Diseño de Sistemas de Internet de las Cosas

Tarea 1.

Profesor: Luciano Radrigan F.

Cada grupo tendrá una Raspberry pi y un microcontrolador ESP32, con estos elementos debe realizar las siguientes tareas:

- A través de la Raspberry generar una red WiFi (con nombre y contraseña).
- En la Raspberry programar un socket **TCP y UDP server**
- En el ESP32 programar un socket **TCP y UDP client**
- El paquete enviado desde el ESP32 deberá cumplir las siguientes características:

Hearder					**Data**			
2 bytes	6 bytes	1 byte	1 Byte	2 bytes	-	-	-	-
ID Device	MAC	Transport layer	ID Protocol	leng msg	Data 1	Data 2	Data …	Data N

- El protocolo se define como:

		1 bytes	1 bytes	4 bytes	1 bytes	4 bytes	1 bytes	4 bytes	4 bytes	4 bytes	4 bytes	4 bytes	4 bytes	4 bytes	4 bytes
ID Protocol	leng msg bytes	Data 1	Data 2	Data 3	Data 4	Data 5	Data 6	Data 7	Data 8	Data 9	Data 10	Data 11	Data 12	Data 13	Data 14
0	6	Val: 1	Batt_level	Timestamp											
1	16	Val: 1	Batt_level	Timestamp	Temp	Press	Hum	Co							
2	20	Val: 1	Batt_level	Timestamp	Temp	Press	Hum	Co	RMS						
3	44	Val: 1	Batt_level	Timestamp	Temp	Press	Hum	Co	RMS	Amp x	Frec x	Amp y	Frec y	Amp z	Frec z
		1 bytes	1 bytes	4 bytes	1 bytes	4 bytes	1 bytes	4 bytes	6400 bytes	6400 bytes	6400 bytes				
									Data 8	Data 9	Data 10				
4	24016	Val: 1	Batt_level	Timestamp	Temp	Press	Hum	Co	Acc_X	Acc_y	Acc z				

- El Flujo de datos es el siguiente: El microcontrolador deberá tener una conexión inicial TCP. Esta conexión realizara una consulta a la base de datos sobre que valores en la variable ID_*Protocol*, para enviar los datos según corresponda. Además se consultara el valor de la variable *Transport layer,* con el cual el microcontrolador enviara los datos pot TCP o UDP (Transport layer = 0 se selecciona TCP y Transport layer= 1, se elecciona UDP). Con la comunicación TCP el microcontrolador enviará un paquete de datos y deberá estar 1 minuto en el modo *Deepsleep*, luego del minuto se repite la rutina. Análogamente para el envió con UDP se envían los datos de forma continua hasta que el valor de *transport_layer* cambie. Este valor se debe modificar de forma manual.
- Los datos recepcionada por los Socket deberán ser almacenados en una base de datos SQL.
- Crea una base de datos la cual contenga: (i) Una tabla de los datos recepcionados con su *timestamp* y el identificador del dispositivo (*Id_device y MAC*), (ii) una tabla con los logs del dispositivo considerando el *Id_protocol* y el *Transport_layer*. Una tabla de configuración en que se cambia los valores de *ID_protocol y Transport layer* para que el microcontrolador cambie su modo de operación.
- Los datos enviados desde el microcontrolador deberán ser programados, para esto se deberán implementar funciones que emulen el funcionamiento de los sensores, dentro de la que destacan:

- Aceloremeter_sensor: Genera un vector de 2000 datos por eje (x,y,z). Los datos son flotantes dados por la siguiente formula
  - Acc_x= 2*sen(2*pi*0.001*n);
  - Acc_y= 3*cos(2*pi*0.001*n);
  - Acc_z= 10*sen(2*pi*0.001*n);
- THPC_sensor:
  - Temp = Valor aleatorio entre 5.0 a 30.0.
  - hum = Valor aleatorio entre 30 a 80.
  - pres = Valor aleatorio entre 1000 y 1200
  - Co = Valor aleatorio entre 30.0 a 200.0
- Batt_sensor:
  - Value = Valor aleatorio entre 1 y 100
- Aceloremeter_kpi
  - RMS = ((Amp_x)^2+ (Amp_y)^2+(Amp_z)^2)^1/2
  - Amp_x = Valor aleatorio entre 0.0059 y 0.12
  - Frec_x = Valor aleatorio entre 29.0 y 31.0
  - Amp_y = Valor aleatorio entre 0.0041 y 0.11
  - Frec_y = Valor aleatorio entre 59.0 y 61.0
  - Amp_z = Valor aleatorio entre 0.008 y 0.15
  - Frec_z = Valor aleatorio entre 89.0 y 91.0

- La entrega de esta tarea consistirá en
  - 1- Una demostración en vivo del funcionamiento del sistema, el 7 de octubre
  - 2- Subir código a U-cursos, el 7 de octubre.