Implementación De Router de Borde Con servidor de Interfaz Web

---

- ## Pasos previos
  1. Para ello realizamos los pasos estipulados en los manuales anteriores (instalación inicial y Nodos con servidor local), en estos se realizarán cambios estipulados a continuación.
- ## Configuración del Borde Router

  1. EL código fuente y los archivos necesarios para el Borde Router están organizados de la siguiente manera:
  -) main.c :Archivo principal con la lógica del Borde Router
  -) Makefile: Archivos de configuración para compilar el proyecto.
  -) Archivos de configuración adicionales: Estos presentan la configuración para las diferentes interfaces (Makefile.ethos.conf, Makefile.slip.cof, etc.)

  2. Modificaciones Realizadas en el archivo principal que contienen el shell básico para los comandos y cola de mensajes.
  3. Configuramos el archivo Makefile para que utilizara ethos como medio de conexión para habilitar el IPv6, ICMPv6 y la funcionalidad del Router Border. **(Estos cambios ya se encuentran presentes en los archivos del repositorio del proyecto)**
  4. Compilación y flasheo del ESP32-WROOM-32
     -) Comando de compilación del Borde Router:

```
USEMODULE='gnrc_rpl' CFLAGS='-DESP_NOW_CHANNEL=6' make BOARD=esp32-wroom-32 BUI
LD_IN_DOCKER=1
```

     -) Código para el flasheo del Border Router:

```
USEMODULE='gnrc_rpl' CFLAGS='-DESP_NOW_CHANNEL=6' make
BOARD=esp32-wroom-32 BUILD_IN_DOCKER=1 PORT=/dev/ttyUSB0 flash
```

     -) Código de inicialización del Border Router:

```
make term BOARD=esp32-wroom-32 BUILD_IN_DOCKER=1 PORT=/dev/ttyUSB0
```

- ## Configuración del Nodo IoT

  1. EL código fuente y los archivos necesarios para el Nodo están organizados de la siguiente manera:
  -) main.c :Contiene la lógica para ejecutar el Shell y gestionar los comandos.
  -) Makefile: Archivos de configuración para compilar el proyecto.
  -) udp.c: Implementa las funciones para enviar datos vía UDP.

  2. Modificaciones realizadas en el archivo udp.c
     Se implemento la lectura del sensor DHT11 y configuración de el envió de datos en formato temperatura, humedad estos se enviarán al servidor central por medio del router border. **(Estos cambios ya se encuentran presentes en**

los archivos del repositorio del proyecto)

3. Compilación y flasheo de la ESP32-WROOM-32:

   -) Compilación del nodo IoT

```
make BOARD=esp32-wroom-32 BUILD_IN_DOCKER=1
```

   -) Flasheo del Nodo IoT

```
make BOARD=esp32-wroom-32 BUILD_IN_DOCKER=1 PORT=/dev/ttyUSB2 flash
```

   -) Ejecutamos el Nodo IoT

```
make term BOARD=esp32-wroom-32 BUILD_IN_DOCKER=1 PORT=/dev/ttyUSB2
```

- ## Configuración del Servidor Central

El servidor central recibe los datos de los nodos IoT, los almacena en una base de datos SQLite y genera una interfaz web para la visualización de los datos en tiempo real

1. EL código fuente y los archivos necesarios para el Servidor Central están organizados de la siguiente manera:

   Utilizamos Flask como firewall web y consta de los siguientes archivos:

   -) server.py: Contiene la lógica principal para recibir datos UDP.
   -) templates/index.html: Contiene la estructura de la página web HTML (muestra de datos y grafico en tiempo real).

2. Instalamos las dependencias necesarias:

```
pip install flask sqlite3
```

-) inicializamos la base de datos, para ello el servidor creara automáticamente las tablas de datos e historial en SQLite al ejecutarse por primera vez.

-) Ejecutamos el servidor

```
python3 server.py
```

-) Salida esperada: Base de datos creada y conectada

```
/path/to/sensor_data.db Servidor HTTP disponible en: http://127.0.0.1:5000 Serv
idor UDP escuchando en [::]:8888
```

- ## Interfaz Web

La interfaz web es accesible desde cualquier navegador en la red local y muestra:

1. Tabla dinámica: muestra los datos de temperatura y humedad en tiempo real
2. Gráficos interactivos: Visualización de un grafico con los datos recibidos de cada nodo.
3. Diseño: Se utilizo Google Charts para los gráficos y CSS personalizados para la tabla.

- # Resultados