

Tarea 2

Profesor: Luciano Radrigan F.
Auxiliar: Ricardo Aravena P.

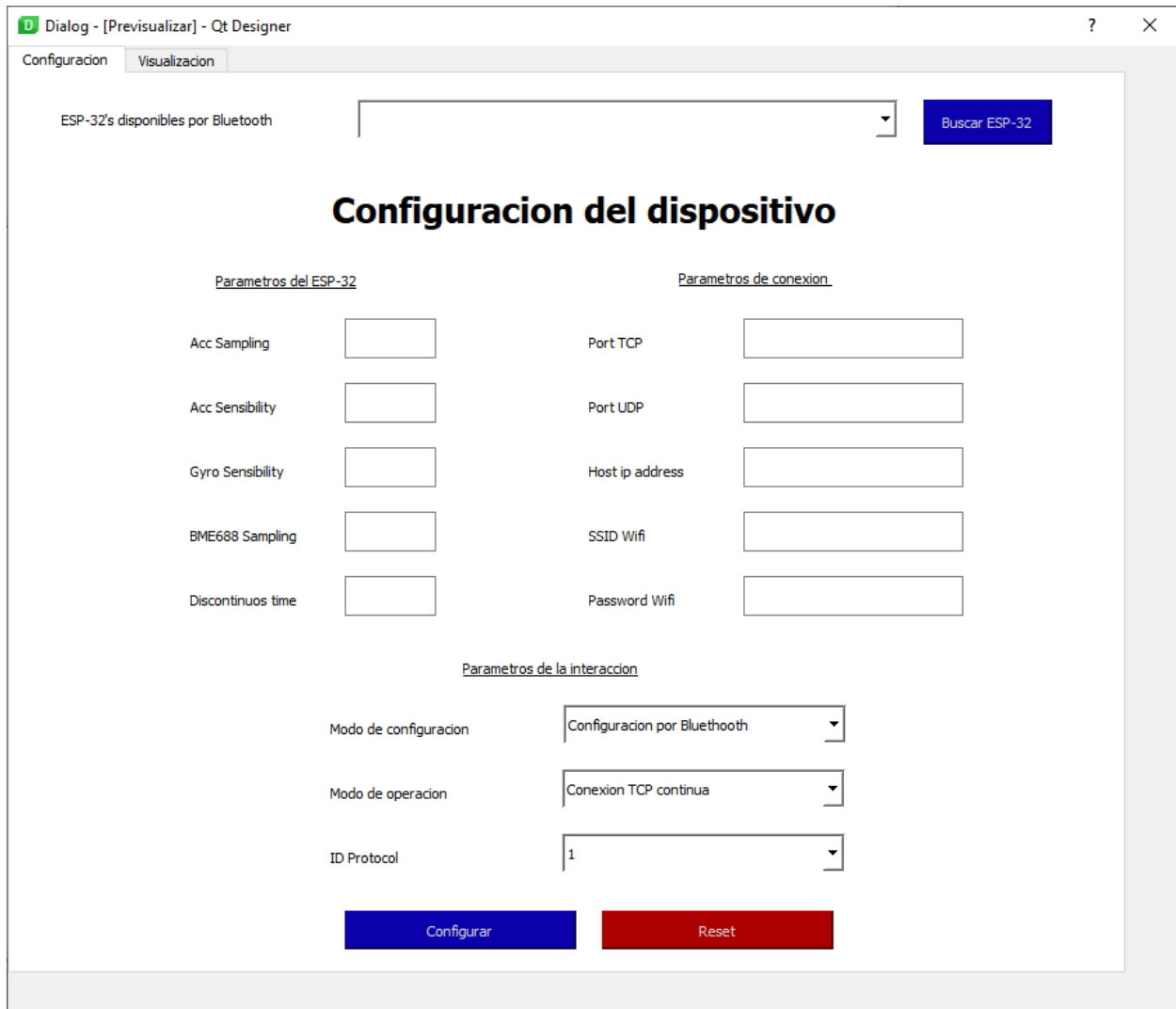
Cada equipo contara con una Raspberry Pi3 y dos microcontrolador ESP32.
A partir de lo que se realizo durante la tarea 1, se busca extender este funcionamiento a incluir conexión Bluetooth, junto con la capacidad de configurar de forma más exacta desde la Raspberry a través de una interfaz gráfica cosas como el tipo de conexión, el protocolo y otros detalles.

Objetivos

- Crear una aplicación de escritorio con **PythonQT 5** que permita realizar una configuración inicial con los ESP32 y las gráficas de los datos recepcionados en operación. (Imágenes de ej de estas interfaces se pueden encontrar en **Sección de Interfaz Gráfica**). Para la recepción de datos deberá estar corriendo un servidor que puede ser creado desde la interfaz o puede tenerlo ya de fondo.
- Los microcontroladores ESP32 deben comenzar en un modo de operación (status 0) que busca esperar una configuración para su modo de funcionamiento, en este modo debe poder realizarse una conexión Bluetooth punto a punto desde la interfaz para hacer la configuración. Toda la configuración se guardara en la memoria no volátil del microcontrolador.
- Los microcontroladores ESP32 tras conseguir un modo de operación (designado como status) deben poder comunicarse con la Raspberry pi 3 para poder guardar toda la información generada en estos (de forma parecida a la tarea 1). Entre estos modos de operación, aparte de poder realizar todos los mismos protocolos de paquete, deben poder conectarse a través de TCP Continuo o Discontinuo, UDP y BLE (Bluetooth Low Energy) Continuo o Discontinuo, en cualquier combinación permitida (Para más información acerca de los Status lea la sección **de Descripción**)

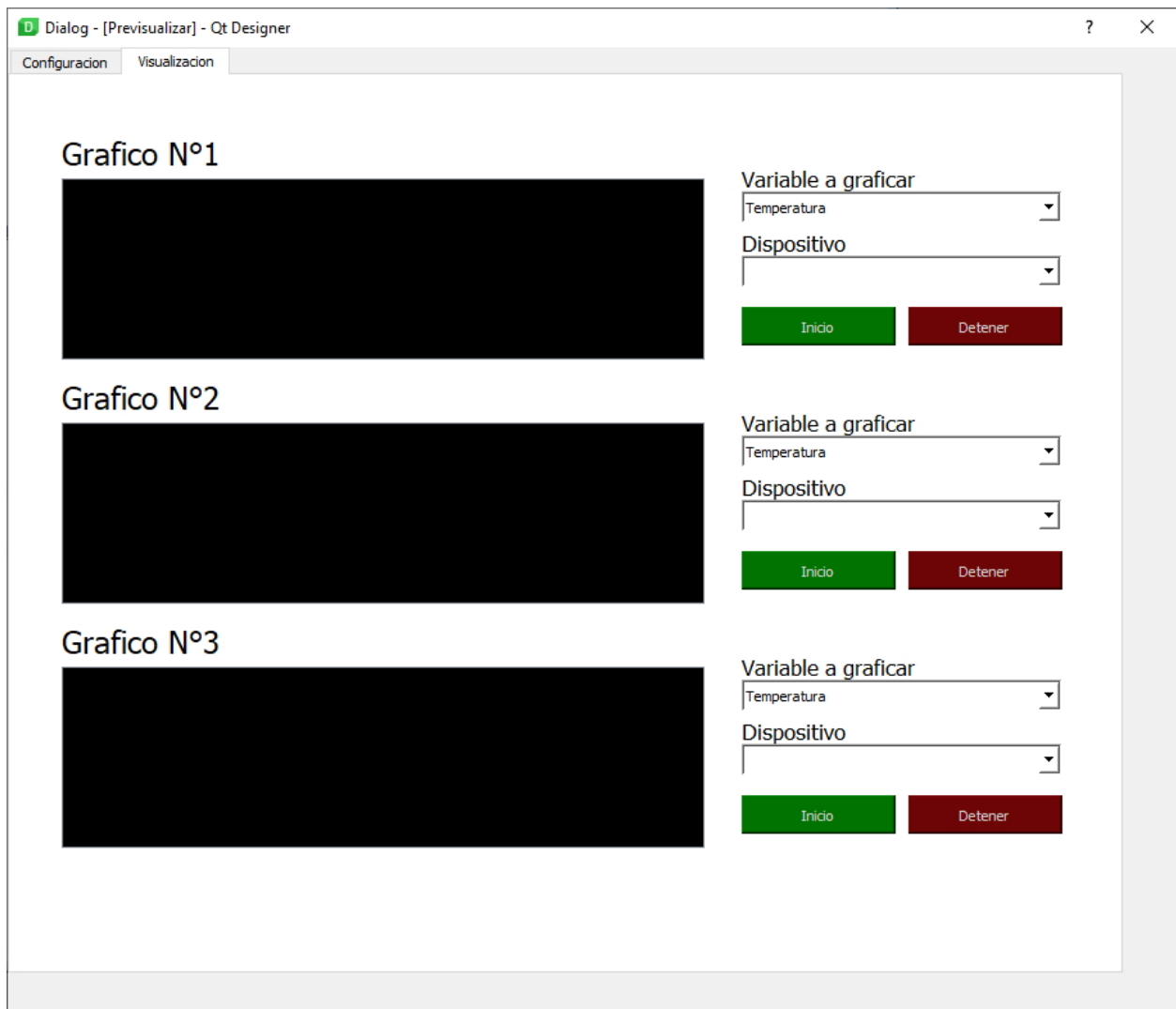
Interfaz Gráfica

Estos son ejemplos: cambielas tanto como necesite para que cumplan con las funciones pedidas. Se incluye un archivo .ui con esta interfaz ya diseñada para que modifique como desee desde esta base.



En esta pestaña debe ser posible primeramente poder seleccionar las ESP32 a traves de una busqueda Bluetooth y configurar todas las variables pedidas del ESP32.

Figura 1: Pestaña de Configuración de la interfaz



En esta pestaña se debe poder seleccionar alguna variable que la ESP32 este enviando y graficarla en vivo (Puede simplemente graficar lo que encuentre en la base de datos mientras se actualiza)

Figura 2: Pestaña para revisar los datos de la interfaz

Descripción de Status

Tabla 1: Modos de operación ESP32

Status	Descripción	Compatibilidad con IP_Protocol
0	Configuración por Bluetooth	1
20	Configuración vía TCP en BD	1
21	Conexión TCP Continua	1-2-3-4-5
22	Conexión TCP Discontinua	1-2-3-4-5
23	Conexión UDP	1-2-3-4-5
30	BLE Continua	1-2-3-4
31	BLE Discontinua	1-2-3-4

1. Configuración por Bluetooth (Status=0)

El ESP32 se configura con los datos seteados por la interfaz QT desde la raspberry a través de una comunicación Bluetooth punto a punto (se recomienda utilizar la librería Pygatt en la Raspberry). Los parámetros para configurar deben guardarse en la Raspberry en una tabla de la base de datos (llamada config) y en el ESP32 en memoria no volátil. Los parámetros para configurar son los siguientes:

Tabla 2: Parámetros de Configuración

Tipo	Descripción
int8_t	Status
int8	ID_Protocol
int32_t	BMI270_Sampling (Valores Posibles: 10, 100, 400, 1000)
int32_t	BMI270_Acc_Sensibility (Valores Posibles: 2,4,8,16)
int32_t	BMI270_Gyro_Sensibility (Valores posibles: 200, 250, 500)
int32_t	BME688_Sampling (1,2,3,4)
int32_t	Discontinuos_Time
int32_t	Port TCP
int32_t	Port UDP
int32_t	Host_IP_Addr
Char	SSID X
Char	Pass

2. Configuración vía TCP en BD (status = 20)

El ESP32 tendrá un Cliente TCP y la Raspberry un Servidor TCP (el Ssid, Pass y Port_TCP se toman de los valores configurados por la interfaz). En este modo el ESP32 puede actualizar cualquiera de los valores de la tabla **Parámetros de Configuración** a través de una conexión TCP. Los valores se adquieren de la tabla config de la DB.

3. Conexión TCP continua (status= 21)

El ESP32 tendrá un Cliente TCP y la Raspberry un Servidor TCP (Este debe poder iniciarse desde la interfaz con la configuración puesta ahí). Según el valor de ID_Protocol es el paquete de datos que se transferirá. (protocolos de datos se observan en la **Tabla 3**). El ESP32 deberá enviar este paquete de forma continua hasta que desde la Raspberry se detenga la conexión (desde la interfaz gráfica).

4. Conexión TCP discontinua (status= 22)

El ESP32 tendrá un Cliente TCP y la Raspberry un Servidor TCP (Este debe poder iniciarse desde la interfaz con la configuración puesta ahí). Según el valor de ID_Protocol es el paquete de datos que se transferirá. (protocolos de datos se observan en la **Tabla 3**). Luego de enviar los datos según el valor de Discontinuous_Time el ESP32 entrara por ese tiempo en modo Deep_sleep. Se recomienda que el Discontinuous_Time tenga como unidad minutos y que su valor mínimo sea 1. Este modo se deberá poder detener desde la Raspberry, deteniendo envíos posteriores de datos (desde la interfaz gráfica).

5. Conexión UDP (status=23)

El ESP32 tendrá un Cliente UDP y la Raspberry un Servidor UDP (Este debe poder iniciarse desde la interfaz con la configuración puesta ahí). Según el valor de ID_Protocol es el paquete de datos que se transferirá. (protocolos de datos se observan en la **Tabla 3**). El ESP32 deberá enviar este paquete de forma continua hasta que desde la Raspberry se detenga la conexión (desde la interfaz gráfica).

6. BLE continua (status=30)

El ESP32 tendrá un Server BLE y la Raspberry un Servidor Cliente. (Se recomienda utilizar la librería pygatt). Según el valor de ID_Protocol es el paquete de datos que se transferirá. (protocolos de datos se observan en la **Tabla 3** y este modo no es compatible con ID_protocol= 5). El ESP32 deberá enviar este paquete de forma continua hasta que desde la Raspberry se detenga el envío (desde la interfaz gráfica).

7. BLE discontinua (status=31)

El ESP32 tendrá un Server BLE y la Raspberry un Servidor Cliente. (Se recomienda utilizar la librería pygatt). Según el valor de ID_Protocol es el paquete de datos que se transferirá. (protocolos de datos se observan en la **Tabla 3** y este modo no es compatible con ID_protocol= 5). Luego de enviar los datos según el valor de Discontinuous_Time el ESP32 entrara por ese tiempo en modo Deep_sleep. Se recomienda que el Discontinuous_Time tenga como unidad minutos y que su valor mínimo sea 1. Este modo se deberá poder detener desde la Raspberry, deteniendo envíos posteriores de datos (desde la interfaz gráfica).

Datos

Los datos enviados desde los microcontroladores deberán ser programados, para esto se deberán implementar funciones que emulen el funcionamiento de los sensores, dentro de la que destacan:

- **Acceloremeter__Sensor:** Un medidor de aceleración, genera un vector de 2000 datos por eje (X,Y,Z) para sus dos parámetros, aceleración y velocidad angular. Los datos son *float* dados por la siguiente formula:
 - **Acc__X** = Valores aleatorios entre -16. y 16.
 - **Acc__Y** = Valores aleatorios entre -16. y 16.
 - **Acc__Z** = Valores aleatorios entre -16. y 16.
 - **Rgyr__X** = Valores aleatorios entre -1000 y 1000
 - **Rgyr__Y** = Valores aleatorios entre -1000 y 1000
 - **Rgyr__Z** = Valores aleatorios entre -1000 y 1000
- **THPC__Sensor**(Temperatura-Humedad-Presión-CO): representa un sensor de cada uno de estos aspectos, genere su medición usando los siguientes valores (enteros, excepto por CO que será float):
 - **Temp** = Valor aleatorio entre 5 y 30
 - **Hum** = Valor aleatorio entre 30 y 80
 - **Pres** = Valor aleatorio entre 1000 y 1200
 - **CO** = Valor aleatorio entre 30 y 200
- **Batt__Sensor:** representando el nivel de batería del aparato, Valor aleatorio entre 1 y 100 (u_int8).
- **Acceloremeter__kpi:** representa un sensor de vibraciones, midiendo en los tres ejes y sacando su promedio (RMS, root mean square), para sus valores (float) use:
 - 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
 - $RMS = \sqrt{(Amp_x^2 + Amp_y^2 + Amp_z^2)}$

Protocolos de Paquete

Los tamaños de estas tablas son referenciales, los predispuestos son los tamaños en los cuales se pueden enviar estos datos, no necesita usar los mismo tamaños si es que no logra 'apretarlos' a este nivel, mientras sea consistente en su envío de datos todo bien.

- Cloud se refiere a un header, el mismo que la tarea anterior.

Tabla 3: Forma de los paquetes

Cloud					Data			
2 bytes	6 bytes	1 byte	1 byte	2 bytes	-	-	-	-
ID Device	MAC	ID Protocol	Status	Leng Msg	Data 1	Data 2	Data ...	Data N

Tabla 4: Protocolos 1 al 3

Tamaño de datos ->				1 bytes	4 bytes	1 bytes	4 bytes	1 bytes	4 bytes	4 bytes
	ID_Protocol	Status	Leng msg	Data 1	Data 2	Data 3	Data 4	Data 5	Data 6	Data 7
Cloud	1	~	6	Batt_Level	Timestamp	-	-	-	-	-
	2	~	16	Batt_Level	Timestamp	Temp	Pres	Hum	CO	-
	3	~	20	Batt_Level	Timestamp	Temp	Pres	Hum	CO	RMS

Tabla 5: Protocolo 4

Tamaño de datos ->				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
Cloud	ID_Protocol	Status	Leng msg	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
	4	~	44	Batt_Level	Timestamp	Temp	Pres	Hum	CO	RMS	Amp x	Frec x	Amp y	Frec y	Amp z	Frec z

Tabla 6: Protocolo 5

Tamaño de Datos ->				1 bytes	4 bytes	1 bytes	4 bytes	1 bytes	4 bytes	Data array de 2000 floats	~	~	~	~	~
	ID_Protocol	Status	Leng Msg	Data 1	Data 2	Data 3	Data 4	Data 5	Data 6	Data 7	Data 8	Data 9	Data 10	Data 11	Data 12
Cloud	5	~		Batt_level	Timestamp	Temp	Pres	Hum	CO	Rgyr x	Rgyr y	Acc z	Rgyr x	Rgyr y	Rgyr z

Detalles finales

- Se adjunta base de datos propuesta. (desde la interfaz se guardan las opciones de configuración en la tabla configuration, y desde ahí el microcontrolador puede ir reconfigurándose). Solo para términos de depuración la variable Conf_peripheral se compone como la suma de string de los valores de configuración de Acc_samplig, Acc_sensibility Gyro_sensibility, BME688_sampling y discontinuos_time.

Entrega

Esta contara de dos partes:

- Se realizara de forma presencial una demostración del funcionamiento de la tarea el día Viernes 25 de Noviembre, a una a acordar con el profesor (cada grupo va tomar una hora que pueda durante ese día)
- Se deberá entregar todo el código como un repositorio Git (teniendo documentación suficiente para correr la tarea) o entregarlo en forma de un zip con todo el código (que este bien ordenado y documentado).

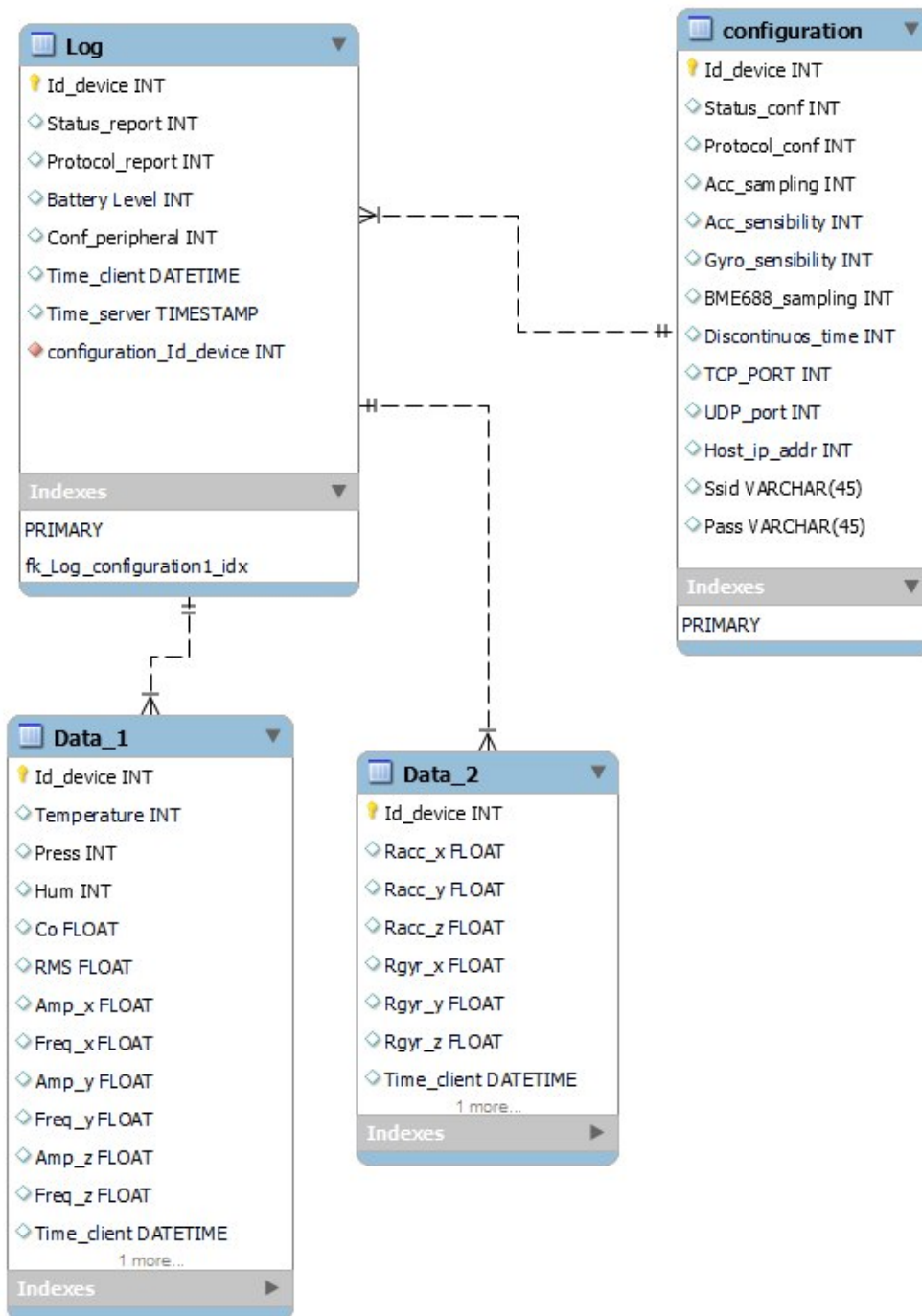


Figura 3: esquema de la base de datos