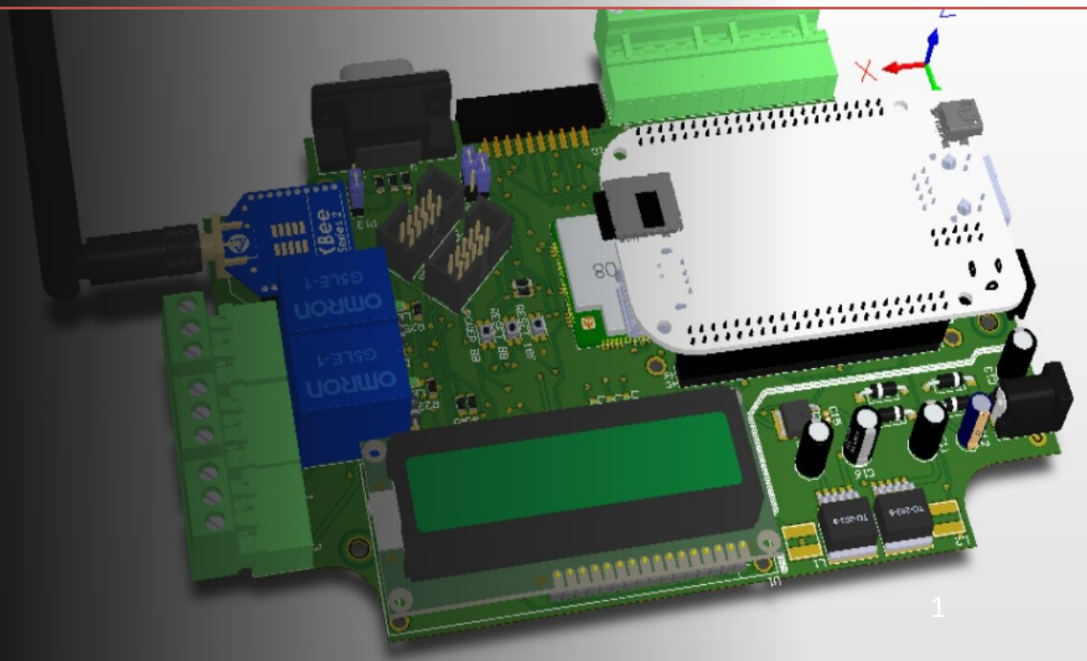
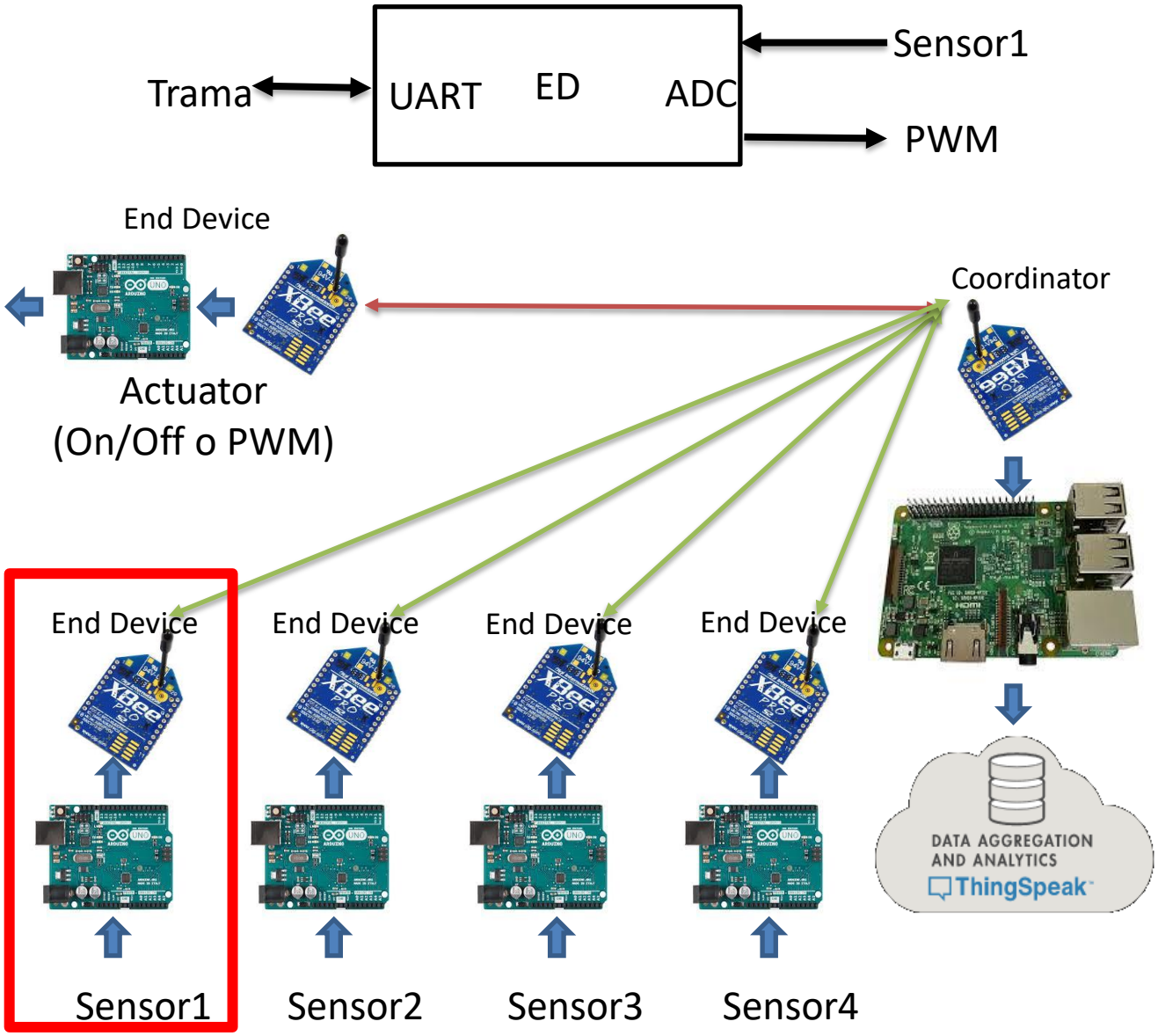
SISTEMAS EMBEBIDOS

END DEVICE (Free RTOS)



End Device



01101010011001010110000101011110

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:

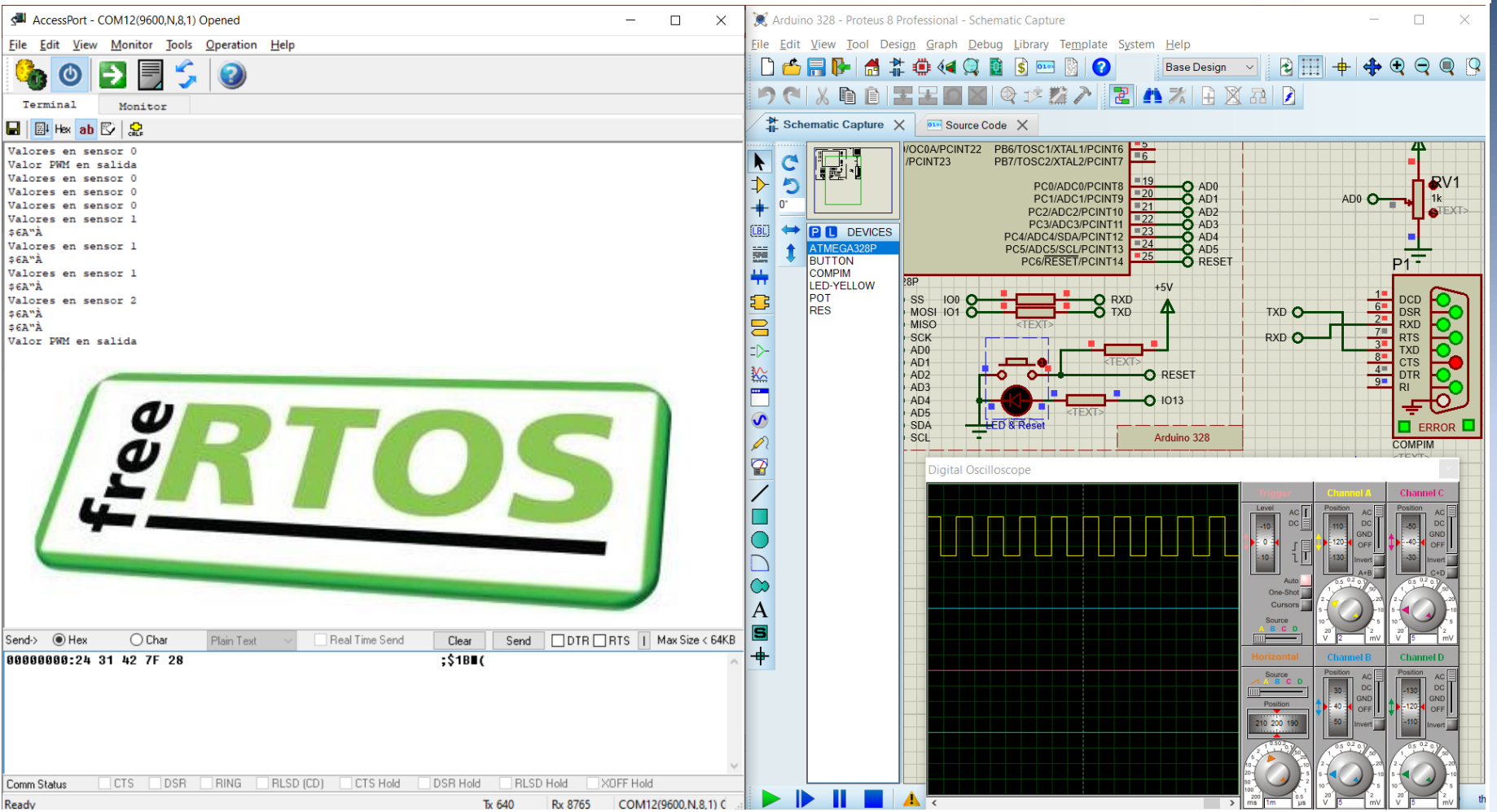
Start byte	ID byte	Byte Task	Byte Data	Byte Checksum
0x24 ($)	0x31 (1) – ED 0x36 (6) – C	0x41 (A) – Read 0x42(B) - PWM	0x00 . 0x30 (0) . 0xFF ()	XOR (Start byte, ID byte, byte Task, byte Data)

- El equipo Coordinador (C) deberá enviar la siguiente trama de solicitud de datos al End Device (ED) con Byte ID de 0x31: $\boxed{0x24}$ $\boxed{0x31}$ $\boxed{0x41}$ $\boxed{0x00}$ $\boxed{0x54}$
- El equipo End Device (ED) deberá enviar la siguiente trama, en respuesta a la solicitud que realizó el Coordinado con Byte ID de 0x36: $\boxed{0x24}$ $\boxed{0x36}$ $\boxed{0x41}$ $\boxed{0xYY}$ $\boxed{0xZZ}$
- Note que el byte $\boxed{0xYY}$ 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 $\boxed{0xZZ}$ deberá ser calculado en cada trama antes de ser enviado. Como ejemplo, el valor $\boxed{0x54}$ = $\boxed{0x24} \oplus \boxed{0x31} \oplus \boxed{0x41} \oplus \boxed{0x00}$

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


AccessPort - COM12(9600,N,8,1) Opened
File Edit View Monitor Tools Operation Help
Terminal
Monitor
Valores en sensor 0
Valor PWM en salida
Valores en sensor 0
Valores en sensor 0
Valores en sensor 0
Valores en sensor 1
$6A"À
Valores en sensor 1
$6A"À
Valores en sensor 1
$6A"À
Valores en sensor 2
$6A"À
$6A"À
Valor PWM en salida
freeRTOS
Send-> Hex Char Plain Text Real Time Send Clear Send DTR RTS Max Size < 64KB
00000000:24 31 42 7F 28 ;$1B■(
Comm Status CTS DSR RING RLSD (CD) CTS Hold DSR Hold RLSD Hold XOFF Hold
Ready Tx 640 Rx 8765 COM12(9600,N,8,1) C
Arduino 328 - Proteus 8 Professional - Schematic Capture
File Edit View Tool Design Graph Debug Library Template System Help
Base Design
Schematic Capture
Source Code
DEVICES
ATMEGA328P
BUTTON
COMPIM
LED-YELLOW
POT
RES
PC0/ADC0/PCINT8
PC1/ADC1/PCINT9
PC2/ADC2/PCINT10
PC3/ADC3/PCINT11
PC4/ADC4/SDA/PCINT12
PC5/ADC5/SCL/PCINT13
PC6/RESET/PCINT14
AD0
AD1
AD2
AD3
AD4
AD5
RESET
RV1
1k
P1
DCD
DSR
RXD
RTS
TXD
CTS
DTR
RI
ERROR
COMPIM
+5V
IO13
LED & Reset
Arduino 328
Digital Oscilloscope
Trigger
Channel A
Channel B
Channel C
Channel D
Horizontal

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

```
#include <Arduino_FreeRTOS.h>

TaskHandle_t TaskLed_Handler;

//variables globales
int led = 13;
int led_pwm=11;
unsigned long ADCsensor = 0;
int Datos1[240]={};//4 horas de datos
int index=0;
int estado =0;
int RXTrama[5];
int TramaS[5] ={0x24,0x31,0x41,0x00,0x54};//sensor
int TramaA[5] ={0x24,0x31,0x42,0x00,0x57};//actuador
int IdCoordinador =0x36;//'6'
// Sensor AD-0
//24 31 41 00 54
// Actuador PWM-11
//24 31 42 FF A8
//24 31 42 7F 28
//24 31 42 00 57

// End Device
void TaskBlink( void *pvParameters );
void TaskRxTrama( void *pvParameters );
void TaskAnalogRead( void *pvParameters );// Sensor
```

Codigo FreeRTOS End Device en Proteus


```
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  }
```


01101010011001010110000101011110

Codigo FreeRTOS End Device en Proteus


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


01101010011001010110000101101110

Codigo FreeRTOS End Device en Proteus


```
117  void TaskAnalogRead(void *pvParameters)  // This is a Main task.
118  {
119    for (;;){
120       ADCsensor=0;
121       for(int j=0;j<6;j++){//durante 60 datos en 1min
122         ADCsensor += analogRead(A0)*100/1023;// [0% - 100%]
123         vTaskDelay( 1000 / portTICK_PERIOD_MS );//cada 1 segundo
124       }
125       ADCsensor /= 6;//60
126       Datos1[index]=ADCsensor;index+=1;
127    }
128  }
```

Codigo FreeRTOS End Device en Arduino

Google Drive