Universitat Autònoma de Barcelona

TEMP&CAP

Autor

Angel García Calleja 1490917

Compañero

Youssef Assbaghi Asbahi 1493477

Índice

Introducción

Como en esta asignatura realizamos 2 prácticas, quisimos realizar una práctica que se pareciera bastante a la que estamos desarrollando más teóricamente.

Temp&Cap es un sistema que registra la temperatura, humedad y personas que entran o salen de un local. La utilidad de este sistema está en que tenemos los datos a tiempo real de la temperatura y humedad, lo cual nos genera el valor de sensación térmica y así decidir si hace falta poner el aire acondicionado o la calefacción, además obtenemos el valor del número de personas que entran y salen del establecimiento, lo que nos permite obtener el aforo del lugar y así compararlo con el aforo máximo permitido.

La realización de esta práctica consistía en 4 partes:

1. Generar el código necesario de la placa que se nos dio el primer día para que enviara los valores mencionados con anterioridad al Edge.
2. Generar el código necesario para que el Edge se conectara a la placa, recibiera esos datos, los enviara al Cloud y en nuestro caso los enseñara a través de la aplicación.
3. Generar el código necesario para que el Cloud recibiera los datos del Edge, los procesara y los guardara en la base de datos.
4. Entrenar un modelo para que dados unos valores de entrada nos hiciera una predicción de la sensación térmica en nuestro caso.

La práctica la fuimos haciendo siguiendo ese orden de puntos, primero código de placa, después la conexión entre Device y Edge, recibir los datos, ir desarrollando la aplicación para que mostrara esos datos, enviar los datos recibidos al Cloud, procesar esos datos y enviarlos de vuelta al Edge para que se mostraran.

A raíz de una tutoría con el profesor para ver si nos estábamos dejando alguna parte, tuvimos que añadir tanto la base de datos como el modelo de predicción de la sensación térmica.

Desarrollo

A continuación, se explicará de cada parte las funcionalidades que se han codificado.

Device

El código de la parte del Device se hizo con Visual Studio Code y C++, utilizando las librería de Arduino y BLEPeripheral para poder realizar la conexión mediante BLE.

Primero creamos un servicio y una característica, que es donde irán nuestros valores.

Luego seteamos el dispositivo con el nombre de Temp&Cap y le añadimos ese servicio y característica como atributos.

Finalmente, entramos a un bucle que enviará datos sin parar si el dispositivo está conectado.

Al no tener los sensores con los que obtener estos valores, hemos considerado apropiado el randomizar todos los dígitos cada vez que se envíen los datos.

Los datos se encuentran en un único array de 7 chars con las siguientes características:

- Los dígitos de signo son valores decimales entre 43 y 45. Aquí obtenemos un +, una , y un -. Consideramos apropiado que la coma también significara valor positivo, ya que es más normal obtener temperaturas positivas.
- Los dígitos de números son valores decimales entre 48 y 57, donde obtenemos valores de 0 a 9.
- Los 3 primeros dígitos del array corresponden a la temperatura, el primero es el signo y los dos siguientes los números.
- Los 2 siguientes dígitos del array corresponden a las personas. Consta de un signo para saber si son las que han entrado o han salido y de un número ya que consideramos que normalmente no entran ni salen más de 9 personas a la vez.
- Los últimos 2 dígitos corresponden a la humedad, ya que, al ser un tanto por ciento, siempre la consideraremos positiva, por lo que solamente necesitamos 2 números.

```
void loop()
{
  BLECentral central = ledPeripheral.central();
  unsigned char dataStream2[] = {'+','2','3','-','2','6','9'};
  if (central) {
    while (central.connected()) {

      dataStream2[0] = random(43,46); // sigo positivo o negativo tmeperatura
      dataStream2[1] = random(48,53); // 0-4
      dataStream2[2] = random(48,58); // 0-9
      dataStream2[3] = random(43,46); // singo personas
      dataStream2[4] = random(48,58);// 0-9
      dataStream2[5] = random(48,58); // 0-9 humedad
      dataStream2[6] = random(48,58); // 0-9

      valores.setValue(dataStream2,7);



    }
  }
}
```

Edge

La parte del Edge he decidido dividirla en Edge y App, para poder explicarlo siguiendo el flujo de la información. El código se ha realizado mediante Android Studio y el lenguaje de programación propio de Flutter, dart.

Conexión con el Device

Primero buscamos en Internet aplicaciones similares a la nuestra para tener una base con la que trabajar (a la larga nos dimos cuenta de que sería mejor si la hubiéramos hecho desde 0 nosotros).

Primero tenemos una pantalla donde se muestran todos los dispositivos Bluetooth a los que nos podemos conectar.

Una vez seleccionemos el dispositivo nos aparecerá la siguiente pantalla, la de datos recibidos, donde tendremos que establecer conexión. Al presionar sobre el botón de conectar, iremos al onReadPressed, función que al principio nos dio bastantes problemas. En esta función se crea el servicio, y dentro de la misma añadimos el código para obtener la información referente al dispositivo, como el nombre y los datos. Aquí utilizamos el utf8.decode() para obtener los valores en decimal que nos enviaba el Device y los guardábamos en sus variables correspondientes por separado.

Pantalla de datos recibidos

Al realizar la conexión con el dispositivo, aparece esta pantalla (no exactamente esta, pero como ya no tenemos la placa lo hemos improvisado un poco):

Primero añadimos el botón para comenzar o parar de recibir datos. Si estamos recibiendo datos, se muestran en el ampliable de debajo.

Lo siguiente es el apartado de medias, en el cual se muestran las medias actualizadas de los valores que se están recibiendo (enseñamos por pantalla lo que nos devuelve el Cloud).

También añadimos la parte de aforo máximo, donde una vez establecido el valor, si el aforo actual lo sobrepasa, este se pondrá en rojo, si está por debajo se muestra en verde.

Los apartados de sensación térmica y estadísticas históricas se explican en el apartado de App más abajo.

Envío de datos al Cloud

En el momento en el que presionemos el botón de recibir datos, comenzaremos a recibir datos de la placa (hemos puesto que recibas datos cada 5s). Una vez tenemos los datos actuales, hacemos una petición al servidor llamada addDatos.

Para realizar esta petición primero hemos de setear nuestra url de conexión, que será del tipo http://direcciónIP:8080. Una vez tenemos la URL base, podemos realizar la petición, que se realizará pasándole al servidor los datos de temperatura, personas, humedad y el id del dispositivo conectado. La petición quedaría tal que: $baseUrl/get_ble?$temperatura?$personas?$humedad?$id".

Cloud

Para la parte del Cloud, decidimos hacer un WebServer en Java con la herramienta IntelliJ, ya que lo habíamos usado en otras asignaturas y ya sabíamos que partes teníamos que tocar.

Usaremos 3 clases: Plantilla para todo lo relacionado con la base de datos, WebServer para temas de sockets y peticiones y BLE_Nordic para temas de datos recibidos y modelo de predicción.

Como antes, he decidido separar la conexión en este caso en dos bloques, para poder estructurarlo mejor.

Conexión con el Edge

Al iniciar el servidor, se creará un serverSocket donde comenzaremos a escuchar peticiones. Para recibir datos del Edge, utilizaremos la petición get_ble donde actualizaremos los valores de nuestra variable de clase BLE_Nordic a los valores recibidos del Edge, calcularemos las medias y llamaremos al modelo de predicción. Finalmente, esos datos se insertarán en la base de datos en si id correspondiente.

Base de datos

Creamos una base de datos en MongoDB, herramienta que ya habíamos usados en asignaturas anteriores. En esta base de datos se encuentran documentos (lo llamaré datos). Estos datos tienen los siguientes campos:

_id es un id propio de mongo.

Id del dispositivo

Fecha del dato

Valores de temperatura, aforo y humedad registrados en ese momento.

Para llenar la base de datos creamos una petición llamada insert, la cual llama a la función de la Plantilla insertDatos. Aquí creamos datos de todo 2020 y 2021 menos diciembre. Dichos datos siguen el patrón inicial de ser aleatorios. Para los datos de diciembre, simplemente creamos documentos hasta el día 11, ya que esto lo hicimos el día 12. Una vez hecho esto podíamos recoger datos históricos con consultas, y también podíamos ver como los nuevos datos se insertaban correctamente.

Modelo entrenado

Esta parte, como está explicado en el apartado de repartición del trabajo, no la he tocado.

En este punto nos repartimos la faena ya que no teníamos tiempo y yo me puse a hacer el tema de base de datos y consultas y él hizo el modelo ya que en su TFG estaba haciendo cosas similares y sabía cómo hacerlo.

Conexión con la App

Una vez ya tenemos todos los datos insertados en la base de datos y el modelo entrenado podemos pasar a ver como se enviaban los datos desde la base de datos al servidor mediante consultas y luego a la App mediante peticiones.

Cuando nos llegaban datos desde la petición get_ble, una vez ya lo tenemos todo asignado se crea un fichero llamado datos.json con los datos de las medias, si habíamos superado el aforo y la predicción de si la sensación térmica estaba por encima o por debajo de la temperatura enviada.

También tenemos el caso en que se actualice el aforo máximo mediante un campo a rellenar en la aplicación. Aquí llamaríamos a la función de comprobar si se supera este nuevo aforo máximo, obviamente guardándonos ese valor.

Para el tema de las estadísticas históricas tenemos dos casos, las históricas y las de un día en concreto, ambas funcionan de la misma manera solo que añadimos un filtro para que solo nos aparezcan los datos del día indicado.

Las estadísticas que recogemos y tenemos consulta para recogerlas son: temperatura, aforo y humedad mínima y máximas.

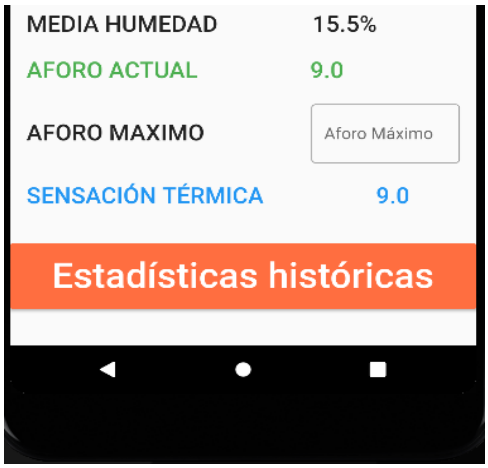
Para el tema de medias tenemos que realizar nosotros la media con los valores obtenidos dividido entre la cantidad de valores.

Una vez tenemos estos 9 valores, creamos un objeto JsonObject para mandárselas a la aplicación.

App

Parte de las primeras pantallas de la aplicación ya ha sido explicada más arriba, ahora se explicarán las partes donde dependíamos de la base de datos y del Cloud.

Lo primero es el aforo máximo, donde ya hemos visto que mandamos al servidor el nuevo valor par que nos devuelva si el aforo actual supera o no el aforo máximo y mostrarnos el texto del aforo actual de un color u otro.



Lo siguiente es el texto de la sensación térmica. Aquí, al mandar los datos al servidor, se llamaría a la función que nos devuelve la predicción sobre si la sensación térmica es de más frío o calor.

Y por último nos encontramos con el botón de las estadísticas históricas. Al presionarlo nos llevará a la siguiente pantalla.

Al presionar el botón, se realizará la petición de getHistoricos, la cual nos devuelve el Json con los valores obtenidos. Simplemente los descodificamos y los mostramos por pantalla.

También aparece el botón de “Datos de un solo día” donde si se presiona, te aparece un calendario. Una vez introducida la fecha deseada, se realizará el mismo funcionamiento que antes, se llamará a la petición getHistoricos_dia, descodificamos el Json obtenido y mostramos los datos por pantalla.

Repartición del trabajo

Hay que decir que, al tener solo una placa, se complicaban las cosas ya que los códigos no podían ser iguales ni podíamos trabajar los dos a la vez. Si que ha habido partes donde los dos estábamos mirando la misma pantalla compartiéndola por Discord, pero los dos intentábamos solucionar los problemas, no estaba uno mirando y el otro trabajando.

Para poder trabajar los dos simultáneamente en estas partes, me he tenido que crear un proyecto a parte donde digamos que hardcodeaba los datos. Una vez lo que estuviera haciendo yo (pantallas de flutter, conversión de datos, asignación de variables, etc...) funcionaba, le pasaba el código a mi compañero y nos poníamos a adaptarlo para que funcionara correctamente en el proyecto bueno.

El Cloud si que por temas de tiempo nos lo repartimos mucho más, ya que si los dos trabajábamos en los mismo no llegábamos. El tema de base de datos y consultas lo hice exclusivamente yo y el tema de entrenar el modelo lo hizo exclusivamente él.

La parte del Device si que lo hicimos compartiendo pantalla en Discord ya que solo hay una placa. Esta parte no lo entendía yo mucho y el no ver que estaba pasando no ayudaba así que esta parte si que la hizo más él.

El Edge, parte más extensa, si que podríamos decir que los dos hemos aportado más o menos lo mismo, o hacíamos lo mismo y ayudábamos al otro o nos repartíamos otras cosas y luego las poníamos en común.

Valoración del grupo

Como valoración grupal, dado el cómo hemos trabajado diría más bien un 60% Youssef y un 40% yo, pero me gustaría pensar que ha sido por lo de no poder hacer los dos lo mismo, ya que en otras asignaturas solemos trabajar de la misma manera y nunca hemos tenido problemas.

Valoración asignatura

En cuanto a la asignatura, la verdad, no me ha gustado, aquí los motivos.

El contenido no ha sido lo que me esperaba (eso ya es cosa mía), además, al no haber exámenes y solo hacer 2 prácticas, no tenía mucho sentido ir a las clases.

Entiendo que la caída del Campus Virtual no ayuda para nada a la situación, pero ha habido un desorden que no había visto en ninguna asignatura previa.

Primero, en las primeras presentaciones de la “otra práctica” (en esta tampoco ha habido mucho problema), a mi personalmente me molestaba mucho que los profesores interrumpieran las presentaciones que estaba haciendo la gente, pero a la vez se entiende el motivo. Muchos de nosotros no sabíamos que se nos estaba pidiendo presentar esa semana, de ahí que dijéramos cosas que no debíamos y los profesores se pasaran toda la puta presentación diciendo “No, no esto no me lo digáis” y cosas así. Finalmente, más o menos se solucionó porque en la explicación de la entrega de la semana se empezó a poner en rojo lo que NO teníamos que incluir.

También quiero añadir que el paso de sesiones presenciales a online no se hizo del todo bien. Hay que decir que es más cómodo presentar online, pero no se puede poner un mensaje en el Teams a las 11 de la noche diciendo si la clase es online o presencial.

El hecho de que uno de los compañeros del grupo deje la asignatura y nos quedemos solo 2 haciendo el trabajo de 3 tampoco nos hizo mucha gracia, aunque tampoco nos importó.

Muchos problemas de estos quiero pensar que no hubieran ocurrido si no pasa lo del Campus Virtual, así que eso, lo que peor me ha parecido de la asignatura es la organización.