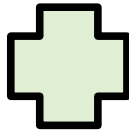


MICRO:CANSAT

Electrónica de CANSAT con MICRO:BIT



Por: *Pedro Ruiz Fernández*
Versión 10/07/2020

Licencia



Introducción

Se trata de implementar la electrónica de un minisatélite cansat con la placa micro:bit en python, aunque también se puede realizar en makecode.

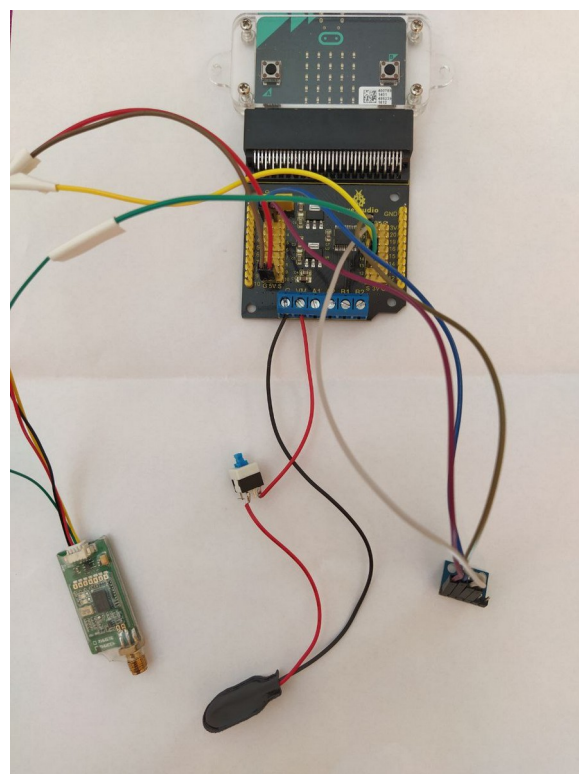
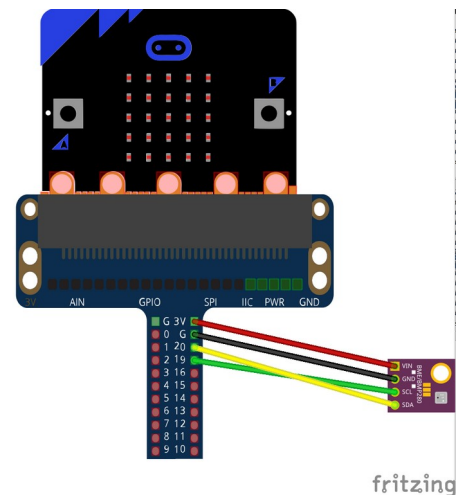
Materiales necesarios

Item	Cantidad	Descripción
1	1	Placa micro:bit
2	1	Shield para obtener pinout completo de micro:bit
3	1	Sensor bmp-280
4	1	Emisor y receptor de radiofrecuencia 3dr robotics a 433 Mhz
5	4	Cables dupont hembra-hembra

Conexionado

Se conecta micro:bit al shield y en el shield se conecta el sensor bmp280 a los siguientes pines del shield, es un conexión tipo I2C.

Pin Shield	Pin bmp280
gnd	gnd
5V	vcc
19 (pin scl del shield)	scl
20 (pin sda del shield)	sda
Pin Shield	Pin 3dr robotics
14	Tx
15	Rx
Gnd	Gnd
5V	5V



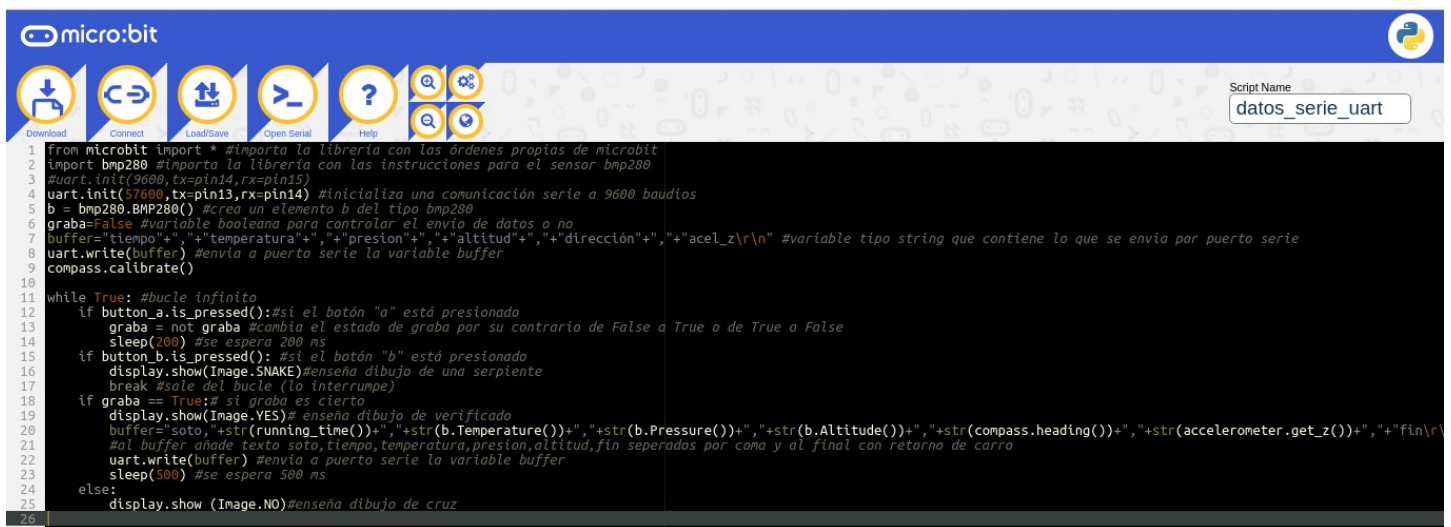
Programación de micro:bit (python)

Vamos a necesitar el [programa principal](#) y una librería para tener las órdenes de lectura del bmp280, que es un sensor que nos da temperatura(°C), presión (Pa) y altitud (m).

Como entorno de programación he utilizado <https://python.microbit.org/v/2.0>, en el mismo he puesto el siguiente código principal:

Funcionamiento del [programa principal](#):

Al pulsar el botón "a" cambia el estado de la variable booleana "graba" por su contrario, y cuando es true muestra en la matriz de leds un símbolo de verificado y envía por puerto serie una cadena de texto compuesta por "soto, tiempo, temperatura, presión, altitud, dirección de brújula, aceleración en z, fin con retorno de carro". El primer y último campo son para realizar el control de los datos enviados. Si la variable "graba" vale False se muestra un símbolo de una cruz en la matriz de leds. Si pulsamos el botón "b" se muestra en la matriz de leds el símbolo de una serpiente y sale del bucle principal interrumpiendo el programa.



```
1 from microbit import * #importa la librería con las órdenes propias de microbit
2 import bmp280 #importa la librería con las instrucciones para el sensor bmp280
3 #uart.init(9600,tx=pin14,rx=pin15)
4 uart.init(57600,tx=pin13,rx=pin14) #inicializa una comunicación serie a 9600 baudios
5 b = bmp280.BMP280() #crea un elemento b del tipo bmp280
6 graba=False #variable booleana para controlar el envío de datos o no
7 buffer="tiempo"+"temperatura"+"presion"+"altitud"+"dirección"+"acel_z\r\n" #variable tipo string que contiene lo que se envia por puerto serie
8 uart.write(buffer) #envia a puerto serie la variable buffer
9 compass.calibrate()
10
11 while True: #bucle infinito
12     if button_a.is_pressed():#si el botón "a" está presionado
13         graba = not graba #cambia el estado de graba por su contrario de False a True o de True a False
14         sleep(200) #se espera 200 ms
15     if button_b.is_pressed(): #si el botón "b" está presionado
16         display.show(Image.SNAKE)#enseña dibujo de una serpiente
17         break #sale del bucle (lo interrumpe)
18     if graba == True:# si graba es cierto
19         display.show(Image.YES)# enseña dibujo de verificado
20         buffer="soto,"+str(running_time())+", "+str(b.Temperature())+", "+str(b.Pressure())+", "+str(b.Altitude())+", "+str(compass.heading())+", "+str(accelerometer.get_z())+", "+str(fin\r\n)
21         #al buffer añade texto soto, tiempo, temperatura, presion, altitud, fin separados por coma y al final con retorno de carro
22         uart.write(buffer) #envia a puerto serie la variable buffer
23         sleep(500) #se espera 500 ms
24     else:
25         display.show(Image.NO)#enseña dibujo de cruz
26
```

```
from microbit import * #importa la librería con las órdenes propias de microbit
import bmp280 #importa la librería con las instrucciones para el sensor bmp280
#uart.init(9600,tx=pin14,rx=pin15)
uart.init(57600,tx=pin13,rx=pin14) #inicializa una comunicación serie a 9600 baudios
b = bmp280.BMP280() #crea un elemento b del tipo bmp280
graba=False #variable booleana para controlar el envío de datos o no
buffer="tiempo"+"temperatura"+"presion"+"altitud"+"dirección"+"acel_z\r\n" #variable tipo
string que contiene lo que se envia por puerto serie
uart.write(buffer) #envia a puerto serie la variable buffer
compass.calibrate()

while True: #bucle infinito
    if button_a.is_pressed():#si el botón "a" está presionado
        graba = not graba #cambia el estado de graba por su contrario de False a True o de True a False
        sleep(200) #se espera 200 ms
    if button_b.is_pressed(): #si el botón "b" está presionado
        display.show(Image.SNAKE)#enseña dibujo de una serpiente
        break #sale del bucle (lo interrumpe)
    if graba == True:# si graba es cierto
        display.show(Image.YES)# enseña dibujo de verificado
        buffer="soto,"+str(running_time())+", "+str(b.Temperature())+", "+str(b.Pressure())+", "+str(b.Altitude())
        +", "+str(compass.heading())+", "+str(accelerometer.get_z())+", "+str(fin\r\n)
```


```

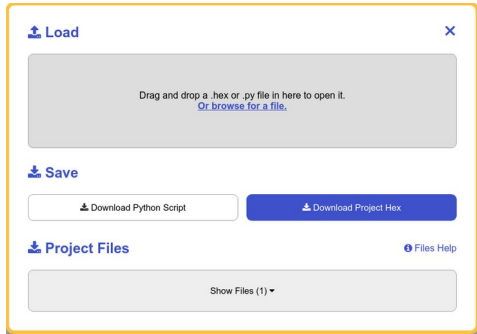
#al buffer añade texto soto,tiempo,temperatura,presion,altitud,fin seperados por coma y al final con
retorno de carro
uart.write(buffer) #envia a puerto serie la variable buffer
sleep(500) #se espera 500 ms
else:
display.show (Image.NO)#enseña dibujo de cruz

```

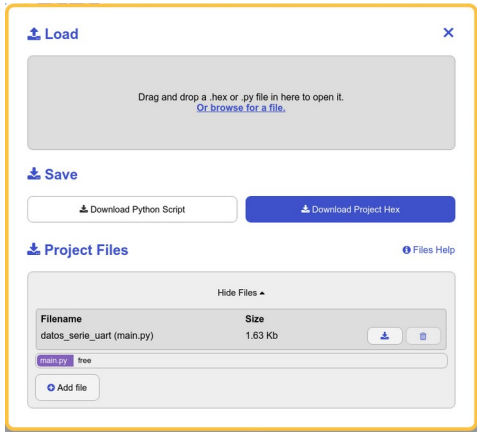
¿Como introducir la librería del bmp280 y cargar todo el programa junto con la librería en microbit?.



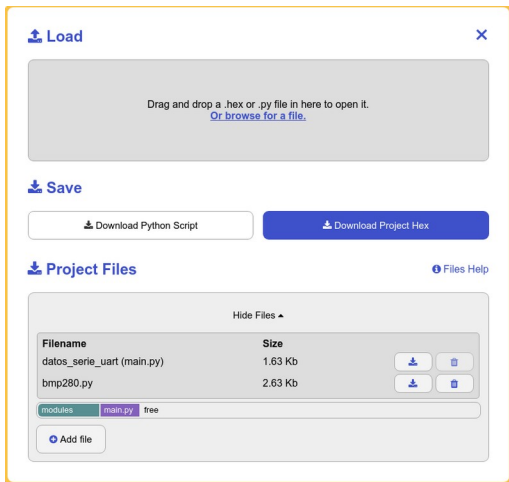
Pulsamos el botón  e incorporamos la librería [bmp280.py](#) previamente descargada en tu PC a los ficheros del programa, para ello pulsamos en la zona “Show Files”




Apareciendo los ficheros que tengo en el programa, por ahora sólo el principal:



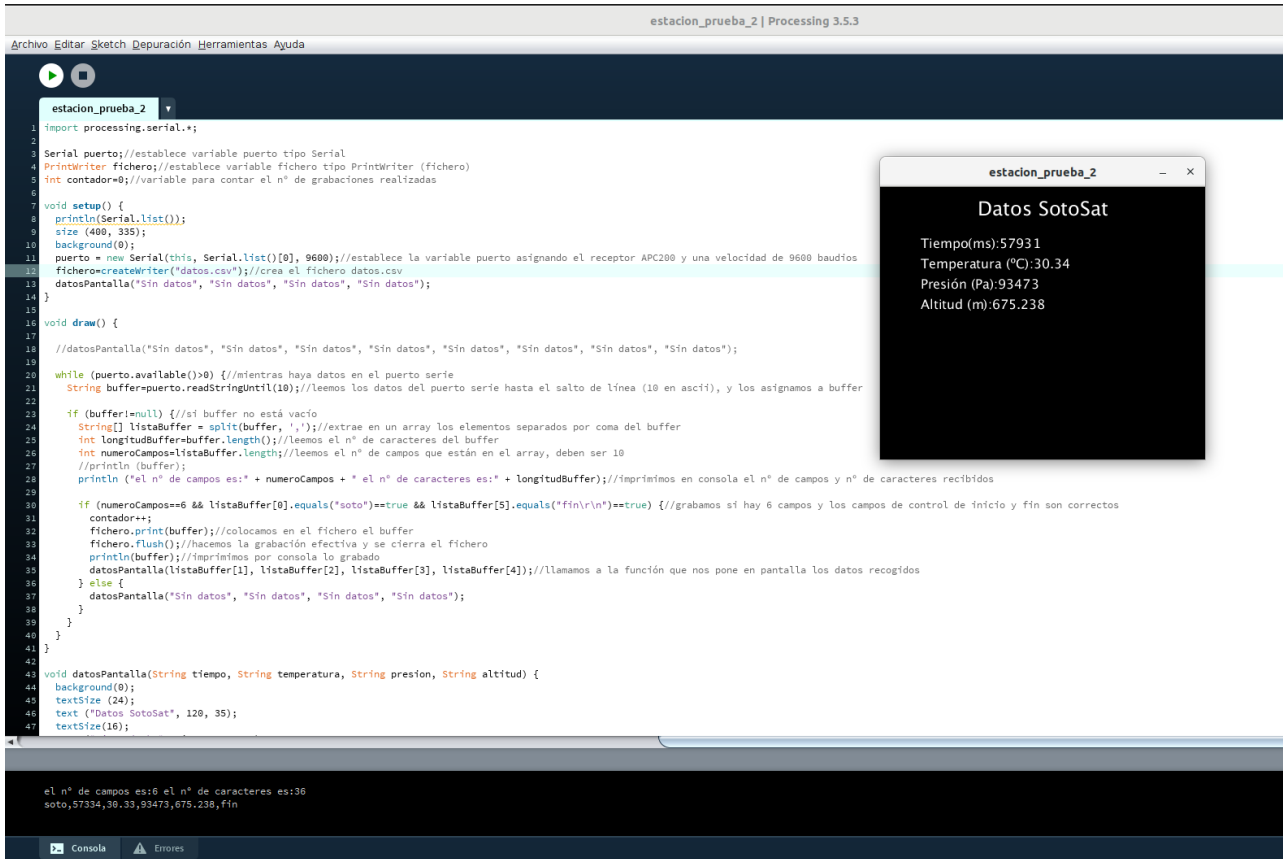
Le damos al botón “Add File”, te sale una ventana para elegir y añadir el fichero [bmp280.py](#) previamente descargado a nuestro ordenador.



Ahora sólo queda descargar el fichero hexadecimal del proyecto en , te aparece una ventana para descargar el fichero en una carpeta en nuestro caso en la de microbit.

Programación de la estación base con Processing

En la estación base tendremos instalado el editor [Processing](#), en el tenemos abierto y ejecutándose un [programa](#) que recoge los datos enviados por puerto serie y comprueba que se reciben correctos, para ello hace tres comprobaciones con la información: que esté integrada por 6 campos, que el primer campo sea el texto "soto" y que el último sea el texto "fin". Con la información que cumple estos requisitos, la guardamos en tiempo real un fichero llamado "datos.csv", el fichero se encuentra en la misma carpeta del programa de estación base de processing, y además se representa en pantalla. Posteriormente este fichero se puede abrir y procesar en una hoja de cálculo o bien graficar en tiempo real con aplicaciones como "[kst](#)".



```
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], 57600);//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");
}
```

```
void draw() {

  //datosPantalla("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==8 && listaBuffer[0].equals("soto")==true && listaBuffer[7].equals("fin\r\n")==true)  
  {//grabamos si hay 6 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]);//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");  
  }  
}
```

```
void datosPantalla(String tiempo, String temperatura, String presion, String altitud, String direccion, String  
acelz) {  
  background(0);  
  textSize (24);  
  text ("Datos SotoSat", 120, 35);  
  textSize(16);  
  text ("Tiempo(ms):"+ tiempo, 50, 75);  
  text ("Temperatura (°C):"+temperatura, 50, 100);  
  text ("Presión (Pa):"+presion, 50, 125);  
  text ("Altitud (m):"+altitud, 50, 150);  
  text ("Dirección (0°-360°):"+direccion, 50, 175);  
  text ("Acel z (mg):"+acelz, 50, 200);  
}
```