

CANSAT accesible con ArduinoBlocks y Processing



Por: *Pedro Ruiz Fernández @pedroruizf*

Con la colaboración inestimable de *Equipo Sotosat (IES Pedro Soto de Rojas)*, *Lorenzo Olmo*, *Manuel Hidalgo @leobotmanuel*, y *Pepe Alcaide*

Versión 28/10/2020

Licencia



Nota Importante: La Información compartida en este documento está realizada por profesor y equipo novato en CANSAT, así como todo los dispositivos y su programación están en fase de pruebas, por tanto, hay que tomarla con cautela y provisionalidad.

| | |
|---|-----------|
| Introducción | 2 |
| Información general | 2 |
| Paracaidas | 2 |
| Cálculo | 2 |
| Trazado | 3 |
| Enlaces complementarios | 4 |
| Estructura | 5 |
| Comunicaciones por radio (antenas) | 6 |
| Software | 7 |
| Misión Primaria | 7 |
| Solución Electrónica con Arduino | 7 |
| Componentes y piezas: | 7 |
| Pin-out de Cansat con Arduino Mega 2560 Pro Mini y esquema con fritzing | 10 |
| Alimentación de Arduino Mega | 12 |
| Programa en Arduinoblocks | 13 |
| Programa de estación base (PC) en processing | 15 |
| Misión Secundaria | 17 |
| Telemetría avanzada, tratamiento y graficado de datos | 17 |
| Proyecto científico | 19 |
| Objetivos | 19 |
| Imágenes | 19 |
| Vídeos | 22 |
| Webs de interés | 22 |

Introducción

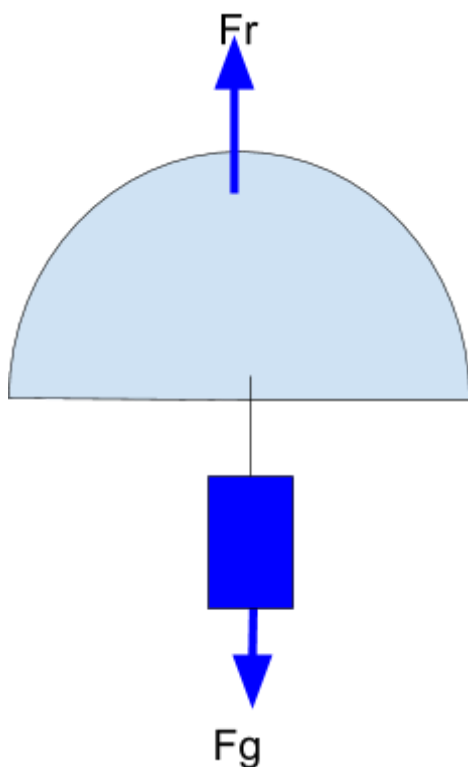
Se trata del desarrollo del proyecto CANSAT de manera que la parte electrónica sea asequible en cuanto a coste y complejidad en cuanto a programación. Para ello hemos elegido una plataforma de control Arduino Mega 2560 pro mini programada bajo la aplicación web “Arduinoblocks”, programación gráfica basada en bloques, además se ha implementado en “Processing” un programa para la estación base que filtra los posibles datos erróneos y los datos correctos los guarda en un fichero, también en la estación base se produce el graficado de los diferentes datos con el graficador libre KST. Esta combinación hace que el coste del proyecto para misiones primarias y secundarias fáciles sean asequibles, así como la programación de las mismas.

Información general

- Web del proyecto: <http://esero.es/cansat/>
- Presentación: <https://docs.google.com/presentation/d/1lyj4ezujqL82Y-b0klbfKxgbNGXrmlXI3QwgTSKNW3c/edit?usp=sharing>
- Requisitos: <http://esero.es/cansat/requisitos/>
- Recursos asociados: <http://esero.es/cansat/recursos-asociados/>

Paracaídas

El paracaídas está en fase de pruebas y todavía no se han realizado lanzamientos de prueba.



Cálculo

Datos de Cálculo: Masa cansat entre 300 y 350 g, Tiempo de vuelo recomendado máximo 120 s, Velocidad de descenso entre 6 y 12m/s (recomendadas entre 8 y 11m/s).

Para calcular vamos a igualar la fuerza de resistencia del paracaídas (F_r) a el peso que mueve el paracaídas ($F_g = mg$) hacia el suelo. Cuando estén igualadas deja de acelerar y la velocidad se convierte en constante, esa es la velocidad de caída.

$$\Sigma F = m.a, F_r - F_g = m.a, \text{ como } F_r = F_g, 0 = m.a; a = 0$$

$F_g = m.g$ (peso cansat).

$F_r = \frac{1}{2} \cdot r \cdot C_d \cdot A \cdot V^2$ (fuerza de resistencia del paracaídas)

- r = densidad del aire (1,22 Kg/m³)
- C_d = coeficiente de resistencia aerodinámico en paracaídas de forma hemiesférico su valor es 0,62.
- A = superficie del paracaídas (en nuestro caso semiesfera)
- V = velocidad de descenso

igualando las dos expresiones $m.g = \frac{1}{2} \cdot r \cdot C_d \cdot A \cdot V^2$ de estas dos expresiones puedo despejar el área del paracaídas, $A = 2 \cdot (m.g) / r \cdot C_d \cdot V^2$

Si la superficie del paracaídas es una semiesfera puedo calcular su diámetro, $A=2\pi R^2$, por tanto igualando

$$\frac{2m \cdot g}{r \cdot C_d \cdot V^2} = 2 \cdot \pi \cdot R^2, \text{ despejando } R = \sqrt{\frac{m \cdot g}{r \cdot C_d \cdot V^2 \cdot \pi}}$$

Para determinar la velocidad de caída y si la velocidad es constante (sin aceleración), tenemos que $V=e/t$, el cohete alcanza 1000 m de altura (espacio a recorrer) y el tiempo dijimos 120 s, por tanto ..., ya tenemos la V, además tenemos la masa de nuestro cansat (de 300g a 350g), con lo cual ya tenemos todo para calcular nuestro radio o diámetro de paracaídas.

Trazado

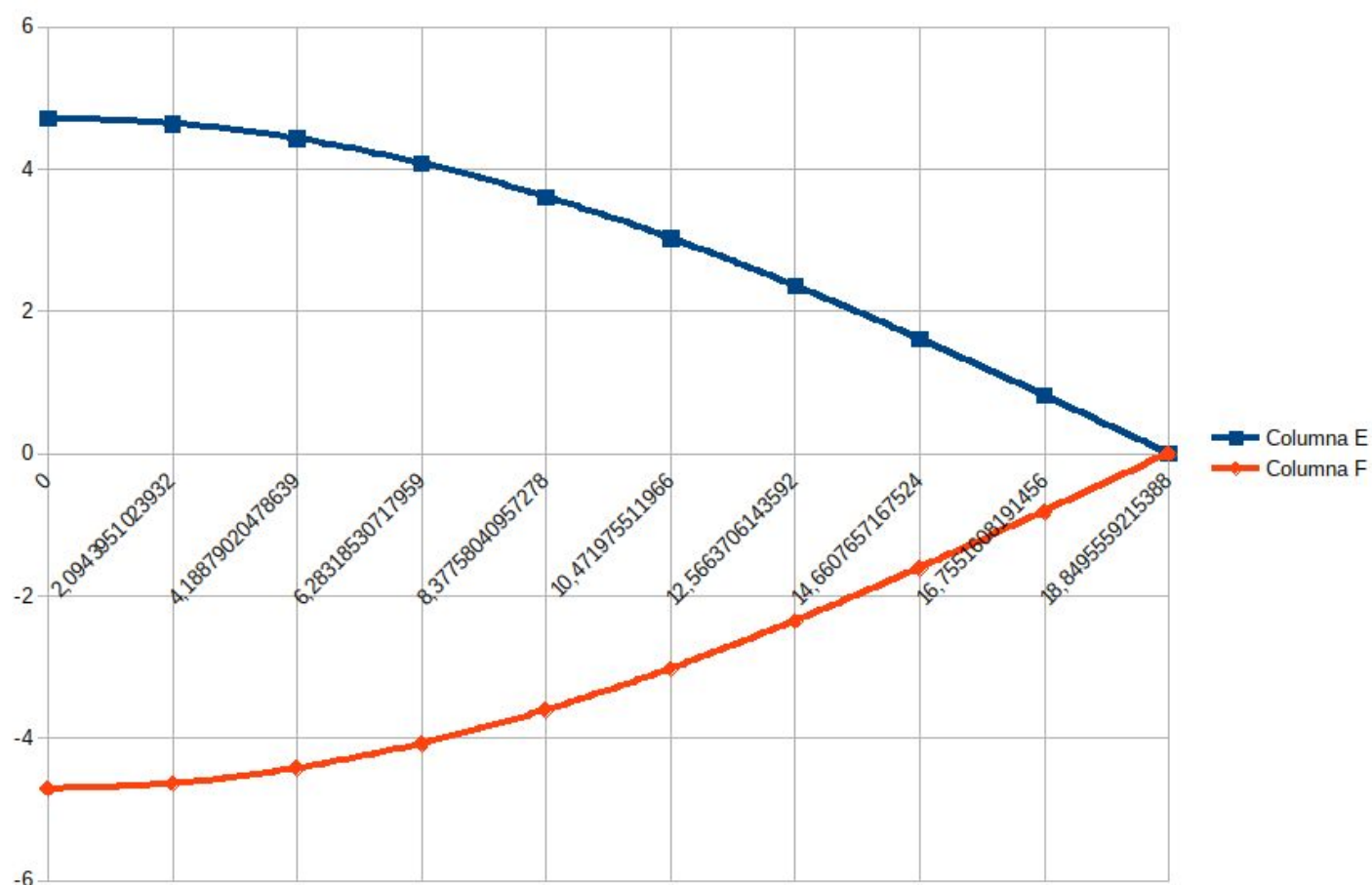
Además hemos realizado una [hoja de cálculo](#) para realizar los cálculos de la forma y dimensiones de nuestros gores (secciones) de nuestro paracaídas, en función del radio elegido y el número de secciones.

Cálculo de coordenadas:

- $X=\alpha \cdot \text{radio}$, posición x de los puntos del gore
- $Y=(2 \cdot \pi \cdot \text{radio} \cdot \cos \alpha) / (n^\circ \text{ gores} \cdot 2)$, posición y de los puntos del gore (hay dos una positiva y otra negativa simétricas)

| radio (cm) | angulos (rad) (α) | angulos | x | y | y- | base |
|------------|-------------------|---------|------------------|------------------|-------------------|------------------|
| 12 | 0 | 0 | 0 | 4,71238898038469 | -4,71238898038469 | 9,42477796076938 |
| n.º gores | 0,174532925199433 | 10 | 2,0943951023932 | 4,64079720309214 | -4,64079720309214 | |
| 8 | 0,349065850398866 | 20 | 4,18879020478639 | 4,42819715114032 | -4,42819715114032 | |
| | 0,523598775598299 | 30 | 6,28318530717959 | 4,08104856952699 | -4,08104856952699 | |
| | 0,698131700797732 | 40 | 8,37758040957278 | 3,6098993922388 | -3,6098993922388 | |
| | 0,872664625997165 | 50 | 10,471975511966 | 3,02906524861466 | -3,02906524861466 | |
| | 1,0471975511966 | 60 | 12,5663706143592 | 2,35619449019235 | -2,35619449019235 | |
| | 1,22173047639603 | 70 | 14,6607657167524 | 1,61173195447747 | -1,61173195447747 | |

| | | | | | | |
|--|--------------------------|----|----------------------|--------------------------|---------------------------|--|
| | 1,396263 4015954 6 | 80 | 16,755160819 1456 | 0,818297758901 526 | -0,818297758901 526 | |
| | 1,570796 3267949 | 90 | 18,849555921 5388 | 2,885506040582 68E-16 | -2,885506040582 68E-16 | |

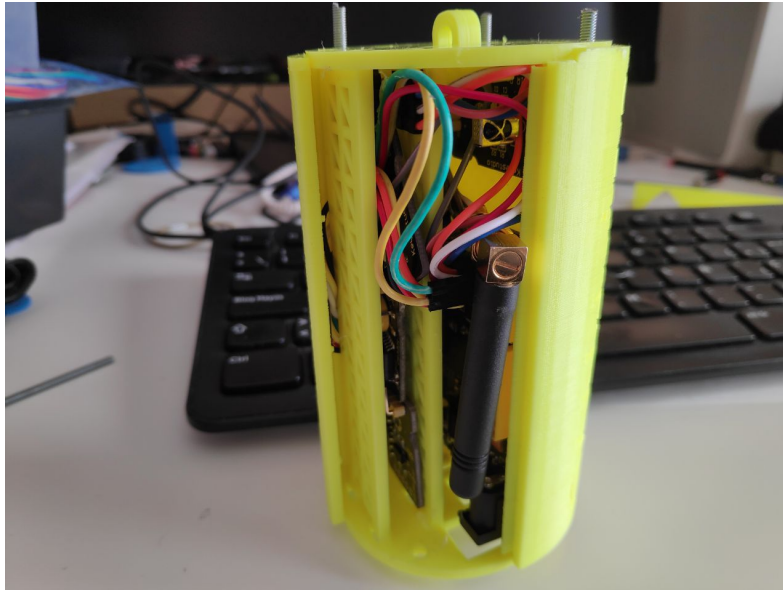


Enlaces complementarios

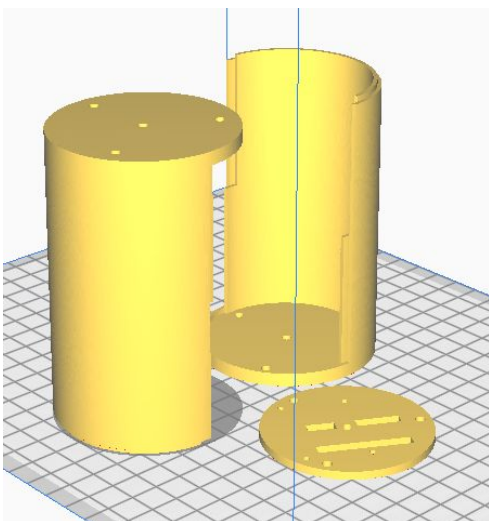
- Documento sobre paracaídas: http://esero.es/wp-content/uploads/2019/10/T10_Parachute_Design.pdf
- Cálculo de paracaídas: <http://www.rocketmime.com/rockets/descent.html>
- Construcción de paracaídas hemisférico: <https://www.rocketreviews.com/making-a-hemispheric-nylon-parachute-9342.html>
- Sitio de compra y cálculo (Fruity Chutes sita en Monte Sereno, CA, US): <https://fruitychutes.com/>
- Diseño y construcción de paracaídas: <http://www.nakka-rocketry.net/paracon.html>
- Hoja de cálculo para diseño: http://www.nakka-rocketry.net/soft/parapat_v1.1.xls
- Compra de tela: [aquí](#) (ancho 1,5 m, precio por metro lineal de ese ancho)

Estructura

- Estructura Canduino: <https://canduino.eu/index.php?id=3d-models> (es la usada en la solución por ahora definitiva).
 - Pequeña modificación a la tapa superior de canduino para pasar cable antena GPS: <https://www.tinkercad.com/things/bsFOAFMVQrr>
 - Pequeña modificación de la tapa inferior para dejar ventana para el receptor de infrarrojo: <https://www.tinkercad.com/things/5xsrRgHKRIN>
 - Pequeña modificación de la base principal (board 1) para dejar hueco para soldar pines Vin u gnd de alimentación: <https://www.tinkercad.com/things/jOn9D70C28O>



- Estructura oficial “cansat” stl: http://esamultimedia.esa.int/docs/edu/3d_printer_files_for_Cansat_case.zip, la estructura tiene los archivos shell (recipiente exterior), que se imprime dos veces y shield que es la bandeja separadora para colocar componentes electrónicos y de control.

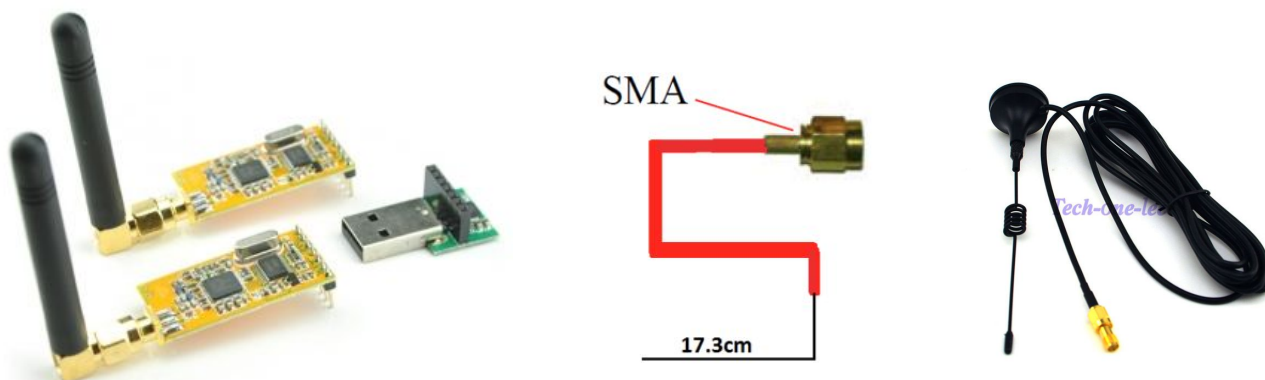


Comunicaciones por radio (antenas)

El sistema está compuesto por emisor de radiofrecuencia APC220 a 434 Mhz situado en el cuerpo del cansat, con una antena con conector SMA y de longitud 17,3 cm (ver web (<https://www.narom.no/undervisningsressurser/the-cansat-book/the-primary-mission/cansat-mechanics-design/antenna-design/>), y un receptor APC200 con adaptador serie-usb para conectarlo al PC. La antena que lleva el receptor es una de ¼ de onda omnidireccional. La transmisión se ha ajustado a una velocidad de 2400 baudios y a nivel de potencia 9 para un mayor alcance, para el ajuste de los dispositivos del kit APC220 hemos usado esta web (<http://beetlecraft.blogspot.com/2015/10/tutorial-apc220.html>).

Para el cálculo de la longitud de la antena emisora se usa la fórmula, $L = \frac{c}{4f}$, donde c es la velocidad de la luz y f la frecuencia de transmisión en Hertzios (Hz), por tanto $L = \frac{3 \cdot 10^8 \text{ m/s}}{4 (434 \cdot 10^6 \text{ 1/s})} = 0,173 \text{ m} = 17,3 \text{ cm}$.

Para construir la antena se corta un cable coaxial con conector SMA y se desenfunda 17,3 cm, dejando sólo el cable central con su funda y quitando el resto.



Documentación:

- Documento: http://esero.es/wp-content/uploads/2019/10/T11_Radio_Communication.pdf
- Construcción antena emisor apc220: <https://www.narom.no/undervisningsressurser/the-cansat-book/the-primary-mission/cansat-mechanics-design/antenna-design/>
- Configuración APC220: <http://beetlecraft.blogspot.com/2015/10/tutorial-apc220.html>
- Piezas para carcasa de receptor de radiofrecuencia APC220: <https://www.thingiverse.com/thing:643721>



Software

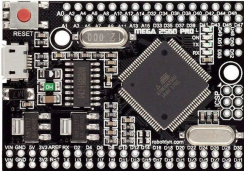

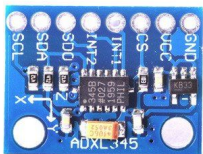
- [Arduinoblocks](#): maravilloso entorno gráfico de programación online de diferentes plataformas arduino.
- [Processing](#): lenguaje de programación en el que se basa Arduino, con el nos comunicaremos por puerto serie para recibir datos en el ordenador y filtrar los correctos.
- [KST](#) (graficado en tiempo real): programa para realizar graficado en tiempo real.




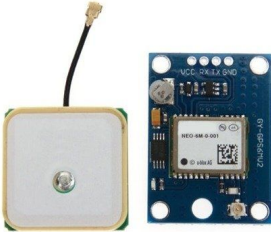
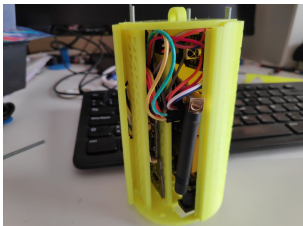
Misión Primaria



Solución Electrónica con Arduino

Dicha solución está operativa y funcionando, queda por realizarle pruebas de alcance en la radiotransmisión.

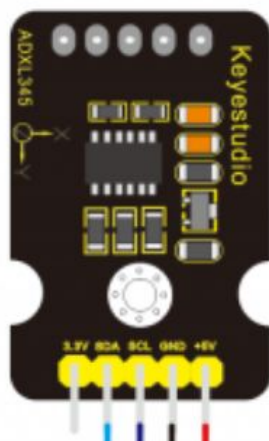
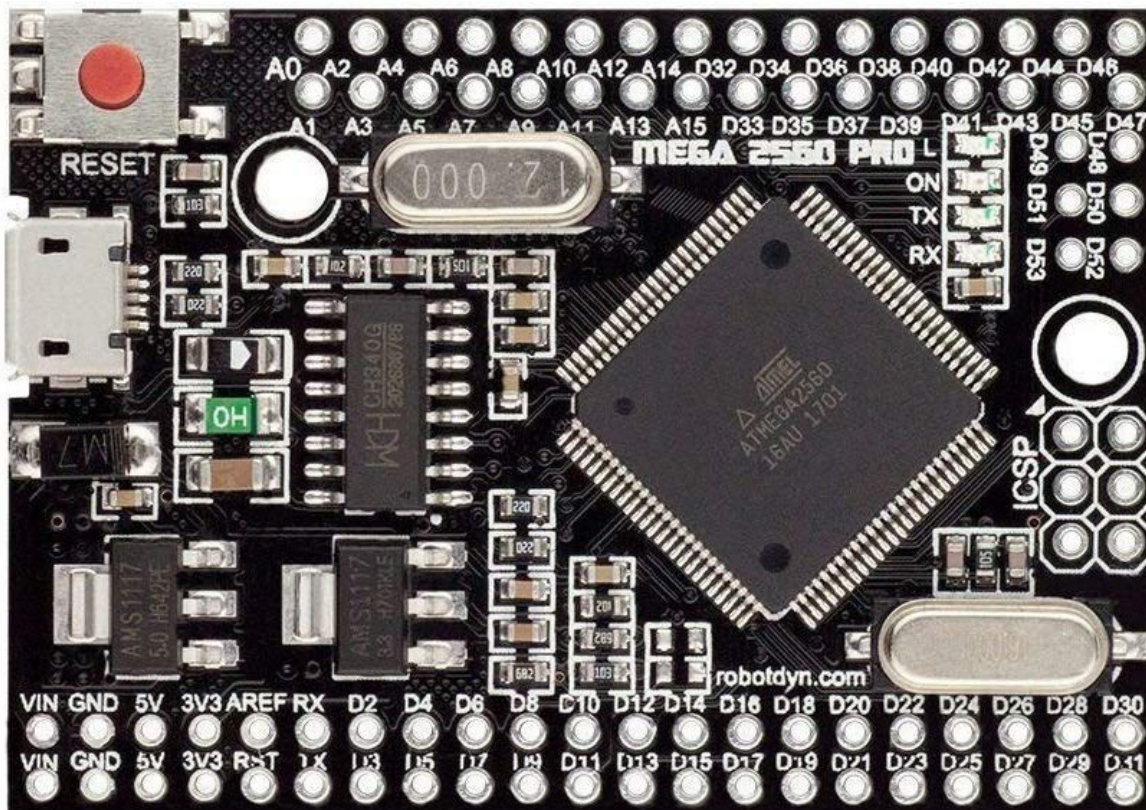
Componentes y piezas:

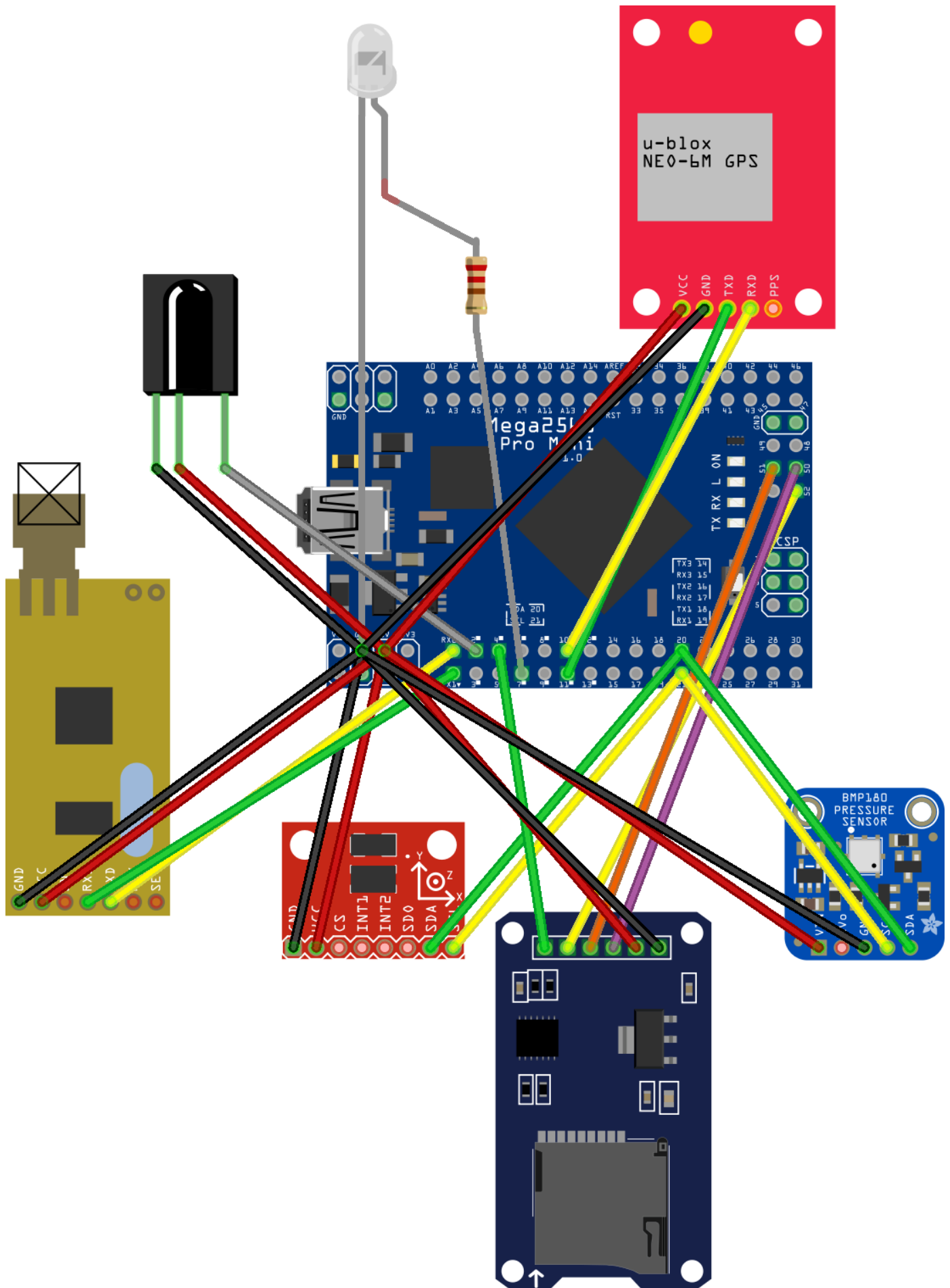
| Item | Descripción | Imagen | Cantidad | Pin OUT | Compr a |
|------|--|---|----------|-----------------------------------|---|
| 1 | Arduino Mega 2560 Pro Mini |  | 1 | . | Aquí |
| 2 | Acelerómetro ADXL345 |  | 1 | I2C: sda (20), scl (21), GND, VCC | Aquí o Aquí |
| 3 | Sensor presión, temperatura y altitud BMP180 |  | 1 | I2C: sda (20), scl (21), GND, VCC | Aquí |

| | | | | | |
|---|---|---|---------|--|---|
| 4 | Receptor de IR |  | 1 | GND,VCC, Signal (D2) | Aquí |
| 5 | Modulo tarjeta microSD por SPI |  | 1 | GND,VCC, CS (D4), MOSI (51), SCK (52), MISO (50) | Aquí |
| 6 | Emisor y receptor (en PC) de radiofrecuencia APC220 |  | 1 | GND, VCC, RX (TX), TX (RX) | Aquí |
| 7 | GPS |  | 1 | GND, VCC, RX (D10), TX (D11) | Aquí o Aquí |
| 8 | Piezas estructura canduino |  | 1 juego | | Aquí |

| | | | | | |
|----|-------------------------|---|------------------|--|----------------------|
| 9 | Carcasa receptor APC220 |  | 1 juego | | Aquí |
| 10 | Varilla roscada M3 |  | 1 m (4 x 125 mm) | | |
| 11 | Tuercas M3 |  | 8 | | |

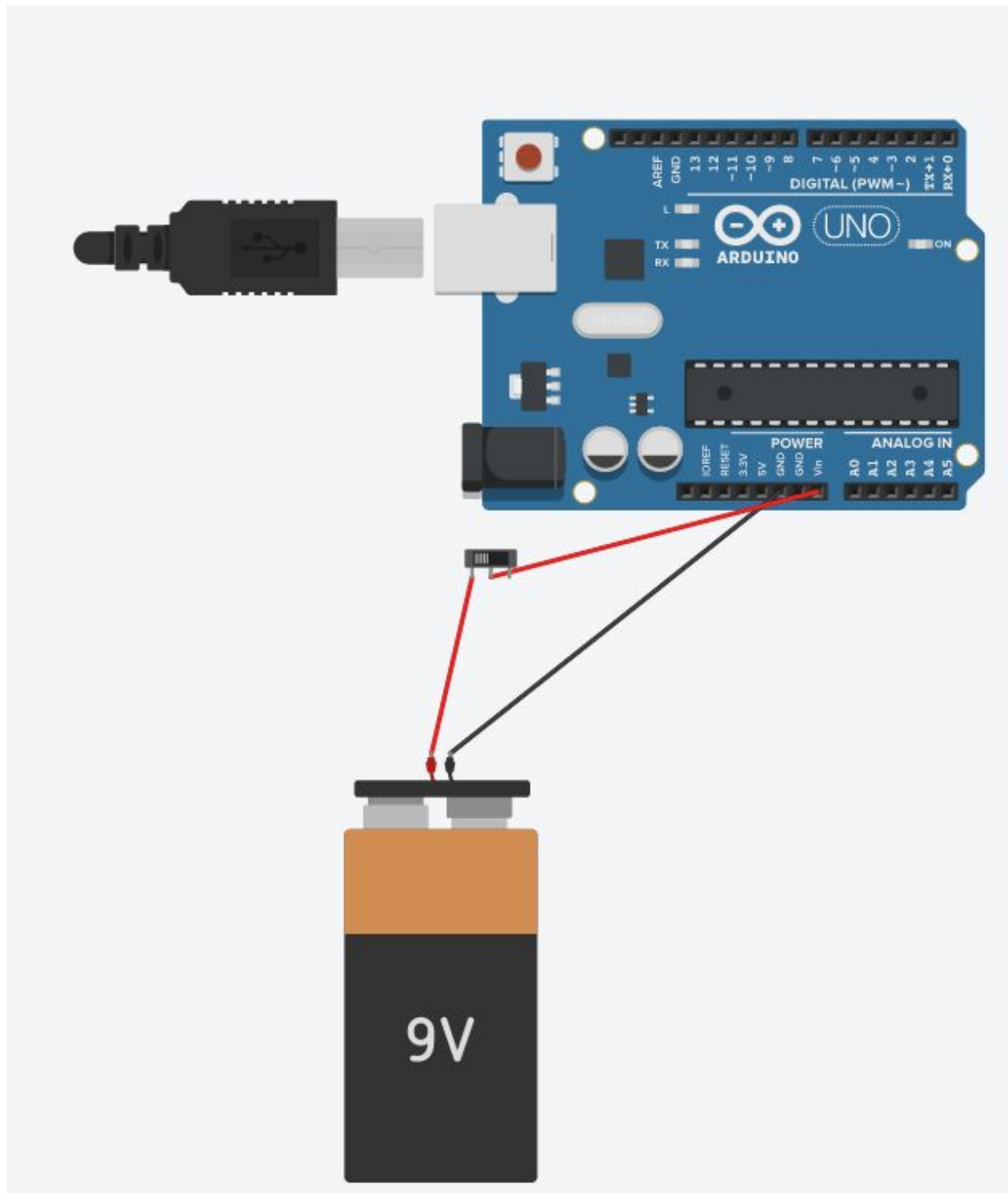
Pin-out de Cansat con Arduino Mega 2560 Pro Mini y esquema con fritzing





Alimentación de Arduino Mega

La alimentación de cansat es a través de una pila de 9V conectada a microinterruptor de conmutador doble, como se muestra en la figura:



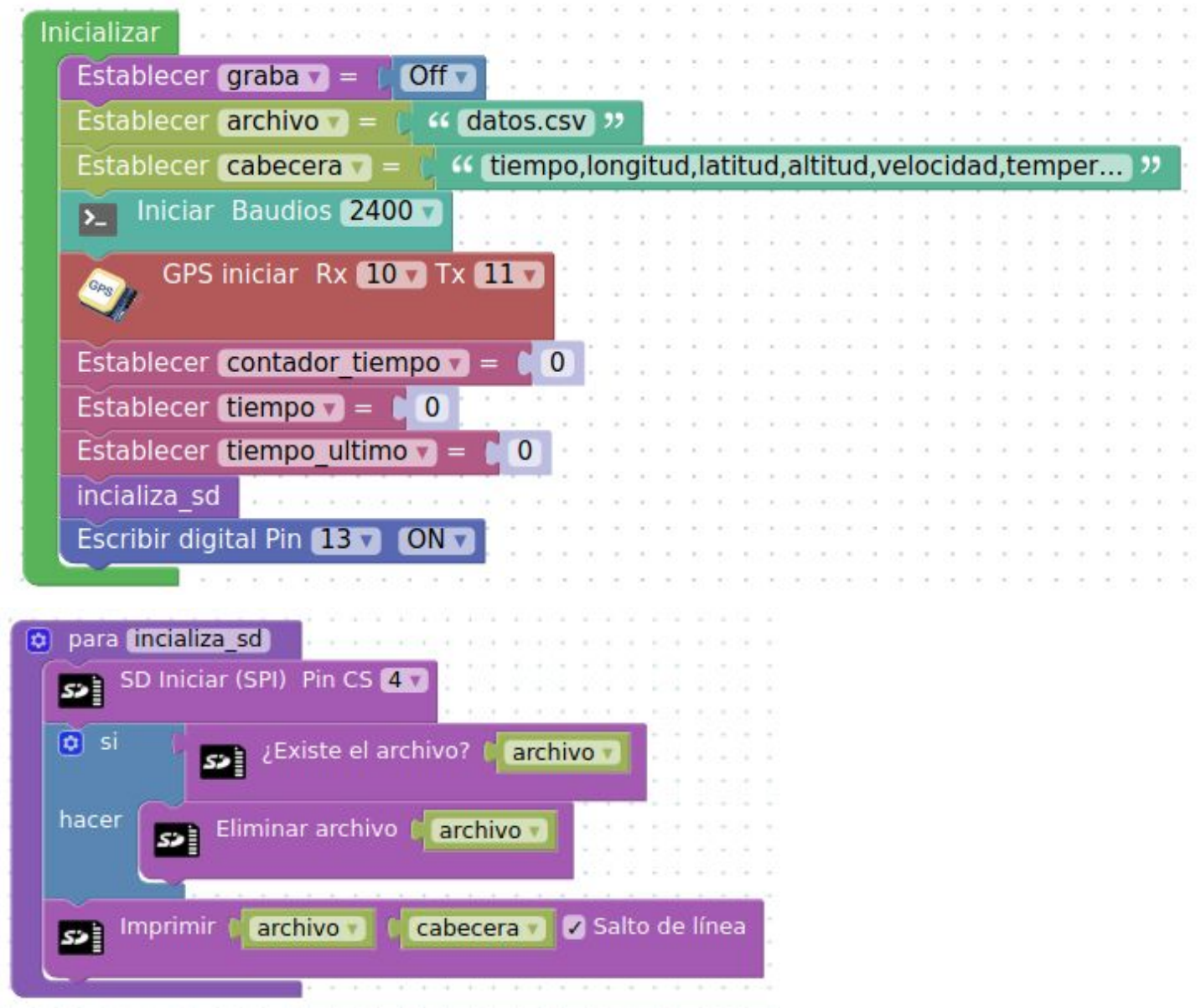
Programa en Arduinoblocks

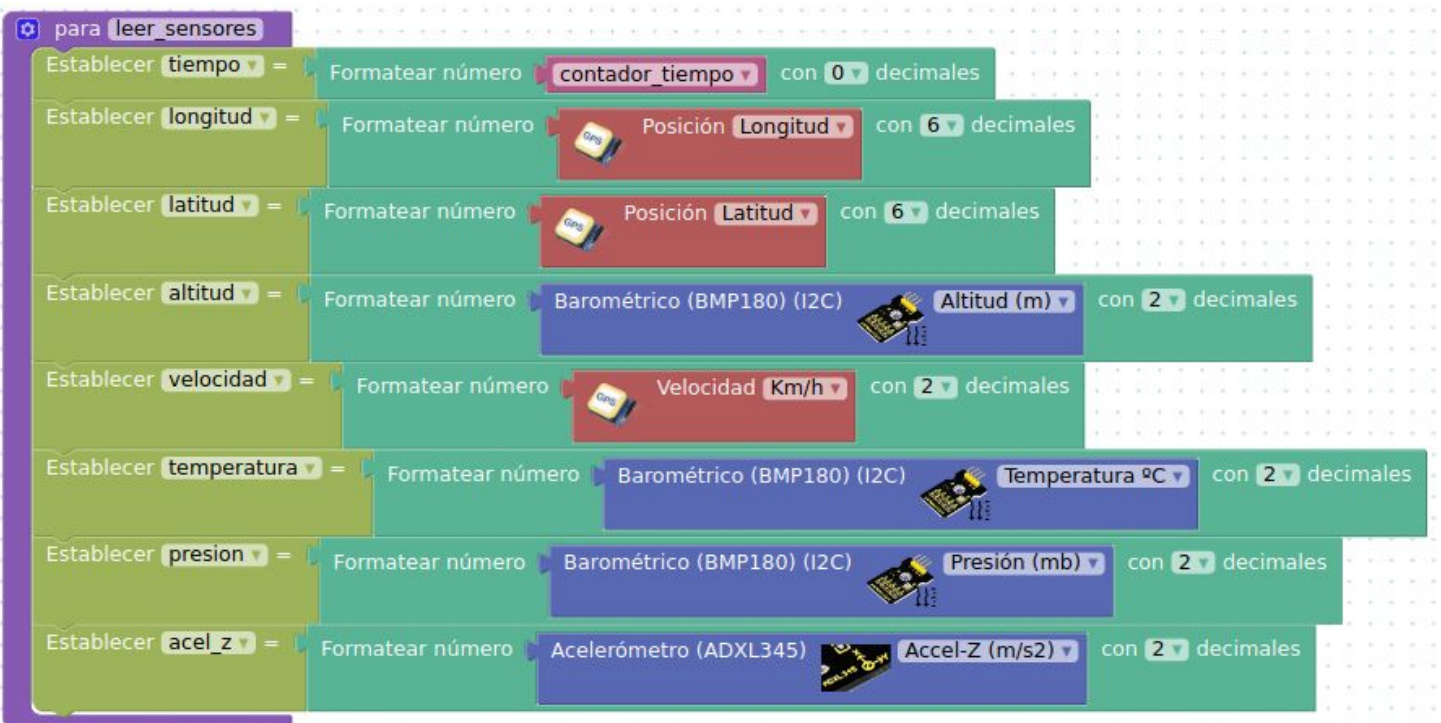
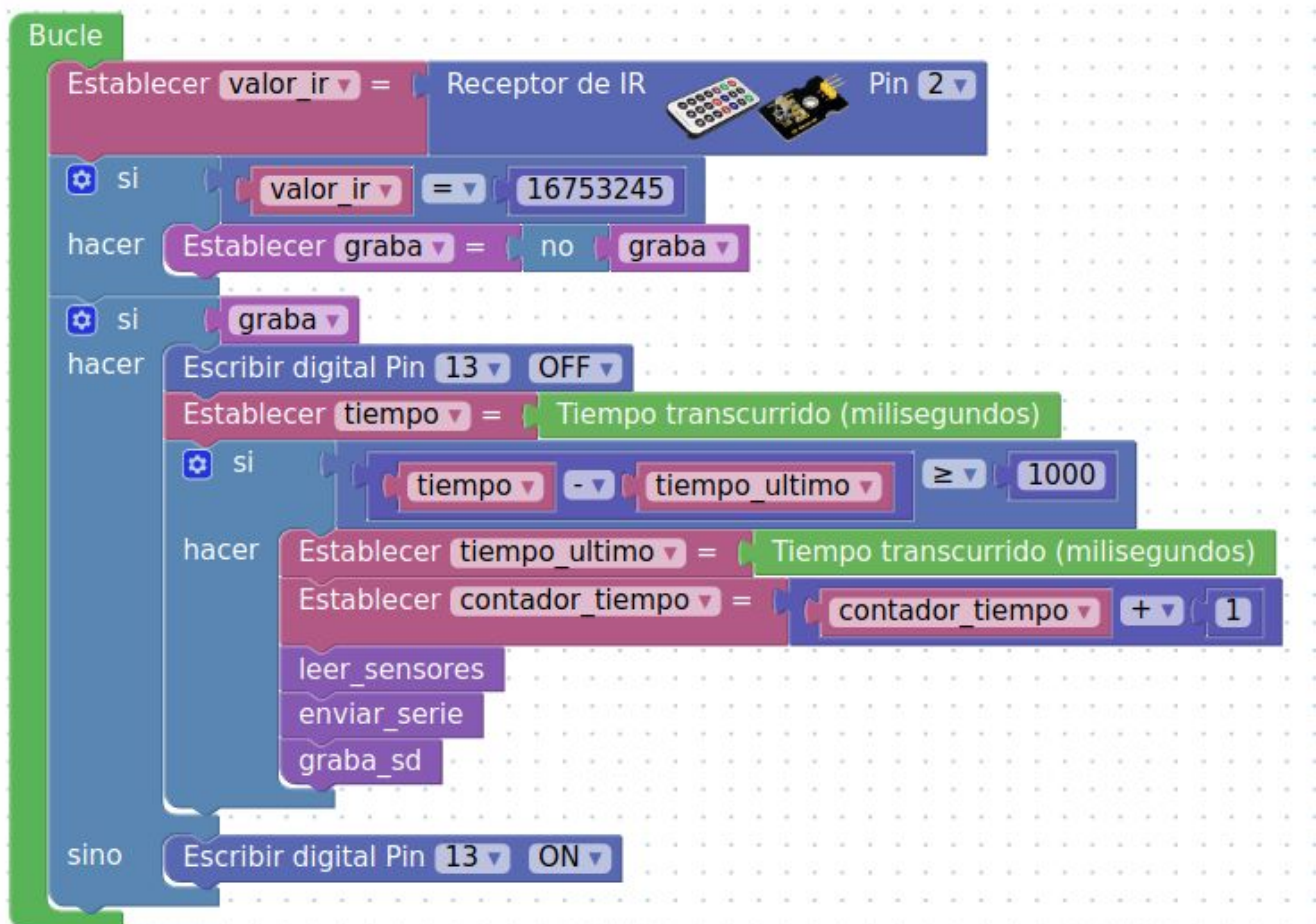
Proyecto compartido para arduino mega (V1): <http://www.arduinoblocks.com/web/project/223835>

Dicho programa inicializa el la función setup (inicializar) parámetros como variables, la velocidad de transmisión, el gps, la inicialización de la SD y enciende el led asociado al pin 13.

En el bucle principal leemos el valor del sensor de infrarrojo, si es un determinado valor cambia el valor de la variable booleana graba. Si graba es cierto apaga el led del pin 13 y deja el sistema preparado para cada segundo leer los sensores, grabarlos en la microsd y enviarlos por puerto serie por la radiofrecuencia.

Si graba es falso enciende el led del pin13.







Programa de estación base (PC) en processing

El siguiente programa (descargar [aquí](#)) se ejecuta en el PC, establece comunicación con el puerto serie (APC200 receptor pinchado en USB), filtra los posibles datos erróneos que provengan de cansat usando tres comprobaciones: verifica que la cantidad de campos que hay en cada registro sean 10, verifica que el primer y el último campo tengan un valor determinado, en nuestro caso primer campo “soto” y el último campo “fin”. Más tarde con los datos ya filtrados los guarda en un fichero y los muestra en consola y ventana.

```
import processing.serial.*;
```

```
Serial puerto;//establece variable puerto tipo Serial
```

```
PrintWriter fichero;//establece variable fichero tipo PrintWriter (fichero)
```

```
int contador=0;//variable para contar el nº de grabaciones realizadas
```

```
void setup() {
```

```
  //println(Serial.list());
```

```
  size (400, 335);
```

```
  background(0);
```



```

    puerto = new Serial(this, Serial.list()[32], 9600);//establece la variable puerto asignando el receptor APC200 y
una velocidad de 9600 baudios
    fichero=createWriter("datos.csv");//crea el fichero datos.csv
    datosPantalla("Sin datos", "Sin datos", "Sin datos", "Sin datos", "Sin datos", "Sin datos", "Sin datos", "Sin datos", "Sin datos");
}

```

```

void draw() {

```

```

    datosPantalla("Sin datos", "Sin datos", "Sin datos", "Sin datos", "Sin datos", "Sin datos", "Sin datos", "Sin datos", "Sin datos");

```

```

    while (puerto.available()>0) {//mientras haya datos en el puerto serie
        String buffer=puerto.readStringUntil(10);//leemos los datos del puerto serie hasta el salto de línea (10 en ascii), y los asignamos a buffer

```

```

        if (buffer!=null) {//si buffer no está vacío
            String[] listaBuffer = split(buffer, ',');//extrae en un array los elementos separados por coma del buffer
            int longitudBuffer=buffer.length();//leemos el nº de caracteres del buffer
            int numeroCampos=listaBuffer.length;//leemos el nº de campos que están en el array, deben ser 10
            //println (buffer);
            println ("el nº de campos es:" + numeroCampos + " el nº de caracteres es:" + longitudBuffer);//imprimimos en consola el nº de campos y nº de caracteres recibidos

```

```

            if (numeroCampos==10 && listaBuffer[0].equals("soto")==true && listaBuffer[9].equals("fin\r\n")==true)
{//grabamos si hay 10 campos y los campos de control de inicio y fin son correctos
                contador++;
                fichero.print(buffer);//colocamos en el fichero el buffer
                fichero.flush();//hacemos la grabación efectiva y se cierra el fichero
                println(buffer);//imprimimos por consola lo grabado
                datosPantalla(listaBuffer[1], listaBuffer[2], listaBuffer[3], listaBuffer[4], listaBuffer[5], listaBuffer[6], listaBuffer[7], listaBuffer[8]);//llamamos a la función que nos pone en pantalla los datos recogidos
            } else {
                datosPantalla("Sin datos", "Sin datos", "Sin datos", "Sin datos", "Sin datos", "Sin datos", "Sin datos", "Sin datos", "Sin datos");
            }
        }
    }
}

```

```

void datosPantalla(String tiempo, String longitud, String latitud, String altitud, String velocidad, String temperatura, String presion, String acelz) {
    background(0);
    textSize (24);
    text ("Datos SotoSat", 120, 35);
    textSize(16);
    text ("Tiempo(s):"+ tiempo, 50, 75);

```

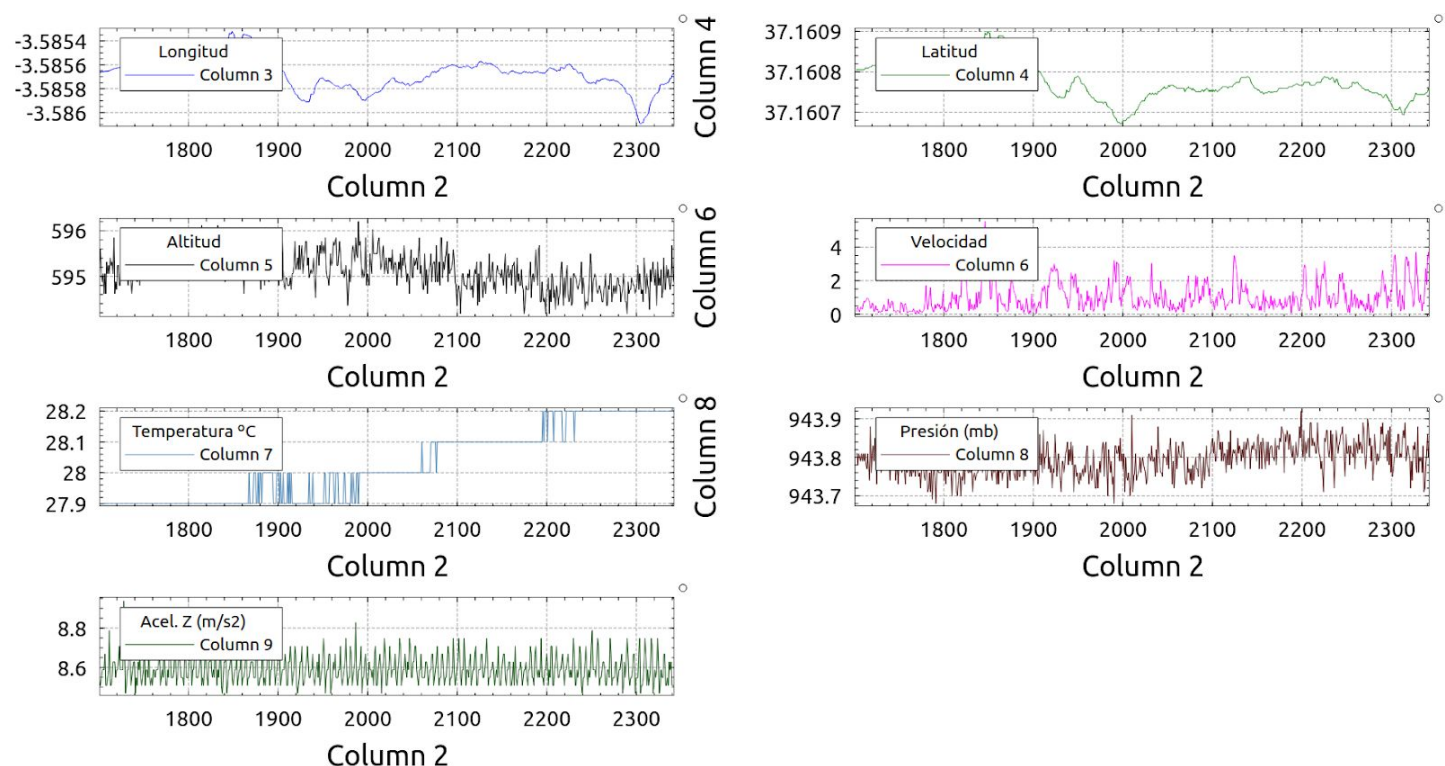
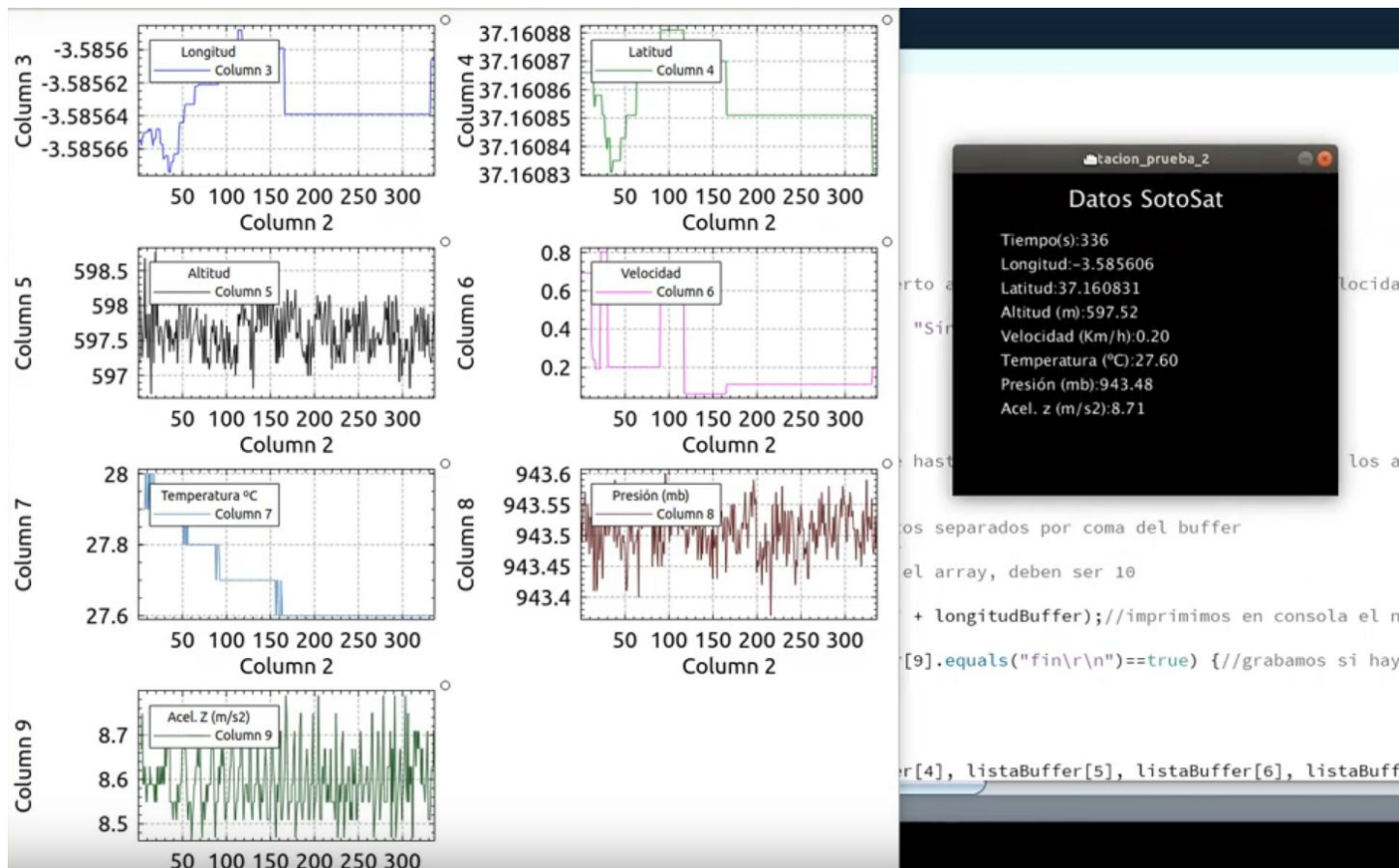
```
text ("Longitud:" + longitud, 50, 100);
text ("Latitud:" + latitud, 50, 125);
text ("Altitud (m):" + altitud, 50, 150);
//text ("Rumbo:" + rumbo, 50, 175);
text ("Velocidad (Km/h):" + velocidad, 50, 175);
text ("Temperatura (°C):" + temperatura, 50, 200);
text ("Presión (mb):" + presion, 50, 225);
text ("Acel. z (m/s2):" + acelz, 50, 250);
//text ("Día:" + dia, 50, 100);
//text ("Hora:" + hora, 50, 125);
}
```

Misión Secundaria

Telemetría avanzada, tratamiento y graficado de datos

La misión secundaria que se aborda con este proyecto pretende ser de bajo coste y exportable al mayor número de situaciones, para ello se han escogido tres vías de actuación:

- Realizar telemetría avanzada usando sensores de bajo costo: posicionamiento gps, aceleración, velocidad, etc.
- Filtrado de datos en la estación base: se trata de un programa de processing en el PC de la estación base que recibe los datos en crudo de la emisión de cansat y los filtra detectando registros (tramas) incorrectas, para ello realiza tres comprobaciones que se deben cumplir a la vez, que el número de campos recibidos sea el correcto, que el campo de control de inicio se cumpla y que el campo de control final se cumpla. Con este filtrado conseguimos que los datos que no son correctos por fallos de transmisión (perdida de comunicación por falta de alcance en la emisión-recepción, interferencias) no lleguen al graficador en tiempo real que provocarían un mal graficado.
- Realizar el graficado con KST de los datos previamente filtrados presentando una gráfica por cada dato de telemetría recibida, siendo el eje X de todas las gráficas el tiempo de ejecución del lanzamiento.



Enlaces de interés:

- Arduino y processing: <https://youtu.be/gETASNUtwps>
- Processing y puerto serie: <https://polaridad.es/processing-serie-comunicaciones-uart/>

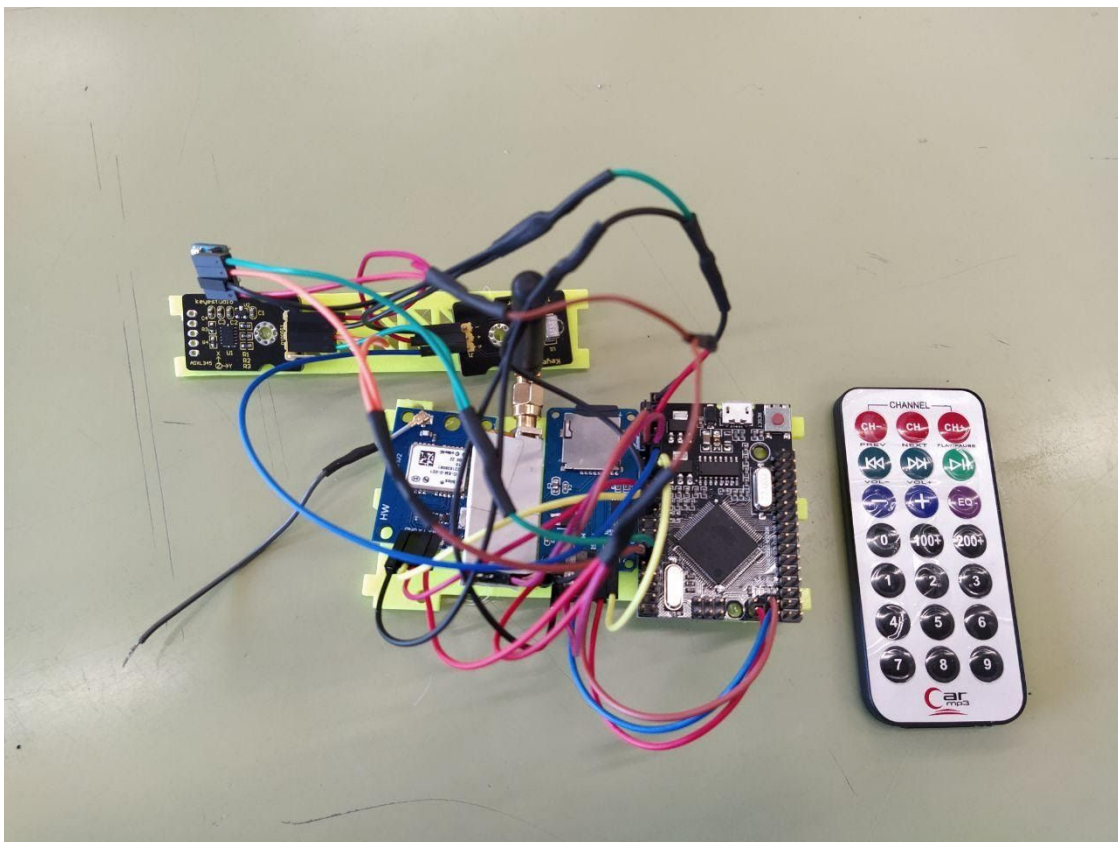
Proyecto científico

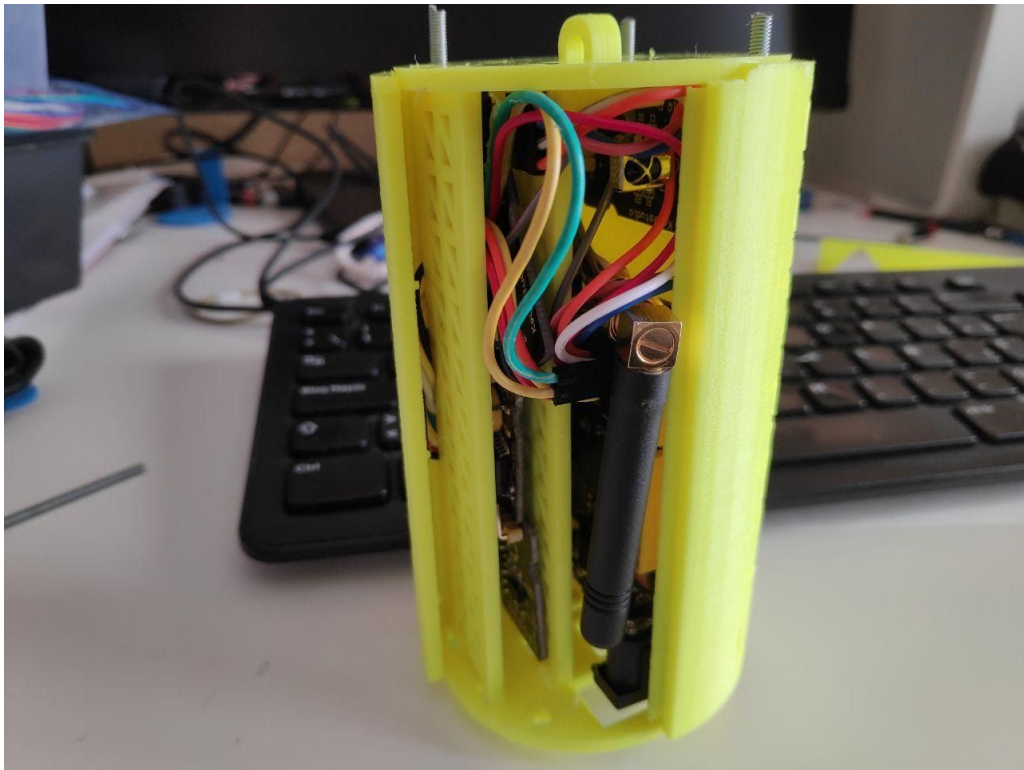
Objetivos

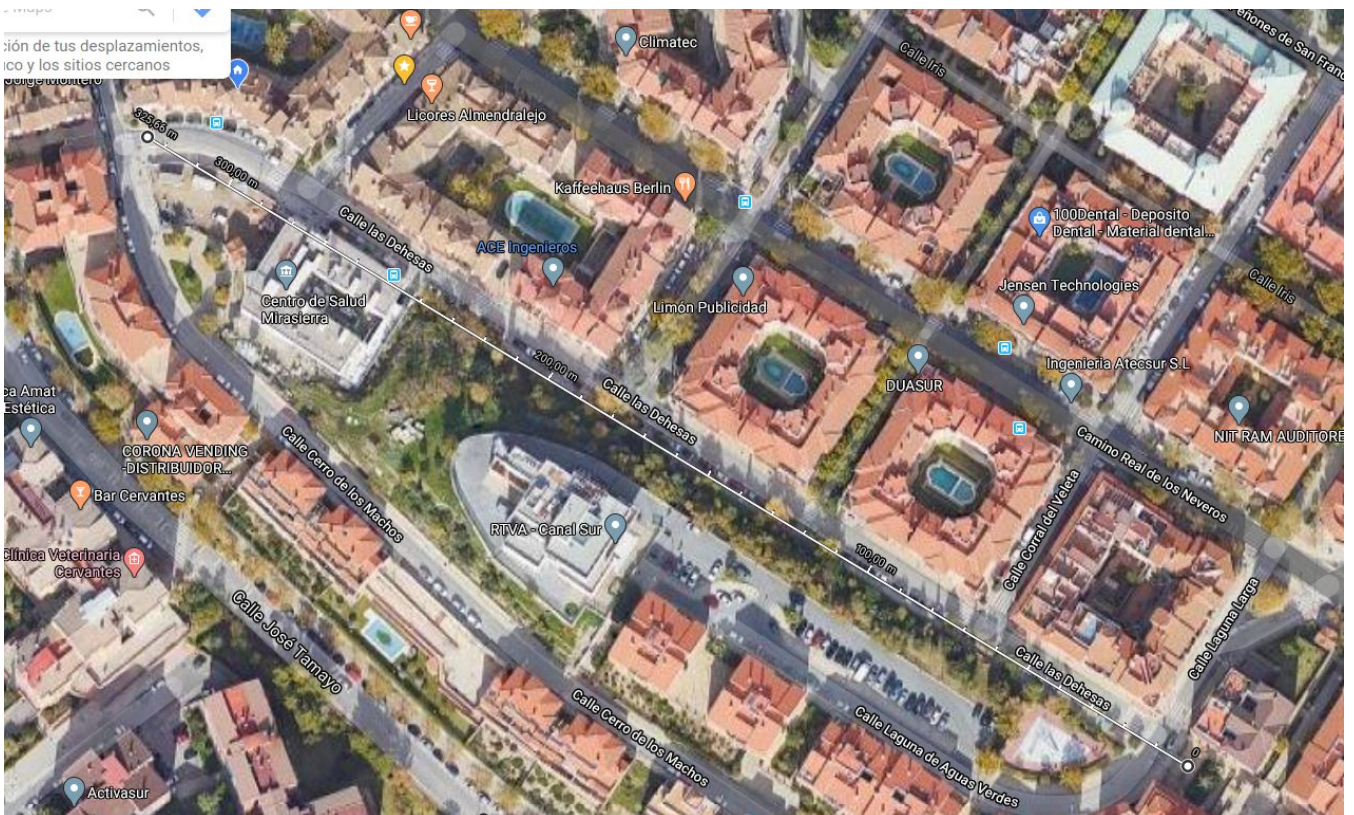
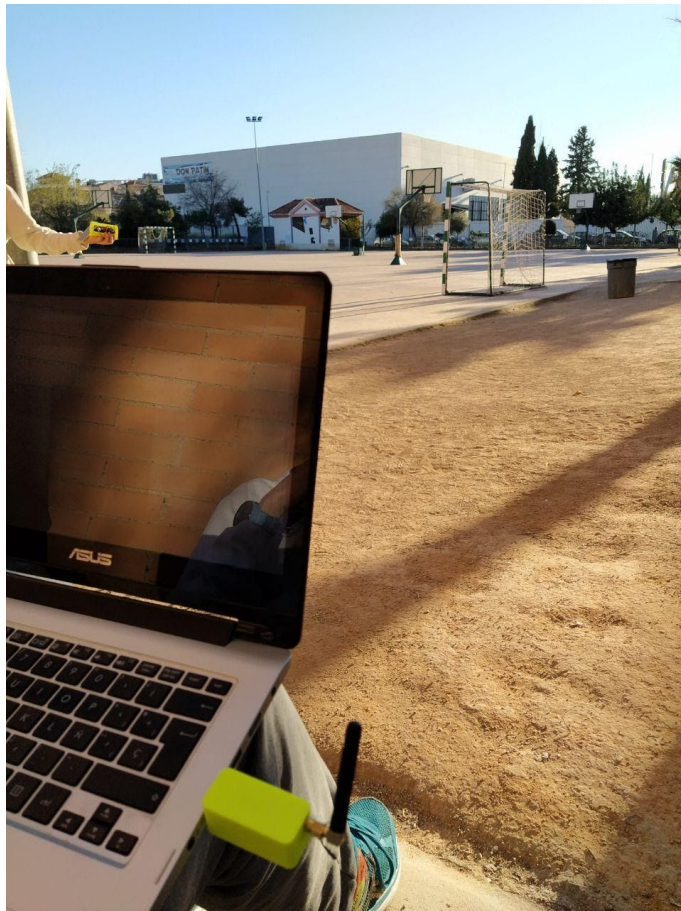
- Graficar e interpretar variaciones de altura y presión: comprobar que a mayor altura menor presión y viceversa.
- Graficar e interpretar variaciones de altura y temperatura: comprobar que a mayor altura menor temperatura.
- Graficar e interpretar la aceleración en Z: comprobamos el diseño del paracaidas, que ha sido calculado para que el movimiento de descenso se M.R.U (movimiento uniforme a velocidad constante) por tanto la aceleración en Z debe ser en torno a 0 m/s².

Imágenes

[Enlaces a Imágenes](#)







Vídeos

- Prueba de graficado de telemetría con filtrado de datos: <https://youtu.be/t9S8PFPLkVA>
- Pruebas de paracaidas:
 - <https://youtu.be/uHOaMot1JTM>
 - <https://youtu.be/Mxxc1IZZjkk>

Webs de interés

- <https://www.youtube.com/playlist?list=PLwagIiYuh5RYOJNwSjlyMwnosqbO2QryI>