Tarea 1

CC5326 – Diseño de Sistemas de Internet de las Cosas

Universidad de Chile
Facultad de Ciencias Físicas y Matemáticas
Departamento de Ciencias de la Computación

Link: `https://iiot-embedded.com/iot/tareas/t1/`

Profesor: Luciano Radrigán F.
Ayudante: Gabriel Díaz C.

Objetivos a cumplir

- Generar una red WiFi a través de la Raspberry Pi (con nombre y contraseña), que otorgue IPs a los ESP32 mediante DHCP.
- Programar un servidor de socket TCP y UDP en la Raspberry Pi.
- Programar un cliente de socket TCP y UDP en el ESP32.
- Enviar un paquete desde el ESP32 con datos generados en el mismo dispositivo, que será recibido por el servidor.
- Almacenar la información en una base de datos PostgreSQL utilizando Docker.
- Posibilitar el cambio del protocolo de envío de datos desde la base de datos.
- Permitir el cambio del tipo de conexión (TCP o UDP) desde la base de datos.
- Los cambios en la base de datos deberán reflejarse en el ESP32 (por ejemplo, si se cambia el protocolo de envío, el ESP32 deberá enviar los datos con el nuevo protocolo).
- Recibir datos simultáneamente de dos ESP32 y almacenarlos en la base de datos.

Indicaciones

Flujo de los datos

1. El ESP32 se inicia mediante conexión TCP.

2. Consulta la base de datos por `ID_protocol` y `Transport_Layer`. Con estos valores, determinará qué protocolo de paquete y tipo de conexión (TCP o UDP) utilizará.

3. Dependiendo del tipo de conexión:

   - **TCP:** Envío de paquete, luego entra en modo Deep Sleep por 60 segundos.
   - **UDP:** Envío continuo de datos hasta cambiar valor de `Transport_Layer`.

4. La Raspberry Pi debe descomponer los paquetes y almacenarlos en la base de datos SQL. En cada envío de datos, se deberá verificar la configuración de la base de datos. Si ha habido algún cambio, se deberá ajustar el protocolo de envío o el tipo de conexión, según corresponda.

5. En cada envío se debe verificar configuración actual.

Encabezados de los paquetes

Cada paquete deberá contar con su propio encabezado, que incluirá la siguiente información. Este será común para todos los protocolos de envío:

- **ID** (2 bytes)
- **Device MAC** (6 bytes)
- **Transport Layer** (1 byte)
- **ID Protocol** (1 bytes)
- **Length** (2 bytes)

Protocolos de envío

- Cada protocolo define los datos a enviar según configuración en la base de datos.
- Todos siguen el formato `Headers + Body`.

Ejemplo de datos por protocolo

Protocolos 0 al 3

- Data 1: Batt level (1 byte)
- Data 2: Timestamp (4 bytes)
- Data 3: Temp (1 byte)
- Data 4: Press (4 bytes)
- Data 5: Hum (1 byte)

- Data 6: CO (4 bytes)
- Data 7: RMS (4 bytes)
- Data 8–13: Amp/Frec en X, Y, Z (4 bytes cada uno)

Protocolo 4

- Igual que anterior hasta Data 6.
- Data 7–12: Vectores de 2000 valores float (8000 bytes cada uno)

Generación de datos

Los datos deben ser generados aleatoriamente dentro de los rangos entregados. Esto con el objetivo de simular la recogida de datos de sensores.

Acceloremeter_kpi

Representa un sensor de vibraciones que mide en los tres ejes y calcula su promedio mediante la fórmula de la raíz cuadrada de la media (RMS). Los valores son float.

- Ampx: 0.0059 – 0.12
- Freqx: 29.0 – 31.0
- Apy: 0.0041 – 0.11
- Freqy: 59.0 – 61.0
- Ampz: 0.008 – 0.15
- Freqz: 89.0 – 91.0
- RMS: $\sqrt{Ampx^2 + Ampy^2 + Ampz^2}$

Acceloremeter_Sensor

Representa un medidor de aceleración que genera un vector de 2000 datos por eje (X,Y,Z) para sus dos parámetros: aceleración y velocidad angular. Los datos son float.

- Acc_X/Y/Z: -16.0 – 16.0
- Rgyr_X/Y/Z: -1000 – 1000

THPC_Sensor

Representa un sensor de cada uno de estos aspectos. Los valores son enteros, excepto para CO que es float.

- Temp: 5 – 30
- Hum: 30 – 80
- Pres: 1000 – 1200
- CO: 30 – 200 (float)

Batt_Sensor

Representa el nivel de batería del aparato.

- Nivel batería: uint8 entre 1 y 100

Base de Datos

Tablas requeridas

- **Datos:** Esta tabla almacenará todos los datos recibidos, incluyendo un sello de tiempo (timestamp) y el identificador del dispositivo (Id_device y MAC).
- **Logs:** Esta tabla registrará la información de cada conexión recibida por el servidor, incluyendo el ID_device, el tipo de capa de transporte (Transport_Layer), el protocolo utilizado y su timestamp.
- **Configuración:** Esta tabla contendrá las variables ID_protocol y Transport_Layer, que se utilizarán para configurar el envío de datos al iniciar la conexión del ESP32. Estas variables deben poder modificarse de alguna manera.
- **Loss:** Esta tabla almacenará el tiempo de demora en la comunicación (tiempo desde el momento de envío hasta el momento de escritura en la base de datos, calculado usando la diferencia entre esos timestamps) y la cantidad de pérdida de paquetes (packet loss) en bytes para cada comunicación que haya ocurrido.

Inicialización con Docker

```
docker-compose up -d
```

Entrega

- Demostración en vivo: por reagendar.
- Subir código a U-Cursos (o enlace a repositorio Git).

Recomendaciones

- Coordinar pruebas físicas en persona.
- Usar repositorio compartido (como GitHub).
- Conexiones físicas: revisar pines antes de usar.
- Verificar componentes antes de culpar al software.
- Consultas: foro o equipo docente.