sensibilidad para recibir datos (-168 dB)
- 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

Todo esto hace que ésta sea una tecnología ideal para conexiones a grandes distancias y para redes de IoT en las que se necesitan sensores que no dispongan de corriente eléctrica de red. Por ello, tiene grandes posibilidades de aplicación para Smart Cities o ciudades inteligentes, lugares con poca cobertura celular como por ejemplo aplicaciones agrícolas o ganaderas en el campo, o para construir redes privadas de sensores y/o actuadores.

3.3 PuTTY

El nombre PuTTY proviene de las siglas Pu: Port unique TTY: Teletipo (por sus siglas en inglés). Su traducción al castellano sería: Teletipo de puerto único.

PuTTY es un cliente SSH, Telnet, rlogin, y TCP raw con licencia libre. PuTTY nos permitirá emular una terminal con soporte SSH, que nos proporcionará acceso remoto a un servidor por medio de un canal seguro en el que toda

la información está cifrada.

Para nuestro objetivo, se utilizará en el receptor para visualizar si el mensaje en código morse se envió correctamente a través del módulo LoRa.

3.4 TinyGS

TinyGS es una red abierta de estaciones terrestres distribuidas por todo el mundo para recibir y operar satélites LoRa, sondas meteorológicas y otros objetos voladores, utilizando módulos baratos y versátiles.

Se puede acceder a la página de TinyGS donde se encuentra: listado completo de las estaciones terrestres actuales, los satélites soportados y todos los paquetes de datos recibidos por la comunidad en tiempo real.

TinyGS nos permitirá enviar (y recibir) datos de satélites en funcionamiento.

3.5 websdr.org

WebSDR es un receptor de radio definido por software conectado a Internet, que permite a muchos oyentes escucharlo y sintonizarlo simultáneamente. La tecnología SDR hace posible que todos los oyentes sintonicen de forma independiente y, por lo tanto, escuchen diferentes señales; esto contrasta con los muchos receptores clásicos que ya están disponibles a través de Internet.

Nos permitirá tomar códigos morse de satélites reales para probar y verificar que los objetivos del proyecto se cumplieron

4 Organización: Kanban y Trello

Kanban es un método para gestionar el trabajo, quiere decir tarjeta con signos o señal visual. El tablero más básico de Kanban está compuesto por tres columnas: “Por hacer”, “En proceso” y “Hecho”. Si se aplica bien y funciona correctamente, serviría como una fuente de información, ya que demuestra dónde están los cuellos de botella en el proceso y qué es lo que impide que el flujo de trabajo sea continuo e ininterrumpido. Kanban es fácil de implementar, y su objetivo principal es que exista un flujo de trabajo rápido para minimizar el riesgo y evitar el coste de retraso y así hacerlo de manera previsible.

Trello es un software de administración de proyectos con interfaz web, un tablón virtual en el que se pueden colgar ideas, tareas, imágenes o enlaces. Empleando el sistema **Kanban**, para el registro de actividades con tarjetas virtuales organiza tareas, permite agregar listas, adjuntar archivos, etiquetar eventos, agregar comentarios y compartir tableros.

5 Desarrollo del proyecto

5.1 Envió y decodificación de señales en código morse

En esta primer etapa, como primera instancia se buscó familiarizarse con el código morse, también conocido como CW (por sus siglas en inglés: Carrier Wave, o Continuous Wave), el sistema de representación de letras y números mediante señales emitidas de forma intermitente (Figura 1). Para esto, se generaron señales en código morse a mano, a través de un circuito pulsador (Figura 2), se las decodificó en tiempo real con arduino, y se mostró la decodificación por monitor serie en la pantalla de un ordenador.

International Morse Code							
A	N	1	.	=	+	-	@
B	O	2	,	!	\$	“	”
C	P	3	?	‘	’	‘	’
D	Q	4	!	‘	’	‘	’
E	R	5	/	;	:	;	:
F	S	6	"	;	:	;	:
G	T	7	(;	:	;	:
H	U	8)	;	:	;	:
I	V	9	&	;	:	;	:
J	W	0	:	;	:	;	:
K	X			;	:	;	:
L	Y			;	:	;	:
M	Z			;	:	;	:
SOS				Break			
New Line				Closing			
New Page				Shift to Wabun code			
New Paragraph				End of contact			
Attention				Understood			
Error				Invitation for named station to transmit			
Wait				Invitation for any station to transmit			

Fig. 1: Código Morse Internacional

Por otra parte, investigando se descubrió e hizo uso de la página web: [Morse Code Translator](#). Dicha página sirve para traducir desde y hacia el código morse internacional, como se observa en la figura 3, y permite reproducir la codificación morse con sonido, o adicionalmente visualizarla mediante una luz intermitente en la pantalla del ordenador. Además, permite controlar velocidad, frecuencia y volumen de los sonidos, y brinda la posibilidad de descargar el sonido que reproduce del código traducido.

Entonces, se creó construyó un amplificador para micrófono Electret (Figura 4), y se lo testeó. Adicionalmente, se utilizó un led para verificar la correcta detección de los sonidos por parte del micrófono Electret.

Luego se desarrolló, adaptó y refinó un programa en arduino para poder decodificar el código morse. En particular se dedicó mucho tiempo en adaptar y

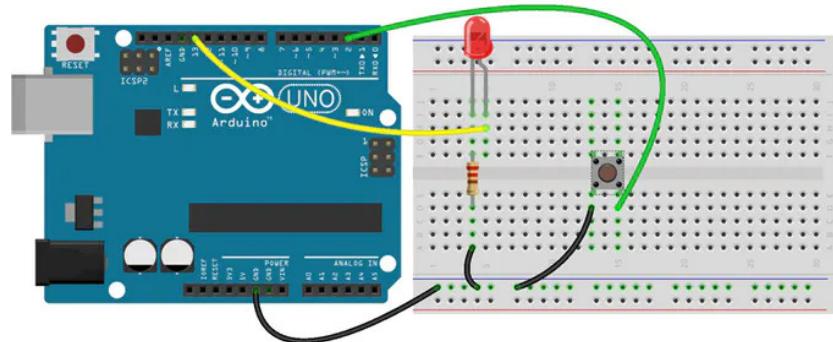


Fig. 2: Circuito para la generación manual de código morse

Translate a Message

Input:
TECNICAS Y HERRAMIENTAS MODERNAS EN MORSE

Output:
- . - . - . - . / - . - / . - . - . - . - . - . - . - . / - - - - . - . - . - . / - . - / - - - - . - . - . - .

 Play  Pause  Stop  Repeat  Sound  Light  Configure  Download

Fig. 3: Morse Code Translator

refinar los valores de los tiempos entre pitidos para lograr una correcta decodificación, considerando adicionalmente también los saltos de línea; y así poder recuperar el mensaje original sin ningún error.

Se utilizó el circuito que se observa en la Figura 5. También se ofrece al lector el código Arduino que se utilizó en esta parte, el cual se encuentre en el Anexo (5.4).

El siguiente paso fue descargar y utilizar la aplicación para móviles (y ordenador) Zello, una aplicación de comunicación instantánea que emula walkie-talkies push-to-talk (PTT) simulando radios tradicionales de dos vías a través de redes de telefonía celular. Permite intercambiar mensajes cortos de audio y hablar individualmente con una sola persona o en un canal con un grupo de personas.

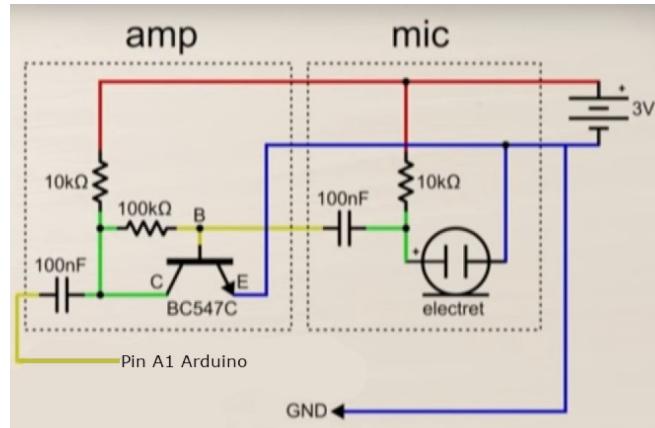


Fig. 4: Circuito amplificador de micrófono Electret

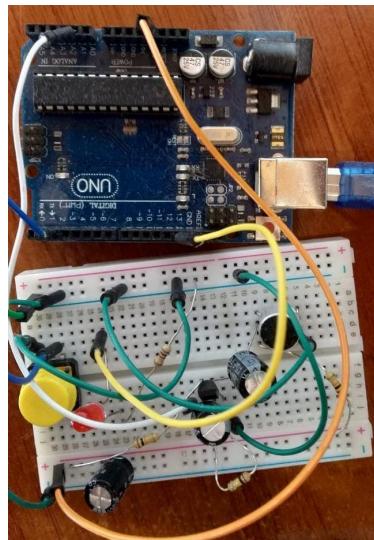


Fig. 5: Circuito decodificador

Una vez instalada la aplicación Zello y creados los usuarios, se generó código morse con la página Morse Code Translator, y se envió vía Zello el sonido para luego en casa de otro integrante del grupo, ser reproducido por el parlante, captado por el micrófono Electret y decodificado con Arduino Uno.

Finalmente, en la imagen 6 podemos ver como se relaciona todo lo explicado en esta subsección.



Fig. 6: Envio y decodificación de señales en código morse

5.2 Decodificación de señales de satélites

Habiendo realizado los primeros pasos con el código morse, esta segunda etapa comenzó con tratar de decodificar un audio en morse proveniente de un satélite, obtenido por el profesor de la Cátedra (el Dr.Ing.Ricardo R.Palma). Para ello se adaptó la frecuencia de detección del código decodificador programado en arduino. Pero no se tuvo éxito en realizar la traducción correctamente debido a el gran ruido de la fuente, que lo imposibilitada la interpretación de los pitidos. Por ello, se optó por aplicar filtros de sonidos para reducir dicho ruido, mediante la página: audiodenoise.com, pero aun así tampoco se logró decodificar el audio de manera correcta.

Dado el inconveniente, se decidió probar realizar la decodificación CW mediante el software FLDIGI (Figura 7) (abreviatura de: Fast Light DIGItal), un programa gratuito y de código abierto que utiliza la tarjeta de sonido de una computadora común como un módem de datos bidireccional simple, para permitir la comunicación mediante una gran cantidad de diferentes modos digitales, y un gran rango de velocidades. para ello se descargó, investigó y configuró adecuadamente el programa para decodificar código morse mediante audio.

Dicho detector morse tiene opciones avanzadas tales como el auto detector de frecuencia, que detecta el cambio de velocidad del código morse mismo, cambios de frecuencia debido al Efecto Doppler¹, ya que los satélites en general se desplazan a una gran velocidad respecto al sensor (observador).

Para ver su correcta configuración, se probó decodificar señales desde la página Morse Code Translator, mediante el programa FLDIGI, y si se la pudo decodificar perfectamente.

Por otra parte, además, se buscó en webDSR (Figura 8), la red de radios receptores que se encuentran en distintos lados del mundo, una señal con menos ruido. Una WebDSR interesante donde se encontró una gran cantidad de señales de CW es [N4HCT's WebDSR](#). Dichas señales en morse se encuentran distribuidas entre las frecuencias de 7MHz hasta los 7,3MHz, el cual es un rango donde

¹ Cambio de frecuencia aparente de una onda producido por el movimiento relativo de la fuente respecto a su observador

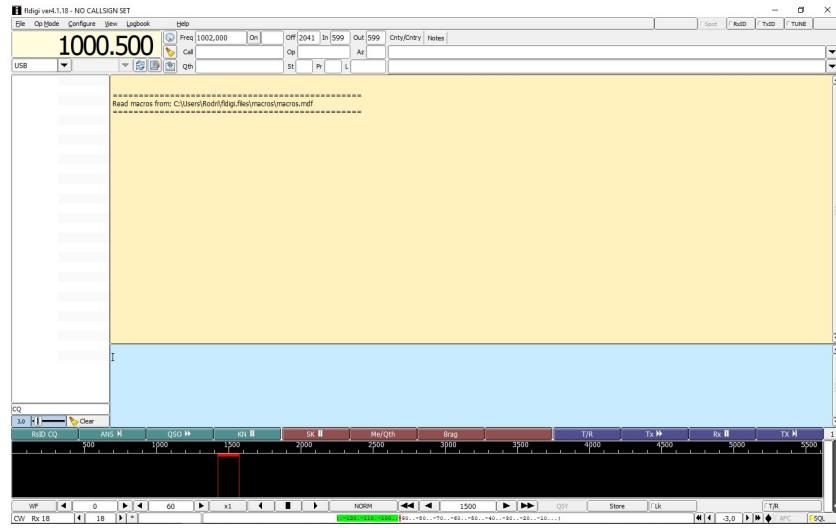


Fig. 7: Programa FLDIGI

aparece mucha gente, y muchos satélites transmitiendo en morse.

Pero luego de probar con distintas transmisiones de código morse en vivo, con banda de 40m RTL y ancho de banda CW-angosto, se obtuvo que la decodificación siempre mostraba basura similar a la que se observa en la Figura 9. Esto probablemente por el ruido de la señales, o también en parte por sintonizar morse creado por una mano humana y no por una unidad de cómputo, lo cual es mucho más difícil de detectar para los programas.

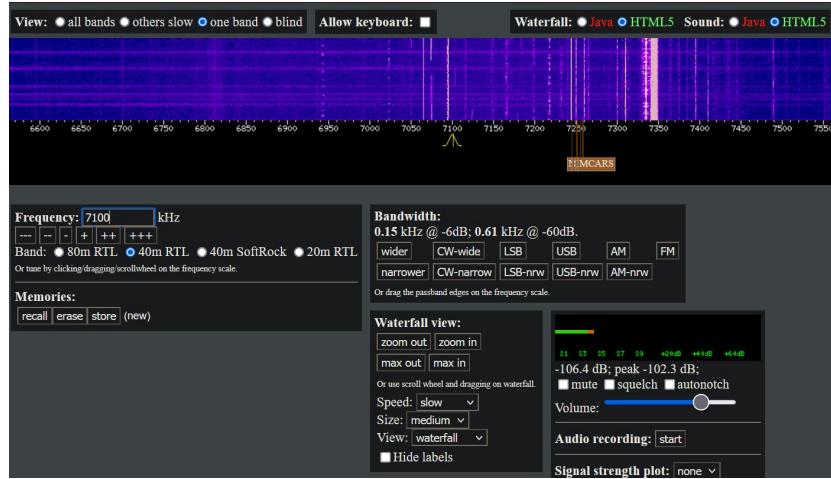


Fig. 8: N4HCl's WebDSR

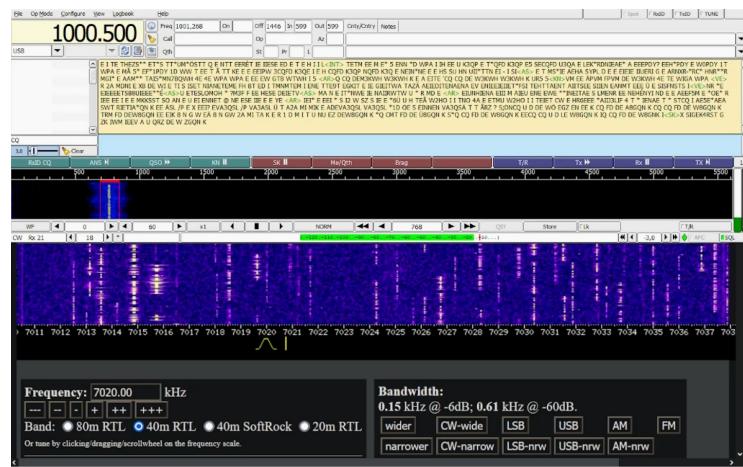


Fig. 9: Decodificación con ruido obtenida

5.3 Comunicación mediante protocolo LoRa

Círculo La cátedra nos brindó 2 LoRa, montados en una placa perforada. El primer paso fue realizar el relevamiento del circuito para identificar los componentes electrónicos y sus conexiones. Entre estos utilizamos el módulo LoRa SX1278 y el conversor bidireccional de nivel lógico de 8 canales, el cual convierte señales de 3,3V a 5V y viceversa, ya que resolvía el problema de que en los arduinos las entradas y salidas de propósito general son de 5V y el módulo LoRa trabaja a 3,3V. Luego se montó el circuito del transmisor y del receptor como se

observa en las figuras 10a y 10b, en estos diagramas no se muestra el adaptador de nivel lógico.

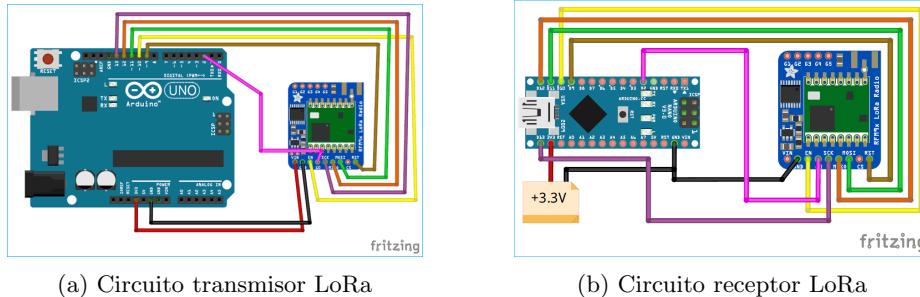


Fig. 10: Circuitos LoRa

Programación y pruebas Como primera prueba para el uso de Lora se procedió a utilizar el código y circuito de ejemplo disponible en documentación del classroom de la cátedra. Este primer acercamiento no fue exitoso dado que el código de ejemplo se encontraba incompleto por lo cual se volvía incompatible con nuestros arduinos.

La siguiente experiencia fue buscar en internet guías de cómo realizar comunicación Lora con los mismos módulos que nosotros teníamos disponibles, todos estos tutoriales se realizaban con Arduino UNO. En un comienzo no contábamos con 2 arduinos UNO, nuestras pruebas fueron intentadas sobre un Arduino Mega el cual tiene un chip ATMEGA2560 y posee distintos registros con respecto al Arduino UNO que posee un chip ATMEGA328p. Se trato de adaptar el código, pero por consola recibíamos el mensaje de que el modulo Lora no se podía iniciar.

Para poner en funcionamiento se procedió a conseguir otro Arduino UNO de manera de contar con dos del mismo. Luego de seguir múltiples guías de internet se procedió a hacer uso de una librería específica de Lora para el entorno de programación de Arduino. Se estudio los ejemplos disponibles y procedió a realizar el cableado básico. Una vez realizadas las conexiones se hicieron pruebas con las cuales se pudo lograr efectivamente la inicialización de los módulos. Se escribió el código para el transmisor y se lo cargó a un arduino y para el otro lo mismo pero con el código del receptor (ver Anexo (5.4)). La comunicación se realizó a una frecuencia de 433MHz y la comunicación entre el arduino y el módulo LoRa fue mediante SPI. Se comenzó enviando un mensaje cada cierto tiempo y una variable de acumulación de la cantidad de veces que el mensaje se había enviado. Se veía por monitor serie en el receptor esta información, como de aprecia en la figura 11. Para todo este procedimiento los arduinos eran alimentados por los

puertos USB de las computadoras también con el fin de poder hacer uso de la consola de comunicación serie y ver los mensajes.

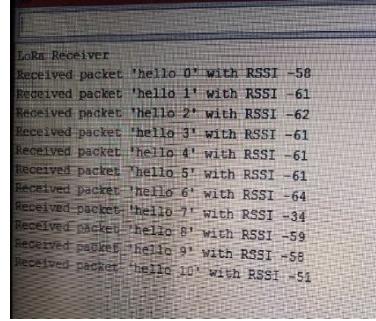


Fig. 11: Receptor monitor serie.

Una vez establecida la comunicación se puso a prueba el alcance que se tenía con las antenas, por lo que se colocó al transmisor en la caja estanco alimentado por un powerbank de los que son utilizados para cargar celulares (figura 12) y se lo alejó en un auto del receptor (figura 13), el cual permanecía en una casa conectado a una computadora, al volver se pudo comprobar que se perdió la comunicación a una distancia en línea recta de aproximadamente 100m. En la figura 14 se observa que al perderse la señal se introdujo ruido en ese momento al receptor.

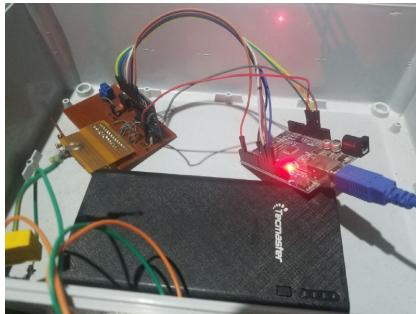


Fig. 12: Transmisor móvil.

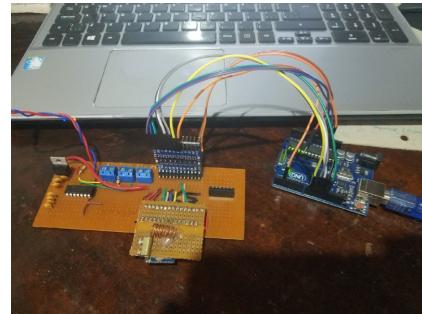


Fig. 13: Receptor fijo.

Envío de información mediante una terminal RS232. Una vez establecida la comunicación y conocido el alcance de las antenas, se procedió a utilizar

```

Received packet 'Tecnicas y Herramientas Modernas 0' with RSSI -99
Received packet 'Tecnicas y Herramientas Modernas 1' with RSSI -93
Received packet 'Tecnicas y Herramientas Modernas 2' with RSSI -109
Received packet 'Tecnicas y Herramientas Modernas 3' with RSSI -111
Received packet 'Tecnicas y Herramientas Modernas 4' with RSSI -116
Received packet 'Tecnicas y Herramientas Modernas 5' with RSSI -113
Received packet 'Tecnicas y Herramientas Modernas 6' with RSSI -101
Received packet 'Tecnicas y Herramientas Modernas 7' with RSSI -106

```

Fig. 14: Receptor monitor serie alcanzado el límite.

una terminal que simula una VT100, en el caso del receptor se utilizó PuTTY y para el transmisor se utilizó Termite. La configuración de la comunicación serie por RS232 para cada caso fue a 9600 baudios, con 8 bits de dato, un bit de stop y ningún bit de paridad. Luego de esto se comenzó a enviar cadenas de texto desde la terminal de Termite al arduino configurado como transmisor, este mediante el módulo LoRa envía la información la cual era recibida por el otro arduino mediante el módulo LoRa, luego por comunicación serie la información es visible en la terminal de PuTTY como se observa en las figuras 15 y 16. Habiendo logrado el correcto funcionamiento del envío y recepción de información esto puede ser utilizado con fines prácticos para el control de datos de dispositivos aislados de conexiones típicas como son las de internet.



Fig. 15: Transmisor Termite.

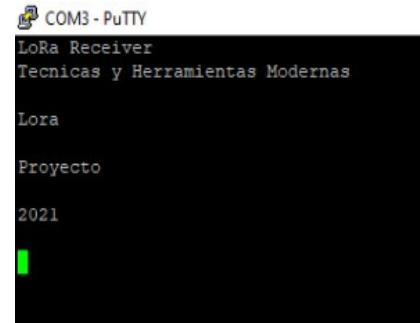


Fig. 16: Receptor PuTTY.

5.4 Agregar nuestra estación a TinyGS

TinyGS es una red abierta de estaciones terrestres a nivel global, que recibe y opera satélites LoRa, sondas meteorológicas y otros objetos voladores. Cuenta con 652 estaciones activas y 2000+ miembros.

Para llevar a cabo la comunicación con un satélite se hizo uso del TTGO que

teníamos disponible e hizo falta el armado de una antena para la recepción de información.

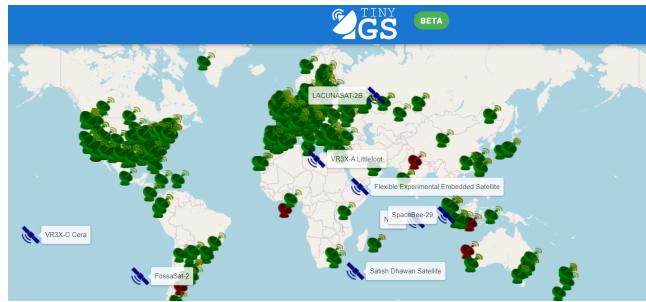


Fig. 17: Página web de TinyGS: tinygs.com

Construcción de la antena Para la construcción de la antena utilizada se hizo uso de los siguientes materiales:

- 1 Conector hembra PL 259
- 2 Conectores macho PL 259
- 1 Conector hembra PL 259
- 1 Conector SMA
- Alambre de cobre
- 1m Cable coaxial rg58

El satélite NORBI a comunicarse funciona en la frecuencia de 915 MHz por lo tanto se tuvo que realizar una antena que se adaptara a dicha frecuencia. Mediante el uso de tutoriales de armado de antenas se construyó una antena del tipo omnidireccional con plano a tierra. El largo de las varillas de cobre se cortaron de 8,2 cm, esto se calculó en función del siguiente cálculo:

$$\lambda = \frac{c}{frecuencia} = \frac{299.792.458 m/s}{915 MHz} = 0,328 m$$

$$Longitud Radiane = \frac{\lambda}{4} = \frac{0,328 m}{4} * 100 \frac{cm}{m} = 8,2 cm.$$

(Nota: Se divide por 4 para así obtener 1/4 de la longitud de onda).

Sobre el conector hembra PL 259 se soldó una de las varillas verticales y las otras 4 a 45º con respecto a la horizontal en la base formando un plano a tierra.

Se armó un cable coaxial rg 58 con conector machos PL 259 en cada extremo, en uno se colocó la antena armada y en el otro un conector SMA dado que el TTGO posee este tipo de entrada.

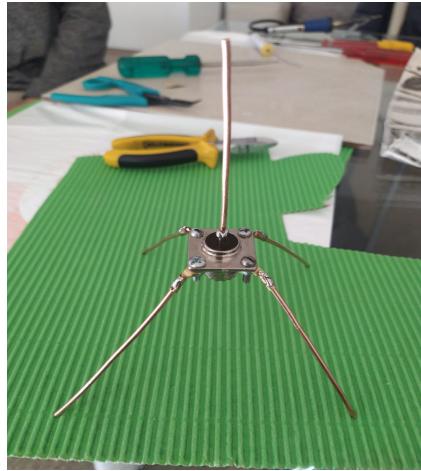


Fig. 18: Conector Hembra PL 259 y radiantes



Fig. 19: Conectores macho PL259 y cable coaxial



Fig. 20: Antena obtenida

Configuración del TTGO Los pasos realizados fueron los siguientes:

- 1 - Descargar "TinyGS Uploader" del siguiente github:
<https://github.com/G4lile0/tinyGS/releases>

2 - Una vez descargado se ejecuta el programa. En caso de warning seleccionar 'More options' y luego 'Run anyway'.

3 - Conectar el TTGO a la PC mediante el cable USB, previamente se le conectó la antena fabricada. Luego dar click en "Refresh" y hacer click en 'Upload tinyGS firmware'. La placa se reiniciará e iniciará el proceso de arranque.

4 - La placa genera un AP ('access point', punto de acceso) de nombre 'My TinyGS'. Luego de conectarse al mismo, se accede a la configuración de los parámetros básicos de la estación (Figura 22):

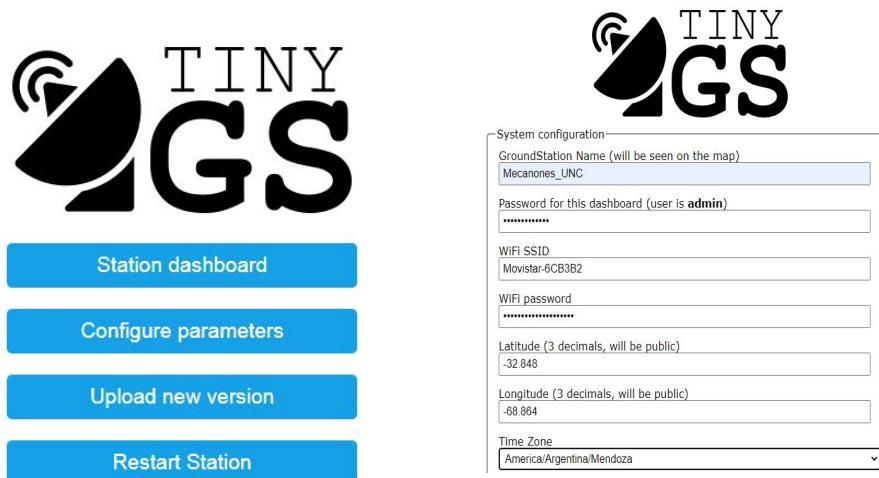


Fig. 21: Menú principal

Fig. 22: Configuración de parámetros básicos

5- Creación de las credenciales MQTT utilizando un bot llamado "Tinygs Personal Bot" de la aplicación de Telegram. El comando que utilizamos es: /mqtt. Este comando nos indica que nuestra estación se agregó al sistema y esta en funcionamiento. Y para poder observar el status de nuestras estaciones se utiliza el comando /stations. Esto se puede apreciar en la Figura 23.

Esto concluye la configuración de TTGO, a continuación hacemos el test con la antena construida.

Se observa que en la imagen 24 que la estación Mecanones UNC es visible en TinyGS, mostrando el Status de Online.

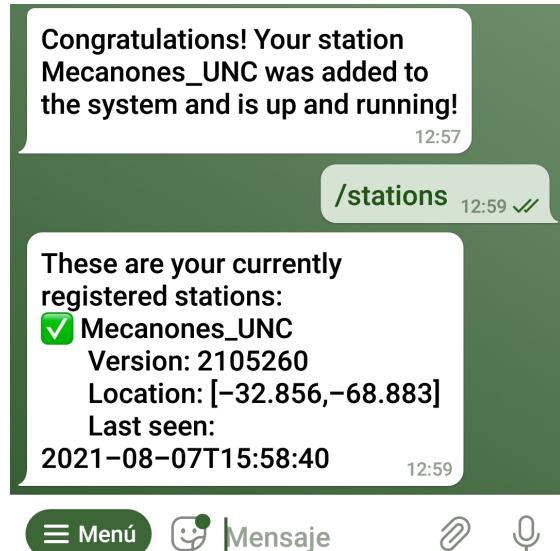


Fig. 23: Paso 3



Fig. 24: Resultado final

Resultados finales Vemos nuestra estación conformada por la antena y el TTGO configurado:

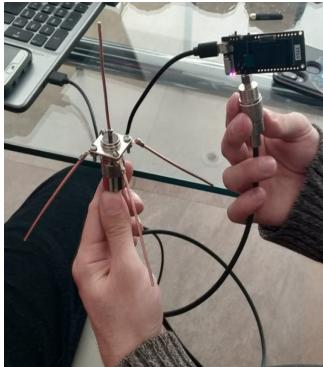


Fig. 25: Estación Mecanones_UNC

Anexo 1

```

1 int audioInPin = A1;

3 bool primerHIGH = true;
4 unsigned long tiempoInicioPitido;
5 unsigned long tiempoCuandoHIGH = 0;
6 unsigned long tiempoCuandoLOW;
7 unsigned long tiempoFinPitido;
8 bool flag1 = false;
9 bool flag2 = false;
10 bool flag3 = false;

11 int average = 0;
12 const unsigned int itersAverage = 77;

15 //-----
16 String code = "";
17
18 unsigned long pres_len = 0;
19 int state = 0;
20 int unit_delay = 120; //250
21 int min_delay = 10;
22 //-----
23 //
24 //-----
25 char MakeString() {
26     if(pres_len < (unit_delay) && pres_len > 50) {
27         return '.';
28             //if button press
29         less than 0.2s, it is a dot
30     }
31     else if(pres_len > (unit_delay)) {
32         return '-';
33             //if button press
34         more than 0.2s, it is a dash
35     }
36 }
37 //-----
38 void Morse_decod() {
39     static String morse[] = {".-", "-...", "-.-.", "-..", ".",
40     "....", "--.", "....", ".-", ".---", "-.-", ".-..", "--",
41     "-.", "-.--", "-.-.", "-.--", ".-.", "...", "-.", "...", "-.-",
42     ".--", "-..-", "-.--", "-.--", "-..-", "-!";
43 };
44 int i = 0;
45 while(morse[i] != "!") {
46     if(morse[i] == code) {
47         Serial.print(char('A' + i));
48         break;
49     }
50 }

```

```

43    }
44    i++;
45  }
46  if(morse[i] == " ! ") {
47    Serial.println(" ");
48    Serial.println("This code is not exist!");
49  }
50  code = ""; //Limpiamos el string: code, para reiniciar el
51  // loop principal.
52 //}
53 void setup() {
54   Serial.begin(2000000);
55   pinMode(LED_BUILTIN, OUTPUT);
56   delay(777);
57   for(int i=0; i<itersAverage; i++) {
58     average = average + analogRead(audioInPin);
59   }
60   average = average/itersAverage;
61   // Serial.print("Valor Promedio de la Entrada de Audio = ");
62   // Serial.println(average);
63   // Serial.println("");
64 }
65
66 void loop() {
67   int sensorValue = analogRead(audioInPin);
68   //Serial.println(sensorValue);
69   if(sensorValue > (average+4)) {
70     if(primerHIGH) {
71       tiempoInicioPitido = millis();
72       TM=tiempoInicioPitido-tiempoFinPitido;
73       Serial.println("TM= "); Serial.println(TM);
74       digitalWrite(LED_BUILTIN, HIGH);
75       primerHIGH = false;
76     }
77     tiempoCuandoHIGH = millis();
78     flag1 = false;
79     flag2 = false;
80     flag3 = false;
81   } else if(!primerHIGH){ //Siempre que: (sensorValue < 430)
82     && (primerHIGH == False)
83     tiempoCuandoLOW = millis();
84     if(tiempoCuandoLOW-tiempoCuandoHIGH > min_delay) {
85       tiempoFinPitido = millis();
86       digitalWrite(LED_BUILTIN, LOW);
87       primerHIGH = true;
88       flag1 = true;
89       flag2 = true;
90       flag3 = true;
91       pres_len = tiempoFinPitido-tiempoInicioPitido;
92     }
93   }
94 }
```

```
91         code += MakeString();
92     }
93 }
94
95 if(flag1){
96     if( (millis()-tiempoFinPitido) > (111) ) { //Y cuando
97         este entre 0.111s y 0.333s:
98             Morse_decod();
99             flag1 = false;
100        }
101    }
102 if(flag2){
103     if( (millis()-tiempoFinPitido) > (333) ) { //Cuando
104         este entre 0.333s y 0.555s:
105             Serial.print(" ");
106             flag2 = false;
107        }
108 if(flag3){
109     if( (millis()-tiempoFinPitido) > (555) ) { //Cuando el
110         tiempo desde el fin del ultimo pitido sea mayor a 0.555s
111         :
112             Serial.println(" ");
113             flag3 = false;
114        }
115    }
116}
117}
```

Listing 1.1: Decodificador_de_Codigo_Morse.ino

Anexo 2

```
1 #include <SPI.h>
2 #include <LoRa.h>
3
4 int counter = 0;
5 String a;
6 char aux;
7 void setup() {
8     Serial.begin(9600);
9     while (!Serial);
10
11     Serial.println("LoRa Sender");
12
13     if (!LoRa.begin(433E6)) {
14         Serial.println("Starting LoRa failed!");
15         while (1);
16     }
17
18     LoRa.setTxPower(20);
19
20 }
21
22 void loop() {
23
24     if (Serial.available() > 0) {
25         aux=Serial.read();
26         if (aux!='\n'){
27             a.concat(aux);
28         }
29         if (aux=='\n')
30         {
31             LoRa.beginPacket();
32             LoRa.print(a);
33             LoRa.endPacket();
34
35             Serial.println(a);
36             a="";
37         }
38     }
39 }
```

Listing 1.2: LoRa_lado_transmisor.ino

```
1 #include <SPI.h>
2 #include <LoRa.h>
3
4 void setup() {
5     Serial.begin(9600);
```

```
7     while (!Serial);
9
11    Serial.println("LoRa Receiver");
13
15 void loop() {
17    // try to parse packet
19    int packetSize = LoRa.parsePacket();
21    if (packetSize) {
23        // received a packet
25        //Serial.print("Received packet ");
27
29        // read packet
31        while (LoRa.available()) {
33            Serial.print((char)LoRa.read());
35
37            Serial.print('\n');
39            Serial.print('\r');
41            // print RSSI of packet
43            //Serial.print(" with RSSI ");
45            //Serial.println(LoRa.packetRssi());
47        }
49    }
51 }
```

Listing 1.3: LoRa_lado_receptor.ino

References

1. "Efecto Doppler" — Wikipedia, La enciclopedia libre, https://es.wikipedia.org/w/index.php?title=Efecto_Doppler&oldid=136062473.
2. "Mini project: Amplified electret microphone" — bitluni, <https://www.youtube.com/watch?v=SToBPCajwc0>.