AUTOMATED WI-FI AND BLUETOOTH CONTACT TRACING SYSTEM WITH NO USER INTERVENTION

[1] SUDHEER RAJA VENISHETTY,[2] BHARATHA SATHEESH,[3] VUPPULA MANOHAR,
[4] DOMAKONDA HARSHINI,[5] BHONAGIRI JYOTSHNA
[1]Associate Professor, [23]Assistant Professor,[45]Students
Department of ECE
Vaagdevi College of Engineering, Warangal, Telangana

ABSTRACT

A custom Wi-Fi and Bluetooth contact tracing system is created to find detailed paths of infected individuals without any user intervention. The system tracks smartphones, but it does not require smartphone applications, connecting to the routers, or any other extraneous devices on the users. A custom Turtlebot3 is used for site surveying, where it simulates mobile device movement and packet transmission. Transmit power, receive power, and round trip time are collected by a custom ESP32C3 router. MAC randomization is defeated to identify unique smartphones. Subsequently, the wireless parameters above are converted to signal path loss and time of flight. Bidirectional long short term memory takes the wireless parameters and predicts the detailed paths of the users within 1 m. Public health authorities can use the contact tracing website to find the detailed paths of the suspected cases using the smartphone models and initial positions of confirm cases. The system can also track indirect contact transmissions originating from surfaces and droplets due to having absolute positions of users.

***K*eywords** Received signal strength indicator (RSSI), Round trip time (RTT), Fine time measurement (FTM), Wi-Fi indoor localization, Bluetooth indoor localization

1 Introduction

When a new outbreak appears with unknown pathogens, vaccines and treatments are not available immediately to reduce the spread of the disease. Therefore, governments and public health agencies use extensive disease testing to identify infected individuals. However, testing the entire population is inefficient because of the limited testing capacity, false negative cases, and the associated costs. Contact tracing has been developed to make efficient use of the limited testing resources, where the closest contacts of the confirmed cases or symptomatic cases are tested and isolated.

1.1 Contact Tracing

Contact tracing is difficult because super-spreaders could infect thousands of people a day [1] and exponentially increase the number of people in the contact tracing list. Traditionally, contact tracing has been done by hand, where the authorities interview each confirmed case to get the contacts and visited places. Afterwards, suspected cases are isolated and tested. Symptomatic cases and high exposure cases get a higher priority in testing. With a high enough contact tracing efficiency, diseases can be locally contained and sometimes be eradicated [2]. However, performing contact tracing manually is very inefficient because the infected people might forget who they met and where they visited. Staff shortages, incorrect training, and slow turnaround times can also cause inefficient contact tracing.

Many countries have moved to automated means of contact tracing [3–5] via smartphones, cameras, custom tracking devices, or genome sequencing. Cameras can be used in-conjunction with facial recognition software to track individual people. Researchers collected a database of faces and applied a convolutional neural network (CNN) to classify the presences of the people in the database [6]. They are able to perform contact tracing via a web interface. Instead of only classifying faces, other researchers have used multiple cameras to track movements in real time [7]. Furthermore, they can determine the actual paths of the confirmed cases for contact tracing.

Genome sequencing enables contact tracing without interviewing patients or requiring tracking devices. This particularly useful for incapacitated or unidentified patients. J. Gardy et al. [8] applied hierarchical clustering to sequenced genomes in order to create a genome tree of a tuberculosis outbreak. Moreover, the genome tree perfectly matches the contact traced social network created from patient questionnaires. The main disadvantage of genome sequencing is the genome tree can only be created after the patients are infected.

On the other hand, smartphones are readily available and can be used for tracking the movements of individuals. Thus, many governments, public health agencies, and software companies have implemented smartphone applications for contact tracing. The Singaporean government released one of the first contact tracing applications for COVID-19 [9]. It

uses Bluetooth Low Energy (BLE) to exchange temporary IDs between two smartphones within a certain range. If there is a confirmed case based on the temporary IDs received, then the application warns the user about a COVID-19 exposure. Other researchers have used similar approaches for contact tracing [10–14]. Alternatively, T. Yasaka et al. [15] used QR codes for tracking social gatherings between groups of people. The host of the social gathering creates a QR code using the application, and the participants scan the QR code to build a time series graph. When a user indicates a positive test result, all users within 3 traversals of the time series graph are notified. A few more research papers have used the QR code approach [16–18]. Moreover, the smartphones' GPS can be used to track users in the outdoor environments [19]. This would provide a higher position accuracy than BLE and QR codes.

Wi-Fi can be a useful tool for localization and contact tracing. A. Trivedi et al. [20] developed a Wi-Fi based contact tracing system without the need to install an application onto the smartphone. They used the access points (APs) of two universities to collect packets from smartphones, where the user's trajectory is built using the closest APs. Furthermore, a graph search algorithm takes the user's trajectory and produces a location and proximity report of the exposed users. Other research groups have used Wi-Fi based smartphone applications [21] to capture beacon frames from nearby APs and upload the data to the cloud. This allows the authorities to track the visited places and the positions of the confirmed cases. Moreover, the lifespan of the disease can be known due to the recorded timestamps of the beacon frames.

1.2 Indoor localization

Localization is fundamental to contact tracing, and it has two major categories: outdoor and indoor localization. Out of all the outdoor localization methods, Global Positioning System (GPS) is the most popular and is robust against signal interference and jamming [22]. However, GPS requires direct line-of-sight (LoS) between the satellites and the handset, which is unsuitable for indoor localization.

Indoor localization has drawn more attention in the industry for its wide variety of use cases, such as autonomous indoor vehicles (AIVs) [23], unmanned aerial vehicles (UAVs) [24], home automation, and smart buildings [25]. Radio Frequency (RF) waves penetrate materials like tables and walls, making RF-based indoor localization the most adopted solution. Moreover, RF performs better than other methods [26]. RF-based systems employ mobile phones for capturing wireless parameters such as angle of arrival (AOA), time of arrival (TOA), and received signal strength indication (RSSI). These parameters act as fingerprints for positioning. With the help of machine learning, the average localization error is around 1 m [27, 28].

Our interests lie in estimating the positions of people to determine COVID-19 exposures. As a result, indoor localization is more useful than outdoor localization due to indoor environments having a higher infection rate [29]. Moreover, indoor localization is extremely helpful for tracking COVID-19 outbreaks in complex environments such as supermarkets and airports.

1.3 Privacy and MAC Address Randomization

Privacy is a very important aspect to keeping collected information safe and within regulations with Canadian and British Columbia Privacy Acts. In this system, phone numbers, email addresses, and names are not collected by the routers and are not stored in the database. Only the MAC addresses of the Wi-Fi chipsets are obtained as the identification of the smartphones. Moreover, users that enter a building with the contact tracing system in place need to be aware of what the system does and actively consent to their data being collected. Users must also have the ability to request the data collected from their phone MAC address to be erased by overwriting the data with random bits. The collected data such as MAC addresses, smartphone model specific information, and phone positions are encrypted with AES256. Note that MAC addresses can not directly identify users due to MAC address randomization.

1.4 Features and Differences of the Proposed Contact Tracing System

Previous Wi-Fi and BLE contact tracing systems require significant user intervention to succeed. Some contact tracing systems require the users to manually install smartphone applications [9–14], of which results in a low uptake and an ineffective system. Commercial off-the-shelf (COTS) routers and COTS contact tracing software [20] require users to manually join wireless networks. However, smartphones often disconnect from wireless networks when they go to sleep. As a result, this creates large position gaps in COTS contact tracing software. Other research groups have developed custom user equipment that periodically transmit beacons to mitigate the problem. However, this method is costly because every user needs to buy a device. The contact tracing systems listed above can only determine the relative distances between two devices, while the absolute positions of the users are unknown. This poses a problem because the public health authorities require absolute positions to track diseases that can survive on surfaces and in ventilation systems for many days.

Unlike the manual and relative positioning systems above, we propose a Wi-Fi and BLE contact tracing system for finding the absolute paths of the infected individuals without any user intervention. The proposed system does not

require the users to install smartphone applications or to join wireless networks. Custom user equipment is not needed as the system captures Wi-Fi and BLE packets from smartphones, smartwatches, tablets, laptops, and BLE headphones. We transmit RTS packets to the devices to make them respond at a higher rate, even if their screens might be off. Moreover, the localization algorithms are able to determine the absolute positions and exact timings of the users. As a result, we can use disease half lives to track indirect contact transmissions originating from surfaces and droplets.

The paper is organized as follows. Section 2 describes the components of the proposed contact tracing system. The collection process of the wireless parameters and the overall database are explained in Section 3. Section 4 shows the results and discussions of identifying unique mobile devices, localization performance, and contact tracing. A conclusion is presented in Section 5.

2 Proposed Contact Tracing System

Fig. 1 shows the contact tracing system diagram. The system has 5 main components: Turtlebot3 [30], ESP32C3 routers, packet processing, localization algorithms, and contact tracing algorithms. Firstly, the Turtlebot3 [30] executes autonomous site surveys by visiting all positions within the indoor environments. A smartphone is mounted on the robot, and it broadcasts wireless packets while moving in order to simulate mobile device trajectories. Secondly, the ESP32C3 routers transmit RTS packets and BLE pings to increase the response rates of the mobile devices. Simultaneously, the routers capture and store all Wi-Fi and BLE packets onto internal SD cards. At the end of the day, the routers move the data to the master server for packet processing. Thirdly, the master server extracts transmit power (TX power), received signal strength indication (RSSI), and fine time measurement (FTM) from the packets. Time of flight (ToF) is computed from FTM, while signal path loss is derived from TX power and RSSI. Fourthly, bidirectional long short term memory (BiLSTM) neural networks are used for predicting the trajectories of mobile devices. One BiLSTM is trained using a time series of ToF, while the other is trained using a time series of signal path loss. Finally, we build a contact tracing graph using the contact tracing algorithms. Every user is assigned to a node on the contact tracing graph. For every trajectory intersection between the trajectory of a confirmed case and the trajectory of a user, we add an edge that connects the node of the confirmed case to the node of the user. After repeating the process above, a graph of the suspected cases is generated.

2.1 Turtlebot3 for Site Survey

Typically, mobile devices transmit many Wi-Fi and BLE packets as they move around the building. By implementing packet sniffing on the router side, mobile devices can be tracked throughout the day. However, the localization algorithms require large amounts of training and testing data. Measurement of the training data is done in the form of a site survey, where a mobile device is placed at every position and the packet information is captured at the router side.

Collecting data by hand is extremely tedious and introduces position errors. Instead, we built a custom Turtlebot3 [30] to autonomously collect data for the site survey. The original Turtlebot3 has a height of 19 cm, which is too short for the height of a smartphone on a table or in a user's pocket. A platform is added to the custom Turtlebot3 in order to increase the smartphone's height to 75 cm. Moreover, the custom Turtlebot3 is also equipped with RPLIDAR A2, Intel D415 RGBD camera, Nvidia Jetson TX2, and Raspberry Pi 3.

Robot Operating System 2 (ROS2) [31] is an open-source robotics framework that collects sensor information, executes data processing, implements inter-process communications, and allows real-time control. ROS2 has four main concepts: nodes, topics, services, and actions. Nodes are individual processes, of which execute a singular task like collecting sensor data or filtering information. Nodes can commence one way communications with other nodes by publishing messages to topics. All nodes that subscribe to a specific topic receive the same messages. Unlike topics, services are a two-way communications channel. Nodes can send service requests and receive service responses once the specific operation is completed. Actions are an extension of services, where the nodes receive periodic feedback status messages instead of not receiving feedback messages.

The RPLIDAR A2 is a 2D laser ranging device that measures the distances to the nearest opaque objects. It is a 360° LIDAR that completes 1 revolution every 0.1 seconds. The 360° scans are divided into 360 angle intervals. For each angle interval, the RPLIDAR A2 returns a distance value. The laser scans feed into ROS2 SLAM_toolbox, of which it produces a 2D grid map and publishes the transform from map to odometry (odom). It essentially determines the position and orientation of the Turtlebot3. On the other hand, the Intel D415 RGBD is used for obstacle avoidance. The D415 produces a RGBD point cloud at 720p 30 frames per second (FPS). Camera sensors have false positive readings, where the sensor outputs a ghost point in the absence of objects. In order to eliminate the false positives, multiple RGBD point cloud frames are joined together and are uniformly decimated. Afterwards, the point cloud is organized into clusters, where the cluster centres and standard deviations are calculated. If a point is 1 standard deviation away from the cluster centre, then it is removed.

Algorithm 1: Robot Path Planing and Navigation

```
Input  :LIDAR laser scans and camera point clouds
Output :Moves the robot to sampling positions along the planned path
VisitedPositionList ← [];
while True do
    Odom ← getOdom();
    LaserScans ← getLIDAR();
    PointCloud ← getCamera();
    Costmap ← newCostmap(Costmap,LaserScans);
    Costmap ← newCostmap(Costmap,PointCloud);
    Rays ← Costmap - Odom;
    StartPositionList ← [];
    for each Ray in Rays do
        Check ← checkObstacles(Costmap,Ray);
        if Check.hasObstacles() == False then
            Position ← selectRandomPosition(Ray);
            StartPositionList.append(Position);
        end if
    end for
    BestPath ← initPath();
    BestPathScore ← 0;
    for each StartPos in StartPositionList do
        for each EndPos in Costmap do
            Path ← createPath(StartPos,EndPos);
            Check ← checkObstacles(Costmap,Path);
            if Check.hasObstacles() then
                continue;
            end if
            PathScore ← computeScore(Path,VisitedPositionList);
            if PathScore > BestPathScore then
                BestPathScore ← PathScore;
                BestPath ← Path;
            end if
        end for
    end for
    for each Position in BestPath do
        Check ← checkObstacles(Costmap,Position);
        if Check.hasObstacles() then
            moveRobotToPosition(Odom);
            break;
        end if
        moveRobotToPosition(Position);
        sleep();
        VisitedPositionList.append(Position);
    end for
end while
```

Algorithm 2: Pseudocode of Contact History Generation Algorithm

```
Input  : TargetTraceList, OtherTraceList
Output : ContactHistoryList
Function GenerateContactHistory();
Initialize Hash map of time for pair matching;
Initialize Hash map of distance for pair matching;
Initialize ContactHistoryList;
Max_Distance ← 15;
Time_Resolution ← 30;
for each TargetTrace in TargetTraceList do
    Find the DistanceMap list where firstCell = TargetTraces.cell and distance < Max_Distance;
    startTime ← TargetTrace.time + 0.5Time_Resolution;
    endTime ← TargetTrace.time - 0.5Time_Resolution;
    for each OtherTrace in OtherTraceList do
        if startTime < OtherTrace < endTime and TargetTrace.siteId = OtherTrace.siteId and the distance between
          TargetTrace.cell and OtherTraces.cell appears in the DistanceMap List then
            add TargetTrace.time and distance to HashMap with key = TargetTrace.macAddress +
              OtherTrace.macAddress + TargetTrace.siteId;
        end if
    end for
    for each Key and value in HashMap do
        Initialize new ContactHistory;
        firstMacAddress ← key.Target TraceMacAddress;
        secondMacAddress ← key.Other TraceMacAddress;
        siteId ← key.siteId;
        contactDuration ← size of value;
        lastContactTime ← last element in value of time HashMap;
        Assign each element in value of distance HashMap to corresponding distance_Range;
        Calculate contact_Distance_Avg and contact_Distance_Min using distance HashMap value;
        Add ContactHistory to ContactHistoryList;
    end for
end for
return ContactHistoryList
```

3 Database

Multiple datasets were collected at the University of Victoria, Victoria, British Columbia, Canada in the Engineering Office Wing (EOW) 3rd floor, EOW 4th floor, Engineering Computer Science (ECS) 1st floor, and ECS 5th floor. At each floor, the Turtlebot3 physically moves along 3 unique trajectories. Every trajectory contains unique positions that the other trajectories do not have. One of the trajectories is randomly selected for the training dataset, while another is selected for the testing dataset. The remaining trajectory is appended to the cross-validation dataset. Furthermore, we artificially generated more training trajectories using the data points from the training dataset as described in Section 2 Subsection D. However, we did not create artificial trajectories using the testing and cross-validation datasets. The quality of the data collected by the ESP32C3 routers is unknown, thus we used COTS routers as a reference to validate the quality of the data from the ESP32C3 routers. The data collected are organized into two main groups: Dataset A is collected using the COTS routers and Dataset B is collected using the ESP32C3 routers.

Dataset A contains the packets collected by COTS routers. At every position, at least 50 samples are obtained by the routers. Each sample contains a timestamp, X position, Y position, θ orientation, Wi-Fi RSSI, Wi-Fi SQI, BLE RSSI, and BLE TX power. For the EOW 3rd floor, 11 Wi-Fi routers and 7 BLE routers are deployed to obtain the dataset. Note that some Wi-Fi routers share the same locations as the BLE routers. The raw dataset contains approximately 500,000 samples at 1,000 different positions, where each sample has 31 wireless parameter features. For the EOW 4th floor, 9 Wi-Fi routers and 7 BLE routers are deployed to obtain the dataset. There are fewer routers on this floor due to the lack of power outlets. The raw dataset contains approximately 300,000 samples at 600 different positions, where each sample has 29 wireless parameter features. For the ECS 1st floor, 7 Wi-Fi routers and 7 BLE routers are deployed to obtain the dataset. The raw dataset contains approximately 200,000 samples at 1000 different positions, where each sample has 24 wireless parameter features. For the ECS 5th floor, 8 Wi-Fi routers and 6 BLE routers are deployed to obtain the dataset. The raw dataset contains approximately 300,000 samples at 600 different positions, where each sample has 24 wireless parameter features extracted from the packets.

Dataset B is sampled at the same locations and with the same procedures as Dataset A. However, Dataset B uses ESP32C3 routers, and it provides a new wireless feature known as Wi-Fi FTM. Moreover, the ESP32C3 routers occupy 40 MHz bandwidth instead of the 20 MHz bandwidth in Dataset A. These new additions increase the localization accuracy of the BiLSTM and the precision of the contact tracing algorithm.

4 Results and Discussion

4.1 Identifying Unique Mobile Devices from Random MAC Addresses

Table 1: Test Results of the Clustering Algorithm for Identifying Unique Mobile Devices from Random MAC Addresses.

Bucket	Device	MAC Addresses
0	Galaxy S4	D0:22:BE:F5:7C:B4
1	HTC One X	E8:99:C4:99:57:24
2	Galaxy S6	4E:0F:A0:57:F8:75, 26:45:19:1E:D5:FE, 1A:5B:0A:B1:7D:4A, 0E:BF:6D:4D:ED:A7, 42:B2:3B:14:49:F9, 1A:CF:16:13:A2:CB, 8C:F5:A3:3D:16:DA, 3A:DC:D3:0A:46:B6
3	Galaxy A11	6A:E0:23:0C:20:0F, 56:5C:AC:D6:13:30, E2:01:19:D0:64:2D, FA:05:BB:EA:47:2D, A0:27:B6:EE:6A:A7, 7E:69:90:C6:C4:04, DA:00:FD:35:82:25, 56:2F:2B:64:BC:C5, F6:08:C4:AF:61:94, 16:0D:FA:80:F8:1F, 5E:99:98:7B:5A:BF, 96:96:27:97:22:4C, FE:CB:1A:2E:F5:9A, B2:78:9D:5C:B9:1A 16:3C:FC:DF:1C:CA, 96:38:7C:5D:20:5C
4	iPhone SE	82:31:01:8A:F3:AD, AE:9E:BE:7A:F3:D3, A6:E9:93:A7:9D:3E, D2:C5:A7:8B:9E:2C, 46:33:10:CE:43:3B, AA:CB:57:97:5E:5F, 1A:40:6D:01:B4:05, 96:C2:5B:09:D8:4E, 3A:E6:E3:9B:8E:6D, 56:D3:41:61:0B:0A, A2:68:13:44:B2:EF, 8E:6A:CF:EF:6E:1F, 56:A2:4A:EE:D4:46, E2:F7:83:DC:1E:E4, 22:29:5A:0D:F3:24, B6:33:3F:4F:89:1A, 9E:3D:78:F4:38:5D, ...
5	iPhone X	86:98:6E:73:89:1D, 86:AD:C7:47:02:39, 3E:6F:2D:B3:4D:BB, 76:8A:CB:74:73:90, 9A:2D:E5:A8:F1:5A, 76:34:D2:C0:89:71, FE:B8:15:02:43:7C, 76:55:81:98:C3:78, CE:56:BC:E7:3E:72, 46:C9:78:16:41:B6, BE:F2:DB:37:1A:8A, 2A:2F:3E:B1:C7:A0, 6E:4C:1E:F1:8E:E8, A2:97:F2:BA:2A:D5, 6A:DD:55:59:2E:68, DA:D2:D1:55:18:60, F2:C5:62:AD:29:04, ...
6	PinePhone	7A:26:59:B4:C6:6D, D6:8A:05:6A:62:F5, 96:61:D9:88:25:45, 7E:53:9C:5F:BE:D0, B6:13:70:3F:28:C9, 0A:70:BB:F2:2D:9D, 52:A2:3B:BD:6D:DF, D6:DB:E0:37:8C:CE, 62:0B:F8:3C:3A:E2, BA:12:FB:78:53:F4, A2:3E:F7:DF:14:03, BE:B6:47:61:BC:31, EA:14:84:75:F9:00, 8E:B7:F1:4D:1A:FC, C6:41:5C:E2:C6:7B, 92:47:59:89:C4:37, CA:DC:C0:CF:39:FB, ...

In this section, the effectiveness of the clustering algorithm for identifying unique mobile devices from random MAC addresses is tested. For the test setup, MAC address randomization is enabled on the devices, and they are forced to join a wireless network. Every time a mobile device joins a new wireless network, the operating system generates a new

random MAC address for that specific network. Ground truth MAC addresses are obtained by looking at the settings menu. Simultaneously, the devices' probe request packets are captured at the router side. Afterwards, the clustering algorithm is applied to the probe requests to identify unique mobile devices from random MAC addresses.

Table 1 shows the results of the clustering algorithm on the testing dataset. Each row of the table contains a single bucket, of which each bucket contains MAC addresses that belong to the same mobile device. Galaxy S4 is loaded with LineageOS 16, of which does not have MAC address randomization. Table 1 shows the clustering algorithm assigning Galaxy S4's single MAC address to a single bucket and MAC addresses from other devices are not present in that bucket. The result matches the Galaxy S4's ground truth MAC address. Similar to the Galaxy S4, the HTC One X does not have MAC randomization, and it results in a single MAC address found in Table 1. However, Galaxy S6 is loaded with LineageOS 18.1, and it generates a new random MAC address upon joining a new wireless network. Galaxy S6 is forced to join 7 different wireless networks, and the clustering algorithm places all 7 of the Galaxy S6's random MAC addresses in the same bucket. Note that the clustering algorithm has 100% accuracy because all the MAC addresses in Galaxy S6's bucket in Table 1 matches all the MAC addresses in the ground truth. On the other hand, Android 11 on Galaxy A11 adds a new feature that randomizes MAC addresses while scanning for nearby SSIDs. The exact same test is performed on the Galaxy A11, of which the ground truth MAC addresses matches the clustering result found in Table 1. Note that the extra MAC addresses of Galaxy A11 are generated when scanning for SSIDs. We have also observed that Android only generates a new random MAC address on the first network connection. Rejoining a previously connected network yields the same MAC address.

The iPhone SE supports MAC address randomization because it has iOS 15.1 firmware. For the test, iPhone SE is forced to join 7 different wireless networks, and the clustering algorithm places all 7 of the iPhone SE's random MAC addresses in the same bucket. Again, the clustering algorithm achieves 100% accuracy because all the ground truth MAC addresses are found at the iPhone SE's bucket in Table 1. The extra MAC addresses found in the iPhone SE's bucket are generated when scanning for nearby SSIDs. The iPhone X's results are the same as the iPhone SE's results because they both have the same iOS 15.1 firmware.

The full Arch Linux distribution is installed onto the PinePhone, of which allows full control over MAC address randomization. We wrote a script to generate 25 new random MAC addresses and to save them into a file as the ground truth. Afterwards, the clustering algorithm's results found in Table 1 are compared to the ground truth. The clustering algorithm is able to place all the PinePhone's MAC addresses into the same bucket without any other MAC addresses from other devices being there. Overall, the clustering algorithm correctly classified every single test device into their respective buckets, even though their MAC addresses are randomized. However, any two mobile devices that have the same model number at the same spacetime might cause the system to produce incorrect results. This is due to identical devices producing indistinguishable probe request information and RSSI information.

4.2 Localization Algorithm Performance on Dataset A

Table 2: Dataset A: BiLSTM's localization performance at different locations.

Location	Method	APs	RMSE (m)	MAE (m)	Training Time (s)	Testing Time (μs)
EOW 3rd Floor	BiLSTM+Wi-Fi	11	0.83	0.58	3.67	420
EOW 3rd Floor	BiLSTM+BLE	7	0.88	0.56	3.12	399
EOW 3rd Floor	BiLSTM+Wi-Fi+BLE	18	0.82	0.58	4.34	445
EOW 4th Floor	BiLSTM+Wi-Fi	9	0.92	0.61	3.55	410
EOW 4th Floor	BiLSTM+BLE	7	0.93	0.63	3.02	402
EOW 4th Floor	BiLSTM+Wi-Fi+BLE	16	0.84	0.60	3.96	405
ECS 1st Floor	BiLSTM+Wi-Fi	7	1.69	1.42	4.72	340
ECS 1st Floor	BiLSTM+BLE	7	2.13	1.73	3.31	361
ECS 1st Floor	BiLSTM+Wi-Fi+BLE	14	1.30	1.03	4.32	417
ECS 5th Floor	BiLSTM+Wi-Fi	8	0.83	0.62	3.72	370
ECS 5th Floor	BiLSTM+BLE	6	0.87	0.68	3.38	373
ECS 5th Floor	BiLSTM+Wi-Fi+BLE	14	0.92	0.63	4.13	391

Note: Wi-Fi implies Wi-Fi RSSI and SQI,while BLE implies BLE RX power and TX power.

The BiLSTM neural networks are applied to many environments, and their RMSE and MAE performances are shown in Table 2. At EOW 3rd floor, the BiLSTM with Wi-Fi has a RMSE of 0.83 m and a MAE of 0.58 m. Moreover, only using BLE information yields a similar RMSE of 0.88 m and a MAE of 0.56 m because the Wi-Fi routers share the same

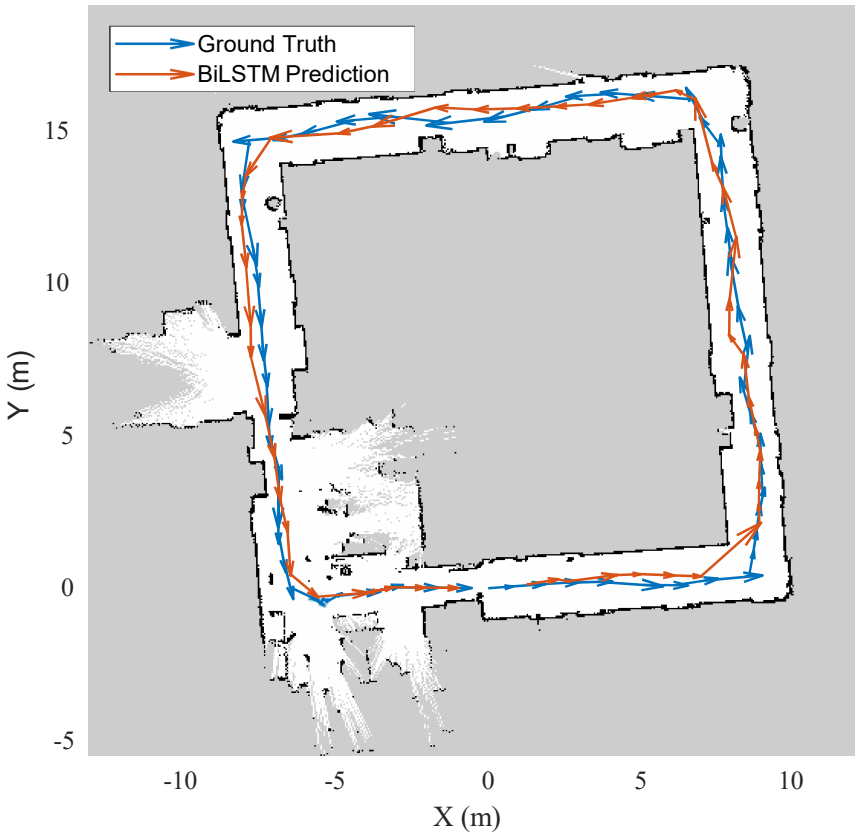


Figure 3: BiLSTM's predicted locations vs ground truth at EOW 3rd floor (Dataset A: Wi-Fi+BLE).

Table 3: Dataset B: BiLSTM's localization performance at different locations.

Location	Method	APs	RMSE (m)	MAE (m)	Training Time (s)	Testing Time (µs)
EOW 3rd Floor	BiLSTM+Wi-Fi FTM	8	0.80	0.57	4.59	360
EOW 3rd Floor	BiLSTM+Wi-Fi RSSI	8	0.82	0.62	5.57	366
EOW 3rd Floor	BiLSTM+Wi-Fi FTM+Wi-Fi RSSI	16	0.75	0.55	5.93	442
EOW 5th Floor	BiLSTM+Wi-Fi FTM	8	0.75	0.57	4.21	461
EOW 5th Floor	BiLSTM+Wi-Fi RSSI	8	0.77	0.59	3.70	428
EOW 5th Floor	BiLSTM+Wi-Fi FTM+Wi-Fi RSSI	16	0.70	0.52	3.52	373
ECS 1st Floor	BiLSTM+Wi-Fi FTM	8	1.52	1.31	11.63	426
ECS 1st Floor	BiLSTM+Wi-Fi RSSI	8	1.63	1.41	11.89	409
ECS 1st Floor	BiLSTM+Wi-Fi FTM+Wi-Fi RSSI	16	0.89	0.70	12.07	446

Note: Wi-Fi RSSI implies Wi-Fi RSSI and SQI, while Wi-Fi FTM implies RTT using IEEE 802.11mc .

positions as the BLE routers. Combining Wi-Fi and BLE information together results in a RMSE of 0.82 m and a MAE of 0.58 m, of which has no discernible difference. For error analysis, the ground truth trajectory is compared against the BiLSTM's predicted trajectory in Fig. 3, while the error heat map is displayed in Fig. 4. The path begins at (X = 0 m, Y = 0 m) with medium error of 1.0 m. However, the error increases to 1.5 m at the corners because that position has the least amount of LoS from the routers. Subsequently, the highest error of 2.5 m occurs in the east hallway because there are multiple objects blocking the signal paths of the routers. Afterwards, the error rapidly drops to 0.5 m as the BiLSTM recovers itself and gets back on the correct trajectory. For the rest of the path, the error predominantly stays below 1.0 m, but there are a few locations where the error jumps above 1.0 m due to corners.

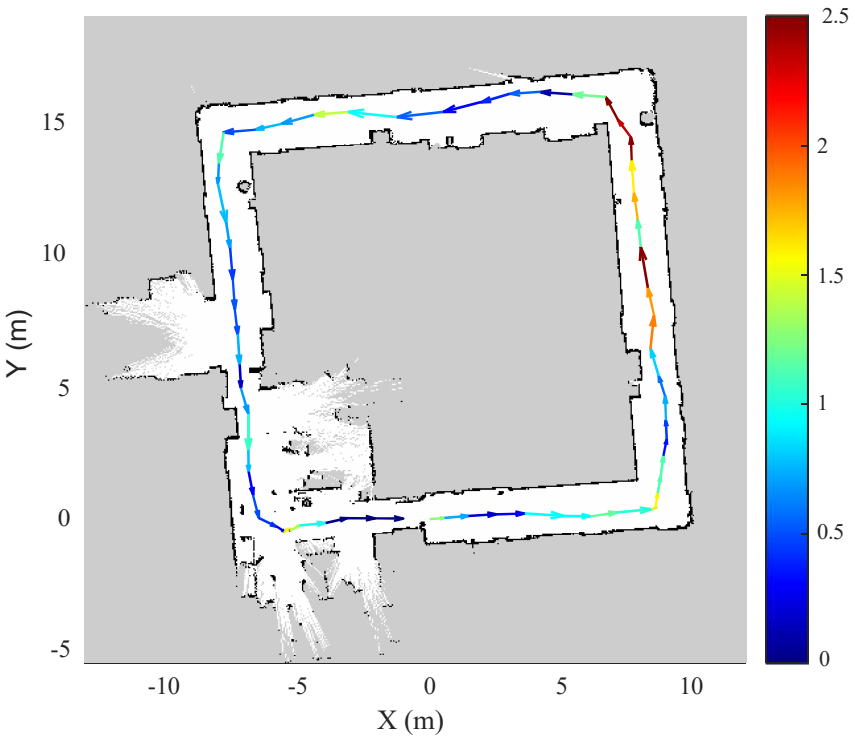


Figure 4: Error heat map of the BiLSTM's predicted locations at EOW 3rd floor (Dataset A: Wi-Fi+BLE).

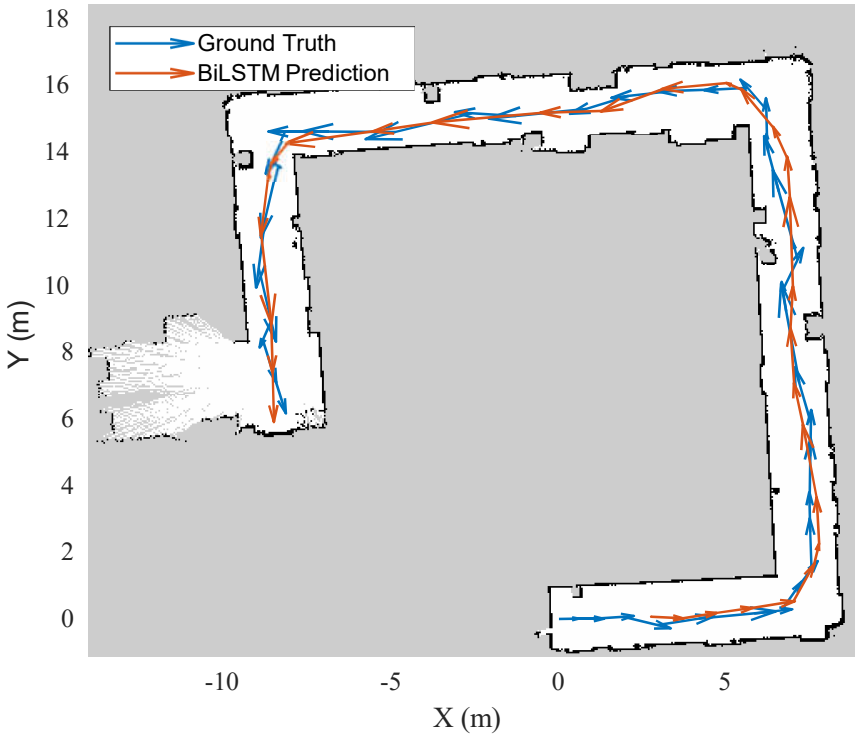


Figure 5: BiLSTM's predicted locations vs ground truth at EOW 4th floor (Dataset A: Wi-Fi+BLE).

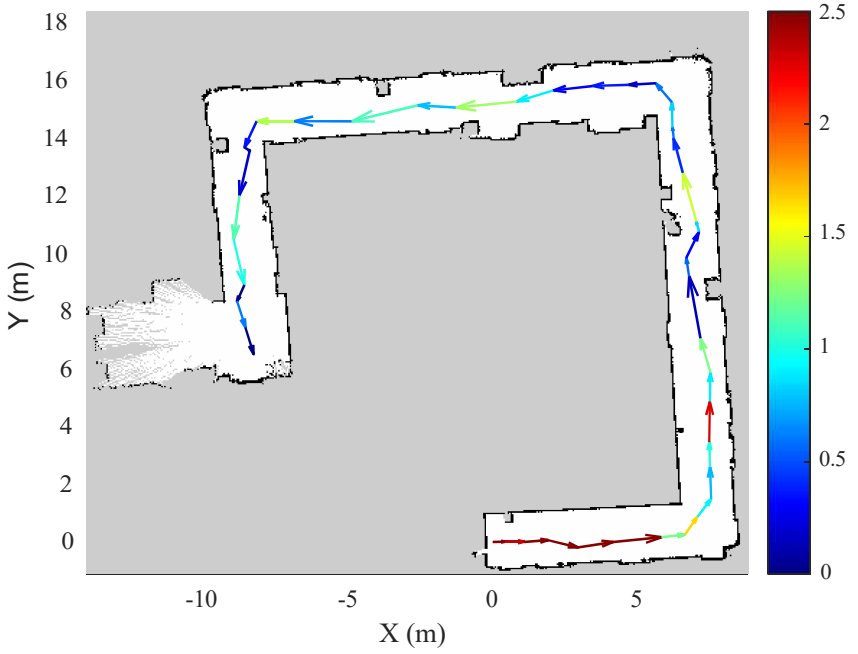


Figure 6: Error heat map of the BiLSTM's predicted locations at EOW 4th floor (Dataset A: Wi-Fi+BLE).

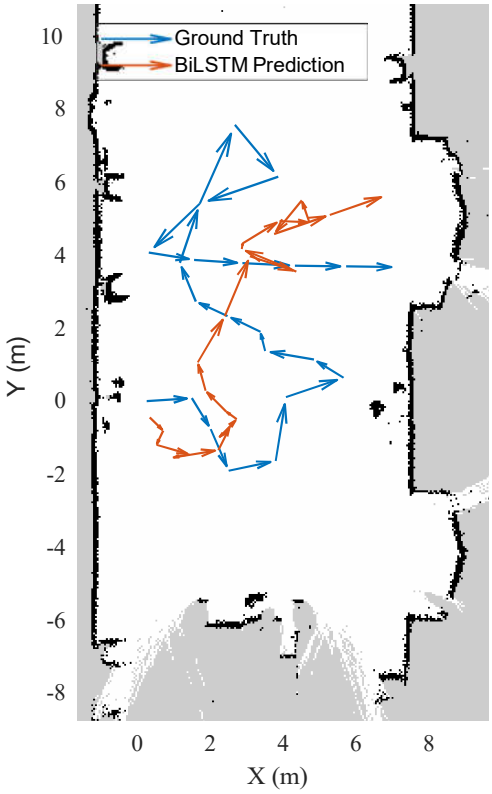


Figure 7: BiLSTM's predicted locations vs ground truth at ECS 1st floor (Dataset A: Wi-Fi+BLE).

At EOW 4th floor, the BiLSTM with Wi-Fi has a RMSE of 0.92 m and a MAE of 0.61 m. The RMSE of EOW 4th floor is slightly higher than the RMSE of EOW 3rd floor because the routers' signal paths in EOW 4th floor are blocked by

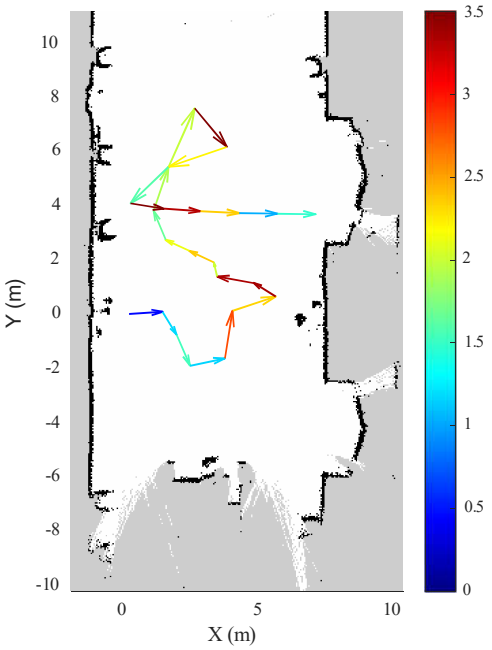


Figure 8: Error heat map of the BiLSTM's predicted locations at ECS 1st floor (Dataset A: Wi-Fi+BLE).

more walls and doors. Moreover, the number of routers is reduced from 11 to 9 due to the lack of power outlets. Using only BLE produces a similar RMSE of 0.93 m and a MAE of 0.63 m due to the number of Wi-Fi and BLE routers being similar. Combining Wi-Fi and BLE information together results in a slightly lower RMSE of 0.84 m and a MAE of 0.60 m. The lower RMSE and MAE is caused by the increased bandwidth and the increased channel diversity. Just as before, the ground truth trajectory is compared to the BiLSTM's predicted trajectory in Fig. 5, while the error heat map is shown in Fig. 6. This time, the highest error of 2.3 m occurs at the starting position of (X = 0 m, Y = 0 m). The large error is caused by not having enough space and power outlets to place more routers at the starting position. Soon after, the error quickly decreases to 1.0 m as the BiLSTM recovers and gets back on the correct trajectory. There are a few instances where the error jumps significantly due to the objects blocking the router's signals, but those errors are lower than the starting position errors.

ECS 1st floor is very different from the other floors because the packets are collected in an open area instead of an enclosed hallway. In an open area, Wi-Fi RSSI and BLE RSSI changes approximately 5 dBm per 10.0 m. The routers are not sensitive enough to detect the small changes in RSSI and creates errors in localization. Moreover, there is an extra degree of freedom compared to the hallways, of which creates more ambiguity in the trajectory. As a result, localization errors on this floor are much larger than the other floors. For Wi-Fi, the RMSE is 1.69 m and the MAE is 1.42 m, of which the localization errors are significantly higher than EOW 3rd floor, EOW 4th floor, and ECS 5th floor. Localization using BLE information has a larger RMSE of 2.13 m and a larger MAE of 1.73 m. This is caused by the BLE having a lower transmit power and worse antennas. Combining Wi-Fi and BLE slightly lowers the RMSE to 1.30 m and the MAE to 1.03 m because the antenna diversity and the channel diversity are increased. The ground truth trajectory is compared to the BiLSTM's predicted trajectory in Fig. 7, while the error heat map is shown in Fig. 8. The trajectory starts off well at the origin of (X = 0 m, Y = 0 m) with an error of 0.5 m. However, the error rapidly increases to above 2.0 m because the predicted trajectory quickly diverges from the ground truth. The BiLSTM never recovers from incorrect predictions, and the error remains above 2.0 m. The large errors are caused by the ambiguity of the wireless features. Multiple unique positions on the map have the same Wi-Fi RSSI, SQI, and BLE RSSI.

4.3 Web Application

We created a website for finding the suspected cases, given the confirmed cases. Moreover, the website displays the trajectories of the suspected cases and their MAC addresses/smartphone model specific information. Fig. 16 shows a page where the health authorities can enter the MAC address/smartphone model specific information of the confirmed case. The search result in Fig. 17 displays the contact history of a confirmed case, showing all the suspected cases

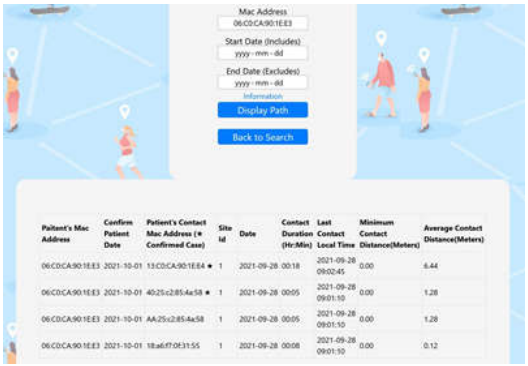
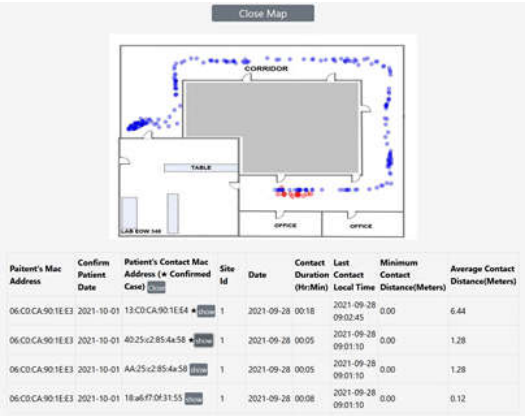


Figure 17: Contact history search result.



Paitent's Mac Address	Confirm Patient Date	Patient's Contact Mac Address (+ Confirmed Case) Show	Site Id	Date	Contact Duration (Hr:Min)	Last Contact Local Time	Minimum Contact Distance(Meters)	Average Contact Distance(Meters)
06:C0:CA:90:1E:E3	2021-10-01	13:C0:CA:90:1E:E4 Show	1	2021-09-28	00:18	2021-09-28 09:02:45	0.00	6.44
06:C0:CA:90:1E:E3	2021-10-01	40:25:c2:85:4a:58 Show	1	2021-09-28	00:05	2021-09-28 09:01:10	0.00	1.28
06:C0:CA:90:1E:E3	2021-10-01	AA:25:c2:85:4a:58 Show	1	2021-09-28	00:05	2021-09-28 09:01:10	0.00	1.28
06:C0:CA:90:1E:E3	2021-10-01	18:a6:f7:0f:31:55 Show	1	2021-09-28	00:08	2021-09-28 09:01:10	0.00	0.12

Figure 18: Path of a confirmed case and suspected case.

with time and distance information. Each row of the contact history table is a contact instance, and each column is the information for each contact instance. To show the path of the confirmed case, the authorities specify the time range between two dates in Fig. 17. The blue dots in Fig. 18 shows the path of a confirmed case for a location. Each blue dot represents a sample instance. The button beside each suspected case allows the user to display the path of the suspected case, shown with the red dot in Fig. 18. A higher density of dots means the person spends more time on that spot.

5 Conclusion

We have created a custom, privacy preserving Wi-Fi and BLE contact tracing system for finding the detailed paths of the infected individuals without any user intervention. The system tracks smartphones, but it does not require smartphone applications, connecting to the routers, or any other extraneous devices on the users. A custom Turtlebot3 is used for simulating user movement and smartphone transmission as described in Section 2. The smartphones' received power, transmit power, and round trip time are collected by a custom ESP32C3 router. Even though MAC randomization is designed to prevent user tracking, we defeated it to track many smartphones, such as the ones listed in Table 1. Afterwards, the wireless parameters above are converted to signal path loss and ToF, of which the BiLSTM takes and predicts the absolute paths of the users. Table 3 shows the localization performance and the RMSE is always below 0.9 m for Wi-Fi RSSI + Wi-Fi FTM. Public health authorities can use our website in Fig. 16 to find the paths of the confirmed cases and suspected cases, together with their MAC addresses/smartphone model specific information. They can also track indirect contact transmissions originating from surfaces and droplets.

References

[1] D. Majra, J. Benson, J. Pitts, and J. Stebbing, "SARS-CoV-2 (COVID-19) superspreader events," *Journal of Infection*, 2020.

[2] K. T. Eames and M. J. Keeling, "Contact tracing and disease control," *Proceedings of the Royal Society of London. Series B: Biological Sciences*, vol. 270, no. 1533, pp. 2565–2571, 2003.

[3] R. A. Kleinman and C. Merkel, "Digital contact tracing for COVID-19," *CMAJ*, vol. 192, no. 24, pp. E653–E656, 2020.

[4] S. Altmann, L. Milsom, H. Zillessen, R. Blasone, F. Gerdon, R. Bach, F. Kreuter, D. Nosenzo, S. Toussaert, J. Abeler *et al.*, "Acceptability of app-based contact tracing for COVID-19: Cross-country survey study," *JMIR mHealth and uHealth*, vol. 8, no. 8, p. e19857, 2020.

[5] R. Hinch, W. Probert, A. Nurtay, M. Kendall, C. Wymant, M. Hall, K. Lythgoe, A. B. Cruz, L. Zhao, A. Stewart *et al.*, "Effective configurations of a digital contact tracing app: a report to NHSX," *Retrieved July*, vol. 23, p. 2020, 2020.

[6] N. Nanthini, B. M. Shankar, S. S. Kumar, and R. S. Ganesh, "Deep learning approach for minimizing disease spread using face identification and contact tracing," *2020 Fourth International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud)(I-SMAC)*, pp. 527–532, 2020.

[7] M. Yaghi, T. Basmaji, R. Salim, J. Yousaf, H. Zia, and M. Ghazal, "Real-time Contact Tracing During a Pandemic using Multi-camera Video Object Tracking," *2020 International Conference on Decision Aid Sciences and Application (DASA)*, pp. 872–876, 2020.

[8] J. L. Gardy, J. C. Johnston, S. J. H. Sui, V. J. Cook, L. Shah, E. Brodkin, S. Rempel, R. Moore, Y. Zhao, R. Holt *et al.*, "Whole-genome sequencing and social-network analysis of a tuberculosis outbreak," *New England Journal of Medicine*, vol. 364, no. 8, pp. 730–739, 2011.

[9] Hallam Stevens and Monamie Bhadra Haines, "TraceTogether: Pandemic Response, Democracy, and Technology," *East Asian Science, Technology and Society: An International Journal*, vol. 14, no. 3, pp. 523–532, 2020. [Online]. Available: https://doi.org/10.1215/18752160-8698301

[10] L. Reichert, S. Brack, and B. Scheuermann, "A survey of automatic contact tracing approaches using Bluetooth Low Energy," *ACM Transactions on Computing for Healthcare*, vol. 2, no. 2, pp. 1–33, 2021.

[11] Q. Zhao, H. Wen, Z. Lin, D. Xuan, and N. Shroff, "On the accuracy of measured proximity of Bluetooth-based contact tracing apps," *International Conference on Security and Privacy in Communication Systems*, pp. 49–60, 2020.

[12] D. J. Leith and S. Farrell, "Coronavirus contact tracing: Evaluating the potential of using bluetooth received signal strength for proximity detection," *ACM SIGCOMM Computer Communication Review*, vol. 50, no. 4, pp. 66–74, 2020.

[13] P. Di Marco, P. Park, M. Pratesi, and F. Santucci, "A Bluetooth-based architecture for contact tracing in healthcare facilities," *Journal of Sensor and Actuator Networks*, vol. 10, no. 1, p. 2, 2021.

[14] E. Hernández-Orallo, C. T. Calafate, J.-C. Cano, and P. Manzoni, "Evaluating the effectiveness of COVID-19 Bluetooth-based smartphone contact tracing applications," *Applied Sciences*, vol. 10, no. 20, p. 7113, 2020.

[15] T. M. Yasaka, B. M. Lehrich, and R. Sahyouni, "Peer-to-peer contact tracing: development of a privacy-preserving smartphone app," *JMIR mHealth and uHealth*, vol. 8, no. 4, p. e18936, 2020.

[16] I. Nakamoto, S. Wang, Y. Guo, and W. Zhuang, "A QR code–based contact tracing framework for sustainable containment of COVID-19: Evaluation of an approach to assist the return to normal activity," *JMIR mHealth and uHealth*, vol. 8, no. 9, p. e22321, 2020.

[17] A. S. Hoffman, B. Jacobs, B. van Gastel, H. Schraffenberger, T. Sharon, and B. Pas, "Towards a seamful ethics of Covid-19 contact tracing apps?" *Ethics and Information Technology*, pp. 1–11, 2020.

[18] D. Mobo, A. L. R. Garcia *et al.*, "Using automated contact tracing system app with QR code to monitor and safeguard parishioners against COVID-19 at St. Anthony of Padua Parish, Subic, Zambales," *American Research Journal of Computer Science and Information Technology*, vol. 4, no. 1, pp. 1–4, 2020.

[19] S. Wang, S. Ding, L. Xiong *et al.*, "A new system for surveillance and digital contact tracing for COVID-19: spatiotemporal reporting over network and GPS," *JMIR mHealth and uHealth*, vol. 8, no. 6, p. e19457, 2020.

[20] A. Trivedi, C. Zakaria, R. Balan, A. Becker, G. Corey, and P. Shenoy, "WiFiTrace: Network-based Contact Tracing for Infectious Diseases Using Passive WiFi Sensing," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 5, no. 1, pp. 1–26, 2021.

[21] G. Li, S. Hu, S. Zhong, W. L. Tsui, and S.-H. G. Chan, "vContact: Private WiFi-based IoT contact tracing with virus lifespan," *IEEE Internet of Things Journal*, 2021.

[22] K. Fallahi, C.-T. Cheng, and M. Fattouche, "Robust Positioning Systems in the Presence of Outliers Under Weak GPS Signal Conditions," *IEEE Systems Journal*, vol. 6, no. 3, pp. 401–413, 2012.

[23] V. Gokhale, G. M. Barrera, and R. V. Prasad, "FEEL: Fast, Energy-Efficient Localization for Autonomous Indoor Vehicles," *arXiv preprint arXiv:2102.00702*, 2021.

[24] J. Tiemann and C. Wietfeld, "Scalable and precise multi-UAV indoor navigation using TDOA-based UWB localization," *2017 international conference on indoor positioning and indoor navigation (IPIN)*, pp. 1–7, 2017.

[25] V. Moreno, M. A. Zamora, and A. F. Skarmeta, "A low-cost indoor localization system for energy sustainability in smart buildings," *IEEE Sensors Journal*, vol. 16, no. 9, pp. 3246–3262, 2016.