Study on how RSSI-Distance correlation can be used to expedite Disaster Recovery process

TEAM 6

Github link:
https://github.com/kskushal/IoTProjectTeam6

Project Objective

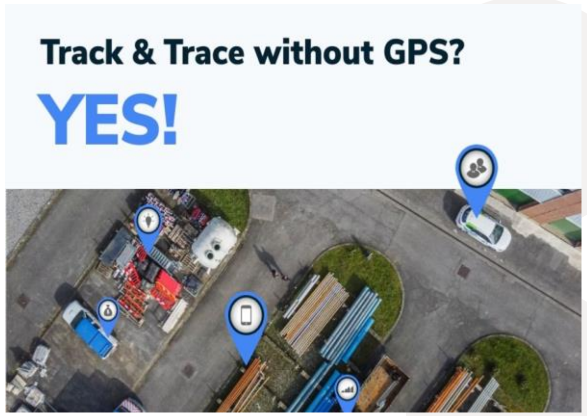
Problem Statement : Most firefighters/Disaster recovery personnel risk their own lives to volunteer and save the victims. One of the challenges faced in these scenarios - Inability to determine if a victim is alive(unconscious) or dead. Timely medical care is the most important factor behind success of a post-disaster operation.

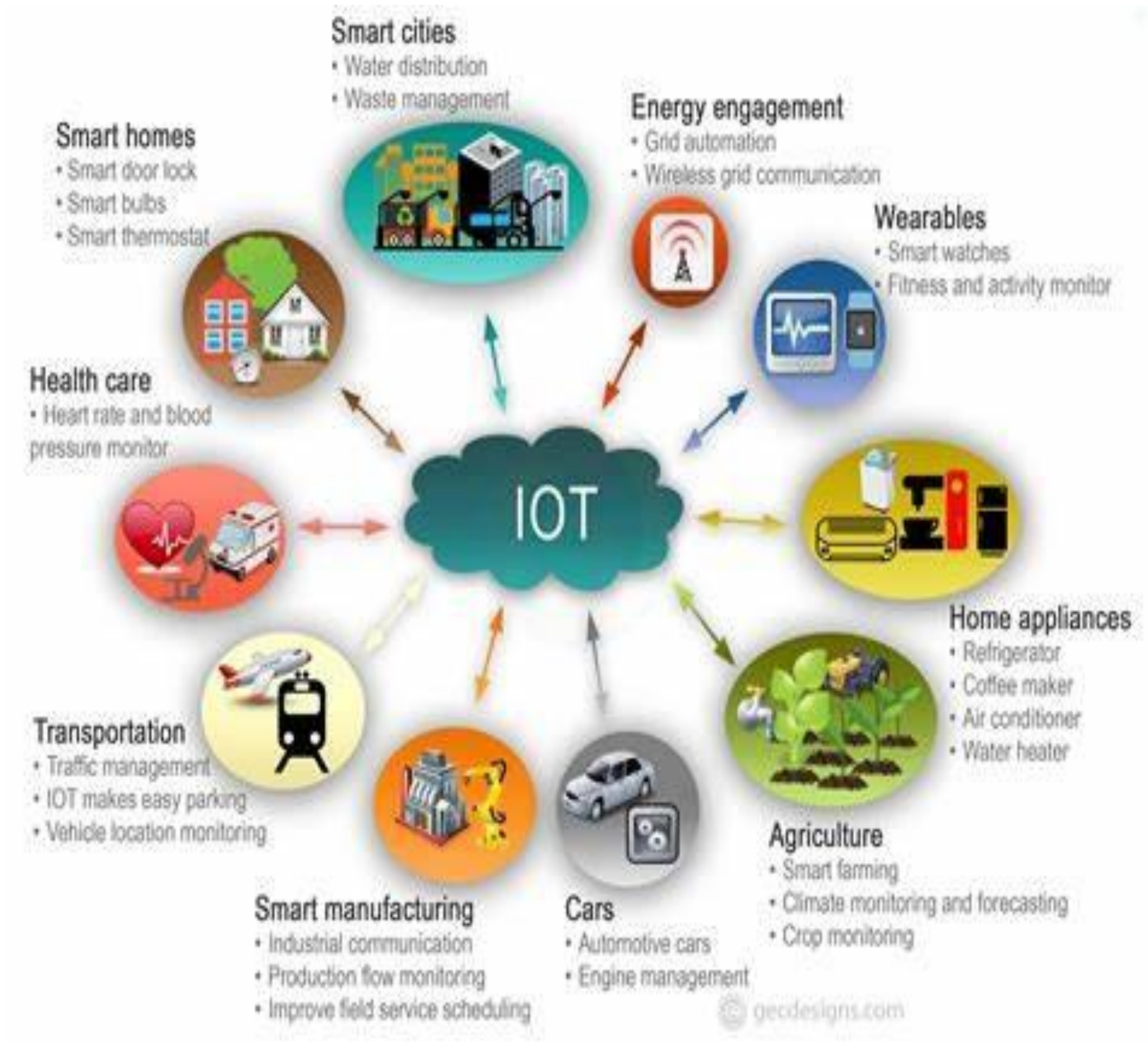
Solution : In Disaster prone zones, we propose implementation of an application that monitors the heartrate/pulse of a person and feeds this information to a Disaster Recovery team. The wearable IOT device (with sensor) will activate only when a disaster occurs. Through our project, we came up with an extensive study on how RSSI strength varies with respect to various types of obstacles. This data helps us compute the accurate distance from beacon while implementing such a system.

Introduction

The main objectives of this project are:

- To develop a study that supports an IoT-based positioning system to support disaster recovery process.
- Analyze RSSI-Distance Correlation with respect to type of obstacles/environment such as water, humans, wall.
- Measure RSSI strength for varying distances - 1 meter and 3 meter to compare N value and determine the accuracy





What is IOT?

The Internet of Things (IoT) is a network of physical objects, devices, sensors, and software that are connected to the internet and can communicate with each other to exchange data and perform various tasks.

Types Of Positioning Systems

There are primarily two types of positioning systems based on their operational principle:

1. Satellite-based positioning systems
2. Terrestrial-based positioning systems

Satellite-based positioning systems:

- These systems use a network of satellites to provide location information to users.
- Examples include GPS, GLONASS, Galileo, BeiDou, and IRNSS.

Terrestrial-based positioning systems:

- These systems use terrestrial infrastructure such as cell towers, Wi-Fi access points, or radio beacons to determine the user's location.
- Examples include WiFi positioning systems, Bluetooth positioning systems, Zigbee positioning systems, RFID positioning systems, and cellular positioning systems.

In general, satellite-based positioning systems tend to be more accurate than terrestrial-based positioning systems. However, terrestrial-based systems can be useful in areas where satellite signals may be obstructed or unavailable, such as indoors or in urban environments with tall buildings.

ESP32 and Bluetooth Low Energy (BLE)

- ESP32 is a microcontroller chip with built-in WiFi, Bluetoot and Bluetooth Low Energy (BLE) capabilities, manufactured if Systems.
- BLE is a low-power, wireless communication technology us for short-range communication between devices.
- When used together, ESP32 and BLE can provide a cost effective, low-power, and accurate solution for location tracking in IoT applications.

Here are some advantages of using ESP32 and BLE for positio

1.Low cost

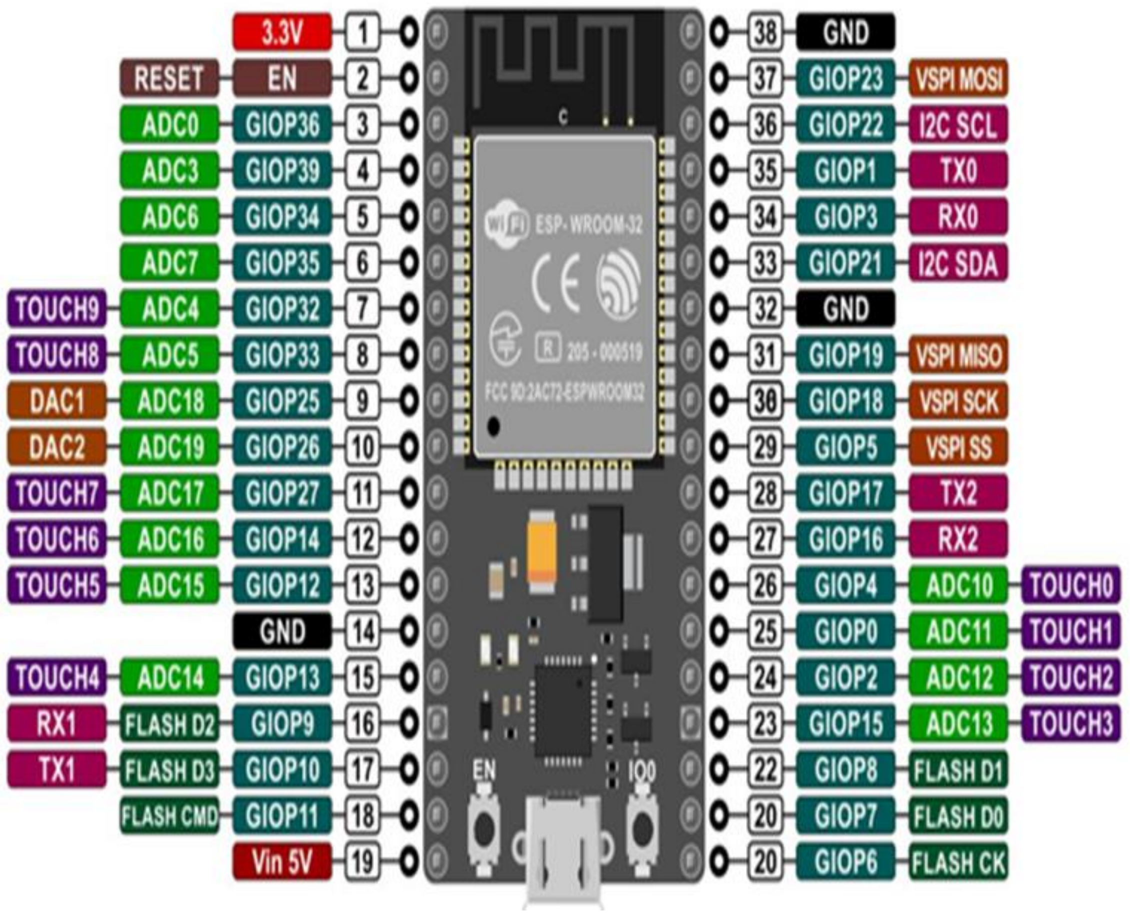
2.Low power consumption

3.Small form factor

4.Wide compatibility

5.High accuracy

6.Versatility



Working Principle of RSSI

- In positioning systems, the working principle of RSSI is used to determine the location of a receiver device based on the strength of the signals received from multiple transmitters or beacons.
- The RSSI values are used to calculate the distance between the receiver and each transmitter using a propagation model. The propagation model takes into account the attenuation of the signal due to factors such as distance, obstacles, and interference.
- Once the distance between the receiver and each transmitter is calculated, the location of the receiver can be determined using triangulation. Triangulation involves using the distance measurements from at least three transmitters to determine the location of the receiver. The location of the receiver can be represented as a point on a map or in a coordinate system.

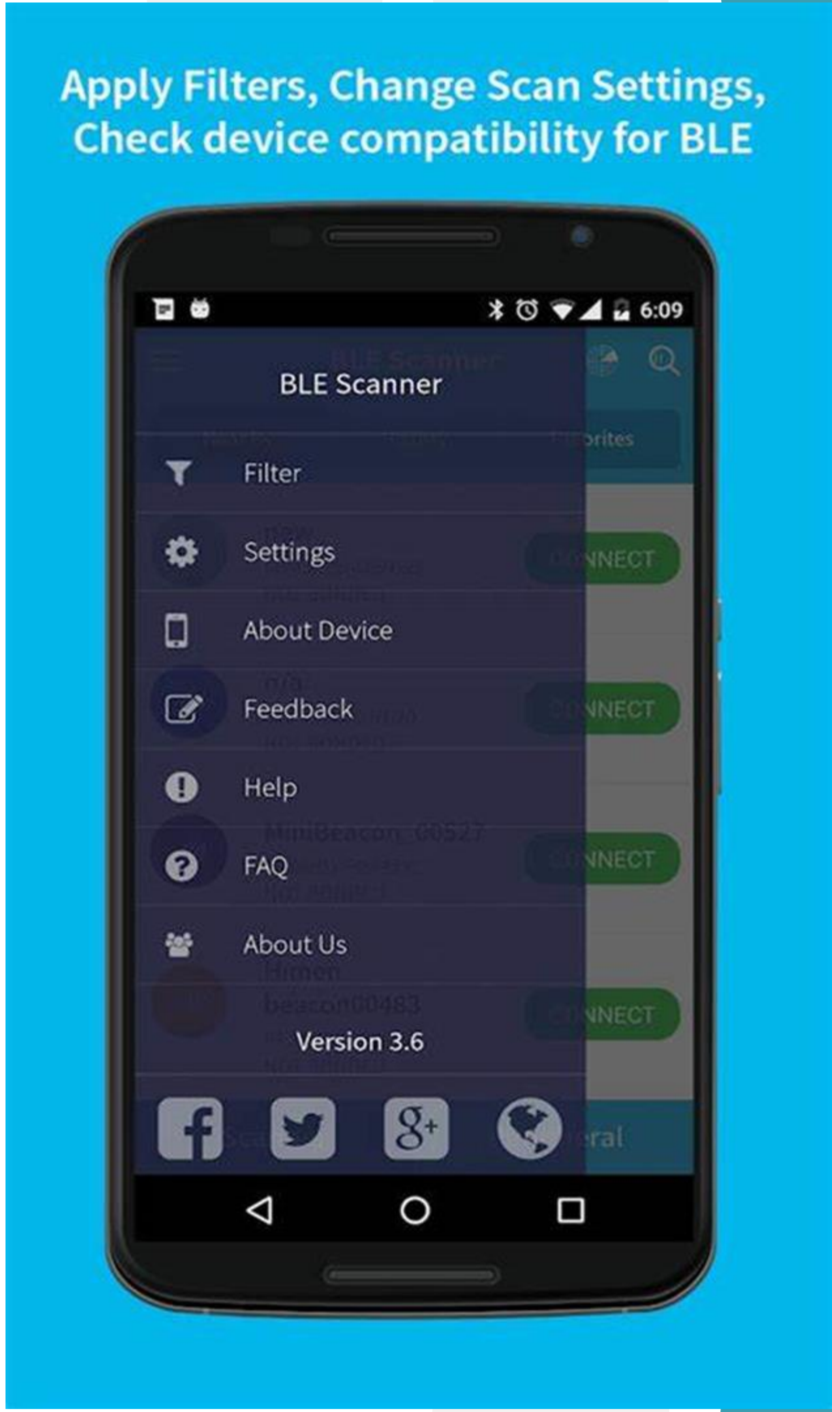
Smartphone Application

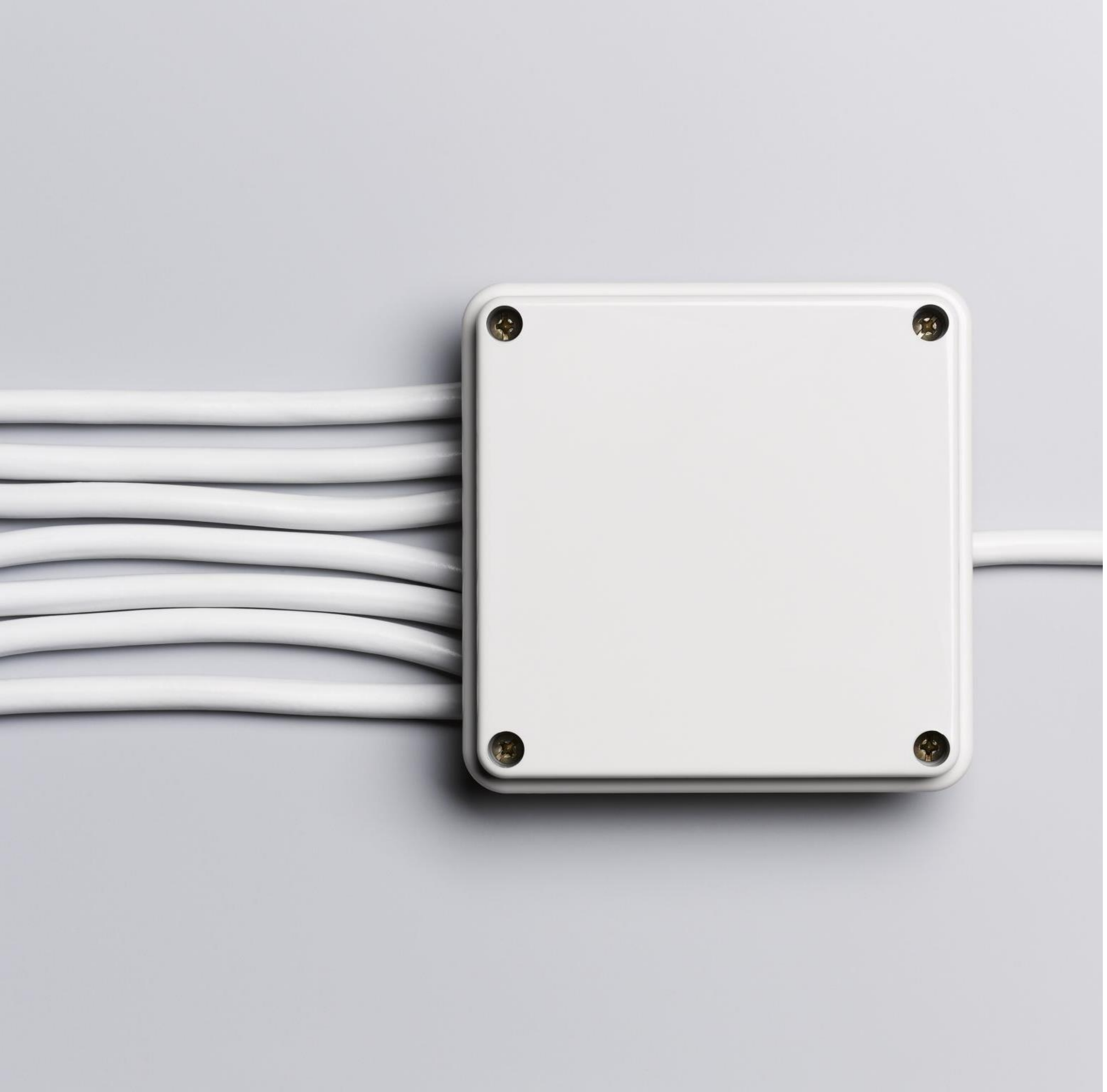
- A BLE (Bluetooth Low Energy) scanner software is a program that can be installed on a device such as a smartphone, tablet, or computer, to detect and receive advertising packets transmitted by BLE devices, such as beacons and other Bluetooth devices.
- BLE scanner software is used in various applications such as asset tracking, indoor positioning, and location-based services.

Features

1. Scanning
2. Signal analysis
3. Range
4. Battery life
5. Compatibility
6. Data transmission
7. Filtering
8. Real-time tracking
9. Alerts

Overall, BLE scanner software is an efficient and cost-effective solution for accurate tracking of BLE devices in various applications. They offer several features and functionalities that enable efficient and accurate positioning, making them a popular choice in various industries.





Hardware Requirements

1.ESP 32 ARDUINO

2. USB CABLE

3.MOBILE DEVICE

4.WIRELESS SENSORS

Software Requirements

In order to connect and program an Arduino board, We should require the following software requirements

1. Arduino IDE
2. USB driver
3. Board-specific software libraries
4. Third-party software libraries or tools

- Arduino IDE: the Arduino IDE is installed on the computer for the purpose of programming the board and uploading the code. This is available for free downloaded from the Arduino website.

- USB driver: the Arduino board is connected to the computer by using the proper USB driver by installing it in the operating system. The computer is allowed identify the board and establish the connection with it.

- Board-specific software libraries: Arduino boards requires the particular software libraries that should be installed to function properly. These libraries are frequently downloaded from the Arduino website or from the Library Manager in the Arduino IDE.

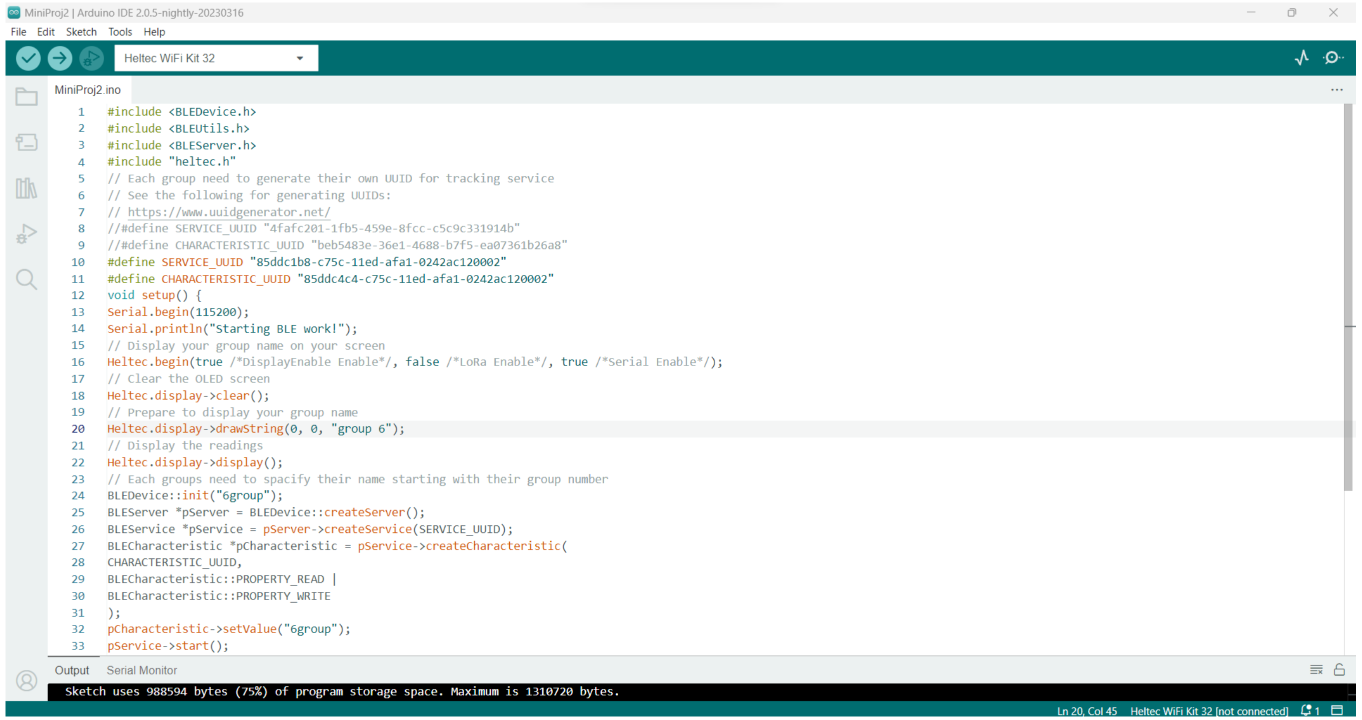
- Third-party software libraries or tools: Based on the particular application, we may need to install additional software libraries or tools to work with sensors or other components.

ESP32 Programming

- The process of writing, compiling, and uploading code to the ESP32 microcontroller which is frequently used in IoT (Internet of Things) applications is known as ESP32 programming.
- The ESP32 is a low-cost, low-power system-on-a-chip (SoC) microcontroller with built-in Wi-Fi, Bluetooth, and BLE (Bluetooth Low Energy) capabilities making it the perfect foundation for creating IoT applications.
- Some of the important elements of ESP32 programming for IoT include Wi-Fi and Bluetooth connectivity, ow-power operation and Support for various sensors and peripherals

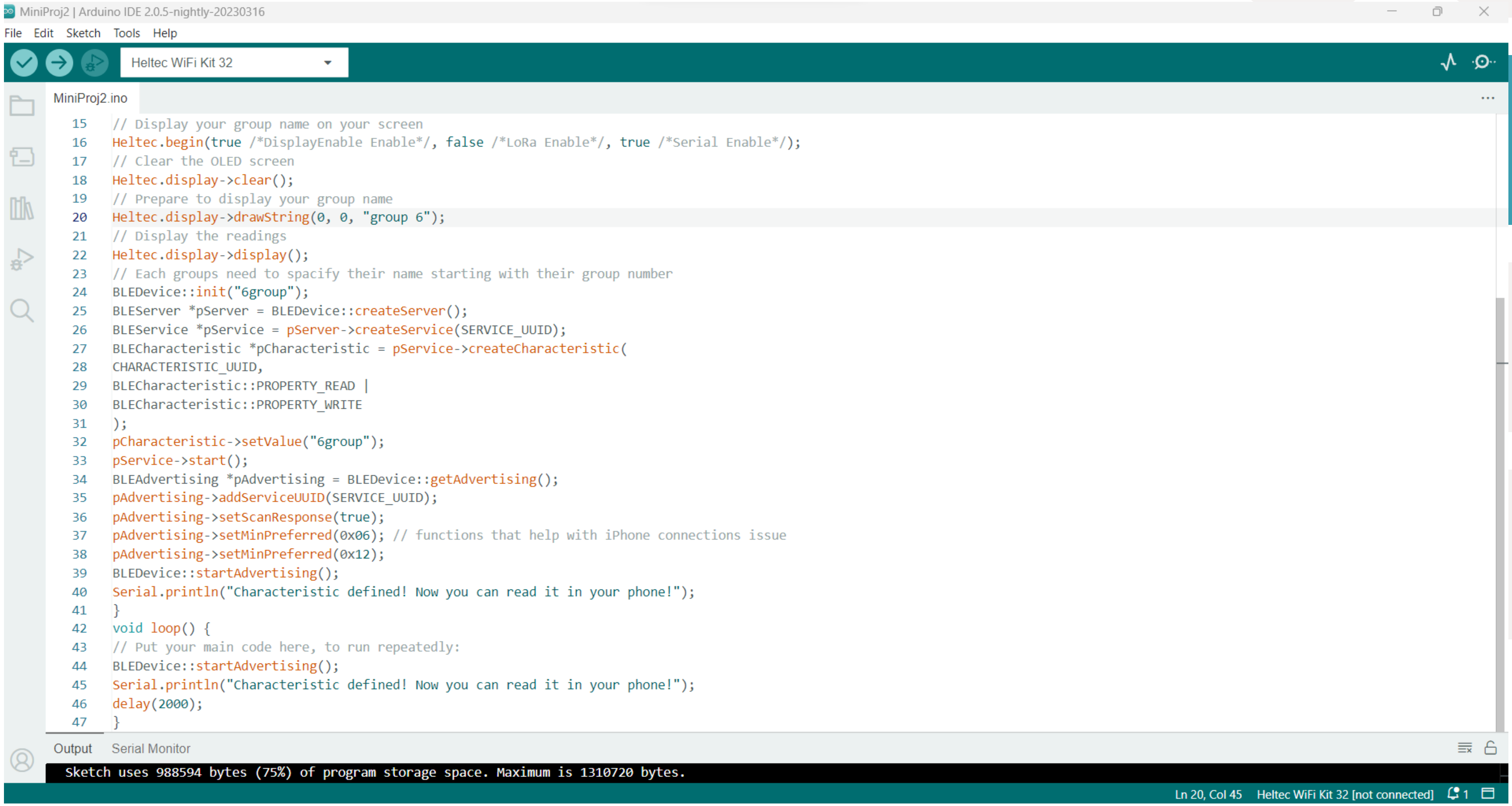
Code Snippet

To observe RSSI (Received Signal Strength Indication) values using ESP32, you can use the WiFi API provided by the ESP32 platform. Here is an example code snippet in Arduino:


MiniProj2 | Arduino IDE 2.0.5-nightly-20230316
File Edit Sketch Tools Help
Heltec WiFi Kit 32
MiniProj2.ino
Output
Serial Monitor
Ln 20, Col 45 Heltec WiFi Kit 32 [not connected]

```
#include <BLEDevice.h>
#include <BLEUtils.h>
#include <BLEServer.h>
#include "heltec.h"
// Each group need to generate their own UUID for tracking service
// See the following for generating UUIDs:
// https://www.uuidgenerator.net/
//#define SERVICE_UUID "4fafc201-1fb5-459e-8fcc-c5c9c331914b"
//#define CHARACTERISTIC_UUID "beb5483e-36e1-4688-b7f5-ea07361b26a8"
#define SERVICE_UUID "85ddc1b8-c75c-11ed-afa1-0242ac120002"
#define CHARACTERISTIC_UUID "85ddc4c4-c75c-11ed-afa1-0242ac120002"
void setup() {
Serial.begin(115200);
Serial.println("Starting BLE work!");
// Display your group name on your screen
Heltec.begin(true /*DisplayEnable Enable*/, false /*LoRa Enable*/, true /*Serial Enable*/);
// Clear the OLED screen
Heltec.display->clear();
// Prepare to display your group name
Heltec.display->drawString(0, 0, "group 6");
// Display the readings
Heltec.display->display();
// Each groups need to spacify their name starting with their group number
BLEDevice::init("6group");
BLEServer *pServer = BLEDevice::createServer();
BLEService *pService = pServer->createService(SERVICE_UUID);
BLECharacteristic *pCharacteristic = pService->createCharacteristic(
CHARACTERISTIC_UUID,
BLECharacteristic::PROPERTY_READ |
BLECharacteristic::PROPERTY_WRITE
);
pCharacteristic->setValue("6group");
pService->start();
```

```
Sketch uses 988594 bytes (75%) of program storage space. Maximum is 1310720 bytes.
```


MiniProj2 | Arduino IDE 2.0.5-nightly-20230316

```
// Display your group name on your screen
Heltec.begin(true /*DisplayEnable Enable*/, false /*LoRa Enable*/, true /*Serial Enable*/);
// Clear the OLED screen
Heltec.display->clear();
// Prepare to display your group name
Heltec.display->drawString(0, 0, "group 6");
// Display the readings
Heltec.display->display();
// Each groups need to spacify their name starting with their group number
BLEDevice::init("6group");
BLEServer *pServer = BLEDevice::createServer();
BLEService *pService = pServer->createService(SERVICE_UUID);
BLECharacteristic *pCharacteristic = pService->createCharacteristic(
CHARACTERISTIC_UUID,
BLECharacteristic::PROPERTY_READ |
BLECharacteristic::PROPERTY_WRITE
);
pCharacteristic->setValue("6group");
pService->start();
BLEAdvertising *pAdvertising = BLEDevice::getAdvertising();
pAdvertising->addServiceUUID(SERVICE_UUID);
pAdvertising->setScanResponse(true);
pAdvertising->setMinPreferred(0x06); // functions that help with iPhone connections issue
pAdvertising->setMinPreferred(0x12);
BLEDevice::startAdvertising();
Serial.println("Characteristic defined! Now you can read it in your phone!");
}
void loop() {
// Put your main code here, to run repeatedly:
BLEDevice::startAdvertising();
Serial.println("Characteristic defined! Now you can read it in your phone!");
delay(2000);
}
```

```
Sketch uses 988594 bytes (75%) of program storage space. Maximum is 1310720 bytes.
```

Working Principle :

- The RSSI values may change based on the distance and obstructions between the ESP32 and the WiFi access point.

- The ESP32 is connected to a WiFi network using this code snippet, which then uses the WiFi.RSSI() function to periodically read the RSSI value from the connected access point. Initializing serial communication for debugging is done with the Serial.begin(115200); line. The ESP32 is linked to the designated WiFi network by the WiFi.begin() function.

- The WiFi.RSSI() function reads the RSSI value and saves it in the rssi variable in the loop() method. The Serial.print() and Serial.println() functions print the RSSI value to the serial monitor for reading. Before reading the RSSI value again, the delay(1000); function inserts a one-second delay.

Distance calculation from RSSI

Distance = 10^((TxPower - RSSI) / (10 * n))

Where,

TxPower is the transmission power of the BLE signal

RSSI is the received signal strength in dBm

n is the path loss exponent, which depends on the environment.

The RSSI values are affected by several factors, such as the transmission power of the BLE signal, the environment, and the obstacles in the signal path. The distance calculation based on RSSI values uses the **Friis transmission equation**, which relates the received power to the transmit power, the distance between the transmitting and receiving antennas, and the environmental factors.

The received power decreases with increasing distance.

Obstacles

- Any physical or natural factors that could prevent or disrupt the accuracy and dependability of indoor positioning systems are referred to as obstacles.

- The signals utilized by positioning technologies like Wi-Fi, Bluetooth, or RFID can be reflected or blocked by walls, floors, and other physical structures that may result in overlapping the signals. This can make it challenging for indoor positioning systems to precisely locate people or objects in real-time.

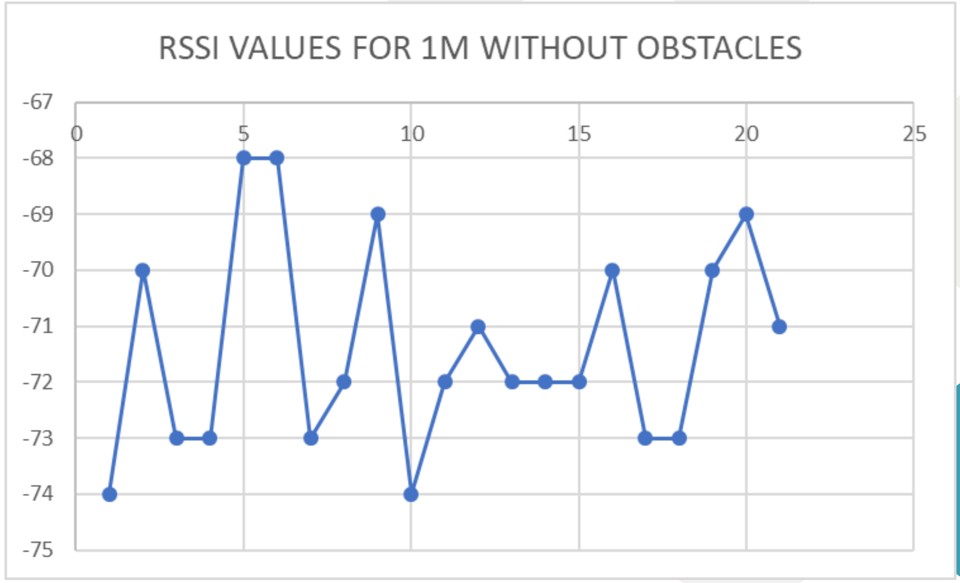
- Other challenges could come in the form of electronic interference from other devices like temperature and humidity variations, which could affect the accuracy of indoor positioning systems.

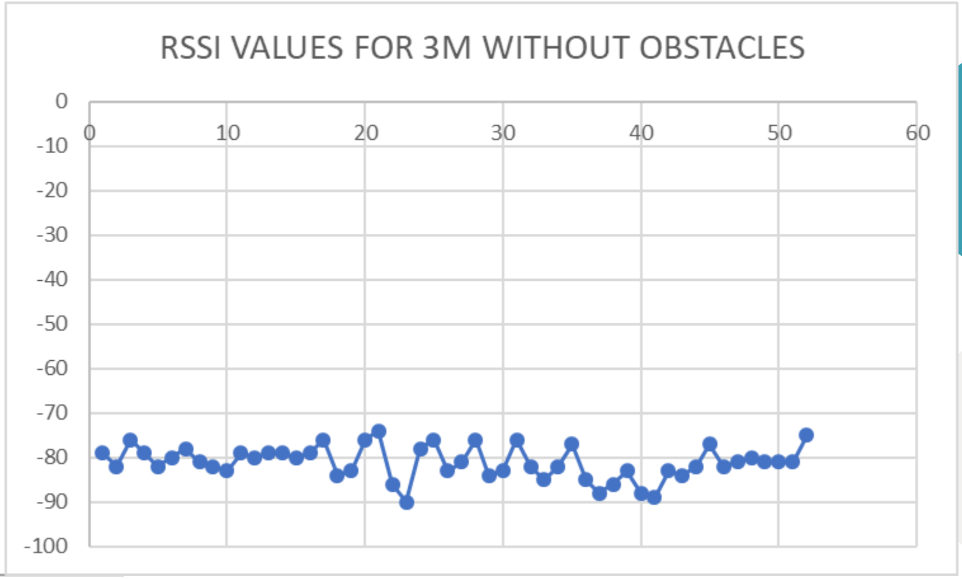
RSSI strength with a varying distance of 1m and 3m has been determined in the following scenarios :

1. Results without obstacles
2. Results with water as an obstacle
3. Results with Wall as an Obstacle
4. Results with 1 Human as an Obstacle
5. Results with 2 Humans as an Obstacle
6. Results with 3 Humans as an Obstacle

Results without Obstacles

- Mean RSSI for 1 meter: -74.66, Standard Deviation: 5.1428.
- Mean RSSI for 3 meters: -84.04, Standard Deviation: 3.765
- Graphs and charts depicting the results



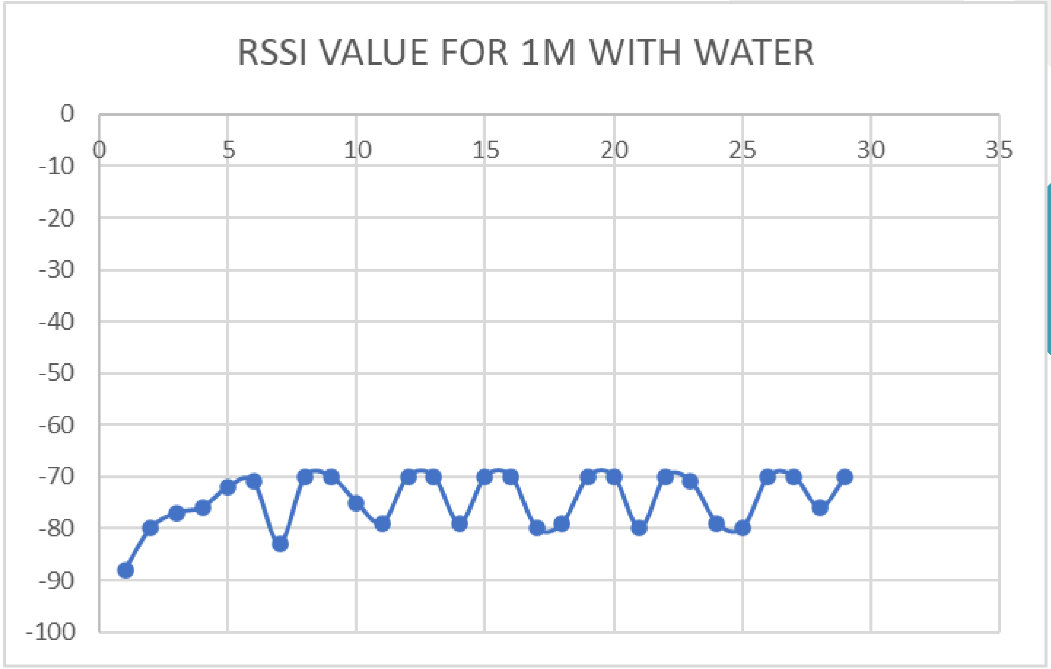


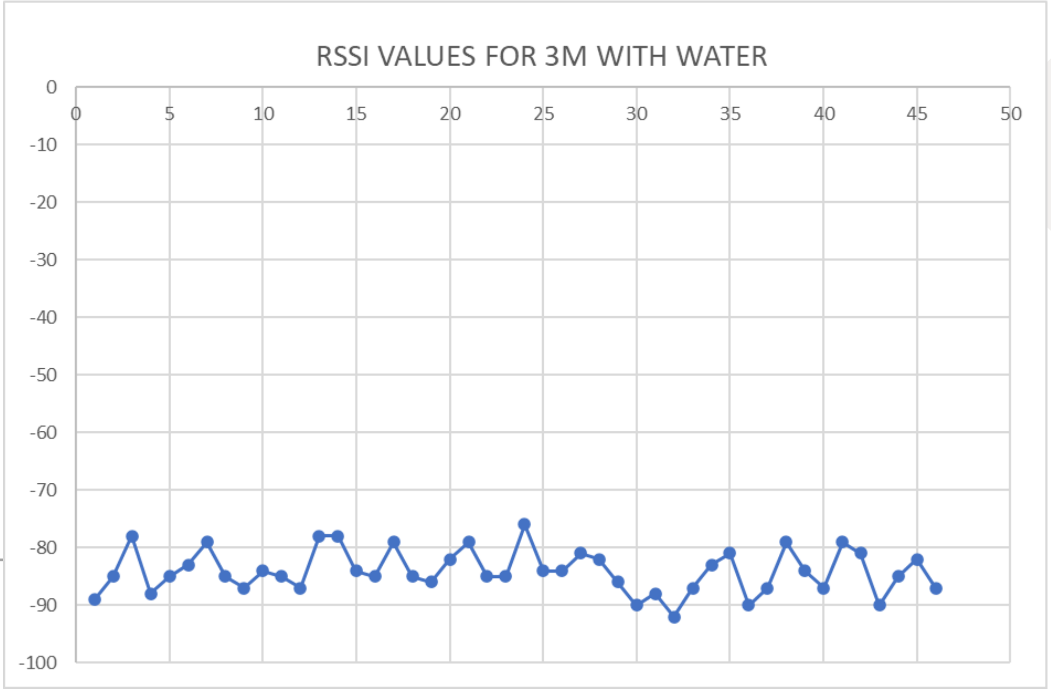
Results with Water as obstacle

- Mean RSSI for 1 meter: -71.381, Standard Deviation: 1.8567.

- Mean RSSI for 3 meters: -81.076, Standard Deviation: 3.6933

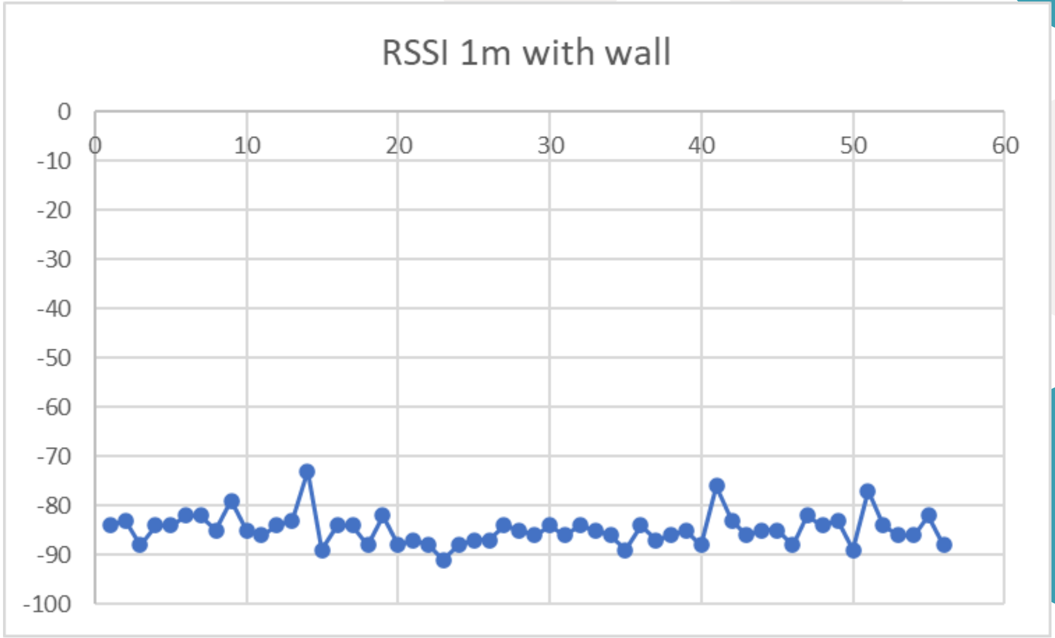
- Graphs and charts depicting the results

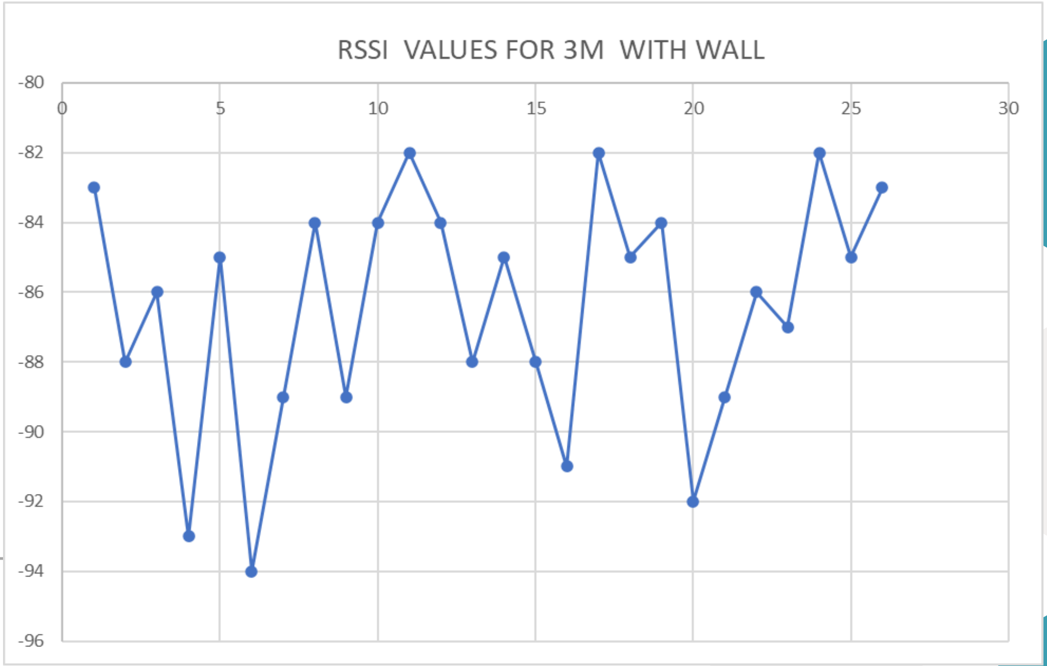




Results with Wall Obstacle

- Mean RSSI for 1 meter: -84.78, Standard Deviation: 3.234
- Mean RSSI for 3 meters: -86.46, Standard Deviation: 3.432
- Graphs and charts depicting the results



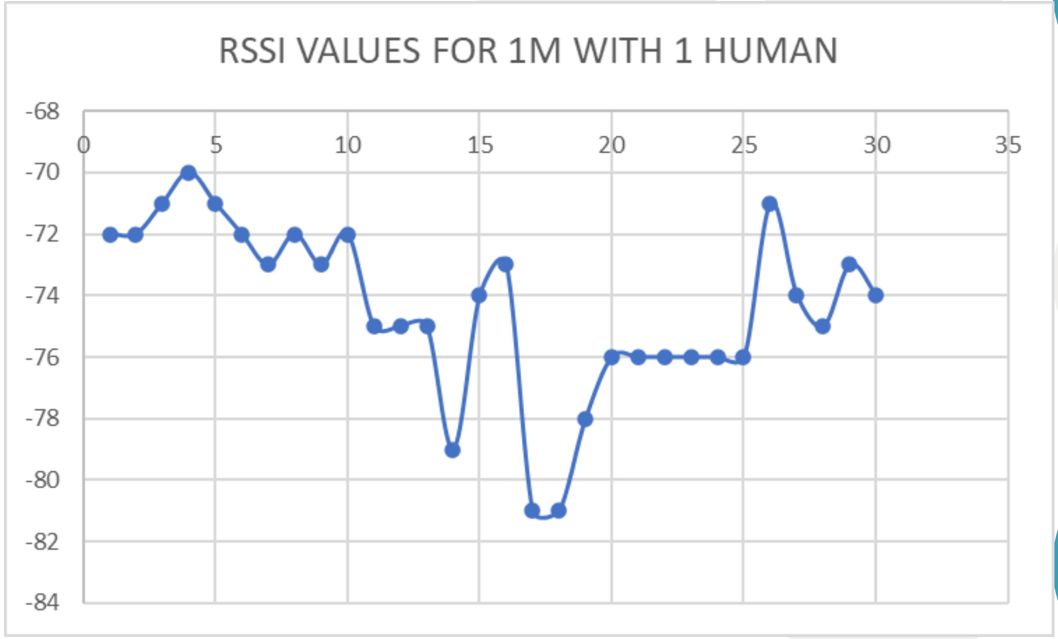


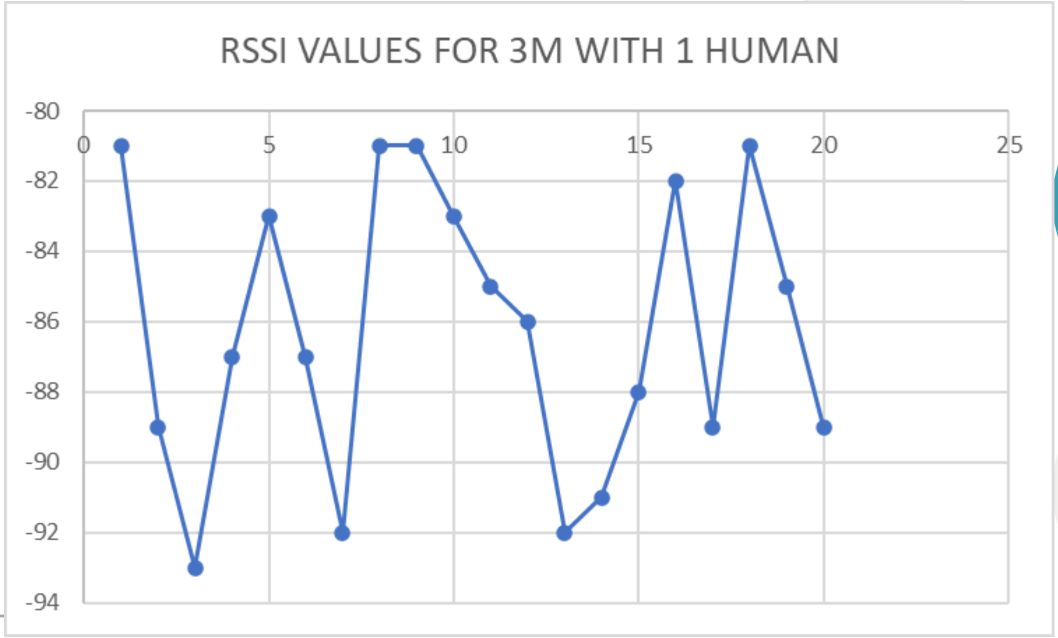
Results with 1 Human Obstacle

- Mean RSSI for 1 meter: -74.4, Standard Deviation: 2.8113.

- Mean RSSI for 3 meters: -86.25, Standard Deviation: 4.064

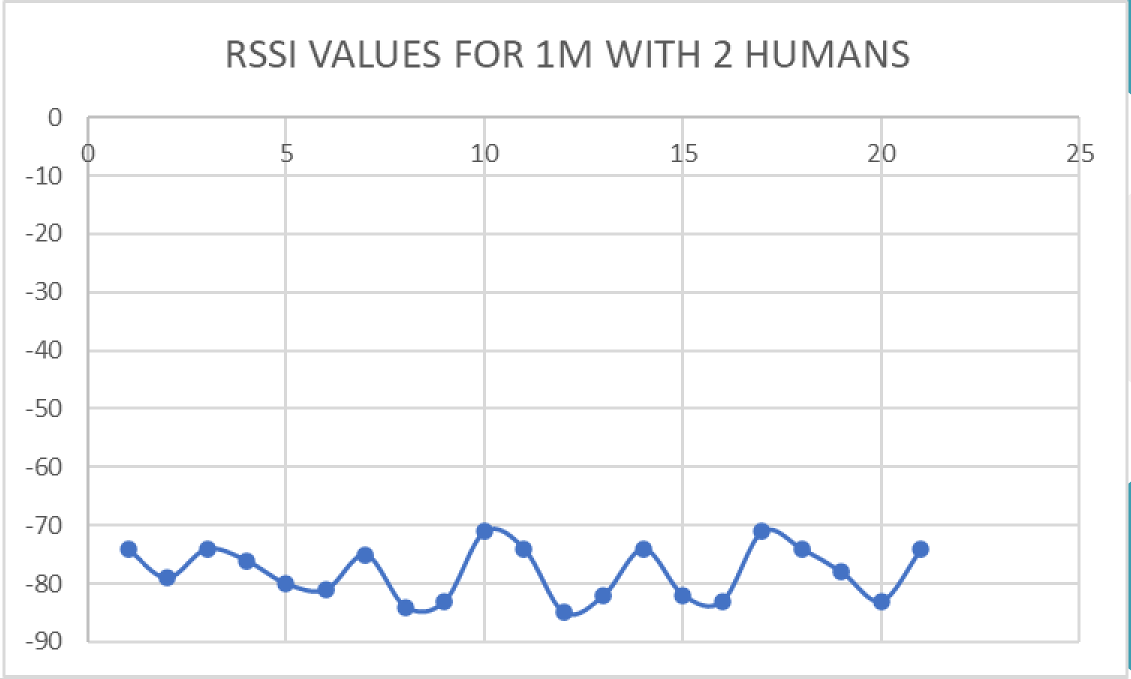
•Graphs and charts depicting the results

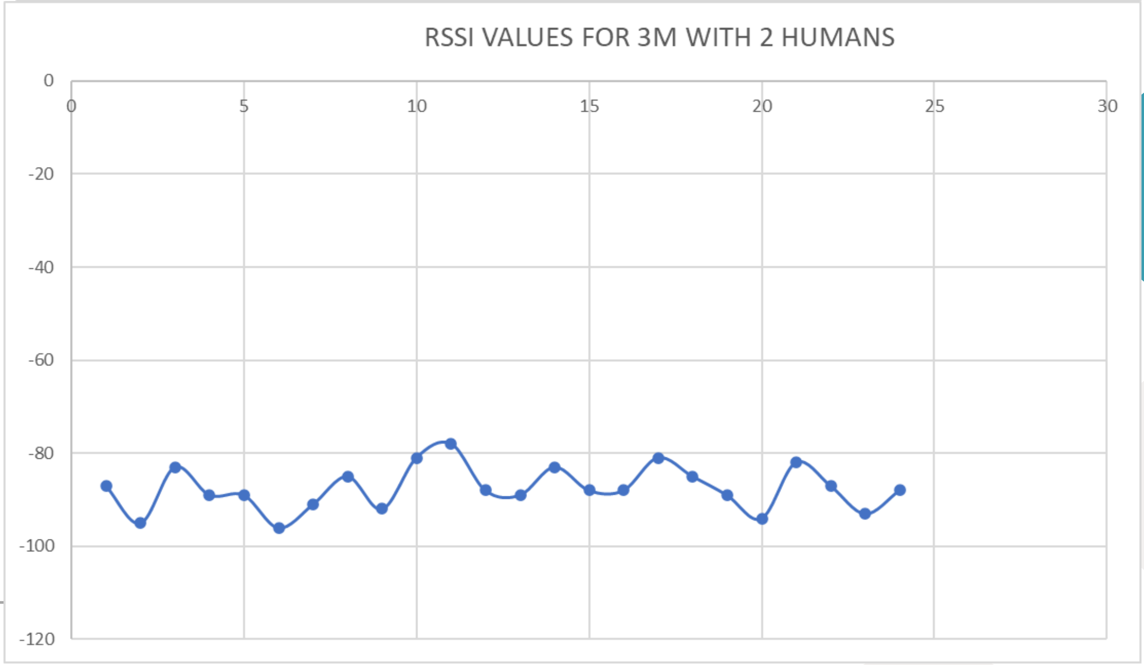




Results with 2 Humans Obstacle

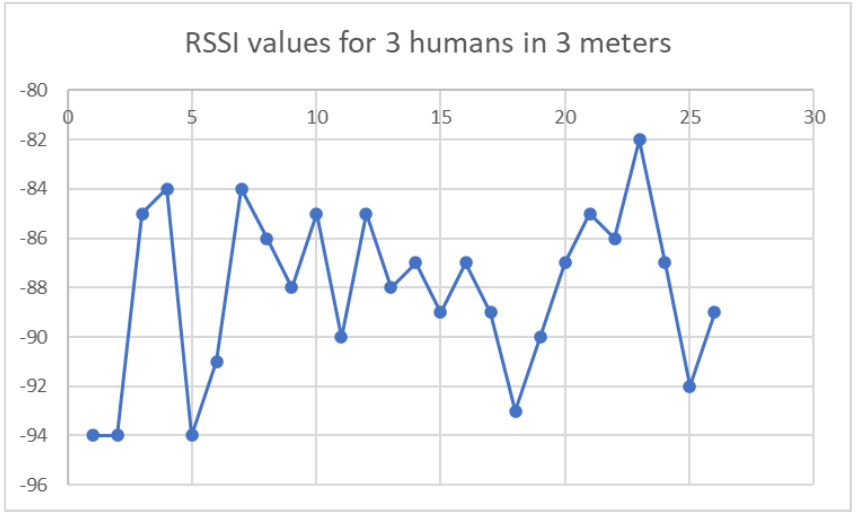
- Mean RSSI for 1 meter: -77.95, Standard Deviation: 4.544.
- Mean RSSI for 3 meters: -87.54, Standard Deviation: 4.672
- Graphs and charts depicting the results

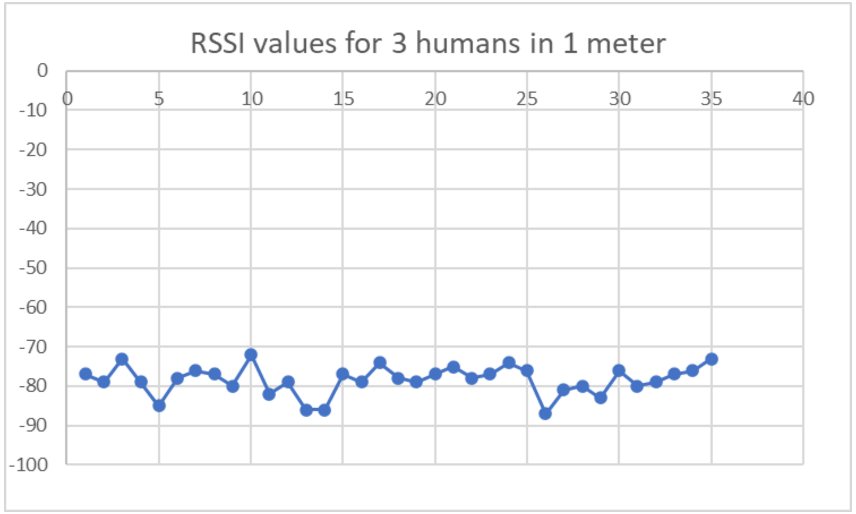




Results with 3 Humans Obstacle

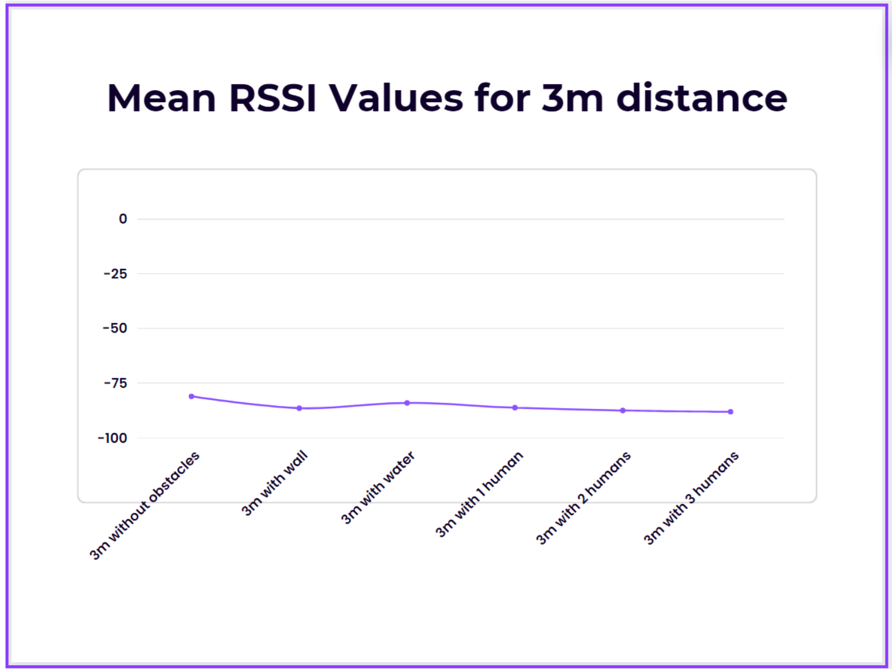
- Mean RSSI for 1 meter: -78.428, Standard Deviation: 3.712
- Mean RSSI for 3 meters: -88.115, Standard Deviation: 3.374
- Graphs and charts depicting the results

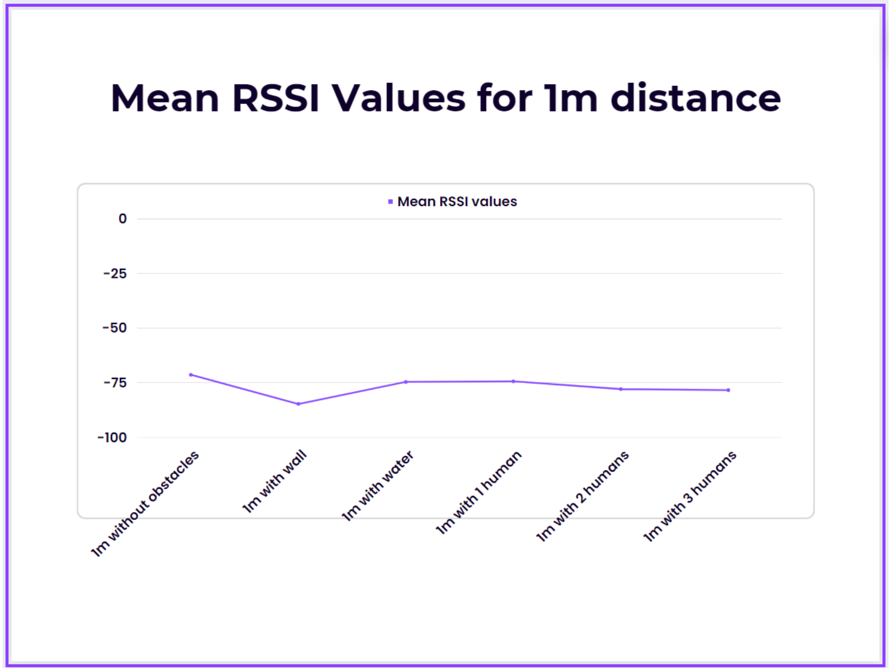




Insights

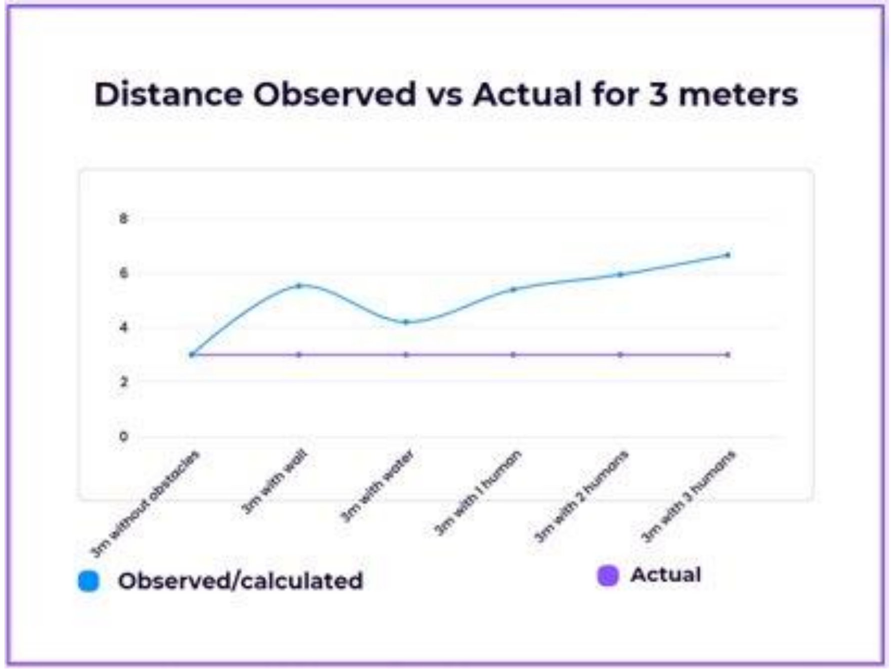
- Comparison of mean values with and without obstacles for 1meter and 3 meters

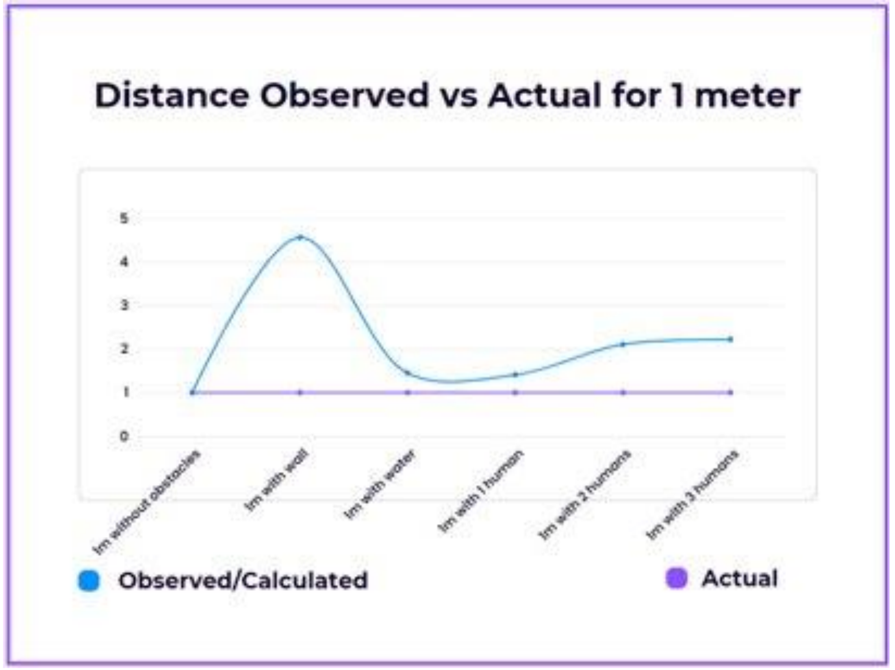




Insights

- Comparison of distance values with and without obstacles for 1 meter and 3 meters





Observations/Inferences

OBSTACLE	1 Meter	3 Meters	Difference in distance(Actual-Observed)
Water	1.4	4.19	~1 m
1 human	1.4	5.3	~1 m
2 humans	2.1	5.9	~1.1 m
3 humans	2.4	6.6	~1.4 m
Wall	4.5	5.5	~2.5 m

Conclusion and Future Scope

- In conclusion, this project has demonstrates the feasibility of using an IoT-based positioning system using ESP32 and a smartphone application to calculate distances based on RSSI values of BLE signals. We have analyzed the effect of interference with respect to obstacles such as water, walls, and humans. This helped us determine an accurate distance computation in a real-time scenario.

- For future scope, this project can be extended in several ways. One possible direction is to use machine learning algorithms to improve the accuracy of the distance calculation based on RSSI values. Another direction is to integrate the positioning system with other IoT devices, such as sensors and actuators, to create a smart environment.

- Additionally, the system can be enhanced to include features such as real-time tracking, mapping, and location-based services. This can be useful in various applications, such as asset tracking, indoor navigation, and healthcare monitoring.

Thank You