HEXACT



Critical DESIGN REVIEW

**IES Seritium - HEXACT**

**Curso 2022-2023**

**Índice**

[**Introducción**](#_y35ddjmx9f9q) **3**

[Organización y roles del equipo](#_vmpfhkt5sc2e) 3

[Objetivos de la misión](#_hrpjl2fsabka) 3

[**Descripción del proyecto CANSAT**](#_k672z3lpt1pl) **3**

[Esquema de la misión](#_n29d9ye47zbc) 3

[Proyecto Científico](#_q0yhy76cov0y) 3

[Diseño de la estructura](#_7mz1ttgdri3p) 4

[Paracaídas](#_c78a53mxfnoc) 4

[Diseño eléctrico](#_25ogskc5ium8) 5

[Software](#_kmwj0u57nlkd) 6

[Sistema de recuperación](#_irmoeo278a3r) 8

[Estación de tierra](#_xcao7ic6gds5) 8

[**Planificación del proyecto CANSAT**](#_kb4ii88j6n75) **8**

[Diagrama de GANTT (cronograma)](#_x30uf421ma30) 8

[Presupuesto del CANSAT](#_tswo68huecte) 9

[Apoyo externo](#_wam9pyisdu0) 10

[Pruebas realizadas](#_kgylnesfgm5w) 10

[**Difusión del proyecto**](#_lgju9tevnr2) **10**

[**Bibliografía / Referencias / Recursos**](#_r17fc31ictfu) **10**

[**Anexos PROGRAMAS**](#_n6dnr7auh8tk) **11**

= = = = = = = = = =

# Introducción

## Organización y roles del equipo

1. Adrián Durán Perdigones → Edición fotografía / vídeo. Grafismo.
2. Iker Espinosa Algeciras → Programación Processing y analista de datos. Responsable redes sociales y difusión.
3. Alejandro Chacón Pérez → Programación módulo CANSAT y estación tierra.
4. Antonio Jesús Suárez Gómez → Diseño 3D y construcción.
5. Miriam Ganaza Gómez → Programación Processing y analista de datos. Responsable paracaídas.
6. David González Diosdado → Programación Processing y analista de datos. Diseño 3D
7. [Aurelio Gallardo Rodríguez](mailto:aurelio@seritium.es) → Profesores

Hemos dedicado 2h semanales, para no interrumpir nuestros estudios en la materia TIC. En clase dedicamos el tiempo en crear el programa de analítica en processing, creando el código de la placa, soldando cables, haciendo el paracaídas y realizando el guión y grabación de vídeos.

## Objetivos de la misión

1. Analizar con precisión el clima y terreno, mandar la información a la estación de tierra y analizarlo.
2. Aprender de telemetría, código, análisis, soldadura, ...
3. Pasarlo bien.

# Descripción del proyecto CANSAT

## Esquema de la misión

Los datos serán recogidos durante el lanzamiento y vuelo gracias a la telemetría por radio. Esperamos que todos los datos nos lo ofrezcan con la mayor exactitud y al momento; sabiendo a cuánta distancia está gracias a la intensidad de señal y llegando a una altura calculada suelte cada cierto tiempo semillas.

1. **Misión primaria:** Presión, temperatura, altura y telemetría.
2. **Misión secundaria:** Aceleración, giroscopio, posición según la intensidad de la señal y dispensador de semillas.

## 

## Proyecto Científico

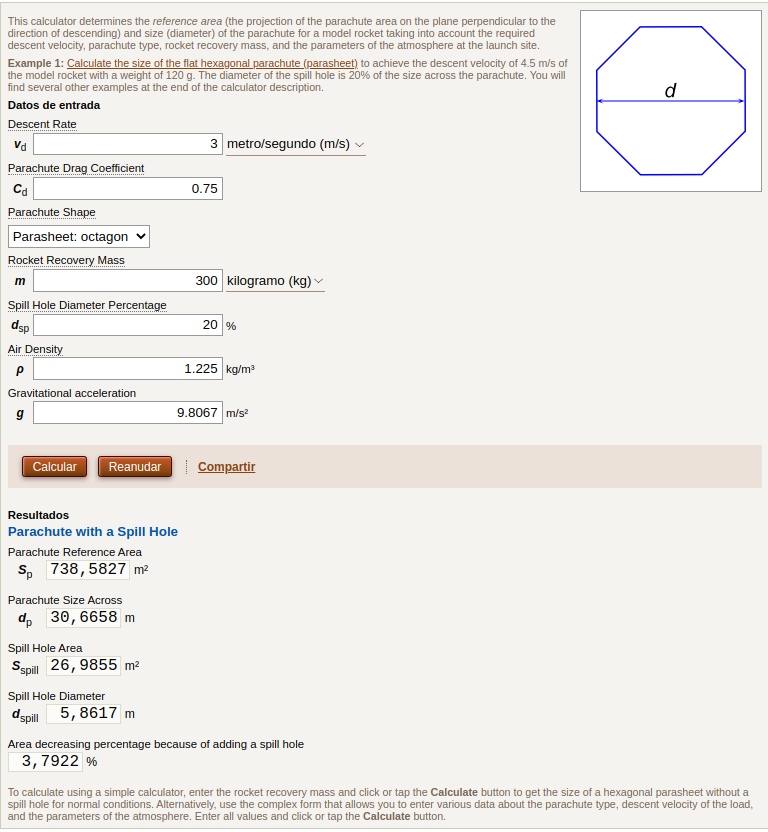
El dispensador de semillas tiene la función de plantar dónde y cuando nosotros queramos semillas en un campo o cultivo para hacerlo de forma automática y rápida.

Usamos la señal de telemetría para saber dónde está el Cansat al caer, el dispensador de semillas para cultivar árboles o plantas; presión, temperatura y altura para analizar el clima y realice una función de satélite; el acelerómetro para calcular la velocidad a la que se encuentra el Cansat y su posible orientación, y la telemetría para que nos llegue los datos en vivo.

## Diseño de la estructura (Antonio)

Planos de la estructura.

|  |  |
| --- | --- |
|  |



### Paracaídas (Miriam)

~~Cordaje: tanza 80LB (36.3Kgf). Tensión máxima que soporta.~~

~~Tela paracaídas: taffta nylon. 70D (unidades deniers) y entre 190/210T (trenzado de líneas horizontales/verticales). Tela especial para paracaídas y tiendas de campaña: ver bibliografía.~~

## Diseño eléctrico (Alejandro)

~~Más cosas que decir:~~

~~-Baterías.~~

~~-Módulo de carga.~~

~~-Placa hecha artesanal~~

~~-Componentes usados normalmente encontrados y no comprados.~~

~~Consumo teórico de componentes importantes:~~

* ~~Arduino Pro Mini: 200mA máximo en el chip~~
* ~~ESP32 CAM: 180mA sin flash~~
* ~~GY-87 (Acelerómetro): MPU6050 → 10 mA máximo. BMP180 → 1mA pico~~
* ~~LORA 868MHz: 130mA máximo transmitiendo~~
* ~~NEO6 GPS: 45mA~~
* ~~LED: menos de 70mA~~

~~I~~~~T~~ ~~teórico < 636mA~~

~~Consumo total de la placa en funcionamiento: I~~~~T~~ ~~= 220 mA max, medido con amperímetro, con LED encendido.~~

~~Carga de las pilas (tantalato litio): (). Con pilas estándar 18650,~~

~~Tiempo de encendido teórico: y con pilas estándar y tantalato de litio, respectivamente.~~

~~Tiempo de encendido práctico:~~

**~~Enlaces a componentes; intensidades máximas teóricas:~~**

**~~-Arduino:~~** [~~https://components101.com/microcontrollers/arduino-pro-mini~~](https://components101.com/microcontrollers/arduino-pro-mini)

**~~-Módulo Lora:~~**[~~https://learn.sparkfun.com/tutorials/rfm69hcw-hookup-guide/all~~](https://learn.sparkfun.com/tutorials/rfm69hcw-hookup-guide/all)

**~~-GY-87:~~** ~~MPU6050 → 10 mA máximo (~~[~~https://www.elecrow.com/crowtail-mpu6050-accelerometer-gyro.html~~](https://www.elecrow.com/crowtail-mpu6050-accelerometer-gyro.html)~~)~~

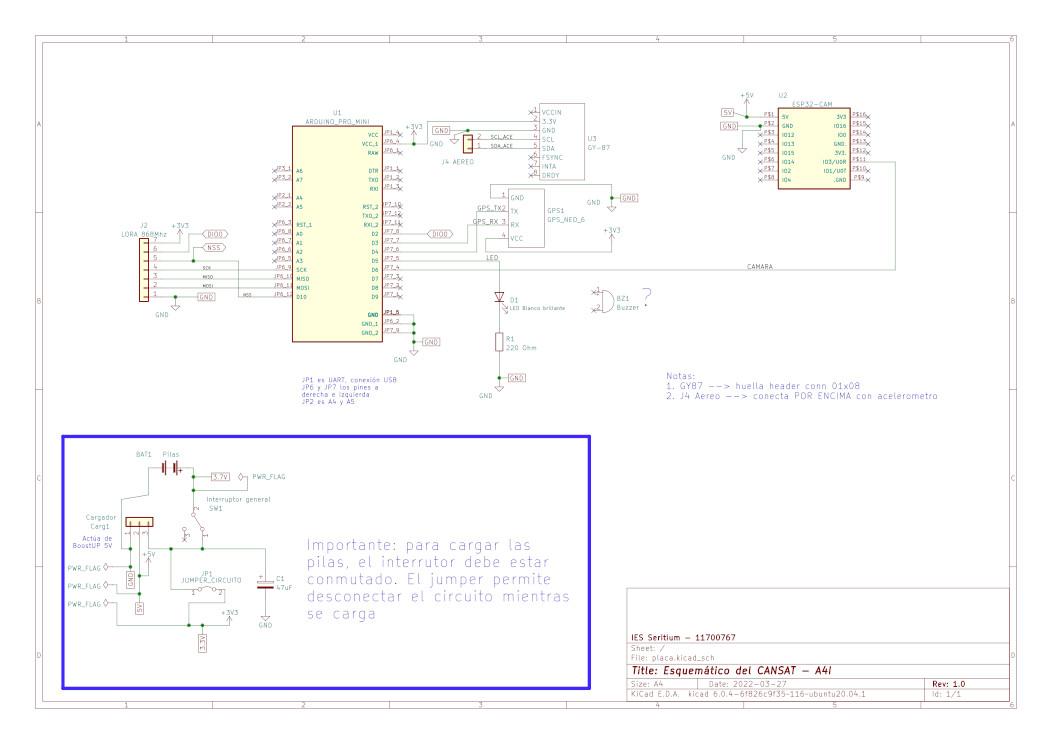
~~BMP180 → 1mA pico (~~[~~https://components101.com/sensors/bmp180-atmospheric-pressure-sensor~~](https://components101.com/sensors/bmp180-atmospheric-pressure-sensor)~~)~~

**~~-ESP32 cam:~~**[~~https://loboris.eu/ESP32/ESP32-CAM%20Product%20Specification.pdf~~](https://loboris.eu/ESP32/ESP32-CAM%20Product%20Specification.pdf)

**~~-NEO6 GPS:~~**[~~https://lastminuteengineers.com/neo6m-gps-arduino-tutorial/~~](https://lastminuteengineers.com/neo6m-gps-arduino-tutorial/)

**~~-Baterías:~~**[~~https://es.aliexpress.com/item/32835795441.html?gatewayAdapt=glo2esp&spm=a2g0o.order\_list.0.0.21ef194dWV3ghg~~](https://es.aliexpress.com/item/32835795441.html?gatewayAdapt=glo2esp&spm=a2g0o.order_list.0.0.21ef194dWV3ghg)

**Esquemático eléctrico / electrónico**



| Pistas cobre | Serigrafía superior / puentes |
| --- | --- |

## Sistema de recuperación

Para evitar cualquier daño en el gps o dificultad en el traspaso de datos hemos preferido usar la señal de telemetría para saber la distancia.

## Telemetría

El cansat tras recopilar los datos los envía al módulo lora el cual modula la señal a 868mHz y los envía por la antena.

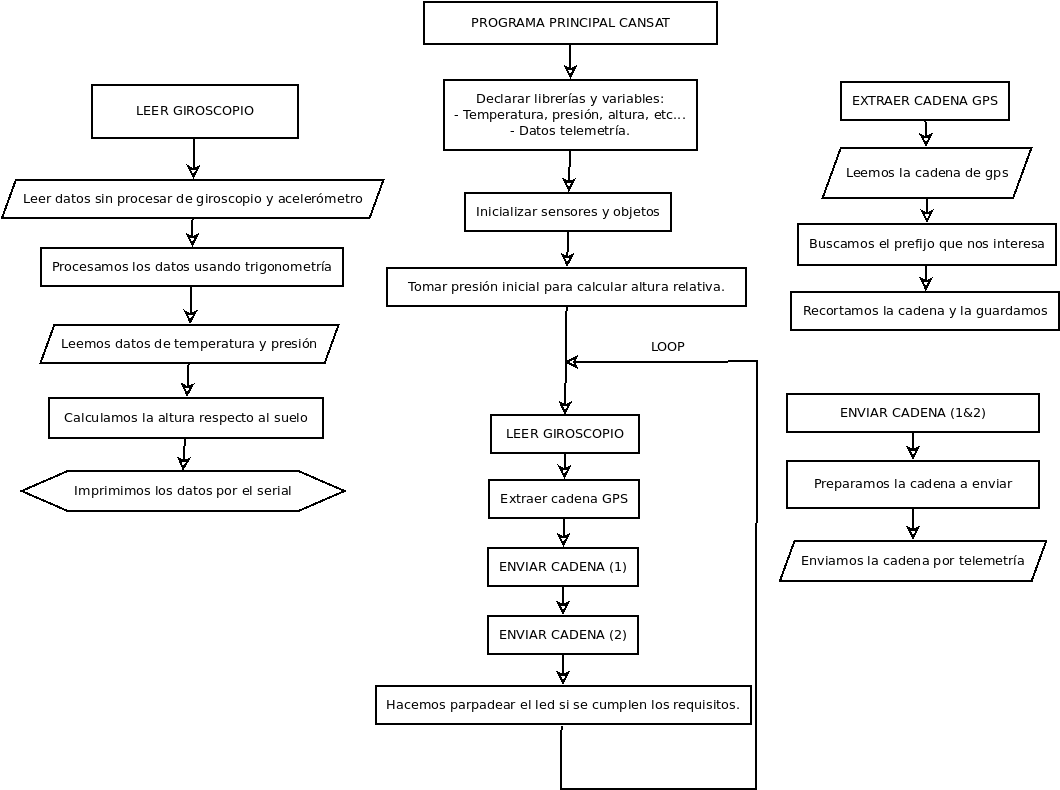
## Estación de tierra

## Software

~~Hemos usado arduino como entorno de desarrollo, programando con su IDE oficial.~~

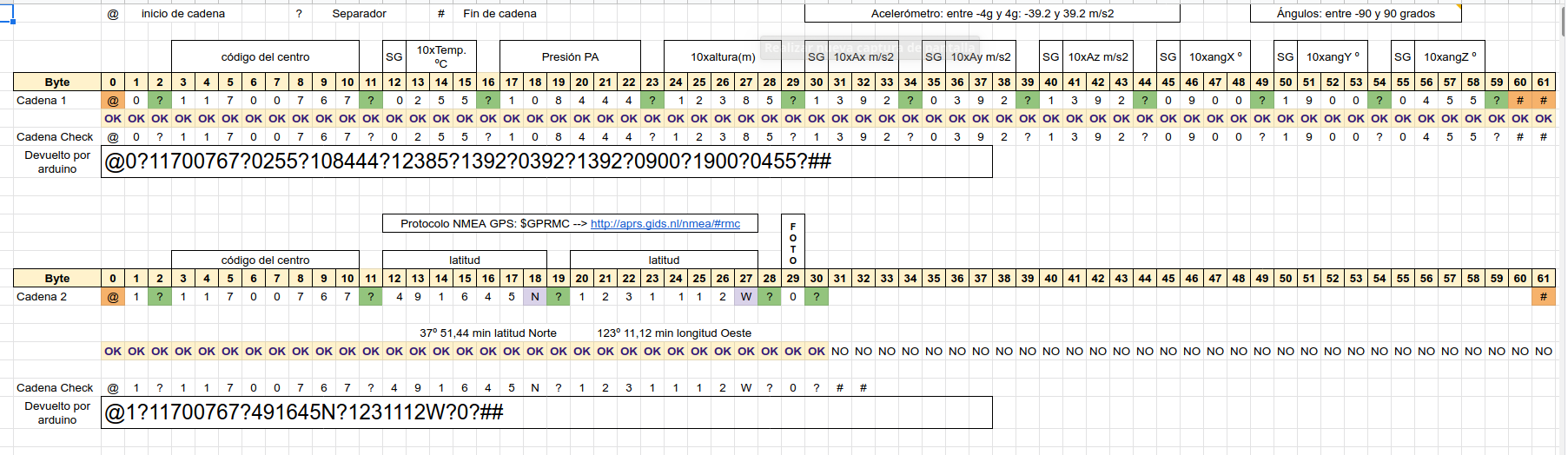
~~Para enviar a tierra los datos, los encriptamos y los enviamos mediante el módulo RFM69 a la estación de tierra. Los datos recopilados son divididos en 2 grupos,que se envían en dos cadenas distintas: La primera, con datos de aceleración y giros en los 3 ejes, presión (Y, derivada de esta, la altitud del CanSat), y temperatura; y la segunda, con datos de posicionamiento GPS y informe sobre si se ha tomado foto o no. ¿~~

**Diagrama de flujo**



**Estructura de datos**

~~Seguir el enlace:~~ [~~https://docs.google.com/spreadsheets/d/1UBlOgZCoYUfe1Wo4PusJnSwNe\_fxC9IOaQ9bJyCvJZQ/edit?usp=sharing~~](https://docs.google.com/spreadsheets/d/1UBlOgZCoYUfe1Wo4PusJnSwNe_fxC9IOaQ9bJyCvJZQ/edit?usp=sharing)

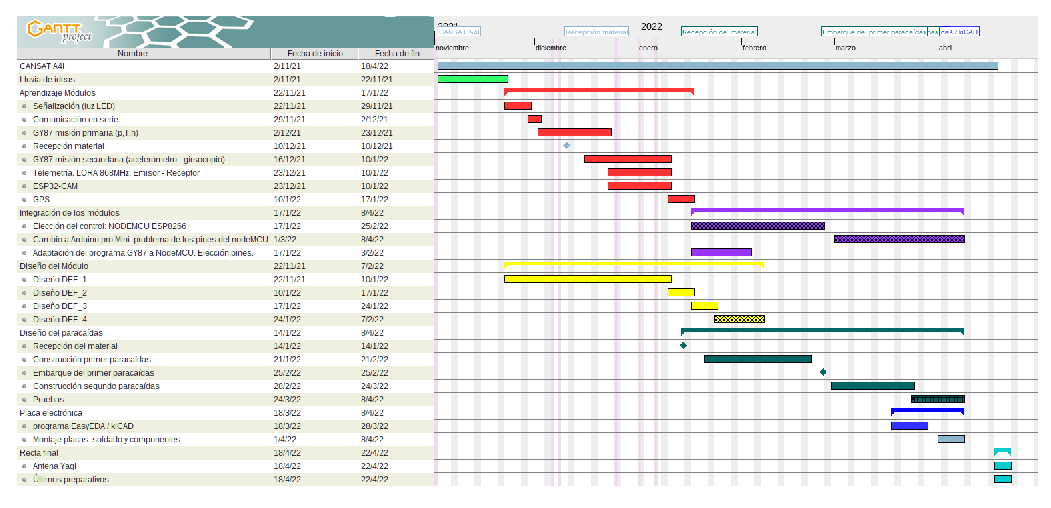
~~~~

~~Se envían dos cadenas alternas. Descripción informaciones:~~

1. ~~Código de cadena, código del centro, temperatura ºC, presión PA, 10xaltura m, 10xAx, 10xAy, 10xAz (m/s2), 10xangX, 10xangY, 10xangZ (º)~~
2. ~~Código de cadena, código del centro,latitud, longitud, Si se ha tomado o no foto.~~

# Planificación del proyecto CANSAT

## Diagrama de GANTT (cronograma)

~~~~

## 

## Presupuesto del CANSAT

~~(\*) Estación de tierra.~~

| ~~Concepto~~ | ~~Valor / Observaciones~~ | ~~Cantidad~~ | ~~Precio (€)~~ |
| --- | --- | --- | --- |
| ~~Led~~ | ~~Para señalización~~ | ~~1~~ | ~~0,01 €~~ |
| ~~Buzzer activo~~ | ~~1~~ | ~~0,1 €~~ |
| ~~Interruptor~~ | ~~Mini para placa~~ | ~~1~~ | ~~0,01 €~~ |
| ~~Resistencia~~ |  | ~~1~~ | ~~0,01 €~~ |
| ~~Cond. electrolítico~~ | ~~47uF~~ | ~~1+1(\*)~~ | ~~0,02 €~~ |
| ~~NodeMCU V2~~ | ~~ESP8266~~ | ~~1 (\*)~~ | ~~3 €~~ |
| ~~Arduino pro mini~~ | ~~Logic 3.3V~~ | ~~1~~ | ~~5 €~~ |
| ~~ESP32-CAM~~ | ~~Con cámara OV2640 75 mm 2MP~~ | ~~1~~ | ~~6€ + 3,5€ = 9,5€~~ |
| ~~LORA 868MHz~~ | | ~~1 + 1(\*)~~ | ~~3€~~ |
| ~~GY-87~~ | ~~acelerómetro + giroscopio + presión + temperatura~~ | ~~1~~ | ~~5,5€~~ |
| ~~NEO6 GPS~~ | | ~~1~~ | ~~4€~~ |
| ~~Piezas 3D~~  ~~45g al 25% densidad + 56g al 50% = 101g~~ | ~~Habitáculo pilas (56g/50%)~~ | ~~1~~ | ~~PLA Amarillo 1 Kg a ~17€~~  ~~~1,7€~~ |
| ~~Separador~~ | ~~2~~ |
| ~~Tapa habitáculo pilas~~ | ~~1~~ |
| ~~Lateral A (Antena)~~ | ~~1~~ |
| ~~Lateral B~~ | ~~1~~ |
| ~~Tornillería, cableado, conectores. Estaño.~~ | ~~Disponibles en el taller~~ | ~~?~~ | ~~1€~~ |
| ~~Antena emisora y receptora, para CANSAT y estación tierra~~ | ~~Construidas con materiales del taller: arandelas y alma de cobre de cables de D1.5~~ | ~~2~~ | ~~2 x 0,5€ = 1 €~~ |
| ~~Placa electrónica~~  ~~41 x 104 mm~~ | ~~70 x 100 mm~~ | ~~1~~ | ~~1 €~~ |
| ~~Pilas 18650~~ | ~~Tantalato de litio~~ | ~~2~~ | ~~15,4€/ 2 = 7,7 €~~ |
| Módulo de carga | Equivalente TP4056 | 1 | 0,5€ |
| Tarjeta microSD | 32GB | 1 | 3€ |
| Tela paracaídas | | 1 | 2,74 € x 1,5m2 |
| Tanza pesca 4 hebras | | 1 | 5,5€ |
| ~~TOTAL:~~ | | | **~~55,29€~~** |

## Apoyo externo

~~En principio, hemos contado con recursos propios y material suministrado por los profesores.~~

## Pruebas realizadas

~~En el enlace:~~ [~~https://www.seritium.es/wordpress/index.php/2022/03/24/proyecto-cansat-2022/~~](https://www.seritium.es/wordpress/index.php/2022/03/24/proyecto-cansat-2022/)

# ~~Difusión del proyecto~~

1. ~~Página web:~~ [~~www.seritium.es~~](http://www.seritium.es)
2. ~~Facebook:~~ [~~https://www.facebook.com/ies.seritium.jerez~~](https://www.facebook.com/ies.seritium.jerez)
3. ~~Twitter: @IesSeritium~~

# ~~Bibliografía / Referencias / Recursos~~

1. ~~Modelización del paracaídas:~~ [~~https://www.translatorscafe.com/unit-converter/es-ES/calculator/parachute-size/~~](https://www.translatorscafe.com/unit-converter/es-ES/calculator/parachute-size/)
2. ~~Cordaje paracaídas:~~ [~~https://es.aliexpress.com/item/1549494151.html?gatewayAdapt=glo2esp&spm=a2g0o.order\_list.0.0.21ef194dWV3ghg~~](https://es.aliexpress.com/item/1549494151.html?gatewayAdapt=glo2esp&spm=a2g0o.order_list.0.0.21ef194dWV3ghg)
3. ~~GY-87:~~ [~~https://www.luisllamas.es/arduino-orientacion-imu-mpu-6050/~~](https://www.luisllamas.es/arduino-orientacion-imu-mpu-6050/) ~~,~~ [~~https://electropeak.com/learn/interfacing-gy-87-10dof-imu-mpu6050-hmc5883l-bmp085-module-with-arduino/~~](https://electropeak.com/learn/interfacing-gy-87-10dof-imu-mpu6050-hmc5883l-bmp085-module-with-arduino/)~~,~~ [~~https://naylampmechatronics.com/blog/45\_tutorial-mpu6050-acelerometro-y-giroscopio.html~~](https://naylampmechatronics.com/blog/45_tutorial-mpu6050-acelerometro-y-giroscopio.html)
4. ~~ESP32-CAM:~~ [~~https://randomnerdtutorials.com/esp32-cam-video-streaming-face-recognition-arduino-ide/~~](https://randomnerdtutorials.com/esp32-cam-video-streaming-face-recognition-arduino-ide/)
5. ~~Telemetría:~~ [~~https://learn.sparkfun.com/tutorials/rfm69hcw-hookup-guide/all~~](https://learn.sparkfun.com/tutorials/rfm69hcw-hookup-guide/all)~~,~~ [~~https://iot.uy/conectar-la-radio-rfm69-hcw-al-esp8266/~~](https://iot.uy/conectar-la-radio-rfm69-hcw-al-esp8266/)~~,~~ [~~https://www.openhardware.io/view/171/Connecting-the-Radio~~](https://www.openhardware.io/view/171/Connecting-the-Radio)~~,~~ [~~https://learn.adafruit.com/adafruit-rfm69hcw-and-rfm96-rfm95-rfm98-lora-packet-padio-breakouts/overview~~](https://learn.adafruit.com/adafruit-rfm69hcw-and-rfm96-rfm95-rfm98-lora-packet-padio-breakouts/overview)~~,~~ [~~https://forum.arduino.cc/t/nodemcu-and-rfm69hw-s2/599219/16~~](https://forum.arduino.cc/t/nodemcu-and-rfm69hw-s2/599219/16)
6. ~~GPS:~~ [~~https://www.luisllamas.es/localizacion-gps-con-arduino-y-los-modulos-gps-neo-6/~~](https://www.luisllamas.es/localizacion-gps-con-arduino-y-los-modulos-gps-neo-6/)
7. ~~Antena Yagi:~~ [~~https://www.changpuak.ch/electronics/yagi\_uda\_antenna\_DL6WU.php~~](https://www.changpuak.ch/electronics/yagi_uda_antenna_DL6WU.php)

# Anexos PROGRAMAS

**Programas en el emisor (satélite CANSAT)**

/\*\*\* CanSat Unificado \*\*\*/

/\* Programa unificado para el proyecto CanSat 2021/2022

\* Dispositivo: Arduino Pro Mini (3.3v)

\* Fecha: 8/1/2022

\*/

/\* -=- Por hacer -=-

\* - Cámara así? Más pruebas?

\*/

/\* -=- Changelog -=-

\* - 31/03/2022: Añadida la lectura de giro en el eje z. Añadido el envío de altura para la cámara. Limpieza de código.

\* - 11/04/2022: Eliminada la función de envio de datos a la cámara. Por ahora lo dejamos así.

\*/

/\* -=- PINOUT -=-

\* RX -----------3

\* TX -----------4

\* SDA ----------A4

\* SCL ----------A5

\* MOSI----------11

\* MISO ---------12

\* SCK ----------13

\* NSS ----------10

\* DIO0----------2

\* ESP32-CAM-----6

\*/

#include <RFM69.h>

#include <SPI.h>

#include <Adafruit\_BMP085.h>

#include "I2Cdev.h"

#include "MPU6050.h"

#include "Wire.h"

#include <SoftwareSerial.h>

// Addresses for this node. CHANGE THESE FOR EACH NODE!

#define NETWORKID 100 // Must be the same for all nodes --> esta es la red.

#define MYNODEID 10 // My node ID --> este es el nodo de mi dispositivo

#define TONODEID 20 // Destination node ID --> este es el nodo del otro dispositivo

#define FREQUENCY RF69\_868MHZ

#define FREQUENCY\_E 868001000UL // frecuencia exacta

// AES encryption (or not):

#define ENCRYPT true // Set to "true" to use encryption

#define ENCRYPTKEY "123456789" // Use the same 16-byte key on all nodes --> clave secreta

// Use ACKnowledge when sending messages (or not):

#define USEACK false // Request ACKs or not

// Según la web https://iot.uy/conectar-la-radio-rfm69-hcw-al-esp8266/

#define SPI\_CS 10 //NSS

#define IRQ\_PIN 2 // DIO0

#define IRQ\_NUM 2 //IRQ igual al pin

#define IS\_RFM69HCW true // Si tu radio es RFM69HCW entonces va "true"

#define POWER\_LEVEL 31 // Valor máximo de potencia

// -=- OBJETOS -=-

RFM69 radio = RFM69(SPI\_CS, IRQ\_PIN, IS\_RFM69HCW, IRQ\_NUM);

MPU6050 sensor; // Iniciamos sensor mpu6050

Adafruit\_BMP085 bmp; // Iniciamos el BMP180

SoftwareSerial gps (3, 4); // Iniciamos el GPS (rxPIN, txPIN)

//SoftwareSerial cam(7, 6); // Iniciamos la comunicación con la cámara (Chequear pines)

const int led = 5;

// -=- DATOS A REGISTRAR -=-

// Cadena del GPS --> $GPRMC,225446,A,4916.45,N,12311.12,W,000.5,054.7,191194,020.3,E\*68

String cadenaGPS = ""; // Cadena GPS

const unsigned int codigoSeritium = 11700767; // Código del centro

float temperatura = -99.9; // Ejemplo de temperatura con signo (2 dígitos + 1 decimal)

unsigned int presion = 999999; // Ejemplo de presión atmosférica. Entero positivo sin decimales.

float altura = 9999.9; // Ejemplo de altura. Positivo sin signo (4 dígitos + 1 decimal)

float alturaPrev = 9999.9;

float ace[3] = {+39.2,-39.2,+39.2}; // valores entre -4g y 4g, en m/s2. Con signo (2 dígitos + 1 decimal)

float ang[3] = {-90.0,+90.0,-45.5}; // valores entre -90 y 90, en grados. Con signo (2 dígitos + 1 decimal)

bool foto = false; // Si el programa ha tomado o no fotografías.

// -=- DATOS DEL GIRSOCOPIO -=-

short int ax, ay, az;

short int gx, gy, gz; // Valores sin procesar del acelarómetro y giroscopio.

long p0; // Valor de la presión inicial.

long tiempo\_prev;

float dt; // Variables que calculan el tiempo (Cálculos aceleromtro/giroscopio)

float ang\_x, ang\_y, ang\_z;

float ang\_x\_prev, ang\_y\_prev, ang\_z\_prev; // Variables para calcular los ángulos relativos de rotación.

//-=- Variables para el LED

unsigned long t = 0;

// ======================

// SETUP

// ======================

void setup() {

delay(2000);

pinMode(led, OUTPUT);

Serial.begin(9600);

Serial.print("Node ");

Serial.print(MYNODEID,DEC);

Serial.println(" ready");

Wire.begin();

gps.begin(9600);

//cam.begin(115200);

sensor.initialize();

if(sensor.testConnection()){

Serial.println("Sensor MPU6050 iniciado correctamente");

}

else{

Serial.println("Error al iniciar el sensor MPU6050");

}

if(!bmp.begin()){

Serial.print("Error al iniciar el sensor BMP180");

}

else{

Serial.println("Sensor BMP180 iniciado correctamente");

} //Inicializamos comunicación Serial y ambos sensores.

p0 = bmp.readPressure();

Serial.print("Presión inicial = ");

Serial.println(p0);

Serial.println("------------------------------------------"); //Tomamos la presión inical, que usaremos para calcular la altura.

radio.initialize(FREQUENCY, MYNODEID, NETWORKID);

radio.setHighPower(); // Always use this for RFM69HCW

// afinar frecuencia

radio.setFrequency(FREQUENCY\_E);

// Saber más en: https://www.aprendiendoarduino.com/tag/banda-ism/

// rango de 863 a 870 MHz

// Turn on encryption if desired:

if (ENCRYPT) {

radio.encrypt(ENCRYPTKEY); }

Serial.println("Terminado SETUP...");

}

// ======================

// LOOP

// ======================

void loop() {

leerGY87();

extraerCadenaGPS();

delay(800); // Cada 500 milisegundos

enviarDatos(cadenaParaEnviar(1)); // Primero llama a la funcion cadenaParaEnviar con valor 1,

// que formatea los datos de temperatura, presión, altura, aceleración, ángulos.

// Los concatena en una cadena que empieza por @ y termina por #. Tamaño máximo 61 caracteres.

// Al principio incluye un 0, primer dato, y separado por ? el código del centro IES Seritium,

// Posteriormente, separado por ?, el resto de los datos. Acaba con #

delay(800); // cada 500 milisegundos

enviarDatos(cadenaParaEnviar(2)); // Idem caso anterior pero con valor 2.

// envía la latitud, la longitud y si se ha sacado o no foto.

t = millis();

digitalWrite(led, ((t/2500)%2)\*((t/200)%2)\*(altura<= 100)); //Hace parpadear al LED 2 veces cada 10 segundos.

//if(((int(altura) /\*% 10\*/) == 0)/\* and (altura =! alturaPrev) and (altura >30)\*/){ //Envio de datos de altura a la cámara.

/\*if((t % 20000) == 0){

cam.print(0);

cam.flush();

alturaPrev = altura;

}\*/

digitalWrite(6, LOW);

}

// ======================

// Funciones

// ======================

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Enviando cadena

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void enviarDatos(String datos) {

// Envío de datos

static char sendbuffer[62];

static int sendlength = 0;

sendlength = datos.length();

datos.toCharArray(sendbuffer,sendlength);

Serial.println(datos);

if (USEACK){

if (radio.sendWithRetry(TONODEID, sendbuffer, sendlength))

Serial.println("ACK received!");

else

Serial.println("no ACK received");

}

// If you don't need acknowledgements, just use send():

else // don't use ACK

{

radio.send(TONODEID, sendbuffer, sendlength);

Serial.println("Enviado...");

}

sendlength=0;

}

void leerGY87(){

sensor.getAcceleration(&ax, &ay, &az);

sensor.getRotation(&gx, &gy, &gz); //Obtenemos las lecturas de aceleración y girsocopio.

dt = (millis()-tiempo\_prev)/1000.0;

tiempo\_prev = millis(); //Calculamos el tiempo pasado entre cada medición.

float ang\_x\_accel = atan(ax/sqrt(pow(ay,2) + pow(az,2)))\*(180.0/3.14);

float ang\_y\_accel = atan(ay/sqrt(pow(az,2) + pow(ax,2)))\*(180.0/3.14);

float ang\_z\_accel = atan(az/sqrt(pow(ax,2) + pow(ay,2)))\*(180.0/3.14); //Calculamos los angulos de inclinación en los ejes usando el acelerómetro.

ang\_x = 0.98\*(ang\_x\_prev+(gx/131)\*dt) + 0.02\*ang\_x\_accel;

ang\_y = 0.98\*(ang\_y\_prev+(gy/131)\*dt) + 0.02\*ang\_y\_accel;

ang\_z = 0.98\*(ang\_z\_prev+(gy/131)\*dt) + 0.02\*ang\_z\_accel; //Calculamos la rotación usando tanto el girsocopio como el acelerómetro para eliminar errores.

ang\_x\_prev = ang\_x;

ang\_y\_prev = ang\_y;

ang\_z\_prev = ang\_z;

ace[2] = az \* (9.81/16384.0);

ace[1] = ay \* (9.81/16384.0);

ace[0] = ax \* (9.81/16384.0); //Calculamos la aceleración en m/s2 en los tres ejes

ang[0] = ang\_x\_accel;

ang[1] = ang\_y\_accel;

ang[2] = ang\_z\_accel;

altura = bmp.readAltitude(p0);

presion = bmp.readPressure();

//float m\_az\_ax = sqrt(pow(sqrt((pow(ax\_ms2, 2)) + (pow(az\_ms2, 2))), 2) + (pow(ay\_ms2, 2))); //Calculamos el módulo de los 3 vectores, para así obtener la aceleración que experimenta.--> NO SE USA POR AHORA.

Serial.print("Rotación en X: ");

Serial.print(ang[0]);

Serial.print("º\tRotación en Y: ");

Serial.print(ang[1]);

Serial.print("º \tAceleración de caída(m/s2): ");

Serial.print(ace[2]);

Serial.print("\tTemperatura = ");

temperatura = bmp.readTemperature();

Serial.print(temperatura);

Serial.print("ºC\tPresión = ");

Serial.print(presion);

Serial.print("\tAltitud respecto al suelo = ");

Serial.print(altura);

Serial.println("m"); //Mostramos por puerto serie los datos obtenidos.

}

void extraerCadenaGPS(){

if(gps.available()){

String cadenaFinal = gps.readStringUntil('\n'); //Leemos lo que nos ha llegado hasta el salto de línea.

if (cadenaFinal.startsWith("$GPRMC")) {

int pos = cadenaFinal.indexOf("$GPRMC"); //Si empieza por el código que queremos, pillamos la posición de inicio.

if (pos == 0) {

cadenaGPS = cadenaFinal;

Serial.println(cadenaFinal); //Si la posición es 1, significa que tenemos toda la cadena, así que la recortamos y la imprimimos.

}

}

}

}

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// cadena ya formateada para enviar

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

String cadenaParaEnviar (int cual) {

char tmp[62]; // variable de 62 bytes, del 0 al 61

switch (cual) {

// en este caso formateo los datos de temperatura, presión, altura, aceleración, ángulos.

case 1:

sprintf(tmp,"@%1d?%8d?%1d%3d?%6d?%5d?%1d%3d?%1d%3d?%1d%3d?%1d%3d?%1d%3d?%1d%3d?#",cual,codigoSeritium

,(temperatura>=0),(int) (abs(temperatura\*10))

,presion,(int) (altura\*10)

,(ace[0]>=0), (int) abs(ace[0]\*10),(ace[1]>=0), (int) abs(ace[1]\*10),(ace[2]>=0), (int) abs(ace[2]\*10)

,(ang[0]>=0), (int) abs(ang[0]\*10),(ang[1]>=0), (int) abs(ang[1]\*10),(ang[2]>=0), (int) abs(ang[2]\*10) );

break;

// en este caso envío los datos de latitud, longitud y si se ha sacado una foto

case 2:

sprintf(tmp,"@%1d?%8d?%6d%1s?%7d%1s?%1d?#",cual,codigoSeritium

,(int) ( 100.0 \* parte(cadenaGPS,',',3).toFloat() ), parte(cadenaGPS,',',4)

,(int) ( 100.0 \* parte(cadenaGPS,',',5).toFloat() ), parte(cadenaGPS,',',6)

,(int) foto);

// Serial.println(parte(cadenaGPS,',',0)); // importante usar simple quotas ' ' ' '

break;

// por defecto

default:

break;

}

return tmp; // devuelve el conjunto de caracteres

}

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// encuentra parte de una cadena

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

String parte(String data, char separator, int index)

{

int found = 0;

int strIndex[] = {0, -1};

int maxIndex = data.length()-1;

for(int i=0; i<=maxIndex && found<=index; i++){

if(data.charAt(i)==separator || i==maxIndex){

found++;

strIndex[0] = strIndex[1]+1;

strIndex[1] = (i == maxIndex) ? i+1 : i;

}

}

return found>index ? data.substring(strIndex[0], strIndex[1]) : "";

}

===============================================

**Programa en el receptor (estación TIERRA)**

/\*

\* Ejemplo de envío de datos por telemetría

\* Dispositivo: NodeMCU con valores lógicos a 3.3V

\* Fecha: 8/1/2022

\*/

// Include the RFM69 and SPI libraries:

#include <RFM69.h>

#include <SPI.h>

// Addresses for this node. CHANGE THESE FOR EACH NODE!

#define NETWORKID 100 // Must be the same for all nodes --> esta es la red.

#define MYNODEID 20 // My node ID --> este es el nodo de mi dispositivo

#define TONODEID 10 // Destination node ID --> este es el nodo del otro dispositivo

// RFM69 frequency, uncomment the frequency of your module:

//#define FREQUENCY RF69\_433MHZ

// #define FREQUENCY RF69\_915MHZ

#define FREQUENCY RF69\_868MHZ

#define FREQUENCY\_E 868001000UL // frecuencia exacta

// AES encryption (or not):

#define ENCRYPT true // Set to "true" to use encryption

#define ENCRYPTKEY "123456789" // Use the same 16-byte key on all nodes --> clave secreta

// Use ACKnowledge when sending messages (or not):

#define USEACK false // Request ACKs or not

// Packet sent/received indicator LED (optional):

// #define LED 9 // LED positive pin

// #define GND 8 // LED ground pin

// Según la web https://iot.uy/conectar-la-radio-rfm69-hcw-al-esp8266/

#define SPI\_CS D8 //NSS o Cable Select al GPIO15 o D8

#define IRQ\_PIN D1 // DIO0 conectado a GPIO4

#define IRQ\_NUM D1 //IRQ igual al pin

#define IS\_RFM69HCW true // Si tu radio es RFM69HCW entonces va "true"

#define POWER\_LEVEL 31 // Valor máximo de potencia

// Create a library object for our RFM69HCW module:

RFM69 radio = RFM69(SPI\_CS, IRQ\_PIN, IS\_RFM69HCW, IRQ\_NUM);

// Datos que deben ser registrados

// GPS -->

String latitud ="";

String longitud="";

// Valores

unsigned int codigoSeritium = 0; // Código del centro

float temperatura = 0.0; // Ejemplo de temperatura con signo (2 dígitos + 1 decimal)

unsigned int presion = 0; // Ejemplo de presión atmosférica. Entero positivo sin decimales.

float altura = 0.0; // Ejemplo de altura. Positivo sin signo (4 dígitos + 1 decimal)

float ace[3] = {0.0, 0.0, 0.0}; // valores entre -4g y 4g, en m/s2. Con signo (2 dígitos + 1 decimal)

float ang[3] = {0.0, 0.0, 0.0}; // valores entre -90 y 90, en grados. Con signo (2 dígitos + 1 decimal)

bool foto = false; // Si el programa ha tomado o no fotografías.

// potencia recibida

int potencia = 0;

// ======================

// SETUP

// ======================

void setup() {

delay(2000);

Serial.begin(9600);

Serial.print("Node ");

Serial.print(MYNODEID,DEC);

Serial.println(" ready");

// Initialize the RFM69HCW:

// radio.setCS(10); //uncomment this if using Pro Micro

radio.initialize(FREQUENCY, MYNODEID, NETWORKID);

radio.setHighPower(); // Always use this for RFM69HCW

// afinar frecuencia

radio.setFrequency(FREQUENCY\_E);

// Saber más en: https://www.aprendiendoarduino.com/tag/banda-ism/

// rango de 863 a 870 MHz

// Turn on encryption if desired:

if (ENCRYPT) {

radio.encrypt(ENCRYPTKEY); }

Serial.println("Terminado SETUP...");

}

// ======================

// LOOP

// ======================

void loop() {

delay(100); // tiene que ser más rápido que el envío de datos... Si se envían cada 500 ms, por ejemplo,

// aquí ponemos la mitad o así...

String cadena = recibirDatos();

if (cadena.length()>2) {analizaCadena(cadena); } // lama a la función que empieza a leer los datos.

} // Fin del LOOP

// ======================

// Funciones

// ======================

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Recibiendo cadena

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

String recibirDatos() {

char temp[63]; // un byte más de lo necesario

// String tmp;

if (radio.receiveDone()) // Got one!

{

// Print out the information:

// Serial.println( (int) radio.DATALEN);

// Serial.println(radio.DATA);

// The actual message is contained in the DATA array,

// and is DATALEN bytes in size:

for (byte i = 0; i < radio.DATALEN; i++) {

char n = (char)radio.DATA[i];

if (n=='#') {

temp[i]=NULL ; break;

} else if (n=='@') {

temp[i]='?';

} else {

temp[i]= n;

}

// Serial.print((char)radio.DATA[i]);

// Serial.print(temp[i]);

// tmp.concat(temp[i]);

}

// int bufferLength = sizeof(temp);

// Serial.println("Tamaño del buffer: " + (String) bufferLength);

// Serial.print("received from node ");

// Serial.print(radio.SENDERID, DEC);

// Serial.print(", message [");

// for (int j=0; j<bufferLength; j++) {

// Serial.print(temp[j]);

// }

// RSSI is the "Receive Signal Strength Indicator",

// smaller numbers mean higher power.

// Serial.print("], RSSI ");

// Serial.println(radio.RSSI);

potencia = (int) radio.RSSI;

// Send an ACK if requested.

// (You don't need this code if you're not using ACKs.)

if (radio.ACKRequested())

{

radio.sendACK();

Serial.println("ACK sent");

}

}

return String(temp);

// return tmp;

}

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// análisis de cadena

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void analizaCadena (String trama) {

String d; // dato que recibe

String comodin; // cadena comodin

d = parte(trama,'?',1); // debe recibir o un 1 o un 2

codigoSeritium = parte(trama,'?',2).toInt();

switch (d.toInt()) {

// recibo la primera cadena

case 1:

Serial.println("Potencia: " + (String) potencia + " // Cadena 1 --> " + trama);

Serial.println("Recibido para el centro: "+ (String) codigoSeritium);

// Temperatura

temperatura = recValorFloat (parte(trama,'?',3), 10, true); // cadena a recuperar, divisor, si tiene signo

// Presión

comodin = parte(trama,'?',4);

presion = comodin.toInt();

// Altura

altura = recValorFloat (parte(trama,'?',5), 10, false);

// Aceleracion

comodin ="Aceleraciones: ";

for (int i=0;i<3;i++) {

ace[i]=recValorFloat (parte(trama,'?',i+6), 10, true); // valores 6, 7 y 8

comodin = comodin + "A"+char(120+i)+" = "+ace[i]+" m/s2 ~ ";

}

comodin = comodin.substring(0,comodin.length()-2);

// Ángulos

comodin = comodin + "\nÁngulos: ";

for (int i=0;i<3;i++) {

ang[i]=recValorFloat (parte(trama,'?',i+9), 10, true); // valores 9, 10 y 11

comodin = comodin + "ANG"+char(120+i)+" = "+ang[i]+" º ~ ";

}

comodin = comodin.substring(0,comodin.length()-2);

// Muestra los datos en el monitor serie

Serial.println("T = "+(String) temperatura +" º C // "+

"P = "+(String) presion +" Pa // "+

"h = "+(String) altura +" m // ");

Serial.println(comodin);

break;

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// recibo la segunda cadena

case 2:

Serial.println("Potencia: " + (String) potencia + " // Cadena 2 --> " + trama);

Serial.println("Recibido para el centro: "+ (String) codigoSeritium);

// latitud

comodin = parte(trama,'?',3);

latitud = comodin.substring(0,2)+"º "+(String) recValorFloat(comodin.substring(2,6), 100, false)

+"' "+ cambioLetra(comodin.substring(6,7));

// longitud

comodin = parte(trama,'?',4);

longitud = comodin.substring(0,3)+"º "+(String) recValorFloat(comodin.substring(3,7), 100, false)

+"' "+ cambioLetra(comodin.substring(7,8));

// Fotografía

comodin = parte(trama,'?',5);

foto = (comodin=="1");

comodin = (foto) ? "Sí" : "No";

Serial.println("LAT: "+latitud+ " // LON: "+longitud);

Serial.println("¿Se ha tomado una foto?: "+ comodin);

break;

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

default:

// Serial.println("No he encontrado una cadena: " + trama);

return; // sale de la función

break;

}

// imprime línea de caracteres

int iguales = 91;

while (--iguales) { Serial.print("="); }

Serial.print("\n");

}

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Recupera valores float

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

float recValorFloat (String num, int divisor, bool signo) {

float valor = 0.0;

int longitud = num.length();

if (signo) {

valor = num.substring(1,longitud).toFloat()/divisor;

valor = (num.substring(0,1)=="0") ? valor = -1\*valor : valor;

} else {

valor = num.toFloat()/divisor;

}

return valor;

}

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// cambio letra en longitud

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

String cambioLetra (String letra) {

return (letra=="W") ? "Oeste" : (letra=="E") ? "Este" : (letra=="N") ? "Norte" : (letra=="S") ? "Sur" : "" ;

}

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// encuentra parte de una cadena

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

String parte(String data, char separator, int index)

{

int found = 0;

int strIndex[] = {0, -1};

int maxIndex = data.length()-1;

for(int i=0; i<=maxIndex && found<=index; i++){

if(data.charAt(i)==separator || i==maxIndex){

found++;

strIndex[0] = strIndex[1]+1;

strIndex[1] = (i == maxIndex) ? i+1 : i;

}

}

return found>index ? data.substring(strIndex[0], strIndex[1]) : "";

}

===============================================

**Programa en el ESP32-CAM**

#include "esp\_camera.h"

#include "Arduino.h"

#include "FS.h" // SD Card ESP32

#include "SD\_MMC.h" // SD Card ESP32

#include "soc/soc.h" // Disable brownour problems

#include "soc/rtc\_cntl\_reg.h" // Disable brownour problems

#include "driver/rtc\_io.h"

#include <EEPROM.h> // read and write from flash memory

// define the number of bytes you want to access

#define EEPROM\_SIZE 1

// Pin definition for CAMERA\_MODEL\_AI\_THINKER

#define PWDN\_GPIO\_NUM 32

#define RESET\_GPIO\_NUM -1

#define XCLK\_GPIO\_NUM 0

#define SIOD\_GPIO\_NUM 26

#define SIOC\_GPIO\_NUM 27

#define Y9\_GPIO\_NUM 35

#define Y8\_GPIO\_NUM 34

#define Y7\_GPIO\_NUM 39

#define Y6\_GPIO\_NUM 36

#define Y5\_GPIO\_NUM 21

#define Y4\_GPIO\_NUM 19

#define Y3\_GPIO\_NUM 18

#define Y2\_GPIO\_NUM 5

#define VSYNC\_GPIO\_NUM 25

#define HREF\_GPIO\_NUM 23

#define PCLK\_GPIO\_NUM 22

int pictureNumber = 0;

void setup() {

WRITE\_PERI\_REG(RTC\_CNTL\_BROWN\_OUT\_REG, 0); //disable brownout detector

Serial.begin(115200);

pinMode(4, OUTPUT);

//Serial.setDebugOutput(true);

//Serial.println();

camera\_config\_t config;

config.ledc\_channel = LEDC\_CHANNEL\_0;

config.ledc\_timer = LEDC\_TIMER\_0;

config.pin\_d0 = Y2\_GPIO\_NUM;

config.pin\_d1 = Y3\_GPIO\_NUM;

config.pin\_d2 = Y4\_GPIO\_NUM;

config.pin\_d3 = Y5\_GPIO\_NUM;

config.pin\_d4 = Y6\_GPIO\_NUM;

config.pin\_d5 = Y7\_GPIO\_NUM;

config.pin\_d6 = Y8\_GPIO\_NUM;

config.pin\_d7 = Y9\_GPIO\_NUM;

config.pin\_xclk = XCLK\_GPIO\_NUM;

config.pin\_pclk = PCLK\_GPIO\_NUM;

config.pin\_vsync = VSYNC\_GPIO\_NUM;

config.pin\_href = HREF\_GPIO\_NUM;

config.pin\_sscb\_sda = SIOD\_GPIO\_NUM;

config.pin\_sscb\_scl = SIOC\_GPIO\_NUM;

config.pin\_pwdn = PWDN\_GPIO\_NUM;

config.pin\_reset = RESET\_GPIO\_NUM;

config.xclk\_freq\_hz = 20000000;

config.pixel\_format = PIXFORMAT\_JPEG;

if(psramFound()){

config.frame\_size = FRAMESIZE\_UXGA; // FRAMESIZE\_ + QVGA|CIF|VGA|SVGA|XGA|SXGA|UXGA

config.jpeg\_quality = 10;

config.fb\_count = 2;

} else {

config.frame\_size = FRAMESIZE\_SVGA;

config.jpeg\_quality = 12;

config.fb\_count = 1;

}

// Init Camera

esp\_err\_t err = esp\_camera\_init(&config);

if (err != ESP\_OK) {

Serial.printf("Camera init failed with error 0x%x", err);

return;

}

//Serial.println("Starting SD Card");

if(!SD\_MMC.begin()){

Serial.println("SD Card Mount Failed");

return;

}

uint8\_t cardType = SD\_MMC.cardType();

if(cardType == CARD\_NONE){

Serial.println("No SD Card attached");

return;

}

}

void loop() {

int t = millis();

if((t%10000) == 0){

camera\_fb\_t \* fb = NULL; // Take Picture with Camera

fb = esp\_camera\_fb\_get();

if(!fb) {

Serial.println("Camera capture failed");

return;

}

// initialize EEPROM with predefined size

//EEPROM.begin(EEPROM\_SIZE);

//pictureNumber = EEPROM.read(0) + 1;

pictureNumber = t/10000;

// Path where new picture will be saved in SD Card

String path = "/picture" + String(pictureNumber) +".jpg";

fs::FS &fs = SD\_MMC;

Serial.printf("Picture file name: %s\n", path.c\_str());

File file = fs.open(path.c\_str(), FILE\_WRITE);

if(!file){

Serial.println("Failed to open file in writing mode");

}

else {

file.write(fb->buf, fb->len); // payload (image), payload length

Serial.printf("Saved file to path: %s\n", path.c\_str());

//EEPROM.write(0, pictureNumber);

//EEPROM.commit();

}

file.close();

esp\_camera\_fb\_return(fb);

// Turns off the ESP32-CAM white on-board LED (flash) connected to GPIO 4

digitalWrite(4, LOW);

}

}