overapp

NODES – Nord Ovest Digitale e Sostenibile

PROTOTIPO BEACON: VERSIONE LABORATORIO [BITE]

SPOKE 3 – Industria del turismo e cultura

DELIVERABLE D 1.3

Version history

No.	Date	Details	Author(s)
0.1			
0.5	Jul 1, 2024	Initial Draft	Emanuele Mastrogirolamo OverApp srl
0.9	Jul 17, 2024	Initial Draft	Lavinia Bertuzzi OverApp srl
1	Jul 20, 2024	First Version	Sandro Tola OverApp srl

This document is part of the project NODES which has received funding from the MUR – Missione 4, Componente 2, Investimento 1.5 – Creazione e rafforzamento di "Ecosistemi dell'innovazione", costruzione di "leader territoriali di R&S" – del PNRR with grant agreement no. ECS00000036

overapp

Contents

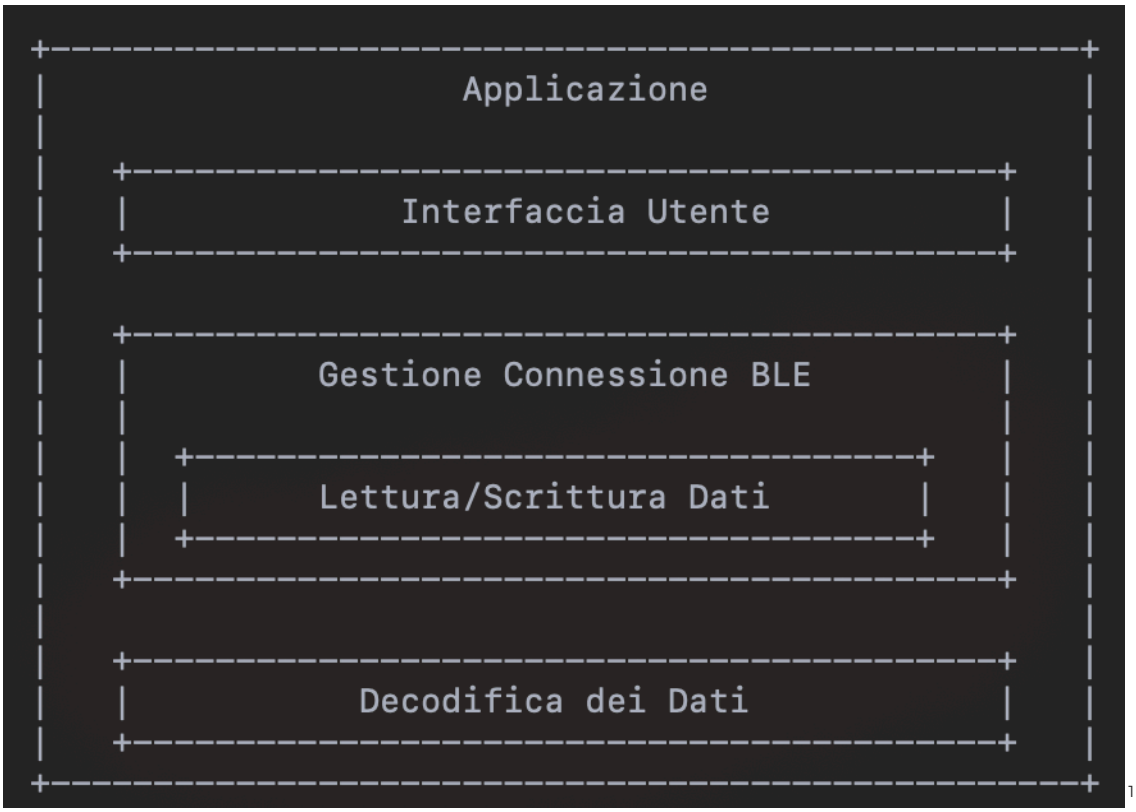
Glossary

	Definition
Hub Coordinator (HC)	The Hub Coordinator represents the single point of contact for the implementation of the innovation ecosystem towards the MUR. It carries out the management and coordination activities of the innovation ecosystem, receives the fundings, verifies, and transmits to the MUR the reporting of the activities carried out by the Spoke and their affiliates.
National Recovery and Resilience Plan (NRRP)	This document uses the Italian acronym for the NRRP, which is PNRR (Piano Nazionale della Ripresa e Resilienza)
Research Program Manager	The person who will be the responsible for the overall scientific contents of the NODES project. The NODES will appoint the Research Program Manager. It refers to "Responsabile del Programma di Ricerca" in the MUR's Call of proposal for "Ecosistemi di Innovazione"
NODES' Research and innovation program	NODES' Research and Innovation program is articulated in specific programs for each Spoke, with the aim to promote and support applied research on topics consistent with the Intelligent Specialization Strategy, with the guidelines of the 2021-2027 partnership agreement scheme, with regional operational plans and regional and national research and innovation priorities. Although NODES' Spokes are concentrated on different themes, they will organize their activities and actions within a common framework – NODES' Booster Methodoloy
Spoke Coordinator	The University in charge of coordinating the Spoke's ecosystem. It refers to "Spoke" in the MUR's Call of proposal for "Ecosistemi di Innovazione"
Spoke Data Manager	The person who will be the responsible for the monitoring and management of data generated at the Spoke level. The Spoke Coordinator will appoint the Spoke Data Manager.
Spoke Partner	The entity associated to the Spoke Coordinator. It can be an Innovation Cluster, Competence Center, Research Center related to the Spoke's ecosystem and contributes to achieve objectives and impact under the Spoke' leadership and management. It refers to "soggetti affiliati" in the MUR's Call of proposal for "Ecosistemi di Innovazione".
Spoke Project manager	The person who will be the responsible for the management, coordination and progress of the project at the Spoke level. The Spoke Coordinator will appoint the Spoke Project Manager.
Spoke research and innovation program	NODES' Research and Innovation program is articulated in specific programs for each Spokes. The spoke will leverage a consolidated collaboration with leading private and public companies and will focus the applied research activity on technological domains and applications that can favour the integration of SMEs into new value chains.
Spoke Scientific and Technical Manager	The person who will be the responsible for the overall scientific contents of the project at the Spoke level. The Spoke Coordinator will appoint the Spoke Scientific and Technical Manager.
Spoke Stakeholders Committee (SC)	Consultation structure formed by relevant stakeholders (Government, universities, companies, civil society, third sector, etc.)
Spoke Thematic	General target focus and domain of the Spoke research.
Spoke Topics	Specific areas/lines of development within the Spoke.
Spoke Work Package Leader	At the Spoke level, Work Packages (WPs) will be organized by WP leaders, who will be responsible for performance evaluation and reporting.
Flagship Project	Main research project at the Spoke level with the goal of prototyping, testing, demonstrating the research activities towards higher TRLs.

A) Introduzione

Questo documento descrive come un'applicazione mobile può leggere i dati dai sensor beacon Teltonika EYE utilizzando la tecnologia Bluetooth Low Energy (BLE).
Di seguito verranno illustrate le fasi di connessione ai sensori, il meccanismo di autenticazione, necessario per la lettura dei dati rilevati dai sensori e la decodifica di questi ultimi.
Il documento è concepito per essere agnostico rispetto alla piattaforma, quindi le tecniche descritte possono essere implementate sia su iOS che su Android.

B) Architettura



[1]

Linguaggi di programmazione

Flutter - Dart

Flutter è la tecnologia utilizzata per lo sviluppo della nostra applicazione. È un framework potente e intuitivo sviluppato da Google che offre la possibilitá di sviluppare con una solo linguaggio (Dart) per tutte le piattaforme (iOS, Android e Web).

Librerie utilizzate

- flutter-reactive-ble

Per l'interazione BLE su Flutter, utilizziamo flutter-reactive-ble, una libreria che semplifica la gestione delle scansioni Bluetooth e la lettura dei dati ricevuti.

[1] Figura 1: Architettura dell'applicativo

Tecnologie coinvolte

Bluetooth Low Energy (BLE)

BLE è una tecnologia di comunicazione wireless progettata per applicazioni che richiedono bassi consumi energetici. È ideale per dispositivi come sensori che trasmettono piccoli pacchetti di dati a intervalli regolari.

Proof of concept

Per dimostrare il funzionamento della nostra applicazione, abbiamo sviluppato un semplice proof of concept che permette di connettersi a un sensore Teltonika EYE, autenticarsi tramite una password e leggere i dati di temperatura e umidità.

Scansione dei dispositivi

La fase di scansione inizia verificando che il Bluetooth del dispositivo sia attivo e pronto. Se il Bluetooth non è disponibile o disattivato, la scansione viene rimandata fino a quando diventa possibile avviarla. Una volta che tutto è pronto, l'applicazione comincia a cercare dispositivi nelle vicinanze, analizzando i segnali trasmessi e filtrandoli in base a criteri come il nome del dispositivo o particolari identificatori.

```dart
// startScan is the method to begin scanning for BLE devices.
 // It takes a callback that will be used to report discovered devices that match criteria.
 // Parameters:
 // - beaconName: optional, if provided, we filter devices by their name.
 // - withServiceUuids: a list of service UUIDs to scan for, defaulting to Eddystone FEAA.
 Future<void> startScan(Function onResult,
     {String? beaconName,
     List<String> withServiceUuids = const [
       'feaa',
       '0000feaa-0000-1000-8000-00805f9b34fb'
     ]}) async {
   _onResultCallback = onResult;

   // If BLE isn't ready, we can't start scanning now.
   // We remember that a scan was requested, so when BLE becomes ready, we start scanning.
   if (!isReadyToScan) {
     BiteLogger().info(
         'Bluetooth is off or not ready. The scan will begin once Bluetooth is turned on.');
     _pendingScan = true;
     return;
   }

   // If we are already scanning, there's no need to start another scan.
   if (_isScanning) {
     BiteLogger().info('A scan is already in progress, ignoring new request.');
     return;
   }

```

Durante il processo, l'app monitora i dispositivi rilevati in tempo reale. Ogni dispositivo viene controllato per evitare duplicati: se è già stato registrato in questa sessione di scansione, viene ignorato. Se invece si tratta di un dispositivo nuovo che risponde ai criteri impostati, i suoi dati vengono elaborati e inviati a una funzione specifica che si occupa di gestirli, ad esempio per mostrarli all'utente o integrarli in altre logiche dell'app.

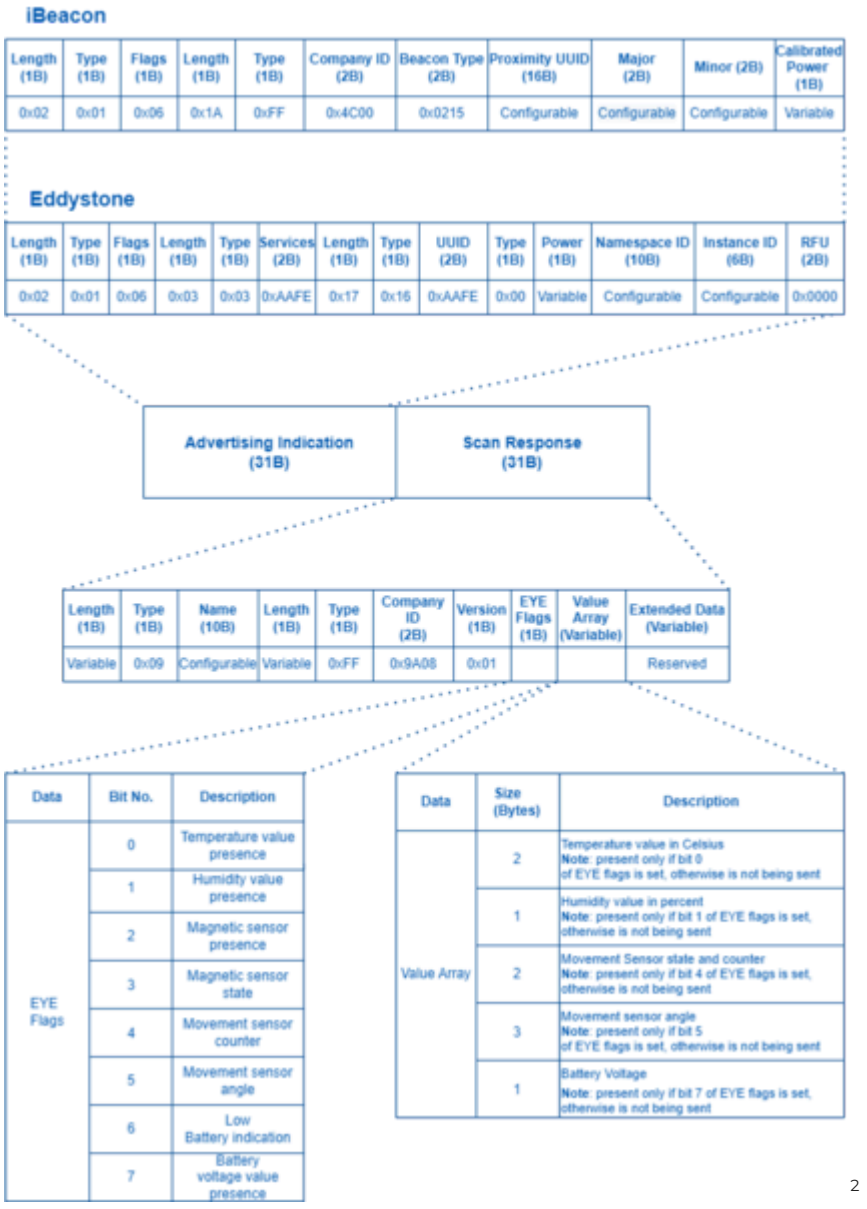
```
_scanSubscription = _ble
          .scanForDevices(
        withServices: withServiceUuids.map((uuid) => Uuid.parse(uuid)).toList(),
      )
          .listen(
        (device) {
          // Each time we discover a device, we check if we've seen it before.
          // If seen, we ignore it. Otherwise, we add it to the set and if the device name matches
          // (or no name filter was given), we pass it to the onResult callback.
          if (_seenDeviceIds.contains(device.id)) {
            // Device already processed this session, skip it.
            return;
          }
          _seenDeviceIds.add(device.id);

          // If beaconName is given, only report devices whose names contain that substring.
          if (beaconName == null || device.name.contains(beaconName)) {
            onResult(device);
          }
        },
        onError: (e) {
          // If an error occurs during scanning, log it.
          BiteLogger().error('Scan encountered an error: $e');
        },
      );
```

La scansione ha una durata predefinita per evitare un utilizzo prolungato che potrebbe consumare inutilmente la batteria del dispositivo. Al termine, la ricerca si interrompe e l'app attende per un intervallo stabilito prima di riavviare automaticamente un nuovo ciclo di scansione, se necessario. Questo permette di bilanciare efficienza energetica e continuità operativa.

Lettura delle caratteristiche e decodifica dei dati

I dati raccolti dai dispositivi rilevati vengono interpretati per estrarre informazioni utili. Il formato dei dati dipende dal dispositivo stesso, nel nostro caso, dipende dalla codifica specifica di Teltonika.



[2]

Possiamo vedere che la prima parte contiene delle informazioni che variano a seconda del protocollo usato, mentre la seconda parte, che comincia dalla trentesima posizione nel protocollo iBeacon e dalla trentunesima nel protocollo Eddystone, è comune per entrambi i protocolli e contiene le informazioni che ci servono, in particolare ci sono due dati che ci interessano: EYE Flags ci permette di rilevare quali sensori sono attivi, Value Array contiene le informazioni rilevate dai sensori.

[2] Figura 2: Schema informazioni restituite dal Beacon una volta rilevato

```dart
// parseBeaconData takes the raw manufacturer or service data (as a Uint8List) from a beacon
// and extracts meaningful information such as temperature and humidity.
// This logic is highly dependent on Teltonika beacon's data format.
Map<String, String?> parseBeaconData(Uint8List data) {
  // Convert the byte data into a human-readable hexadecimal string.
  String hexString =
      data.map((byte) => byte.toRadixString(16).padLeft(2, '0')).join();

  // For this particular beacon, we know that flags and sensor data start at a specific offset.
  // The offset and parsing logic come from the beacon's documentation and prior knowledge.
  int offset = 6; // According to our understanding, flags start at index 6

  // Extract the flags that indicate which sensors are included in the payload.
  String flagsHex = hexString.substring(offset, offset + 2);
  int flags = int.parse(flagsHex, radix: 16);
```

Per prima cosa, i dati vengono convertiti in una rappresentazione esadecimale. Questo consente di lavorare più facilmente con le porzioni di dati specifiche. Una volta ottenuta la stringa esadecimale, si individua la posizione in cui iniziano le informazioni significative, che in questo caso è determinata da un offset predefinito, noto grazie alla documentazione del dispositivo e a una conoscenza approfondita del suo protocollo.
Nel caso specifico del codice, il primo elemento da decodificare è un campo di **flag** che indica quali dati sensoriali sono inclusi nel pacchetto.

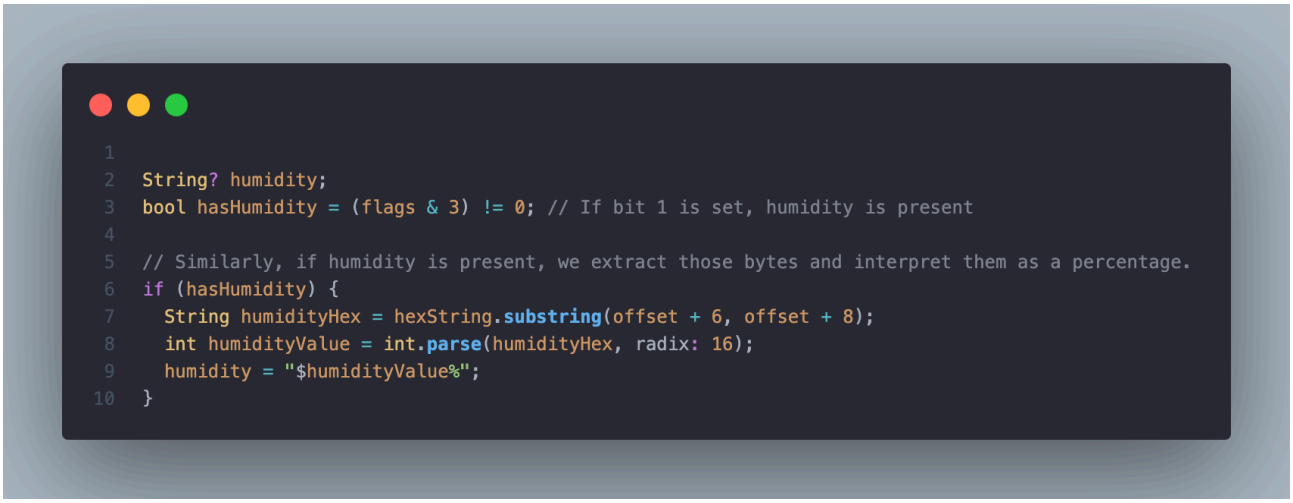
Se il campo di flag indica la presenza di un dato di temperatura, si procede estraendo i byte successivi che rappresentano questo valore. Nel caso specifico, il valore esadecimale corrispondente viene convertito in un numero decimale e poi diviso per un fattore di scala (ad esempio, 100.0) per ottenere il valore reale in gradi Celsius.

```dart

String? temperature;
// Determine if temperature and humidity data are present based on the flags.
bool hasTemperature =
    (flags & 1) != 0; // If bit 0 is set, temperature is present

// If the beacon includes temperature, we extract the relevant bytes and convert them.
// The method of conversion (dividing by 100.0) depends on how the beacon encodes the data.
if (hasTemperature) {
  String tempHex = hexString.substring(offset + 2, offset + 6);
  int tempValue = int.parse(tempHex, radix: 16);
  temperature = "${(tempValue / 100.0).toStringAsFixed(2)}°C";
}
```

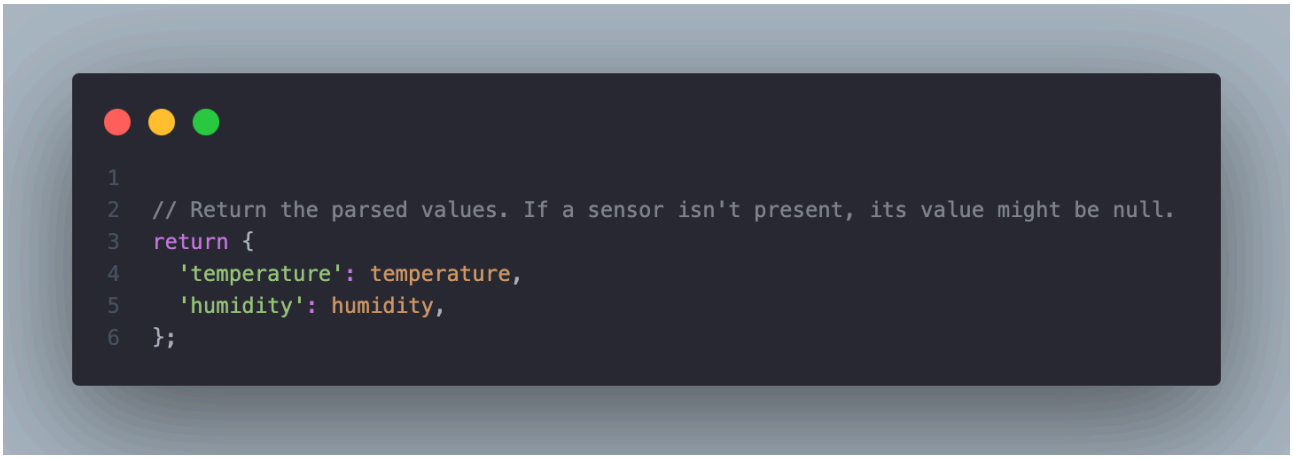
Lo stesso processo viene seguito per altri tipi di dati, come l'umidità. Si estrae la porzione del flusso corrispondente, la si converte in decimale e si applicano eventuali trasformazioni per ottenere un valore leggibile, come una percentuale.



```
String? humidity;
bool hasHumidity = (flags & 3) != 0; // If bit 1 is set, humidity is present

// Similarly, if humidity is present, we extract those bytes and interpret them as a percentage.
if (hasHumidity) {
  String humidityHex = hexString.substring(offset + 6, offset + 8);
  int humidityValue = int.parse(humidityHex, radix: 16);
  humidity = "$humidityValue%";
}
```

Alla fine del processo, tutti i dati decodificati vengono raccolti in una struttura che associa ogni tipo di informazione a un valore leggibile, pronto per essere utilizzato nell'applicazione.



```
// Return the parsed values. If a sensor isn't present, its value might be null.
return {
  'temperature': temperature,
  'humidity': humidity,
};
```

C) Conclusione

Questo documento ha illustrato come configurare un'applicazione mobile per leggere i dati dai sensori Teltonika EYE utilizzando BLE. Seguendo le fasi descritte, è possibile autenticare, leggere e decodificare i dati dei sensori su entrambe le piattaforme iOS e Android.