2021

Eric García Carrizo

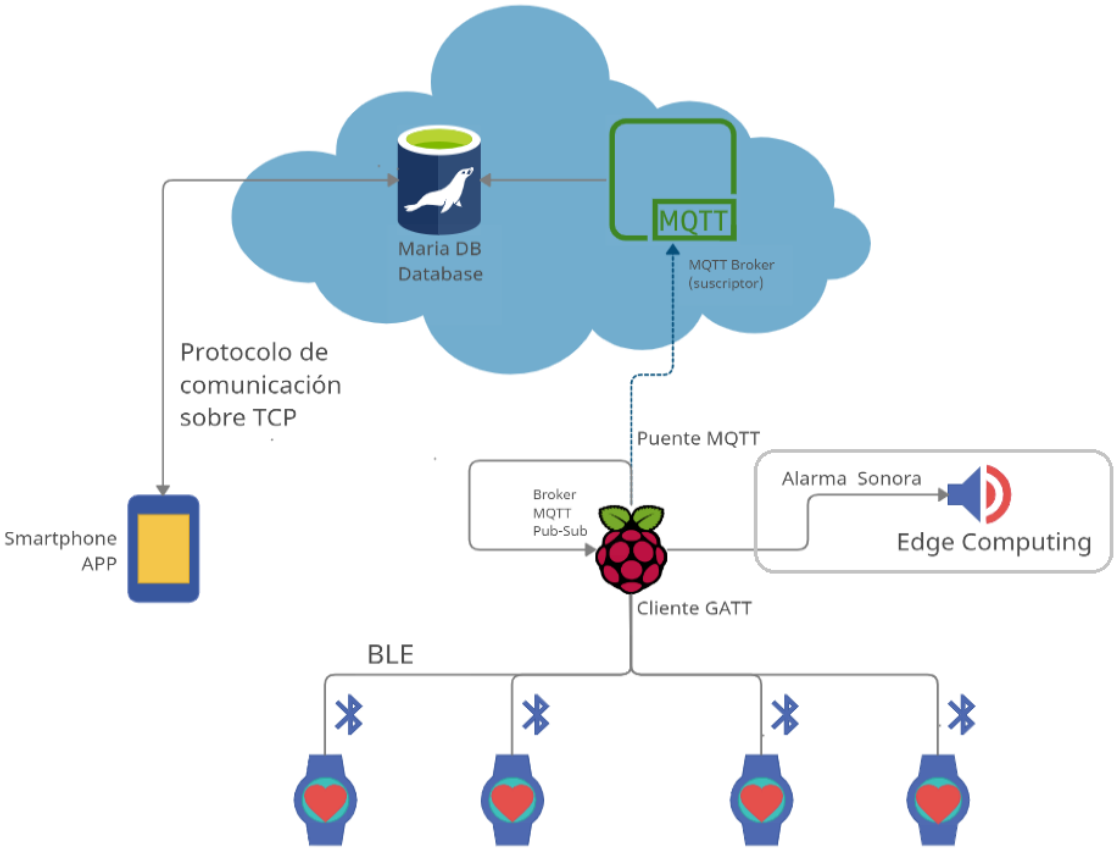
[DOCUMENTO DE ESPECIFICACION DE REQUISITOS]

1.-Introduccion

1.1.-Estructura de Red

Se pretende crear un sistema capaz de medir el pulso y la saturación de oxigeno en sangre de una persona a lo largo de un día y mostrarlo en una aplicación móvil.

La estructura de comunicación es la que se muestra a continuación y consta de los siguientes elementos.



- Varios dispositivos que actúan como servidor GATT y que publican a través de BLE los datos que recopilan de un sensor de pulso y saturación de O2.
- Una raspberry pi que actúa como cliente GATT, recibiendo los datos de los dispositivos BLE. Y que a su vez es un bróker MQTT, el cual tiene configurado un puente para enviar todos los datos que recibe a otro bróker en la nube. Como ultima funcionalidad esta también lanza una alarma local cuando se deja de detectar pulso en alguno de los sensores.
- Un Servidor Cloud que recibe todos los mensajes MQTT de la red local y en función de una serie de criterios los almacena en una base de datos

- Una aplicación móvil que solicita al servidor en la nube los datos de un determinado dispositivo y los muestra en una grafica.

1.2-Criterios de selección de herramientas de desarrollo

Dentro de cada bloque se aborda la elección de las herramientas, tanto de hardware como de software, y métodos de trabajo en base a las siguientes características.

- Facilidad de uso: como de difícil es el uso de las herramientas o la implementación de la solución planteada.
- Soporte: cuál es el respaldo que tienen las herramientas elegidas.
- Coste: cuánto cuesta adoptar la solución planteada. Aquí se tienen en cuenta tanto los costos de licencia como los tiempos de desarrollo.
- Escalabilidad: Capacidad de aumento del número de dispositivos, conexiones y usuarios de la solución.
- Otras características dependiendo de cada bloque: interoperabilidad facilidad de integración de otros sistemas, etc.

2.-Gateway

2.1.-Interacción entre el modulo bluetooth y el Gateway

El envío de información del modulo se hace mediante BLE, para poder interactuar con los dispositivos, usaremos el protocolo GATT el cual os permitira conectarnos, leer y enviar información a los dispositivos bluetooth, para hacer esta interacción usando el protocolo GATT se han barajado tres posibilidades.

- Automatizarla consulta a los dispositivos usando los comandos que nos proporciona el stack de bluetooth para linux (DBus y Bluez) que es una capa de abstracción que incluye el protocolo GATT.
- Modificación del programa de desarrollo de st para interactuar con la sensortile box
- Creación de un cliente GATT que interactué directamente con los dispositivos.

De las tres opciones se ha decidido usar la tercera pues en el caso de la primera aunque es la que menos tiempo requiere para ser implementada, es una solución que únicamente seria válida para sistemas Linux que usen el mismo stack bluetooth (el cual además es de actualización frecuente y con cambios sustanciales entre versiones).

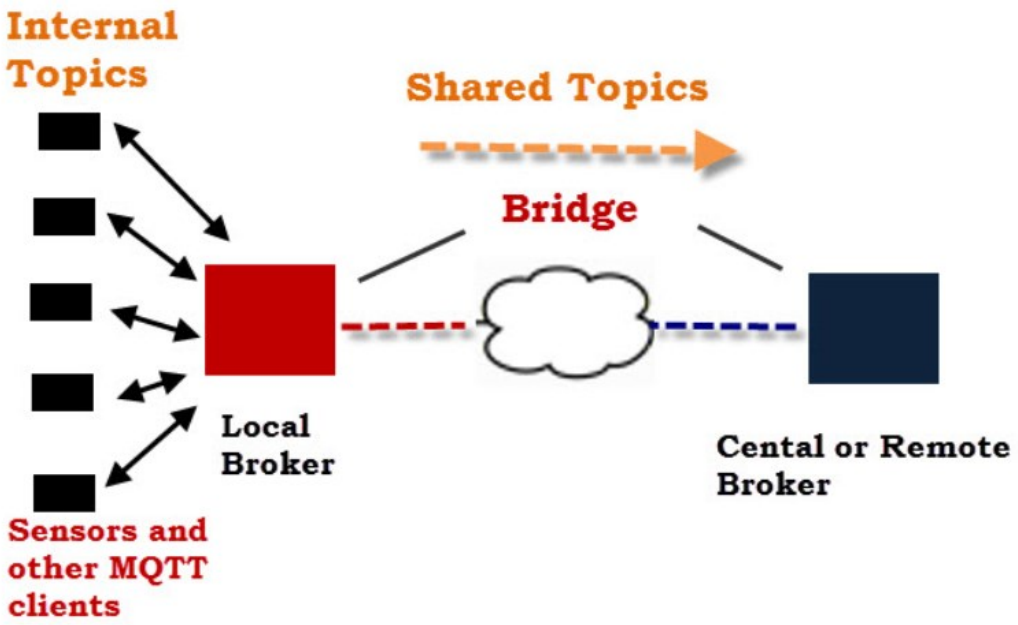
La segunda opción se ha descartado porque, dada la complejidad del código de st, se ha estimado que el tiempo necesario de estudio del código para hacer las adaptaciones pertinentes a nuestra solución muy superior respecto a la última solución.

Así pues finalmente se ha decidido usar la tercera opción, pues, aunque requiere más tiempo que la primera es más probable que funciones en otros sistemas operativos y con otros dispositivos no propios de st. Además al ser creada desde cero se puede adaptar totalmente a nuestras necesidades (numero de dispositivos, numero de características, numero de servicios etc).

2.2- Comunicación MQTT entre la instalación local y el servidor Cloud

La comunicación entre el Gateway y aws se hará utilizando el protocolo MQTT debido a su gran escalabilidad para multitud de dispositivos, facilidad de uso y soporte, para esta comunicación se dispondrá de dos brókers, uno instalado en la maquina en aws y otro en el Gateway, se ha decidido esta configuración porque a diferencia de la alternativa de un único bróker en la nube. Nuestra solución permite que en caso de pérdida de conexión con el servidor de aws, el intercambio de información en nuestra red local continúe funcionando por ejemplo para lanzar una alarma cuando se produzca un paro cardiaco, que será la funcionalidad que le daremos.

Para poder hacer esta conexión y transmitir la información desde nuestro bróker en la red local al bróker en la maquina virtual, hay que construir lo que se denomina un bridge (explicación en la documentación técnica del desarrollo).



ELl dato que se va a enviar usara el formato JSON (JavaScriptObjectNotation) debido a su simpleza y facilidad de uso, además

es un formato de envío de información muy extendido, lo que seguramente nos ayude en el futuro en caso de necesitar hacer integraciones con otros sistemas.

La información que se enviara será la siguiente en función del tipo información que estemos transmitiendo, **prestar especial atención a las mayúsculas y separadores pues son importantes, recordar ademas que en la base de datos todas las columnas empiezan con mayúscula y se separan con "_", pero identificadores dentro de los JSON empiezan con minúscula y cada nueva palabra se indica con una mayúscula.** (Los valores indicados con excepción del tipo de dato son simples ejemplos).

La agrupación de la información de cada tipo de dato responde a la frecuencia de actualización que queremos en la base de datos. Por ejemplo la batería se manda a la vez que la saturación de oxigeno porque la frecuencia de actualización que deseamos de estos dos datos es la misma.

- Información del pulso y saturación de oxigeno del paciente se debe actualizar cada segundo

  {
  "idDispositivo" :FF:FF:FF:FF:FF:FF,
  "tipoDato": "pulso",
  "pulso": 69,
  "o2":98,
  "tiempo": '2012-06-18 10:34:51'
  }
- Información de la batería del dispositivo, se debe actualiar cada 10 minutos

  {
  "idDispositivo" : FF:FF:FF:FF:FF:FF,
  "tipoDato" : bateria,
  "batería": 95,
  "tiempo": '2012-06-18 10:34:51'
  }
- Información de una anomalía, se actualiza cuando se detecte

  {
  "idDispositivo" : FF:FF:FF:FF:FF:FF,
  "tipoDato" : "anomalía",
  "infarto": 1,
  "anomalía": 1,
  "tiempo": '2012-06-18 10:34:51'

  }

2.3.-Otros requisitos en el Gateway

Se deberá establecer un sistema de comunicación de alarmas tanto acústico como visual y que pueda operar sin la necesidad de que ese conectado el servidor cloud, sistema se activara cuando se detecte que se ha producido un infarto, es decir cuando el pulso sea igual a cero.

3.-Servidor Cloud en AWS

3.1-Interacción entre el bróker MQTT y la base de datos

Para guardar la información recibida en la base de datos se desarrollara un script, para desarrollarlo se ha barajado la posibilidad de realizarlo en Python y en Nodejs, pero debido a la dificultad de debug de Nodejs se ha decidido el uso de Python.

El script debe cumplir los siguientes criterios.

- Recibir la información en formato JSON como se ha explicado anteriormente
- Lectura de los parámetros de conexión a la base de datos y al bróker MQTT desde un archivo externo que se encontrara en el mismo directorio que el script con el nombre config.txt, debido al funcionamiento de la lectura de ficheros en python se ha llegado a la conclusión de que en el archivo debe figurar la información de la siguiente manera.

```
[DB]
user= root
password= root
host= 127.0.0.1
port= 3306
database= pulsioximetrodb


[BROKER]
broker_address = 192.168.250.59
borker_port = 1883
topic = #
broker_username= user1
broker_password= pass1
```

- Escritura en una base de datos MariaDB.

3.2.-Envio de información del servidor a la aplicación final

Debido a que el acceso a la base de datos está permitido por cuestiones de seguridad exclusivamente al localhost. Se tiene que desarrollarse un sistema de acceso a esta de forma indirecta por parte de un cliente externo.

Así pues se ha decidido establecer un socket de comunicación tcp entre el servidor, que hará la consulta a la base de datos y el cliente, que

será aquel que ejecute la aplicación final, esta comunicación debe cumplir los siguientes requisitos.

- Usar un puerto que este abierto de forma normal (por ejemplo el puerto 80) y no requiera su apertura por parte del cliente
- El cliente inicia la conversación, indicando dispositivo, usuario o conjunto de datos que desea.
- La información se envía y recibe en formato JSON.

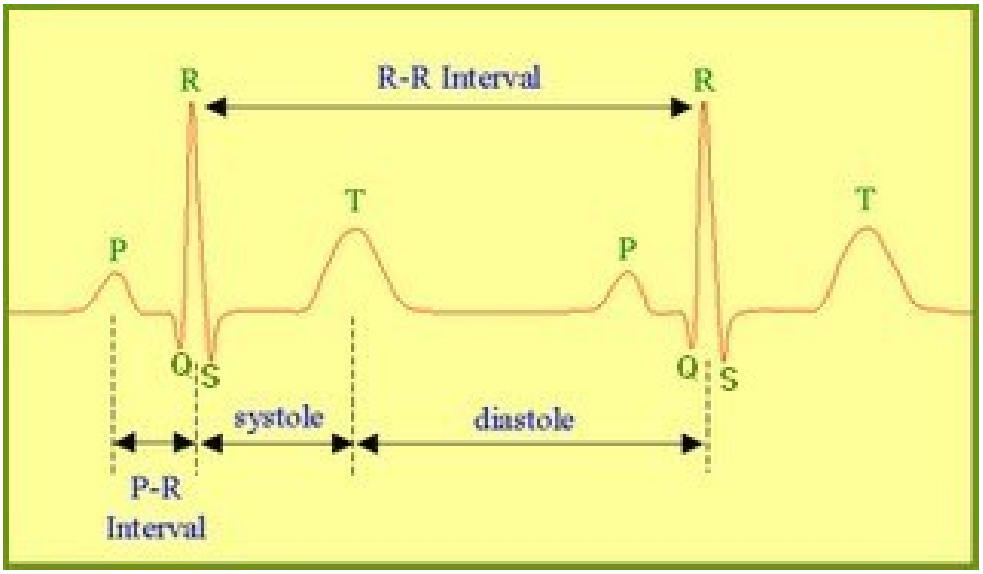
4.-Modulo de sensores con emisor BLE

4.1.-recepcion de los datos del sensor MAX30102

Dada la dificultad para integrar una conexión i2c en el dispositivo Sensortilebox de St se ha decidido hacer un prototipo con un esp32 para demostrar que el resto del sistema, desde el sensor hasta la aplicación final funciona correctamente.

Se han barajado dos opciones a la hora de establecer esta comunicación.

Usar exclusivamente la librería wire.h (la propia para el protocolo i2c) y establecer de forma manual el sistema de recepción de datos desde el sensor. Si bien este sistema nos da más opciones, pues nos da la posibilidad de recibir una señal digital puy similar a la señal analógica que detecta el sensor, lo que nos abre un abanico de opciones, para por ejemplo detectar, además del pulso, fibrilaciones o anomalías en el ritmo cardiaco observando el posicionamiento de los distintos puntos en esta señal, que sería parecida a la siguiente.

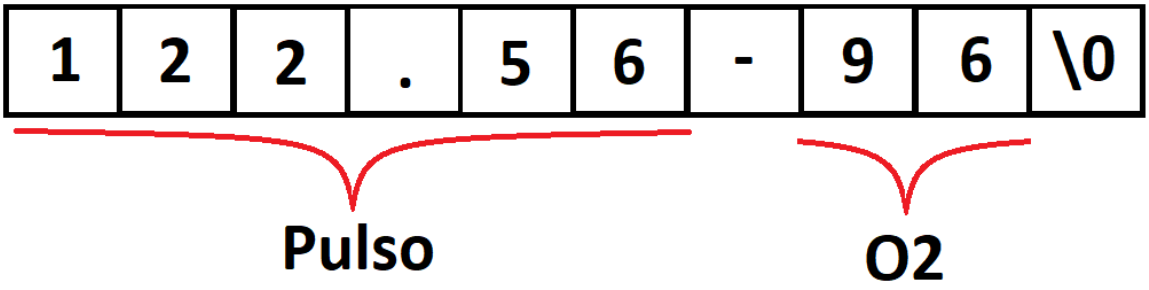


La otra opción es usar la librería MAX30100.h preparada para el propio sensor, esta a diferencia de la anterior opción, solo nos da numéricamente los valores de latidos por minuto y saturación de oxigeno en sangre, pero debido al escaso tiempo de desarrollo del que se dispone se ha decidido esta opción pues el coste de tiempo es mucho menor.

4.2.-Envio de los datos del sensor

Dado que el sistema establecido en la raspberry para la recepción de datos y su posterior envío por MQTT se ha hecho para recibir los datos usando BLE (Bluetooth Low Energy) se va a crear un servidor BLE que permita suscribirse a el por parte de un cliente (raspberry pi). La trama de datos que se enviará, está formada por un array de char y tiene la siguiente estructura.

- Entre 4 y 6 bytes para el valor del pulso (1 o 3 para las unidades decenas y centenas, uno para el punto decimal y dos para los decimales).
- 1 byte para indicar dónde acaba el dato del pulso
- Entre uno y dos bytes para el valor de la saturación de oxigeno
- 1 byte '\0' para indicar el final de la trama.



5.-APP Móvil

Dado que se trata de una aplicación enfocada a la salud se ha decidido que el medio más adecuado para ver los datos es en una aplicación móvil, para adaptarse al mercado actual en el que la gran mayoría de las aplicaciones relacionadas con la salud muestran su información a través de aplicaciones móviles.