An Indoor Positioning System using IoT Beacons and Machine Learning

Project Report

Adarsh Shukla
[Department of EEE/RGIPT]

April 27, 2025

Contents

Abstract

Global Navigation Satellite Systems (GNSS) like GPS provide reliable outdoor positioning but fail indoors due to signal blockage. This limitation hinders numerous applications requiring indoor navigation and location-based services. This report proposes an Indoor Positioning System (IPS) leveraging Internet of Things (IoT) beacons, specifically Bluetooth Low Energy (BLE) beacons, and machine learning techniques. The system architecture involves deploying multiple BLE beacons within an indoor environment. A receiver device (e.g., a smartphone) scans for beacon signals and measures the Received Signal Strength Indicator (RSSI). These RSSI values, along with beacon identifiers, form the input features for a machine learning model. The primary proposed method is RSSI fingerprinting, where a model (e.g., k-Nearest Neighbors, Support Vector Machine, or a Neural Network) is trained on a dataset of RSSI vectors collected at known locations (fingerprints). During operation, the model predicts the user's location based on the currently observed RSSI vector. This report details the system design, methodology for data collection and model training, implementation plan, and expected performance evaluation metrics, aiming to provide a feasible solution for accurate and cost-effective indoor positioning.

1 Introduction

1.1 Background

Location awareness has become ubiquitous in modern applications, driven primarily by the success of Global Navigation Satellite Systems (GNSS) like GPS, GLONASS, and Galileo. These systems enable a wide range of outdoor location-based services (LBS), from vehicle navigation to mobile check-ins. However, GNSS signals are weak and easily obstructed by building materials, rendering them ineffective for indoor environments where people spend a significant portion of their time (e.g., airports, shopping malls, hospitals, office buildings, warehouses). This gap necessitates alternative technologies for accurate and reliable Indoor Positioning Systems (IPS).

1.2 Problem Statement

The lack of a universal, accurate, and cost-effective indoor positioning solution poses a significant challenge. Existing approaches based on Wi-Fi RSSI, Ultra-Wideband (UWB), computer vision, or inertial sensors each have limitations regarding accuracy, cost, infrastructure requirements, privacy concerns, or device compatibility. There is a growing demand for IPS solutions that can provide room-level or even sub-meter accuracy within complex indoor spaces, are relatively easy to deploy, and can leverage existing user devices like smartphones.

1.3 Proposed Solution

This project proposes an IPS based on Bluetooth Low Energy (BLE) beacons and machine learning. BLE beacons are small, low-power, low-cost transmitters that broadcast unique identifiers (UUID, Major, Minor) and their transmission power. A receiver device, such as a smartphone equipped with Bluetooth capabilities, can scan for these beacons and measure the Received Signal Strength Indicator (RSSI) from each detected beacon. The core idea is that the pattern of RSSI values received from multiple beacons changes predictably based on the receiver's location relative to the beacons.

The proposed methodology utilizes **RSSI fingerprinting**. This involves two phases:

1. **Offline Phase (Training):** Systematically collect RSSI vectors (sets of RSSI values from detectable beacons) at numerous known reference points (RPs) within the target indoor area. This creates a "radio map" or fingerprint database where each location is associated with one or more RSSI fingerprints. A machine learning model is then trained on this database to learn the mapping between RSSI patterns and locations (coordinates or zones).
2. **Online Phase (Positioning):** When a user needs their location, their device scans for nearby beacons, measures the current RSSI vector, and feeds this vector into the trained ML model. The model predicts the user's location based on the learned patterns from the offline phase.

Machine learning models suitable for this task include k-Nearest Neighbors (k-NN), Support Vector Machines (SVM), Random Forests, or Neural Networks (e.g., Multi-Layer Perceptrons - MLPs).

1.4 Motivation

BLE beacon-based IPS offers several advantages:

- **Cost-Effectiveness:** BLE beacons are relatively inexpensive compared to technologies like UWB.

- **Low Power:** Beacons can operate for months or years on small batteries.
- **Infrastructure:** Leverages the widespread availability of Bluetooth in smartphones, reducing the need for specialized user hardware.
- **Deployment:** Relatively easy to deploy beacons within an environment.
- **Accuracy Potential:** While raw RSSI is noisy, ML techniques can effectively filter noise and learn complex signal propagation patterns to achieve reasonable accuracy (often room-level or better).

1.5 Objectives

The primary objectives of this project are:

- To design an IPS architecture utilizing BLE beacons and a central processing unit/backend.
- To develop a methodology for collecting an RSSI fingerprint database within a defined indoor area.
- To select, implement, and train appropriate machine learning models (e.g., k-NN, MLP) for location estimation based on RSSI fingerprints.
- To outline an evaluation strategy to assess the positioning accuracy of the system using metrics like mean distance error.
- To propose a method for visualizing the estimated user location on an indoor map.

1.6 Scope

This project focuses on the design and methodology for a 2D indoor positioning system using BLE beacons and RSSI fingerprinting with machine learning.

- **Technology:** BLE beacons (e.g., iBeacon or Eddystone format), Bluetooth-enabled receiver (smartphone assumed), Machine Learning (e.g., Scikit-learn, TensorFlow/Keras).
- **Positioning Method:** RSSI Fingerprinting. Trilateration/Multilateration based on RSSI-to-distance models may be discussed but is not the primary focus due to RSSI volatility.
- **Environment:** A specific indoor area (e.g., a floor of an office building, a lab). Environmental factors affecting RSSI (e.g., obstacles, human presence) are acknowledged but not explicitly modeled beyond their impact on the collected fingerprints.
- **Output:** Estimated user coordinates (x, y) or zone/room identification.
- **Exclusions:** 3D positioning, sensor fusion with inertial sensors (IMU), handling dynamic changes in the environment after fingerprinting, beacon deployment optimization algorithms.

2 Literature Review

Indoor Positioning Systems (IPS) have been an active area of research for decades due to the limitations of GNSS indoors. Various technologies have been explored. Wi-Fi based positioning, often using RSSI fingerprinting similar to the proposed BLE approach, was an early popular method due to existing infrastructure [1]. However, Wi-Fi access point density and signal stability can be limitations. Ultra-Wideband (UWB) offers very high accuracy (centimeter-level) through time-of-flight measurements but requires specialized hardware for both tags and anchors, increasing cost and complexity [2]. Vision-based systems use cameras and image processing, often requiring pre-mapped environments or SLAM (Simultaneous Localization and Mapping) algorithms, facing challenges with lighting changes and computational cost. Inertial Measurement Units (IMUs) in smartphones can track movement (Pedestrian Dead Reckoning - PDR) but suffer from cumulative drift error and require an initial known position [3].

Bluetooth Low Energy (BLE) beacons emerged as a promising technology due to their low cost, low power consumption, and compatibility with modern smartphones [4]. Early BLE IPS often focused on proximity detection or simple trilateration/multilateration based on converting RSSI to distance using path loss models. However, RSSI is notoriously noisy and sensitive to environmental factors (multipath fading, obstacles, body blockage), making direct distance estimation unreliable [5].

To overcome RSSI volatility, **RSSI fingerprinting** became the dominant approach for BLE-based IPS [6]. This method treats the RSSI vector as a location-specific signature. Machine learning algorithms are well-suited for learning the complex, non-linear mapping between these noisy RSSI vectors and physical locations. Common algorithms applied include:

- **k-Nearest Neighbors (k-NN):** A simple yet often effective non-parametric method that finds the k closest fingerprints in the database (based on RSSI vector distance) and estimates the location based on the locations of these neighbors [7].
- **Support Vector Machines (SVM):** Can be used for classification (predicting a zone/-room) or regression (predicting coordinates).
- **Probabilistic Methods:** Techniques like Bayesian filtering or Gaussian processes model the probability distribution of RSSI values at different locations.
- **Neural Networks (NNs):** Multi-Layer Perceptrons (MLPs) can learn complex mappings. More advanced architectures like Convolutional Neural Networks (CNNs) or Recurrent Neural Networks (RNNs, e.g., LSTMs) have also been explored, sometimes treating sequences of RSSI readings over time as input [8].

Challenges in fingerprinting include the labor-intensive data collection (offline phase) and the sensitivity of the fingerprint map to environmental changes (requiring updates). Research continues on reducing collection effort (e.g., crowdsourcing, semi-supervised learning) and improving robustness.

This project focuses on the well-established RSSI fingerprinting approach using BLE beacons, planning to investigate standard ML models like k-NN or MLP due to their balance of performance and implementation complexity for a proof-of-concept system.

3 Proposed System Architecture

The proposed IPS architecture consists of deployed beacons, a mobile receiver, and a backend server for processing and location estimation. Figure **??** illustrates the components and data flow.

3.1 Component Description

3.1.1 BLE Beacons

- **Function:** Periodically transmit advertising packets containing unique identifiers and transmission power information.
- **Technology:** Standard BLE beacons compatible with iBeacon (Apple) or Eddystone (Google) formats.

- **Parameters:** Configurable transmission power and advertising interval (trade-off between detection range/frequency and battery life).
- **Deployment:** Strategically placed throughout the indoor environment (e.g., grid pattern on ceilings or walls) to ensure adequate coverage. Density depends on desired accuracy and building layout.

3.1.2 Receiver Device

- **Function:** Scan for BLE advertising packets, extract relevant data (Beacon ID, RSSI), and potentially perform initial filtering or aggregation before sending data to the backend.
- **Hardware:** Smartphone (iOS or Android) with Bluetooth capability is the primary target. Alternatively, a dedicated device like a Raspberry Pi with a Bluetooth dongle could be used for specific applications (e.g., asset tracking).
- **Software:** A mobile application or script responsible for:
  - Enabling Bluetooth scanning.
  - Filtering for relevant beacon UUIDs/Identifiers.
  - Recording RSSI values and timestamps for detected beacons over a short window (e.g., 1-3 seconds).
  - Averaging or processing RSSI values (optional).
  - Communicating with the backend server via network (Wi-Fi/Cellular).

3.1.3 Backend System (Server/Cloud)

- **Function:** Store the fingerprint database, host the trained ML model, receive RSSI data from receiver devices, perform location estimation, and potentially serve map data or location results to the user application.
- **Components:**
  - **API Endpoint:** Receives incoming RSSI vectors from receiver devices (e.g., via HTTP POST requests). (Implemented using Flask/Django, or cloud functions like AWS Lambda + API Gateway).
  - **Fingerprint Database:** Stores the collected radio map (Reference Point ID/Coordinates, associated RSSI vectors). (e.g., SQL database like PostgreSQL, NoSQL like MongoDB).
  - **ML Model Hosting:** Loads and serves the trained positioning model (e.g., Scikit-learn model file, TensorFlow/Keras model file). Inference can be done within the API handler or via a dedicated ML serving platform.
  - **(Optional) Map Server:** Stores and serves indoor map tiles or vector data.
  - **(Optional) User Database:** Manages user accounts or device registrations if needed.
- **Platform:** Cloud platform (AWS, GCP, Azure) or a dedicated server.

3.1.4 User Interface (Mobile Application)

- **Function:** Initiates scans, sends data to the backend, receives estimated location, and displays it on an indoor map.
- **Features:** Real-time location dot on a map, potentially basic navigation instructions (if pathfinding is added).

4 Methodology

The core methodology is RSSI fingerprinting, involving distinct offline and online phases.

4.1 Offline Phase: Fingerprint Database Collection

This crucial phase builds the radio map used for training the positioning model.

1. **Environment Setup:** Deploy BLE beacons at fixed, known locations within the target indoor area. Record the ID (UUID/Major/Minor) and coordinates of each beacon.

2. **Reference Point (RP) Selection:** Define a grid of reference points covering the area of interest. The density of RPs impacts accuracy (more points generally yield better accuracy but require more effort). Record the precise coordinates (x, y) of each RP.

3. **Data Acquisition:** At each RP, use the receiver device (e.g., smartphone app) to scan for BLE beacons for a defined duration (e.g., 10-30 seconds).

4. **Feature Extraction:** For each scan period at an RP, record the RSSI values from all detected beacons. Typically, the average or median RSSI for each beacon over the scan window is used. Beacons not detected during a scan are assigned a very low RSSI value (e.g., -100 dBm or -110 dBm) or handled via imputation.

5. **Database Construction:** Store the collected data, linking each RP's coordinates (label) with the corresponding RSSI vector (features). An RSSI vector might look like: `[RSSI_Beacon1, RSSI_Beacon2, ..., RSSI_BeaconN]`. The database contains many such (Location, RSSI Vector) pairs. Data might be collected multiple times or facing different directions at each RP to improve robustness.

4.2 Machine Learning Model Training

1. **Data Preparation:** Load the fingerprint database. Preprocess the RSSI vectors if necessary (e.g., normalization, handling missing values if not already done). Split the data into training and validation sets.

2. **Model Selection:** Choose an appropriate ML algorithm. Options include:

   - **k-Nearest Neighbors (k-NN):** Simple, often effective. Finds the k fingerprints in the database whose RSSI vectors are closest (e.g., Euclidean distance) to the currently measured vector. Predicts location by averaging the coordinates of these k neighbors. Requires choosing an optimal 'k'.
   - **Multi-Layer Perceptron (MLP):** A type of neural network that can learn complex non-linear mappings. Input layer size equals the number of beacons, output layer size equals the number of coordinates (2 for x, y) or the number of zones (if classifying zones). Requires careful tuning of architecture (layers, neurons) and hyperparameters.
   - **Other Algorithms:** Random Forest, SVM, etc., can also be applied.

3. **Training:** Train the selected model(s) on the training portion of the fingerprint database. Use the validation set to tune hyperparameters (e.g., 'k' in k-NN, network architecture/learning rate in MLP).

4. **Model Saving:** Save the trained model parameters/object for use in the online phase.

4.3 Online Phase: Positioning

This phase occurs when a user requests their location.

1. **RSSI Scan:** The user's device performs a short BLE scan (e.g., 1-3 seconds) to detect nearby beacons and measure their RSSI values.
2. **Vector Creation:** An RSSI vector is constructed from the scan results, using the same format and handling of missing beacons as during the offline phase.
3. **Backend Communication:** The RSSI vector is sent to the backend server.
4. **Prediction:** The backend loads the trained ML model and inputs the received RSSI vector. The model outputs the predicted location (coordinates or zone).
5. **Result Transmission:** The predicted location is sent back to the user's device.
6. **Visualization:** The user's application displays the estimated location on an indoor map.

4.4 Evaluation Strategy

The performance of the trained positioning model needs to be evaluated quantitatively.

- **Test Data:** Use a separate set of fingerprint data (collected at known RPs but not used for training/validation) as the test set.
- **Primary Metric: Mean Distance Error (MDE):** For each test fingerprint, predict the location using the model. Calculate the Euclidean distance between the predicted coordinates and the true coordinates of the test RP. The MDE is the average of these distances over all test points. Lower MDE indicates higher accuracy.
- **Other Metrics:** Standard deviation of distance error, Root Mean Squared Error (RMSE), accuracy within specific distance thresholds (e.g., percentage of predictions within 1m, 2m, 3m of true location), confusion matrix (if predicting zones/rooms).

5 Implementation Plan

5.1 Hardware

- BLE Beacons: Off-the-shelf iBeacon or Eddystone compatible beacons (e.g., Estimote, Kontakt.io, or generic modules based on nRF5x or ESP32 chips). Number depends on area size and desired density.
- Receiver Device: Android smartphone with Bluetooth 4.0+ for development and testing.
- Backend Server: Cloud VM (e.g., AWS EC2 t2.micro, GCP e2-micro) or local machine for development.

5.2 Software

- **Beacon Configuration Tool:** Manufacturer's app or standard tool (e.g., nRF Connect) to configure beacon UUID, Major/Minor, Tx Power, Interval.
- **Data Collection App (Android):** Custom app using Android's BluetoothLeScanner API to scan for beacons, record RSSI, allow user to tag scans with RP coordinates/ID, and upload data to backend/save locally.

- **Backend Application:** Python (3.8+) with Flask/Django for API; libraries for database interaction (e.g., SQLAlchemy, psycopg2); Scikit-learn (for k-NN, SVM, RF) and/or TensorFlow/Keras (for MLP); NumPy, Pandas.
- **Positioning App (Android):** Uses BluetoothLeScanner, communicates with backend API to send RSSI and receive location, displays location on a map view (e.g., using Android Maps SDK with custom indoor map overlay or a dedicated indoor map library).
- **Development Environment:** Android Studio for mobile apps; Python IDE (VS Code, PyCharm) for backend; Git/GitHub for version control.

6 Expected Results and Discussion

Based on literature [6, 7, 8], the proposed BLE RSSI fingerprinting system is expected to achieve reasonable indoor positioning accuracy.

Anticipated Performance: The primary metric, Mean Distance Error (MDE), is expected to be in the range of **1 to 5 meters**, depending heavily on beacon density, environmental complexity, quality of fingerprint data, and the chosen ML model. k-NN often provides a strong baseline, while tuned MLPs might offer slightly better accuracy by capturing more complex signal patterns. Accuracy sufficient for room-level identification is highly likely. Sub-meter accuracy is generally challenging with RSSI fingerprinting alone.

Factors Influencing Results:

- **Beacon Density and Placement:** Higher density generally improves accuracy but increases cost. Placement strategy affects coverage and signal geometry.
- **Fingerprint Density and Quality:** More reference points and multiple readings per point improve the radio map's resolution and robustness. Consistent data collection procedures are crucial.
- **Environmental Dynamics:** Changes in the environment (moved furniture, number of people) after fingerprinting can degrade accuracy. Human body blockage significantly affects RSSI.
- **Device Heterogeneity:** Different smartphones may report different RSSI values even at the same location, posing a challenge if the system needs to work across various devices. Model calibration or device-specific adjustments might be needed.
- **Choice of ML Model and Hyperparameters:** Performance varies between algorithms (k-NN vs. MLP) and requires proper tuning (e.g., 'k' value, network architecture).

Discussion Points: The results will be analyzed by comparing the MDE achieved by different models (if multiple are tested). The distribution of errors (e.g., error CDF plot) will provide more insight than just the mean. Challenges encountered during data collection (e.g., signal fluctuations) and their impact will be discussed. The trade-offs between accuracy, deployment effort (fingerprinting), and cost will be considered. The suitability of the achieved accuracy for target applications (e.g., general navigation vs. precise asset tracking) will be assessed.

7 Conclusion

This report presented the design and methodology for an Indoor Positioning System utilizing BLE beacon technology and machine learning-based RSSI fingerprinting. The system aims to overcome the limitations of GNSS indoors by leveraging the low cost and ubiquity of BLE technology. The proposed architecture involves deploying beacons, collecting RSSI fingerprints using a receiver device (smartphone), training an ML model (e.g., k-NN, MLP) on this data, and using the model to estimate user location during the online phase.

The methodology detailed the critical offline fingerprint collection process and the subsequent ML model training and online positioning steps. Key performance metrics, primarily Mean Distance Error, were identified for evaluation. The implementation plan outlined the necessary hardware and software components, favoring readily available technologies like standard BLE beacons, smartphones, Python backend frameworks, and common ML libraries (Scikit-learn/TensorFlow).

While RSSI-based methods face inherent challenges due to signal noise and environmental sensitivity, the fingerprinting approach combined with appropriate ML algorithms is expected to yield accuracy suitable for many indoor navigation and LBS applications (typically 1-5 meter MDE). This project provides a feasible blueprint for developing a practical and cost-effective IPS solution. Future work can focus on improving robustness, reducing deployment effort, and integrating sensor fusion techniques.

8 Future Work

Potential extensions and improvements to the proposed system include:

- **Sensor Fusion:** Integrate IMU data (accelerometer, gyroscope, magnetometer) from the smartphone for Pedestrian Dead Reckoning (PDR). Fusing PDR with BLE fingerprinting (e.g., using Kalman filters or particle filters) can significantly improve accuracy, smoothness, and robustness against temporary signal loss.
- **Handling Environmental Changes:** Investigate methods to automatically update or adapt the fingerprint map without requiring complete re-surveying (e.g., using semi-supervised learning, transfer learning across time, or detecting environmental changes).
- **Reducing Fingerprinting Effort:** Explore crowdsourcing techniques or path-based fingerprinting (collecting data along paths instead of just static points) to reduce the manual effort of the offline phase.
- **Device Calibration:** Develop methods to mitigate the impact of RSSI variations between different receiver devices.
- **Advanced ML Models:** Experiment with more complex deep learning models (CNNs, RNNs/LSTMs) on RSSI sequences or spatial RSSI representations.
- **3D Positioning:** Extend the system to estimate the user's floor level in multi-story buildings, potentially using barometric pressure sensors or analyzing RSSI patterns across floors.
- **Beacon Placement Optimization:** Develop algorithms to determine the optimal number and locations of beacons for a given environment and accuracy requirement.
- **Pathfinding and Navigation:** Integrate the positioning output with indoor map data and pathfinding algorithms (e.g., A*) to provide turn-by-turn navigation instructions.

References

References

[1] H. Liu, H. Darabi, P. Banerjee, and J. Liu, "Survey of wireless indoor positioning techniques and systems," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 37, no. 6, pp. 1067–1080, Nov. 2007.

[2] I. Alarifi, A. Al-Salman, M. Alsaleh, A. Alnafessah, S. Al-Hadhrami, M. Al-Ammar, and H. S. Al-Khalifa, "Ultra Wideband Indoor Positioning Technologies: Analysis and Recent Advances," *Sensors*, vol. 16, no. 5, p. 707, May 2016.

[3] R. Harle, "A Survey of Indoor Inertial Positioning Systems for Pedestrians," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 3, pp. 1281–1293, Third Quarter 2013.

[4] K. K. C. Liu and C.-H. Lee, "An experimental study of Bluetooth low energy (BLE) technology for indoor positioning," in *2015 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, Oct 2015, pp. 1–6.

[5] Z. Farid, R. Nordin, and M. Ismail, "Recent advances in wireless indoor localization techniques and system," *Journal of Computer Networks and Communications*, vol. 2013, Article ID 185138, 12 pages, 2013.

[6] O. Zafari, A. Gkelias, and K. K. Leung, "A survey of indoor localization systems and technologies," *IEEE Communications Surveys Tutorials*, vol. 21, no. 3, pp. 2568–2599, thirdquarter 2019.

[7] F. Zafari, M. Papapanagiotou, and K. K. Leung, "IoTFrankenstein: Providing location-awareness to low-power IoT devices," in *2016 IEEE International Conference on Communications (ICC)*, May 2016, pp. 1–6.

[8] H. T. T. Binh and N. T. H. Lien, "Deep Learning Models for Indoor Positioning based on Wi-Fi Fingerprinting: A Survey," *Electronics*, vol. 10, no. 2, p. 128, Jan. 2021.

A Appendix: Potential Code Snippets

This appendix could contain selected code snippets relevant to the project, such as:

- Python code for training a k-NN or MLP model using Scikit-learn/Keras on the fingerprint data.
- Android (Java/Kotlin) code snippet for scanning BLE beacons and extracting RSSI.
- Python (Flask/Django) snippet for the backend API endpoint receiving RSSI data and returning a location prediction.

Appendix: Code Snippets

A.1 Python: Train k-NN Model

```python
# Python: Train a k-NN model for Indoor Positioning
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsRegressor
import joblib

# Load fingerprint dataset
data = pd.read_csv('fingerprints.csv')

# RSSI readings as features, coordinates as labels
X = data.drop(columns=['x', 'y'])
y = data[['x', 'y']]

# Split dataset
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train k-NN
knn = KNeighborsRegressor(n_neighbors=5)
knn.fit(X_train, y_train)

# Save model
joblib.dump(knn, 'knn_model.pkl')
```

A.2 Python: Flask API for Location Prediction

```python
# Python: Simple Flask API for Indoor Position Prediction
from flask import Flask, request, jsonify
import joblib
import numpy as np

app = Flask(__name__)
model = joblib.load('knn_model.pkl')

@app.route('/predict', methods=['POST'])
def predict():
    data = request.get_json(force=True)
    rssi_vector = np.array(data['rssi']).reshape(1, -1)
    prediction = model.predict(rssi_vector)
    return jsonify({'x': prediction[0][0], 'y': prediction[0][1]})

if __name__ == '__main__':
    app.run(debug=True)
```

A.3 Android Java: BLE Beacon Scanning

```java
// Android Java: Scan BLE Beacons and Read RSSI
BluetoothAdapter bluetoothAdapter = BluetoothAdapter.getDefaultAdapter();
BluetoothLeScanner bluetoothLeScanner = bluetoothAdapter.getBluetoothLeScanner();

ScanCallback scanCallback = new ScanCallback() {
    @Override
    public void onScanResult(int callbackType, ScanResult result) {
        String beaconId = result.getDevice().getAddress();
        int rssi = result.getRssi();
```

Appendix: Code Snippets

```
        // Process beaconId and rssi
    }
};

public void startScanning() {
    ScanSettings settings = new ScanSettings.Builder()
        .setScanMode(ScanSettings.SCAN_MODE_LOW_LATENCY)
        .build();
    bluetoothLeScanner.startScan(null, settings, scanCallback);
}
```