



ThingSpeak

IoT Analytics  
with MATLAB

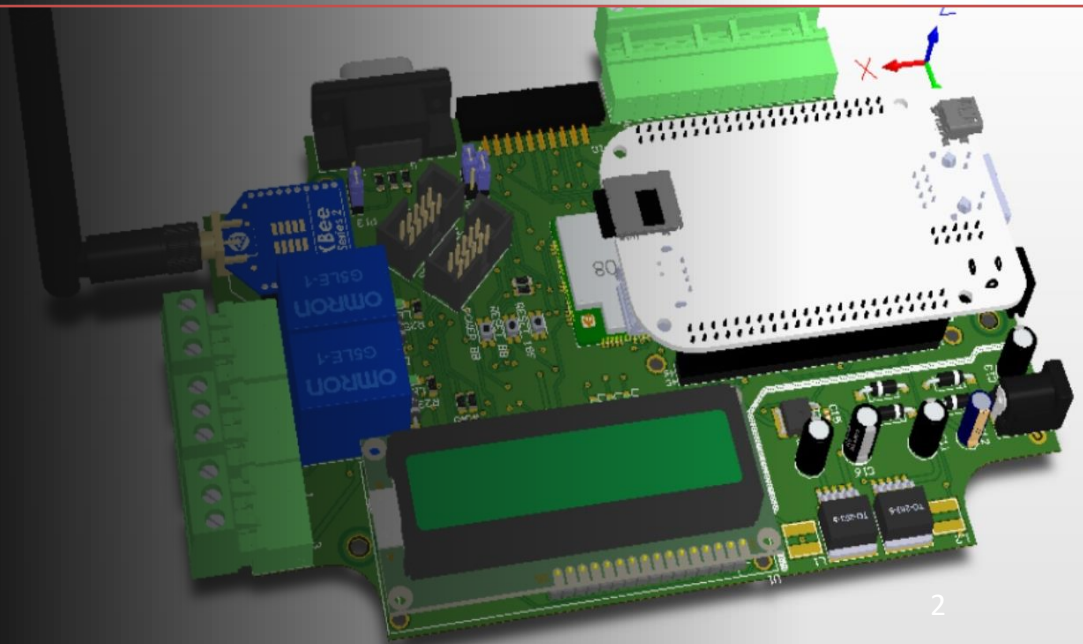


# SISTEMAS EMBEBIDOS

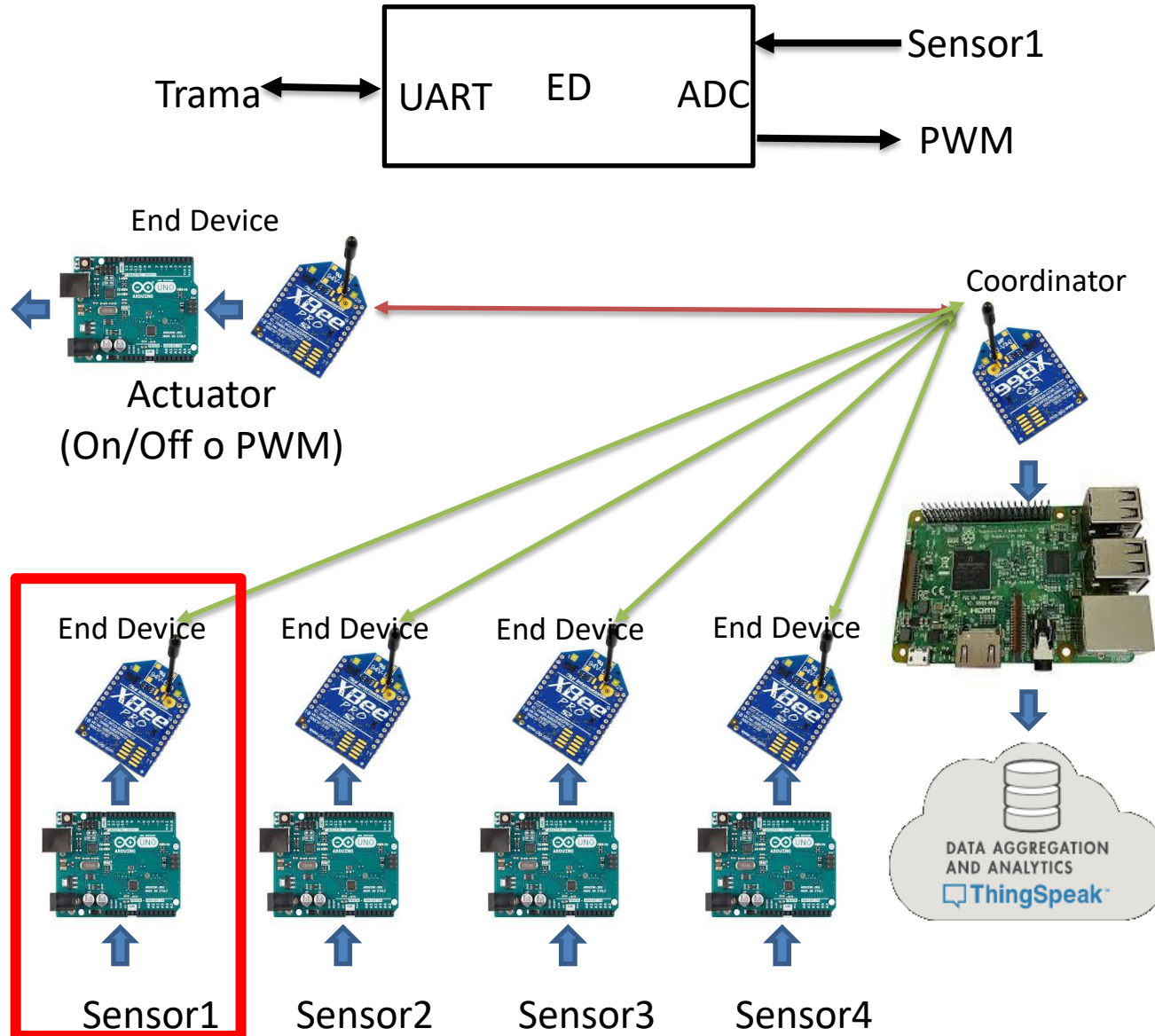
## End Device & Coordinator



# SISTEMAS EMBEBIDOS END DEVICE (Free RTOS)



# End Device



Del anterior sistema ciberfísico, para este deber usted solo debe implementar el End Device del sensor 1; para ello, deberá utilizando conceptos de maquina secuencial, interrupción por desborde del Timer 0, conversión analógico a digital con el bloque ADC y comunicación serial asíncrona con el bloque UART. La trama de comunicación que deberá utilizar para recibir la solicitud de lectura de datos y para enviar los valores almacenados, es la siguiente tabla:

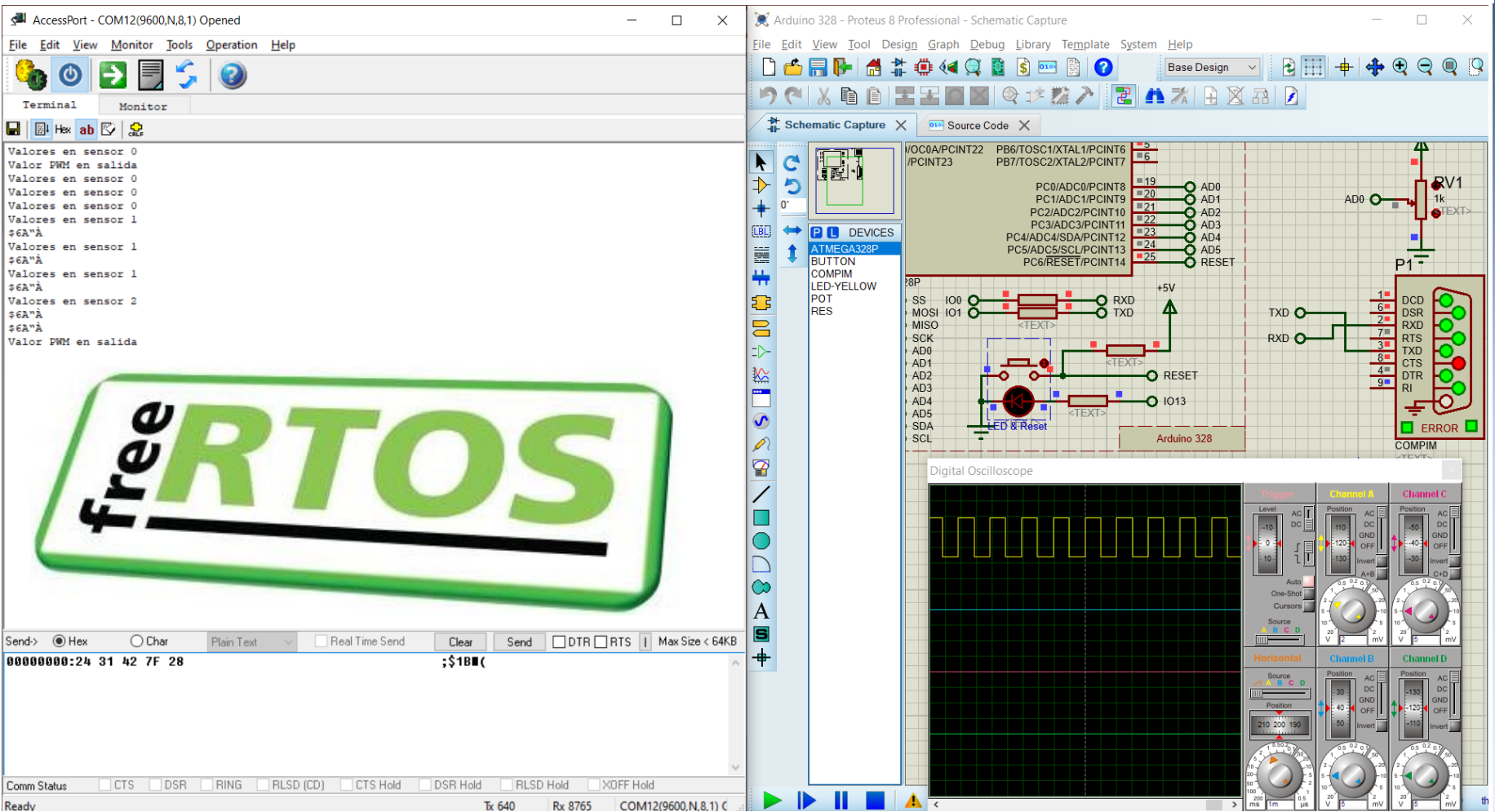
Start byte	ID byte	Byte Task	Byte Data	Byte Checksum
0x24 (\$)	0x31 (1) – ED 0x36 (6) – C	0x41 (A) – Read 0x42(B) - PWM	0x00 . 0x30 (0) . 0xFF ()	XOR (Start byte, ID byte, byte Task, byte Data)

- El equipo Coordinador (C) deberá enviar la siguiente trama de solicitud de datos al End Device (ED) con Byte ID de 0x31: 0x24 0x31 0x41 0x00 0x54
- El equipo End Device (ED) deberá enviar la siguiente trama, en respuesta a la solicitud que realizó el Coordinado con Byte ID de 0x36: 0x24 0x36 0x41 0xYY 0xZZ
- Note que el byte 0xYY representa uno de los datos enviados, es decir, si el ED necesita enviar 100 datos de lecturas, entonces deberá enviar 100 tramas similares.
- Además, note que el byte 0xZZ deberá ser calculado en cada trama antes de ser enviado. Como ejemplo, el valor 0x54 = 0x24  $\oplus$  0x31  $\oplus$  0x41  $\oplus$  0x00

El End Device deberá realizar las siguientes tareas:

1. Hacer un muestreo de la señal analógica del sensor 1 cada 1 segundo.
  1. Almacenar cada 1 segundo el valor analógico del sensor 1, de forma estática en memoria SRAM durante 60 segundos.
2. Al terminar los 60 segundos de adquisición (equivalente al almacenamiento de forma estática 60 valores en SRAM), se deberá calcular el valor promedio de esos valores.
  1. El valor promedio calculado deberá ser almacenado de forma dinámica en memoria SRAM, hasta que el equipo Coordinador haga la solicitud de los valores almacenados de forma dinámica.
3. En todo momento, el End Device deberá estar pendiente de toda trama de comunicación serial UART que éste reciba.
  1. Usted deberá asumir que el equipo Coordinador puede hacer una solicitud de los datos almacenados en el End Device, en intervalos de tiempo aleatorios que pueden variar desde unos 15min hasta 4 horas (240 min). Esto por motivos de conectividad a internet, números de ED en la red, mantenimiento, etc.
  2. En caso de que reciba una solicitud de los valores almacenados, el End Device deberá responder uno a uno los valores promedio almacenados, haciendo uso de la trama de comunicación indicada en la tabla.
  3. Recordar que los valores promedios fueron almacenados de forma dinámica en SRAM, por tanto, se deberá liberar el puntero luego de completar el envío de todos los valores.

# Proteus



01101010011001010110000101101110



## Codigo FreeRTOS End Device en Proteus



## Trama Serial

```
// Sensor AD-0
//24 31 41 00 54
// Actuador PWM-11
//100% -> 24 31 42 FF A8
//50%-> 24 31 42 7F 28
//0%-> 24 31 42 00 57
```

```
1 #include <Arduino_FreeRTOS.h>
2
3 TaskHandle_t TaskLed_Handler;
4
5 //variables globales
6 int led = 13;
7 int led_pwm=11;
8 unsigned long ADCsensor = 0;
9 int Datos1[240]={}; //4 horas de datos
10 int index=0;
11 int estado=0;
12 int RXTrama[5];
13 int TramaS[5] = {0x24, 0x31, 0x41, 0x00, 0x54}; //sensor
14 int TramaA[5] = {0x24, 0x31, 0x42, 0x00, 0x57}; //actuador
15 int IdCoordinador = 0x36; //6'
16 // Sensor AD-0
17 //24 31 41 00 54
18 // Actuador PWM-11
19 //24 31 42 FF A8
20 //24 31 42 7F 28
21 //24 31 42 00 57
22
23 // End Device
24 void TaskBlink( void *pvParameters );
25 void TaskRxTrama( void *pvParameters );
26 void TaskAnalogRead( void *pvParameters ); // Sensor
27
```

## Codigo FreeRTOS End Device en Proteus



Google Drive

```

29 void setup() {
30     pinMode(A0, INPUT);
31     pinMode(led, OUTPUT);
32     pinMode(led_pwm, OUTPUT);
33     Serial.begin(9600);
34     xTaskCreate(
35         TaskBlink
36         , "Blink" // A name just for humans
37         , 128 // Stack size
38         , NULL
39         , 0 // priority
40         , &TaskLed_Handler); // Variable que apunta al task (opcional)
41
42     xTaskCreate(
43         TaskRxTrama
44         , "Blink" // A name just for humans
45         , 128 // Stack size
46         , NULL
47         , 1 // priority
48         , NULL );
49
50     xTaskCreate(
51         TaskAnalogRead
52         , "AnalogRead"
53         , 128 // This stack size can be checked & adjusted by reading Highwater
54         , NULL
55         , 3 // priority
56         , NULL );
57
58     vTaskSuspend(TaskLed_Handler);
59 }

```

```

61 void loop()
62 {
63 }
64 /*----- Tasks -----*/
65 void TaskBlink(void *pvParameters) // This is a task.
66 {
67     for (;;) {
68         digitalWrite(led, HIGH);
69         vTaskDelay( 500 / portTICK_PERIOD_MS );
70         digitalWrite(led, LOW);
71         vTaskDelay( 500 / portTICK_PERIOD_MS );
72         vTaskSuspend(TaskLed_Handler);
73     }
74 }

```





# Codigo FreeRTOS End Device en Proteus

```

76 void TaskRxTrama(void *pvParameters) // This is a Main task.
77 {
78     for (;;)
79     {
80         if(Serial.available()){
81             if(estado==0 and Serial.read () == 0x24){
82                 RXTrama[estado]=0x24;estado=1;
83             }else if(estado>0){
84                 RXTrama[estado]=Serial.read ();
85                 estado++;
86             }if(estado>4){
87                 estado=0;
88                 if(RXTrama[0]==TramaS[0] and RXTrama[1]==TramaS[1] and RXTrama[2]==TramaS[2] and RXTrama[3]==TramaS[3]and RXTrama[4]==TramaS[4]){
89                     //Serial.println("Valores en sensor "+String(index));
90                     Serial.println(String(index));
91                     //Enviar todos los datos almacenados usando la trama de envio al coordinador*****
92                     for(int k=0;k<index;k++){
93                         //vTaskDelay( 500 / portTICK_PERIOD_MS );
94                         Serial.write(0x24);
95                         Serial.write(0x36);
96                         Serial.write(0x41);
97                         Serial.print(Datos1[k]);
98                         Serial.write(0x53 ^ int(Datos1[k]));
99                         Serial.println();
100                     }
101                     index=0; //<-----
102                     //vTaskResume(TaskLed_Handler);
103                     //vTaskDelay( 2000 / portTICK_PERIOD_MS );
104                 }
105                 if(RXTrama[0]==TramaA[0] and RXTrama[1]==TramaA[1] and RXTrama[2]==TramaA[2]){
106                     if(RXTrama[4] == (((RXTrama[0] ^ RXTrama[1]) ^ RXTrama[2]) ^ RXTrama[3])){
107                         //Serial.println("Valor PWM en salida");
108                         analogWrite(led_pwm,RXTrama[3]); //PWM output
109                         //vTaskResume(TaskLed_Handler);
110                         //vTaskDelay( 2000 / portTICK_PERIOD_MS );
111                     }
112                 }
113             }
114         }
115     }
116 }

```

# Codigo FreeRTOS End Device en Proteus



```
118 void TaskAnalogRead(void *pvParameters) // This is a Main task.
119 {
120     for (;;) {
121         ADCsensor=0;
122         for(int j=0;j<15;j++){//durante 60 datos en 1min
123             ADCsensor += analogRead(A0)*9/1023;// [0% - 100%]
124             vTaskDelay( 1000 / portTICK_PERIOD_MS );//cada 1 segundo
125         }
126         ADCsensor /= 15;//60
127         Datos1[index]=ADCsensor;index+=1;
128     }
129 }
```



# SISTEMAS EMBEBIDOS COORDINATOR (Phyton)

011000010111001101100001011011100111101001100001

## Codigo Python Coordinator en Raspberry



**ThingSpeak™**  
IoT Analytics  
with **MATLAB®**



01101010011001010101100000101101110

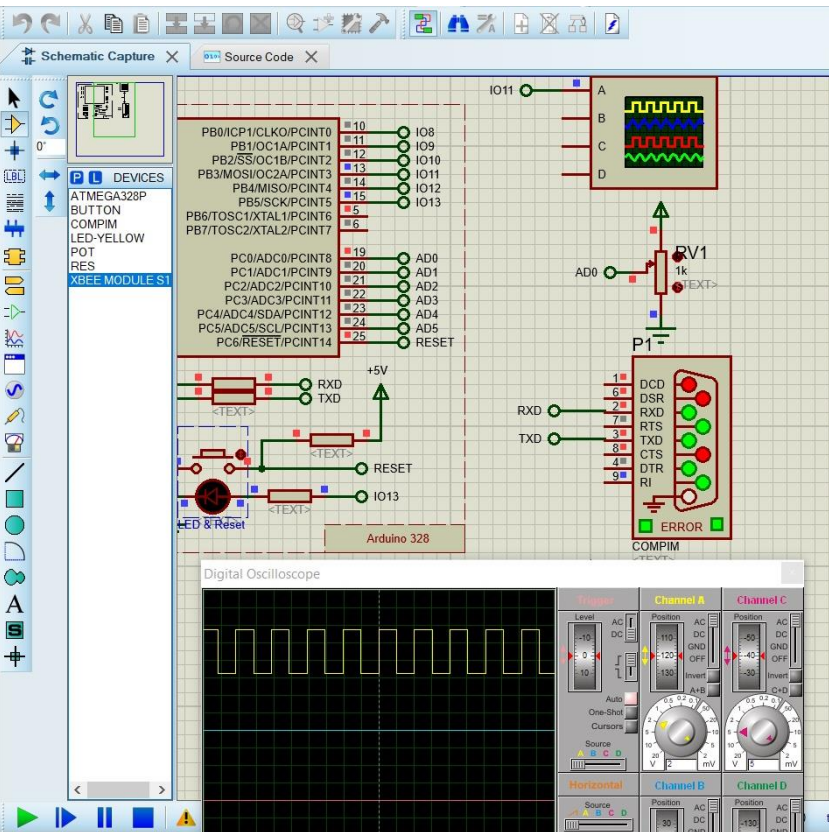




# Codigo Python Coordinador en Raspberry

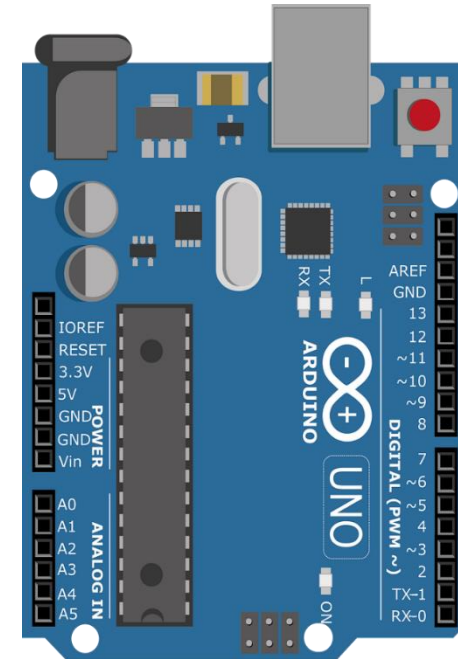


Coordinador con validación de datos, envío a ThingSpeak y activación de salida PWM  
(proyecto.py)

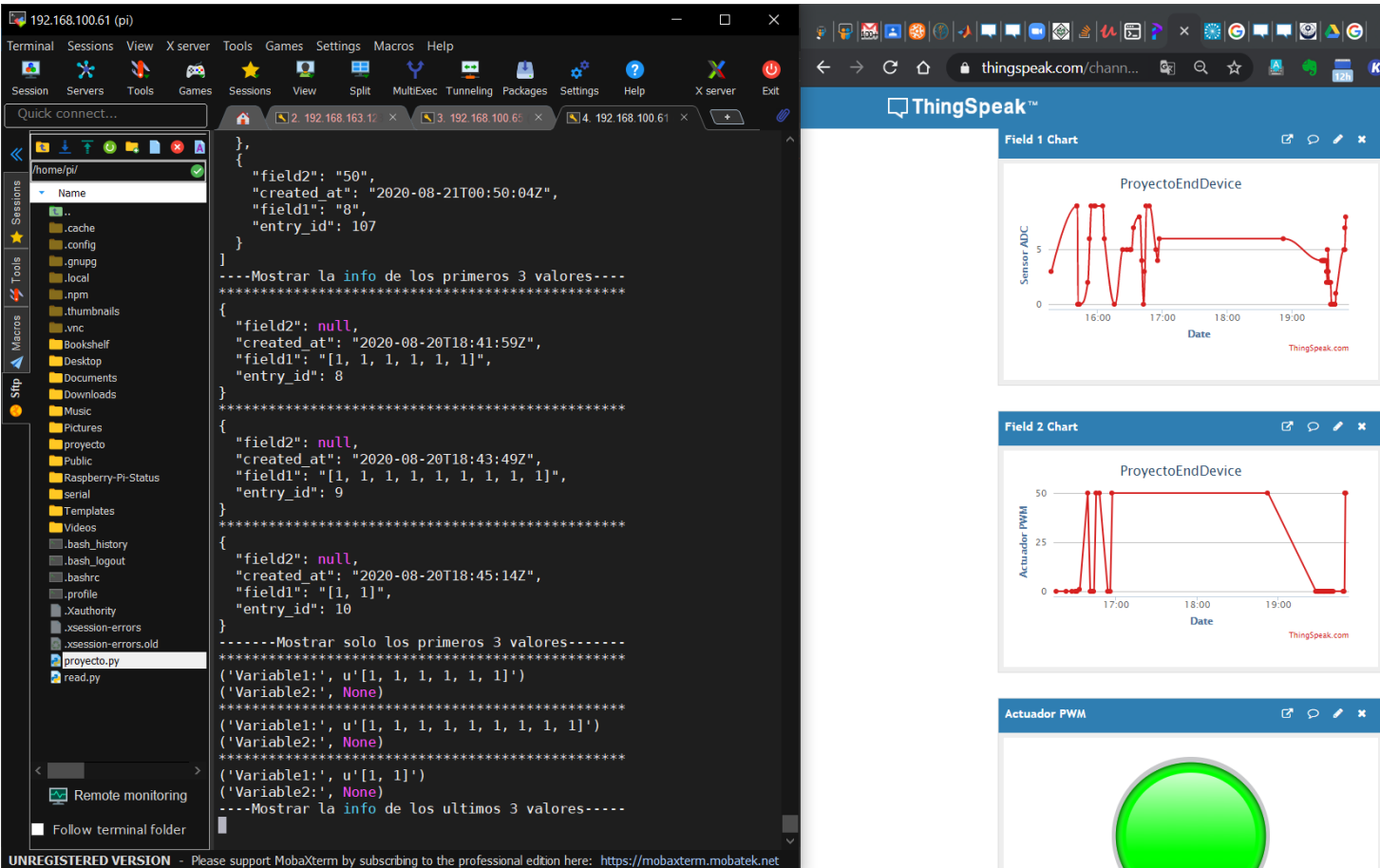




# Codigo FreeRTOS End Device en Arduino



Coordinador con validación de datos, envío a ThingSpeak y activación de salida PWM (proyecto.py)



**ThingSpeak™**  
IoT Analytics  
with **MATLAB®**

