Understanding of BLE Communication

---

1. Overview of BLE (Bluetooth Low Energy)

BLE is a wireless protocol designed for low energy consumption, ideal for IoT devices like your smart lock system. It provides:

- **Low Power**: Ideal for battery-powered devices like smart locks.
- **Short-Range Communication**: Typically, within 10-100 meters.
- **One-to-Many Communication**: BLE supports both 1-to-1 (e.g., mobile app to Raspberry Pi) and 1-to-many communication.

2. BLE Device Roles: Central vs Peripheral

In BLE, devices act in two roles:

- **Central (Client)**: The mobile app acts as the central device, scanning for nearby peripherals like the Raspberry Pi.
- **Peripheral (Server)**: The Raspberry Pi advertises its availability and waits for the central device to connect and send data.

Backend Process:

- Mobile app scans for BLE devices.
- Raspberry Pi advertises its availability.
- Once a connection is established, the app sends the password, and the Raspberry Pi processes it and responds back accordingly.

3. BLE Services and Characteristics *(We use custom UUIDs)*

- **Services**: Collection of characteristics for specific functionality.
- **Characteristics**: Data exchanged between devices, either readable, writable, or both. In our smart lock system:
    - The mobile app writes a password to a characteristic on the Raspberry Pi.
    - The Raspberry Pi compares the password with the stored value/from cloud and writes back the lock status.

4. Advertising and Connection

The Raspberry Pi advertises itself, sending periodic packets with device information.

The mobile app:

- Scans and lists available devices.
- Initiates a connection once the user selects the Raspberry Pi.

[5. BLE Security and Pairing

To ensure security:

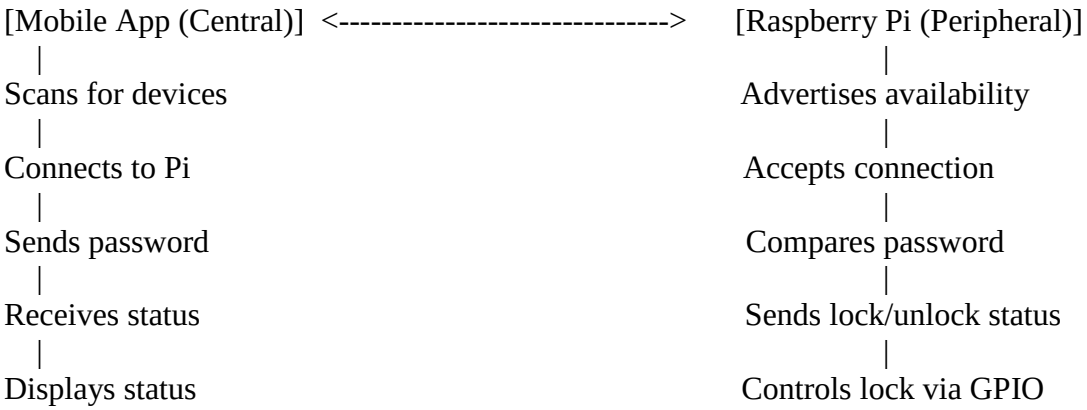
- **Pairing** is the process where two BLE devices establish a secure connection and exchange encryption keys.

- **Bonding** is a step further where the encryption keys are saved for future connections, allowing devices to reconnect securely without going through the pairing process again.**]**

6. Data Flow in Smart Lock System

- **Advertising**: Raspberry Pi broadcasts availability.
- **Connection**: Mobile app initiates connection and sends password.
- **Validation**: Raspberry Pi compares the password and sends lock/unlock status back.
- **Status Display**: The mobile app displays whether the lock is opened or denied.

7. BLE Communication Flow Diagram



8. Advantages of BLE in Smart Lock System

- **Low Power**: Keeps energy usage minimal.
- **Quick Connections**: Fast device discovery.
- **Security**: Protects against unauthorized access.

Programming Model Behind MIT App Inventor

Event-Driven Programming

MIT App Inventor follows an event-driven model where the flow of the app is triggered by user actions or changes in device state (e.g., Bluetooth connection).

Underlying Technology

The app is interacting with **Android's native Bluetooth libraries** (such as android.bluetooth.BluetoothAdapter and android.bluetooth.BluetoothDevice) to manage device discovery, pairing, and communication.

- **Blocks Translated into Java**: The visual blocks you use are translated into Java code, which interacts with Android's APIs.
- **Pre-existing Android APIs**: MIT App Inventor leverages Android's Bluetooth and network APIs to handle BLE communication and cloud interactions without requiring you to write Java code.

Cloud Integration via APIs

- **RESTful APIs**: The app uses HTTP requests for cloud communication (e.g., updating passwords on ThingSpeak).
- **Database and Web Services**: Web components like Web1.Get or Web1.Post interact with external services like ThingSpeak.

Similarly, if you use web components (Web.Get, Web.Post), App Inventor interacts with Android's **HttpURLConnection API** to send HTTP requests and receive responses.

These APIs are "pre-existing" in the sense that MIT App Inventor is built on top of Android, and Android provides a vast number of libraries to access hardware (Bluetooth, camera, GPS) and network services (web, cloud APIs).

IoT Platform: ThingSpeak

---

1. Overview of ThingSpeak

ThingSpeak is an open-source IoT platform that provides APIs to store, process, and visualize data sent from IoT devices. It is widely used in projects where sensors or devices communicate over the internet. Key features of ThingSpeak that are relevant to IoT communication with a microcontroller include:

- **HTTP API** for sending data to the cloud and receiving responses.
- **MQTT Protocol Support**, which is crucial for real-time communication.
- **Cloud-based Data Storage** for sensor data.
- **Data Visualization Tools** such as graphs, which help monitor device performance.

2. REST API Communication

- **REST (Representational State Transfer)** is a protocol-independent architecture that operates over HTTP/HTTPS. It uses URLs (like the one you are using) to perform actions such as reading or updating data.
- When you send the following request:

https://api.thingspeak.com/update?api_key=YOUR_WRITE_API_KEY&field1=1234

The system (RPI) communicates with ThingSpeak using RESTful APIs:

- **HTTP GET Request**: Sends password data from ThingSpeak to Raspberry Pi.

- **Response Handling**: ThingSpeak confirms the update with a unique entry ID ID if successful or an error code if the request fails.

How REST API Works in our Case:

1. **Client (Raspberry Pi)** sends an HTTP GET request to ThingSpeak.
2. **Server (ThingSpeak)** processes the request and updates the respective field with the value provided (e.g., password).

Security and Authentication:

- **API Keys**: Ensure only authorized devices can send (Write API Key) or retrieve (Read API Key) data.

3. Python Programming for BLE and Cloud Integration

- **BLE Communication**: Python libraries like bluez manage device discovery and data exchange between the mobile app and Raspberry Pi.
- **Cloud Integration**: Python scripts fetch or update passwords from ThingSpeak's cloud using HTTP requests.

NOTE:

For microcontrollers like the Raspberry Pi, **WiFi** or **Ethernet** modules are used to send data to ThingSpeak over the internet.

The microcontroller runs a lightweight program (written in Python for Raspberry Pi, or C/C++ for Arduino) that uses libraries such as requests (for HTTP) or paho-mqtt (for MQTT) to send or receive data from ThingSpeak.

Hardcode a password in Cloud-

1. Using Web-Browser-

https://api.thingspeak.com/update?api_key=YOUR_WRITE_API_KEY&field1=1234

2. Using Terminal/cmd-

curl -X GET
https://api.thingspeak.com/update?api_key=YOUR_WRITE_API_KEY&field1=1234

3. Python Code-

```
import requests

# Replace with your ThingSpeak Write API Key

WRITE_API_KEY = "YOUR_WRITE_API_KEY"

password = "1234"  # The password you want to store

# Send the update request to ThingSpeak

url =
f"https://api.thingspeak.com/update?api_key={WRITE_API_KEY}&field1={password}"

response = requests.get(url)

if response.status_code == 200:

    print(f"Password updated successfully: {response.text}")

else:

    print(f"Failed to update password: {response.status_code}")
```

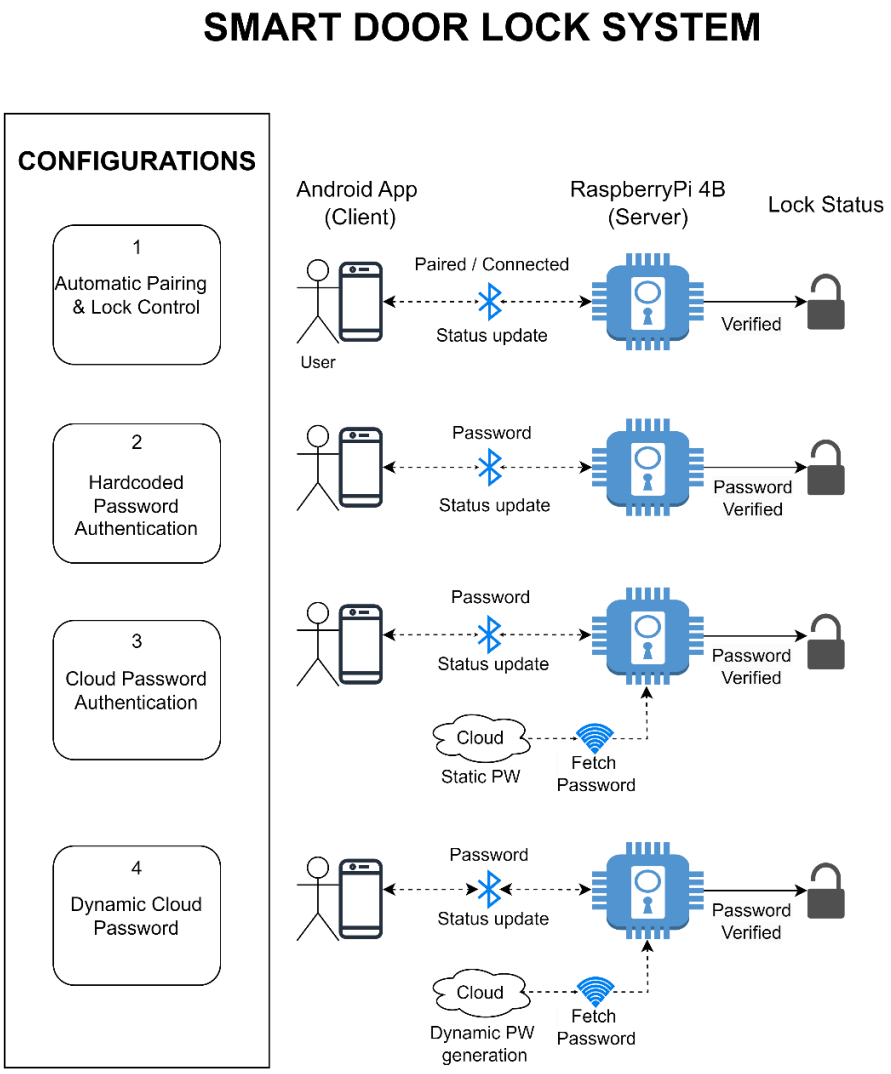
Python Programming for IoT and BLE

- **BLE with Python**:
  - Using Python libraries (bluez, pybluez) to manage BLE communication.
  - Handling device discovery, connection, and data exchange between the mobile app and the Raspberry Pi.
- **Cloud Integration with Python**: Writing Python scripts to fetch, store, and process cloud data (passwords) from platforms like ThingSpeak.

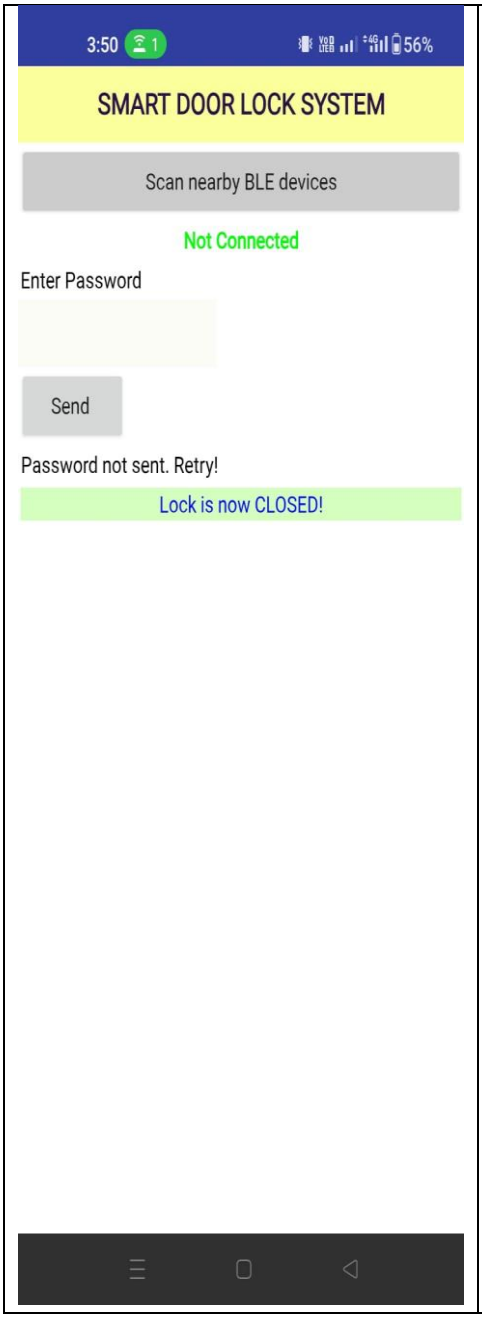
Conclusion

So far, I've gained experience in integrating BLE communication with ThingSpeak. Additionally, I've developed a solid understanding about BLE roles, characteristics, event-driven programming in MIT App Inventor, and RESTful API communication, all of which are helping me build efficient IoT systems.

Flow-Chart:


SMART DOOR LOCK SYSTEM
CONFIGURATIONS
1
Automatic Pairing & Lock Control
2
Hardcoded Password Authentication
3
Cloud Password Authentication
4
Dynamic Cloud Password
Android App (Client)
RaspberryPi 4B (Server)
Lock Status
User
Paired / Connected
Status update
Verified
Password
Status update
Password Verified
Password
Status update
Password Verified
Cloud
Static PW
Fetch Password
Password
Status update
Password Verified
Cloud
Dynamic PW generation
Fetch Password

Android App (MIT)



ThingSpeak-IoT Platform