INTRODUCCIÓN

Gesinen App, es una aplicación diseñada con el fin de poder comunicarse a través de una conexión bluetooth con los sensores que dispone esta empresa, además de poder acceder a la plataforma web y Gateway.

FUNCIONAMIENTO

1. LOGIN

Temporalmente deshabilitado

2. MENÚ

Una vez accedemos con los datos correctos en la parte de Login, nos encontramos con una pantalla con tres botones. El primero es 'SENSORES', este te redirige a la sección de la app donde se van a realizar las conexiones bluetooth con los sensores. El segundo es 'GATEWAY', y tercero es 'PLATFORM' .(Estos dos botones están deshabilitados temporalmente).

La primera pantalla que encontramos después del login, es un menú con 3 botones: SENSORES, GATEWAY y PLATAFORMA. (Gateway y plataforma se encuentran temporalmente deshabilitados).

Ilustración 1 - Layout Menú

3. SENSORES

Esta sección es la encargada de establecer la conexión Bluetooth y de configurar genéricamente los sensores a los que se conecte. El layout que podemos ver cuando accedemos a este apartado es el siguiente:

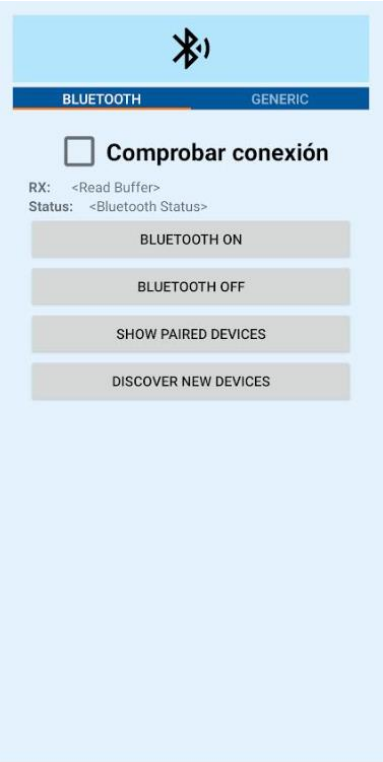


Ilustración 2 - Tab Bluetooth

Ilustración 3 - Tab Generic

Como se puede observar esta sección está compuesta por dos tabs. El primero se llama 'Bluetooth', donde se van a realizar las conexiones bluetooth con los sensores. Y el segundo, se llama 'Generic', donde se podrán configurar los datos más básicos que comparten todos los sensores.

Para crear la conexión bluetooth:

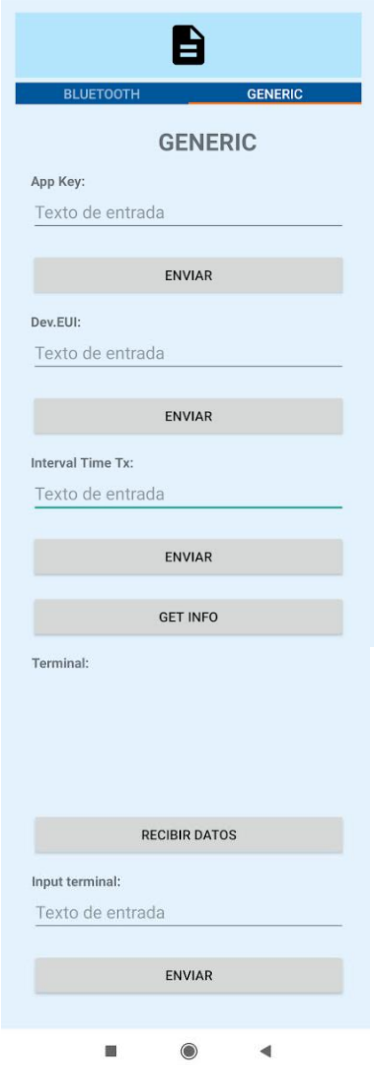
- Primero debemos de comprobar que el bluetooth está activado. Para ello se dispone de dos botones: **'BLUETOOTH ON'** y '**BLUETOOTH OFF'.**

- Una vez activado, podemos buscar los sensores con:
-'SHOW PAREID DEVICES': este botón permite listar los dispositivos emparejados con tu teléfono móvil (para ello desde los ajustes del teléfono, dar los permisos necesarios para que la app obtenga estos datos)
- '**DISCOVER NEW DEVICES'**: el funcionamiento de este botón es buscar los dispositivos cercanos con bluetooth.

- Cuando hayamos hecho la búsqueda del sensor al que nos queremos conectar, debemos pulsar sobre su nombre y esperar unos segundos para que cree la conexión.

- Para comprobar que la conexión ha sido establecida, saldrá un mensaje a continuación de **‘Status’** *: Connected to Device: (Nombre del sensor)*. De lo contrario pondrá : *Conexión Failed.*
- Si se ha hecho correctamente la conexión, también podemos comprobar que todo funciona bien con la CheckBox llamada ‘**Comprobar conexión’**, ya que su función es enviar un mensaje, en este caso ‘Carla’, al terminal del sensor.
- Y por último en esta pantalla tenemos ‘**RX’**, TextView que nos muestra los mensajes enviados por el sensor.

Una vez realizada la conexión bluetooth, pasamos a GENERIC. En esta sección se pueden extraer datos del sensor, o cambiar dichos valores por otros, es decir:



- Si pulsamos el botón de **‘GET INFO’**, obtenemos los valores de APPKEY, DEVEUI y TIME TX del sensor.

- Si queremos cambiar el valor en memoria de la APPKEY, DEVEUI o TIME TX, solo hay que escribir en el texto de entrada de cada uno los nuevos valores y pulsar su respectivo botón de enviar. Una vez enviado, podemos volver a pulsar el botón de ‘**GET INFO’,** y nos sacará los valores actualizados.

- El texto de **‘Input terminal’,** es una entrada de texto que sirve para enviar mensajes al terminal del sensor.

- Y por último el terminal, es una bandeja de mensajes, donde cada vez que pulsamos el botón ‘RECIBIR DATOS’, nos mostrará el mensaje que estamos recibiendo del sensor.

3.1 FUNCIONAMIENTO DEL CÓDIGO.

En este apartado se va a explicar como es el funcionamiento de emisión y recepción de mensajes.

Una vez se ha establecido la conexión bluetooth con el sensor, cuando cambiamos del tab de 'Bluetooth' al de 'Generic', se envia un mensaje desde la app al sensor de tipo A000 en formato base64 (oAA=).

- A0 : cabecera de información
- 00 : cabecera set_data

Cuando el sensor recibe este mensaje realiza la siguientes funciones:

```
void enroute(String arrivedMsg){
  scheduleRXMessageGeneric(arrivedMsg);
  //scheduleRXMessageDevice(arrivedMsg);
  scheduleRXMessageUpdate(arrivedMsg);
  uptadeTimeTx(arrivedMsg);
}
void loop() {
  if (Serial.available()) {
    SerialBT.write(Serial.read());
  }
  if (SerialBT.available()) {
    String arrivedMessage  = SerialBT.readString();
    Serial.println(arrivedMessage);
    enroute(arrivedMessage);
  }
  delay(20);
}
```

```
void scheduleRXMessageGeneric(String s){
  if( s.startsWith("oAA=")){
   infoGeneric();
  }
}
```

```
void infoGeneric(){
    SerialBT.print("A0_");
    String appkey=bytesToString(APPKEY_DEF, sizeof(APPKEY_DEF));
    SerialBT.print(appkey);
    SerialBT.print("_");
    String deveui=bytesToString(DEVEUI_DEF, sizeof(DEVEUI_DEF));
    SerialBT.print(deveui);
    SerialBT.print("_");
    SerialBT.print(TIME_TX);
    Serial.println("A0_APPKEY_DEVEUI_TIME");
}
```

Primero recoge el mensaje recibido, comprueba si el mensaje es oAA=, y si lo es, realiza la función que va imprimiendo y enviando los valores de APPKEY,DEVEUI y TIME TX, acompañados de la cabecera A0, así cuando pulsamos el botón de 'GET INFO' , obtenemos los valores de dichas variables.

Para el caso de cuando queremos cambiar los valores de las variables, en GENERIC, lo que hace el código es:

- Los datos que introducimos deben estar ya en Hexadecimal, sino el dato no será válido. (Menos en Time TX que se trata de un int).
- Cuando enviamos el dato, el sensor lo recibe en base64, ya que antes de ser enviado se hace la conversión.
- Cuando el sensor recibe el mensaje, pasa por una función que hace la conversión de base64 a hexadecimal y comprueba por que empieza:
  -Cuando queremos actualizar el valor de APPKEY, nosotros introducimos el nuevo valor en la app y el sensor recibe A1+(nuevo valor de appkey) en base64. Cuando el sensor

comprueba que empieza por *A1*, realiza la función para actualizar en memoria el valor de APPKEY. Después de la actualización, vuelve a hacer la función de infoGeneric para actualizar el mensaje del botón **'GET INFO'**.

```
void scheduleRXMessageUpdate(String arrivedMsg){
    const char* cs = arrivedMsg.c_str();
    char hex_str[100];
    char* messageHex = base64_to_hex(cs, hex_str);
    std::string str(messageHex);
    Serial.println(messageHex);
    if (str.find("A1") == 0) {
      updateAppKey(str.substr(2));
      infoGeneric();
    }
    if (str.find("A2") == 0) {
      updateDeveui(str.substr(2));
      infoGeneric();
    }

}
```

```
void updateAppKey( std::string hex_str){
  std::vector<uint8_t> bytes;
    for (size_t i = 0; i < hex_str.length(); i += 2) {
        std::string byte_str = hex_str.substr(i, 2);
        uint8_t byte = std::stoi(byte_str, nullptr, 16);
        bytes.push_back(byte);
    }
   std::cout << "Bytes:";
    for (uint8_t byte : bytes) {
        std::cout << " " << std::hex << static_cast<int>(byte);
    }
    std::cout << std::endl;
 std::memcpy(&APPKEY_DEF[0], &bytes[0], sizeof(APPKEY_DEF));
   std::cout << "APPKEY_DEF = ";
     for (int i = 0; i < 16; i++) {
         std::cout << std::hex << static_cast<int>(APPKEY_DEF[i]) << " ";
     }
  std::cout << std::endl;

}
```

- Para actualizar el valor de DEVEUI, es el mismo procedimiento que en APPKEY, pero con la cabecera *A2.*

```
void updateDeveui( std::string hex_str){
   std::vector<uint8_t> bytes;
    for (size_t i = 0; i < hex_str.length(); i += 2) {
        std::string byte_str = hex_str.substr(i, 2);
        uint8_t byte = std::stoi(byte_str, nullptr, 16);
        bytes.push_back(byte);
    }
   std::cout << "Bytes:";
    for (uint8_t byte : bytes) {
        std::cout << " " << std::hex << static_cast<int>(byte);
    }
    std::cout << std::endl;
  std::memcpy(&DEVEUI_DEF[0], &bytes[0], sizeof(DEVEUI_DEF));
   std::cout << "DEVEUI_DEF = ";
     for (int i = 0; i < 8; i++) {
         std::cout << std::hex << static_cast<int>(DEVEUI_DEF[i]) << " ";
     }
   std::cout << std::endl;

}
```

- Y para TIME TX , tenemos el mismo procedimiento pero sin conversiones y con la cabecera *A3.*

```
void uptadeTimeTx(String arrivedMsg){
   if (arrivedMsg.startsWith("A3")) {
    String intNum = arrivedMsg.substring(2);
    if (intNum.toInt() != 0) {  // comprueba que el número es válido
      TIME_TX = intNum.toInt();
      infoGeneric();
    }
  }
```

-

4. GATEWAY

Temporalmente deshabilitado.

5. PLATAFORMA

Temporalmente deshabilitado