International Journal of Engineering & Technology, 7 (4.35) (2018) 26-34

International Journal of Engineering & Technology

Website: www.sciencepubco.com/index.php/IJET

Research paper

Iot based Parking Sensor Network for Smart Campus

Joseph Jesawanth Singh[1], Nanthiine Nair Ravi[2]*, Prajindra Sankar Krishnan[3]

[1]Department of Electronics and Communication Engineering,
[2]College of Engineering, Universiti Tenaga Nasional,
[3]43000 Kajang, Selangor, Malaysia.
**Corresponding author E-mail: nanthiinenair@gmail.com*

Abstract

The difficulty in finding parking spaces is growing into a more serious problem especially in urban areas. As the number of vehicles on the road continues to increase substantially, the difficulty in finding parking spaces increases even more. To combat this rising issue many cities have adopted a more organised and intelligent parking system which is also commonly known as smart parking system. This is to improve the efficiency of locating a parking spot, especially in an urban area. The smart parking system will often have sensors at parking spaces to determine the occupancy of the respective parking space. This occupancy value is relayed to a server by some specific communication protocol. The parking occupancy can then be displayed to the public by a Graphical User Interface (GUI) design which gets the occupancy values from a server. This project simulates the parking sensor network design in a smart campus setting by implementing the concept of the Internet of Things (IoT). This project uses ultrasonic sensors that are connected to a NodeMCU microcontroller to send the parking occupancy values to an online database implemented using Google Firebase. A mobile application with a GUI (Graphical User Interface) is also created using MIT App Inventor 2 to display the vacancy of parking by communicating with the online database. The results obtained in this project were promising. The successful implementation of this idea will allow users to save more time and money, not to mention that it can also help reduce carbon emissions from vehicles resulting in a more sustainable environment.

Keywords: *Internet of Things; Mobile Application; Sensor; Smart Parking; Wi-Fi*

1. Introduction

Internet of Things or IoT refers to billions of physical devices around the world that are now connected to the internet, collecting and sharing data. These devices are embedded with electronics, software, sensors, actuators, and connectivity and thus enabling them to communicate without a human being involved, and merging the digital and physical worlds.The connectivity of these devices will then allow users to access real time information, while providing insights and analytics to help them make smarter and informed decisions in their daily lives. Thus, this project presents on 'IoT based parking sensor network for smart campuses. This project will be catered to implement the concept of IoT mentioned above to design a sensor network in a smart campus setting.

Students and staff members in a university similarly spend their valuable time finding parking in a university campus which also leads to an increase in the carbon emissions level from these vehicles. This parking issue typically involves the entire management of the university and students, thus causing problems for the university. This reduces the efficiency of the parties involved and also results in congestion in the campus environment. This usually happens during peak hours and might continue to worsen if effective measures are not taken. These issues addressed above can be curbed by the introduction of a more sustainable parking management solution. Such a solution is in-line with this project's topic which is the 'IoT Based Parking Sensor Network for Smart Campus'. The proposed design will utilize the sensor nodes placed at the intended car parking spaces to detect or sense the presence of parked vehicles. The sensors will then send information to a microcontroller which transmits those data wirelessly to an online database using Wi-Fi (Wireless Fidelity). The online database which acts as a gateway between the sensor nodes and the mobile application will then collect all of the current sensor data.

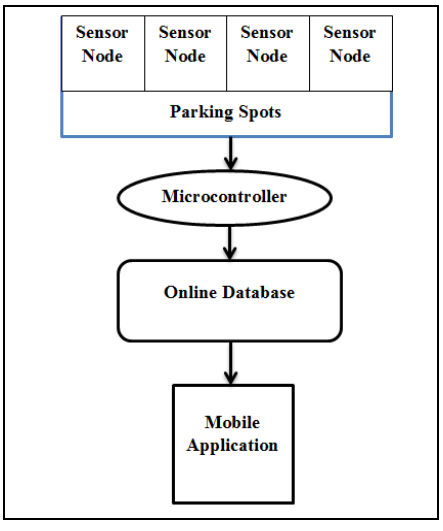

Fig. 1: Block Diagram of the Parking Sensor Network.

This mobile application will be able use the information available in the server to display the vacant parking spots. Thus, users will be able to access real-time information from any desired location. The proposed design is hoped to improve the convenience and efficiency of finding a parking space in a smart campus environment with the help of a real time data displayed on mobile applica-

tion. The block diagram of the overall design is shown in Figure 1. The remainder of the paper is organised as follows.
Section II discusses on the literature review. Research methodology is done is Section III whereas section IV focuses on results and discussion. Finally, conclusion is done in section V.

2. Literature Review

This section covers the literature review of some commonly used parking sensor network systems around the world. Besides that, it also focuses on the review of potential sensors, microcontrollers and software that can be used in this project which then leads to a more informed decision making when designing the project.

2.1. Review of Parking Sensor Network Systems

2.1.1 Parking Sensor using Magnetic Sensor, Distance Sensor and NodeMCU Microcontroller.

The processing unit used in this paper is the NodeMCU which is embedded with an ESP8266 unit. The magnetic sensor used is the HMC5983, magnetoresistance sensor and the distance sensor used is sharp GP2Y0A21YK0F infrared sensor. The system begins with the first stage which is the system initialization [1].
The distance measurement is initialized and woken up. NodeMCU is set to a sleep mode with an interval of 1 minute, when NodeMCU wakes up it measures the magnetic field strength. The recent measured value is then compared to the previous value. If the difference is bigger than a predefined threshold value, then NodeMCU turns on the distance sensor to measure the distance. If distance is within a predefined threshold value, it is assumed that a vehicle is present. If a vehicle is present a variable of '1' is sent, if a vehicle was previously present data is not sent. If distance is not within threshold value no data is sent and NodeMCU goes to sleep for a minute [1]. Figure 2 below shows the sensor mode circuit with NodeMCU.

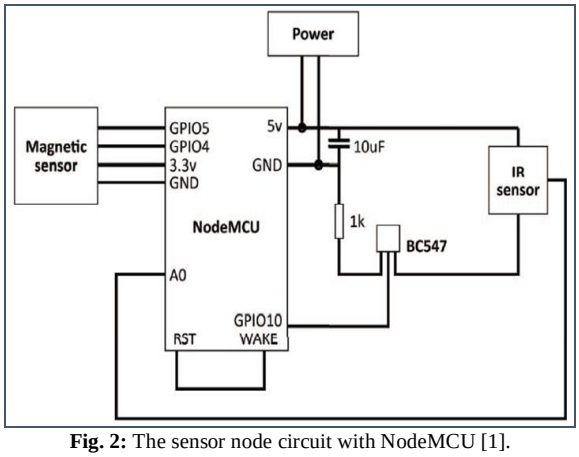

Fig. 2: The sensor node circuit with NodeMCU [1].

The sensor network was implemented using Message Queuing Telemetry Transport (MQTT) protocol. MQTT protocol follows publisher/subscriber architecture; message broker is the central unit between the subscriber and publisher. Furthermore, Elastic Computer Cloud of Amazon Web Service was used to deploy database and web application. MongoDB was used as the database and connection was implemented by Mongoose which is a node.js library. Socket.io middleware was used as the real-time communication part. The server acts as the subscriber of the message broker. When the Server receives the data, the server will store the data, process it and send them to the mobile application using socket.io middleware [1].
The mobile application is called "ParkMe" it enables users to locate any vacant parking space. The app is available on android and also provides directions to guide users to the parking spots. The app is developed using Java, and Google Map API is used to show Google map in the app. Parking types as well as Google map settings can be changed on this app [1]. Figure 3 shows the mobile application interface.

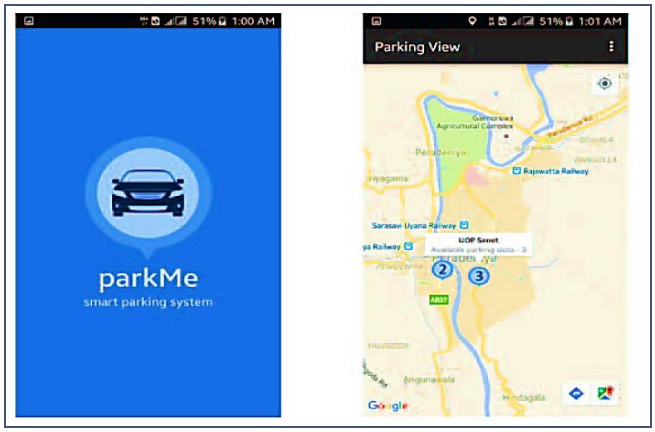

Fig. 3: "Park Me" mobile application interface

2.1.2 Smart Parking Sensor using Infrared Sensors and ARM Microcontrollers.

The system begins with a Parking Inquiry Module which is integrated in the vehicle, when the user requires information about a parking spot, a button in the module will be pressed. This request will be sent through ZigBee and it will then send a message to the vehicle where the information will be displayed on the LCD screen of the vehicle [2]. The parking inquiry module is shown in Figure 4 below.

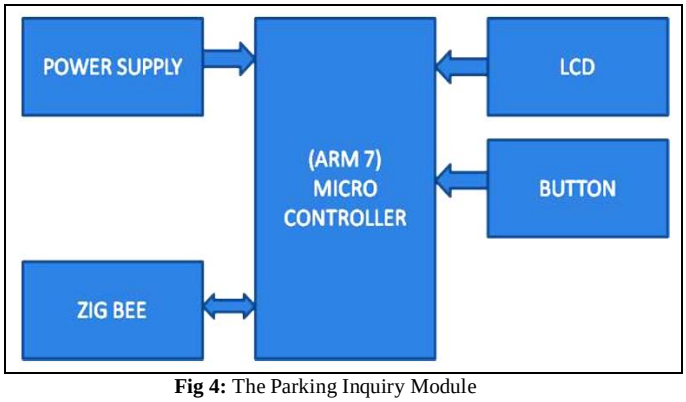

Fig 4: The Parking Inquiry Module

The second part of the system is the Parking Monitoring System (1st floor) which uses RF and ZigBee communications. Upon receiving the request from the vehicle, the monitoring system will collect the status of the car park from the parking slot detector. It checks availability on 1st floor first. If no spots are available, the system proceeds to check the 2nd floor. The RF receiver receives information from the parking spot detector. IR sensors are used to detect at parking spaces to detect vacant spots in the 1st floor and sends it to the monitoring system.
The Third part of the system is the Parking Slot Detector which is located on the 2nd floor. The information from the parking spaces is detected from IR sensors. If any spots are empty, the ARM controller is notified and the vacancy is displayed on the LCD screen [2].
The work flow of the system is such that it begins with RF receiver module on the 1st floor which receives parking status from 2nd floor's RF transmitter. When the 1st floor receives information through the ZigBee, it sends this information to the vehicle. The 1st floor continues to receive signal from the 2nd floor through RF communication and is repeated to eliminate traffic [2]. Figure 5 shows the workflow algorithm of the parking system.

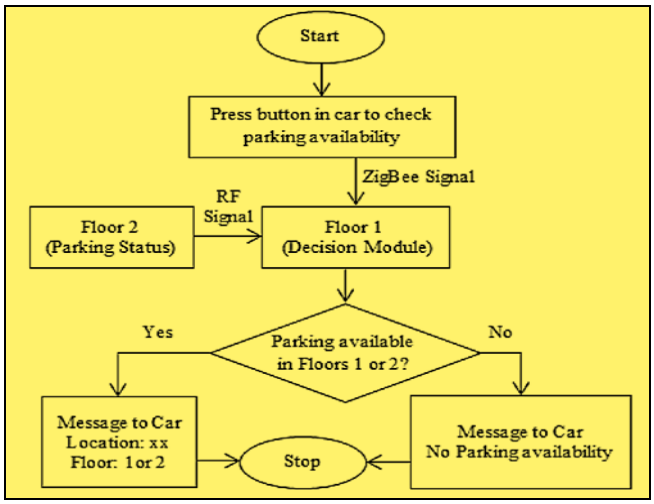

Fig 5: The work-flow algorithm

2.1.3 Smart Parking Sensor using Ultrasonic sensor and Raspberry Pi Camera using Raspberry Pi Microcontroller.

In this system, each parking space is monitored with an ultrasonic distance sensor and a raspberry pi camera using a raspberry pi microcontroller. The presence of a vehicle is posted to the parking system mediator with plate details using the internet enabled raspberry pi. The camera and sensor will be connected to the GIPO pins to receive and send details from ultrasonic sensor and to receive capture image of vehicle plate. Raspberry PI with wiring Pi APIs will use OCR4J for rasterizing image feed to look for plate details [3].

The Raspberry Pi initiates sound waves from the sensor to determine object's distance. The sensor will measure the distance and sends it to Raspberry Pi, if a similar parameter is sent for a few seconds then a vehicle is present. Raspberry Pi then instructs Pi camera in front of vehicle to take a picture of plate details. Wiring Pi receives image and uses OCR4J framework for rasterizing image to determine numbers. This information is sent to the web service to display to users [3].

2.1.4 Smart Parking Sensor using a Light Sensor and IRIS WSN Motes.

An IRIS WSN motes with MTS420/400 light sensor board with a base station connected to a PC is used. When a vehicle enters the parking space it covers the entry node and light sensor, this causes the node to create a unique ID which is sent to the network. Sensor node will also measure intensity of light every second and a program running on the PC will use this information to display the vacant areas so that users will know where to park. When a vehicle exits a parking space the sensor will detect a higher light intensity and this will indicate to the system that the parking spot is vacant. The parking charges can also be calculated from the duration that the car parked and exited [4]. Figure 6 shows the flow diagram of the parking module and graphical display is shown in Figure 7.

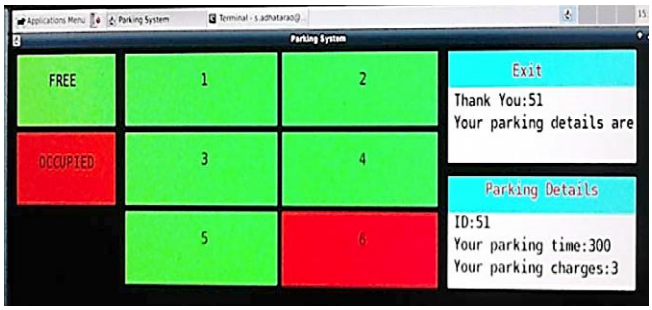

Fig 6: The graphical display

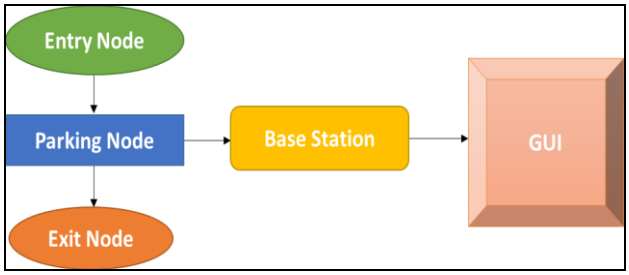

Fig 7: The flow diagram

2.2. Review of Core Components

This section analyses the potential core components such as the sensors, microcontrollers, database and application development tools and the selection of the best potential components based on the comparative analysis done.

2.2.1 Sensors

There are several types of sensors that can be used in this project. Firstly, the Inductive Proximity Sensor (IPS) is very common sensor used in detecting the presence of vehicles. This sensor is usually placed underground and able to detect changes in the earth's magnetic field, thus detects a vehicle when the magnetic field changes by a certain threshold value. One of the disadvantages of using this sensor in a parking sensor network is that it has to be buried underground and the parking space would need to be out of use during the installation process and it would incur more cost.

Ultrasonic sensor is another common sensor that can be used in the parking sensor networks. One of the popular ultrasonic sensors is the HC-SRO4 sensor as depicted in Figure 8. This sensor has four pins which are the Vcc, Ground, Echo and Trigger pins. The sensor works turning on the Trig pin for 10µs this will cause the transmitter of the sensor to generate an eight cycle sonic burst of 40kH which will start a timer. This pulses that are generated by the sensor will travel at the speed of sound in the air until it encounters an object along its path. The pulses will then be reflected of the object and return to the receiver end of the sensor. The timer will stop the moment the ultrasonic pulse returns to the sensor [5].

Figure 9 shows the timing module for the ultrasonic sensor. The timer mentioned previously is used to calculate the time for the pulse to return to the receiver. The time taken can be used to calculate the distance of the object. Equation 1 shows the conversion of the time to distance, taking into account that the speed of sound is v = 340m/s or v = 0.034cm/µs and given time = distance/speed where the distance has to be divided by two as the pulse travels to and from an object [5]:

$$d = (t \times 0.034)/2 \quad (1)$$

Fig 8: HC-SRO4 Ultrasonic Sensor

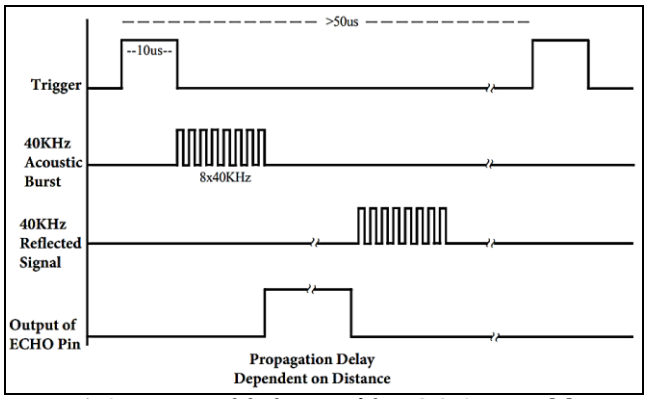

Fig 9: Timing Module diagram of the HC-SR04 sensor [5].

Apart from that, RFID is also one of the popular sensors used to detect the presence of vehicles. RFID would first have to be placed on every vehicle using the parking service. Transmitters that are placed in the parking space will transmit a signal and RFID chips on the vehicles would receive power from the signal and transmit their own signal back. These RFID sensors work by placing them on all the vehicles and only vehicles with RFID chips would be detected and other objects or living things would not provide a false positive result [6].

Another type of potential sensor is LIDAR. This sensor sends out a pulse of high-frequency signal and awaits the response of the pulse to determine the distance. LIDAR and ultrasonic sensors work in similar manner in determining distance of objects. However, LIDAR uses light pulse instead of sound wave. This makes LIDAR more accurate and has better range but it is costly [6].

Based on the comparison as shown in Table 1 and the literature review conducted on the four sensors, it has been decided that this project will select the ultrasonic sensor (HC-SRO4) as the sensor in the parking network. This is because it is very cheap and relatively has good sensing of distance and requires only two digital pin connections. Although this sensor cannot differentiate between vehicles and non-vehicles, this project will not be using an actual vehicle but a toy vehicle in a parking model to recreate a typical parking lot in a smart campus. Thus, the ability to detect metal is not of utmost importance to satisfy the objective of this project.

Table 1: Comparison of Sensors

Sensor	Advantage	Disadvantage
Inductive Proximity Sensor (M18DPO)	-Very compatible to be used in the parking network as the IPS works very well in detecting metal objects like cars. -Not very expensive. (RM 30)	-Sensor has to be buried underground in a real-world parking application. -Has a very low sensing range. (5mm)
Ultrasonic Sensor (HC-SR04)	-Works well in sensing objects with a range of 4m. -Very cheap. (RM 11) -Requires digital pin and easy to convert values to distance.	-Will detect any object, unable to differentiate between vehicles and non-vehicles.
RFID Sensor (MFRC522)	-Relatively cheap. (RM 20)	-Every vehicle requires a RFID tag. -Not very good sensing range. (0.5m)
LIDAR Sensor (LIDAR-Lite)	-Very good sensing distance range (40m). -Distance values are highly accurate.	-Very expensive. (RM 508)

2.2.2 Microcontrollers

There are several types of microcontroller that can be used in this project. Firstly, is the Texas Instruments CC3200 Simple Link microcontroller. This microcontroller has built in Wi-Fi connectivity on a single chip microcontroller which integrates a powerful ARM Cortex-M4 processing unit (240 MHz) with 1MiB ROM memory and 256KiB RAM memory. This controller is designed and tailored specifically to meet the needs of the Internet of Things (Iot) application. The Internet of Things application ranges from Security Systems, Wireless Audio, Access Control, Cloud Connectivity, etc. [7].

Node MCU is an affordable development controller with ESP8266 which is a Wi-Fi microchip designed by Expressif Systems. The ESP8266 is a networking solution while being able to run its stored application simultaneously. It also acts as a bridge between the microcontroller it is connected to and its own network connectivity. Figure 10 below shows the image of the NodeMCU microcontroller.

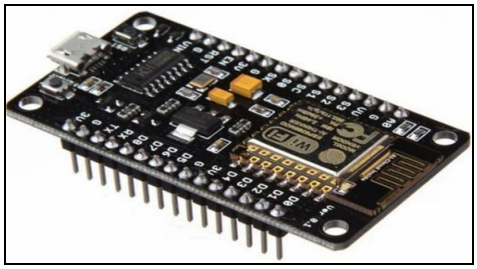

Fig 10: The NodeMCU microcontroller by Devkit [8].

The microcontroller runs on a Tensilica Xtensa L106 32-bit processing unit (80~160MHz) with 64KiB ROM memory and 32 + 80KiB RAM memory. The NodeMCU is also breadboard friendly with a micro USB female port located on the controller for easy connectivity. The microcontroller also has 11 digital and 1 analogue input/output pins. It also has 4 ground and VCC pins of which three are 3.3V and one 5V. The NodeMCU controller can be programmed by using the Arduino IDE software [8].

Apart from that, the Arduino MKR1000 is an IoT development board that is ideal for beginners. The controller comes with a Wi-Fi shield with 2.4GHz 802.11 b/g/n Wi-Fi WINC1500. It also comes with SAMD21 Cortex-M0+ 32bit low power ARM processing unit (560 MHz) with 256KiB RAM and 32KiB ROM.

Based on the literature review and the comparison conducted on the three microcontrollers as shown in Table 2, it has been decided that this project will select the NodeMCU (ESP8266) as the microcontroller in the parking sensor network. This is because it has a relatively good processing speed, has Wi-Fi connectivity, has a high number of digital input/output pin required for the ultrasonic sensor, and it is very cheap. Although, it is lacking in processing power and RAM/ROM memory compared to the other microcontrollers the memory is still sufficient for the parking sensor network implementation.

Table 2: Comparison of Microcontrollers

Microcontroller	Advantage	Disadvantage
Texas Instruments (CC3200)	-Very powerful and fast processing speed (240MHz). -Very high RAM and ROM memory.	-Very expensive (RM 230)
NodeMCU (ESP8266)	-Moderate processing speed. (80 -160MHz). -Very cheap. (RM 28)	-Average RAM and ROM memory.
Arduino (MKR1000)	-Extremely powerful and fast processing unit (560 MHz). -Moderate RAM and ROM memory.	-Very expensive (RM 280)

2.2.3 Database

A database is basically a data collection in an organized manner. The location of the database typically can either be on cloud stor-

age or on a basic desktop storage drive. This project requires the use of a database and it attempts to create it using a cloud storage website. There are several databases identified. Firstly, is Google Firebase. It is an API (Application Programming Interface) that is created by Google mainly for synchronizing with android's web application. Firebase can also be used as database storage as it provides a real-time database feature that allows data to be collected from and stored to database quickly and efficiently. Apart from being database storage, Firebase can also be used to develop android applications [9]. The other features of Firebase include storage, notification, hosting and authentication. Firebase has an API that is provided by Google for the database creation. Real-time data typically can be obtained with only a couple of lines of programming code. Firebase provides free data storage up to 200MB and databases exceeding this size will require paid subscription services [9]. In Firebase data will be stored as JSON (JavaScript Object Notation) and this stored data can be accessed by a wide variety of application platforms. In implementing a Firebase database in an android application the JSON file of the database can be downloaded and included in a project. The Firebase can be introduced in projects that have a minimum requirement of android version 2.3 which is Gingerbread and Google Play Services version 9.6.1 [9]. One of the plus points of Firebase is that custom events can be created by the developer to prevent applications from crashing. It is important to note that Firebase uses the HTTP (Hypertext Transfer Protocol) and XMPP (Extensible Messaging and Presence Protocol) protocol serving both plain text and JSON type data [9]. In order to implement Firebase in a project using Arduino IDE (Integrated Development Environment) software, a Firebase library can be downloaded and specific programming lines can be used to publish and subscribe data from a microcontroller [9].

Similar to Firebase, Adafruit IO is a real-time database that provides primary focus on allowing simple data connectors, little difficulty to use and a little amount of programming requirement. The Adafruit IO primarily uses the MQTT API, HTTP and REST API however can also be used depending on the developer preference. Adafruit IO is very easy to use and ideal for IoT projects applications. Adafruit IO can be used with a variety of programs such as Node.js, Python, Ruby, etc. Figure 11 shows the typical dashboard from Adafruit database. In order to use Adafruit IO with the Arduino IDE software, some libraries can be downloaded. Two popular libraries are the Adafruit MQTT Client Library and the PubSubClient MQTT Library. Certain programming lines can be used depending on the library used to publish and subscribe to data from Adafruit IO.

Fig 11: A typical dashboard from Adafruit IO

Table 3: Comparison of Online Databases

Database	Advantage	Disadvantage
Google Firebase	-Very suitable to be used as database for mobile application. -Has many features that can effectively support applications. (Security, Crash reports) -Data can be sent by a microcontroller by using the Arduino IDE software. -Data updated and secured in real-time.	Does not use MQTT protocol. (XMPP and HTTP)
Adafruit IO	-Uses the MQTT protocol. -Very suitable for IoT projects. -Very easy to analyse data using Blocks. -Data sent in real-time. Data can be sent by a microcontroller by using the Arduino IDE software.	Not suitable to be used along with mobile applications.

Based on the literature review and comparison conducted as depicted in Table 3, it is decided that this project will select the Google Firebase as the Online Database. This is because it is very suitable to be used as database for mobile applications, has many features that can effectively support applications, data can be sent by a microcontroller by using the Arduino IDE software and data is updated and secured in real-time. Although Firebase does not use the MQTT protocol, it does have the XMPP protocol which has the publish-subscribe client that is favourable in IoT projects. Since this project requires the use of a mobile application, it is indeed clear that Google Firebase is the best choice.

2.2.4 Mobile Application Development Tools

There are several tools that can be used in developing the mobile application in this project. Firstly, it is the MIT App Inventor 2. The App Inventor is a tool that works on a web browser and allows developers to develop their applications by either an on-screen emulator or a connected mobile phone. The App Inventor also has a server that auto-saves any work and makes it easy for developers to keep track of their projects. The App Inventor has various features that help developers to design applications [10].

The first feature is the App Inventor Designer where developers can choose and drag any required components for their applications. It is important to note that Firebase component is available for easy connectivity of application to a Firebase database. The second feature is the App Inventor Block Editor where developers can organise their programming blocks that will instruct the components on how to function. The block programs are constructed similar to a puzzle. The changes made will be updated in real-time to application in an Android mobile. Alternately, an android emulator can be installed to the developer's pc (Personal Computer) and used if the developer does not have an Android phone [10]. Figure 12 shows the dashboard of App Inventor tool.

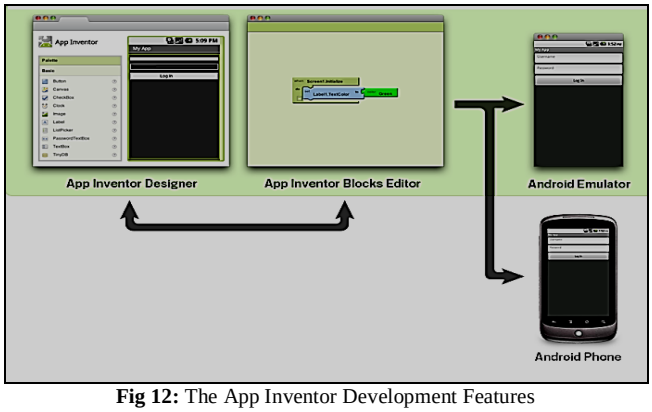

Fig 12: The App Inventor Development Features

The App Inventor development environment is supported by various operating systems like the Windows, Mac and GNU/Linux. Many varieties of applications like games and quizzes can be built by developers using the App Inventor. The App Inventor is also a very powerful tool and has many components like GPS (Global Positioning System) location sensor which can build apps that uses GPS location and allows users to communicate with web browsers like Amazon and Twitter [10]. The App Inventor has many different types of block programming components and Event Handlers is one of them. Event Handlers are basically programs that describe how the phone should respond to certain events. When an Event Handler is triggered, it executes a sequence of commands in its body. Commands are blocks that specify instruction to be performed on the phone. Some commands will also require parameters, arguments or input values. Another component in the block designer is expressions. Expressions are program block that will produce a certain value similar to mathematical expressions. Aside from these three mentioned block components, the App Inventor has many other block components to help develop more complex mobile applications. After creating the application an APK (Application Package Kit**)** file can be generated from the build menu [10].

Another development tool identified was the Android Studio IDE which was developed by Android and was first introduced in May 2013. Android Studio is developed based on the Java IDE Java IDE IntelliJ. The Android Studio is an alternative to the already popular Eclipse, and the tools and usage in the software is very similar to Eclipse. The software can be downloaded for free from the Android Studio website, and the software is able to run on various operating systems like Windows, Linux, and Mac. Android Studio comes with its own preconfigured SDK (Software Development Kit), which are tools that can be used to build a custom application. Different SDK can also be downloaded and used depending on the range of application. The Android Studio IDE contains a wide amount of features that can be used in developing a mobile application. These features are panels, windows and toolbars. The panels consist of the common panels that developers will use on a common basis. Based on the literature review and the comparison done this project will select the MIT App Inventor 2 as the tool to develop the Mobile Application. This is because it is easy to program, does not require software installation, can be easily connected to the Firebase database by dragging the Firebase component from the Application Inventor Designer, Applications can be tested immediately while developing the App, very friendly user interface, ideal for beginners, applications can be downloaded to the Android phones and published on Google Playstore. Although in MIT App Inventor the applications cannot extensively be modified and edited as similar to the Android Studio, this project however will not require such complex modifications and simple mobile application design is sufficient to meet the requirement of this project. Table 4 shows the comparison of the application development tools.

Since this project will be using the NodeMCU from Devkit which can also be programmed by using the Arduino IDE software, thus this project will be utilizing the Arduino IDE software. Another reason why the Arduino IDE is selected for this project is because the data can be easily sent to the Firebase database from the microcontroller using the Arduino IDE software by downloading the Firebase library. The Arduino IDE software was originally intended to program Arduino microcontrollers; however, there are other microcontrollers like the NodMCU that can be programmed from the Arduino IDE software. The Arduino IDE software uses the C programming language. The basic structure of the Arduino programming format consists of the setup () and the loop () functions. The setup () function is the preparation or initialization of the program. It should contain the declaration of variables used in the program like int, the pinMode declaration on the type of pin (input or output) and it should also include the serial communication initialization.

Table 4: Comparison of Mobile Application Development Tools

Tool	Advantage	Disadvantage
MIT App Inventor 2	-Does not require software installation (Web based). -Easy to program using program blocks. -Can be easily connected to the Firebase database by dragging the Firebase component from the Application Inventor Designer. -Applications can be tested immediately while developing the App. -Very friendly user interface, ideal for beginners. -Applications can be downloaded to the -Android phones and published on Google Playstore.	-Components given cannot be extensively modified or changed. -Cannot perform advanced graphical editing on the Application.
Android Studio	-Very suitable for commercial applications in Android. -Moderately easy to use. -Applications can be downloaded to the Android phones and published on Google Playstore. -Components can be extensively modified or changed. -Can perform advanced graphical editing on the Application.	-Requires software to develop App. -Requires programming language (XML).

3. Research Methodology

This section consists of the methodology of the parking sensor network project. The methodology is basically a series of detailed steps on how the project is conducted. These steps will include the hardware and software implementation based on the methodology flowchart as shown in Figure 13.

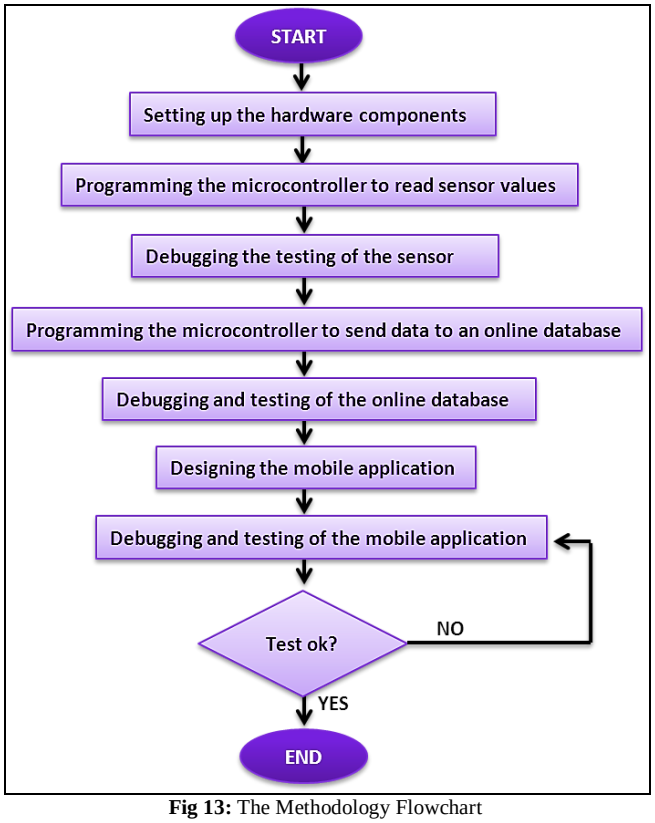

Fig 13: The Methodology Flowchart

3.1. Setting up the Hardware Components

As discussed in the literature review, the NodeMCU has four VCC pins of which three are 3.3Vdc and one 5Vdc. From the review, of the ultrasonic sensor (HC-SR04) it is known that the sensor requires 5Vdc to operate. Thus, the first step will be to test the microcontroller to check whether the VCC values from the microcontroller are accurate. In order to check the VCC values of the microcontroller the black probe of the multimeter is connected to the ground pin and the red probe to VCC. The testing was conducted and it was observed that the value displayed on the multimeter was accurate and meets the initial assumption. The Figure 14 shows the connection of the ultrasonic sensors and the NodeMCU microcontroller using the Fritzing software.

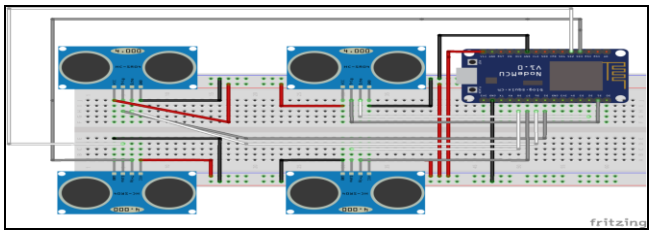

Fig 14: The connection of the ultrasonic sensors and the NodeMCU microcontroller using the Fritzing software.

3.2 Programming the Microcontroller to Read Sensor Values

As discussed in the literature review, this project will be using the Arduino IDE software to program the NodeMCU microcontroller. In order to program the NodeMCU using the Arduino IDE a board package will first have to be installed on the software.

3.3 Programming the Microcontroller to Send Distance Values to an Online Database

As discussed in the literature review, this project will be using Google Firebase as the online database in the implementation. Thus, before attempting to send values to the database, the database will first have to be created and setup. In order to create a project on Firebase an account has to be created, or any existing Google account can also be used to login and then a new project can be created.

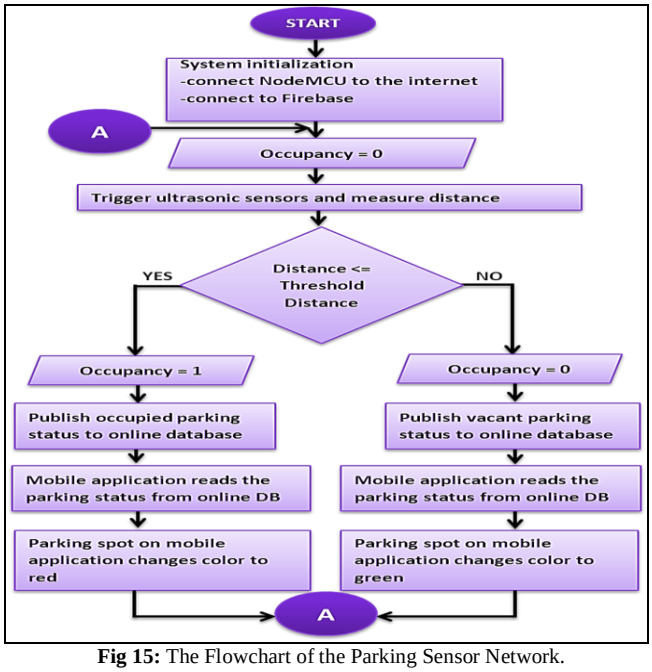

Fig 15: The Flowchart of the Parking Sensor Network.

In order to implement the parking sensor network, the program would need to be designed and planned accordingly. Thus, the overall program flowchart has to be constructed to enable proper implementation of the parking network. The program flowchart is shown in Figure 15. The program begins with the system initialization which includes the input and output pin definitions, Wi-Fi ID and password, Firebase authentication and link details, and library definitions. The program then creates a variable that stores the occupancy value of the parking space, 0 indicates a vacant parking space and 1 indicates an occupied parking space. The program then triggers the sensors to measure the distance values and store them in a separate variable. The program will then compare the distance value measured from the sensors to a threshold distance value of 5cm. If the distance measured from the sensor is less than or equals to 5cm, the occupancy variable will be updated to a value of 1. Otherwise it will remain 0. Each of the sensors will have their respective occupancy variable. After the respective occupancy value is updated, the microcontroller will send the occupancy value of the respective sensor to Firebase. The mobile application then proceeds to read the occupancy value of the respective sensors. When the value of the occupancy variable is 1, the parking spot displayed in the GUI (Graphical User Interface) of the mobile application will be red in colour. When the value of the occupancy variable is 0, the parking spot displayed in the GUI of the mobile application will be green in colour. The program will then return to trigger the sensors and measure the distance values. The new distance values are then compared with the threshold distance of 5cm and the occupancy is updated once again. This cycle is repeated continuously in an infinite loop process.

3.4 Designing the Mobile Application

As discussed in section II after conducting the literature review, the tool that will be used to design the Mobile Application will be the MIT App Inventor 2. Before designing the App, the App can be connected to an Android phone to observe the real-time changes in the GUI design and the functionality can be tested. Once connected, the App can be built. In order to test the implementation of the parking sensor network two buttons will be used to resemble parking space 1 and 2. Once the buttons have been selected, another important component needed is the Firebase component to allow the App to connect with Firebase. Once the components design has been completed, the App design can proceed with the programming part. From the App Inventor Blocks Screen the program blocks can be dragged into the blocks designer window. The blocks can then be used to form blocks of programming code which like a puzzle. Different blocks can be selected by navigating the Blocks tab on the left side of the screen.

4. Results and Discussion

After taking the relevant steps needed to design the parking sensor hardware components, creating and setting up the database and designing the mobile application, the results of the parking sensor can then be obtained and analysed. The results and discussion session focuses on the initial parking sensor network using 2 sensors and the final parking sensor network using eight sensors. This chapter will also cover a brief comparison between the actual distance value and the distance value obtained from the sensors. Table 5 shows the detailed comparison. In order to develop the parking sensor network using the ultrasonic sensors, the sensors will have to provide a close to accurate distance value. This is important in reducing the overall error in the system and validating the accurate of the occupancy value. A simple experiment was conducted using one ultrasonic sensor and having a side-by-side comparison between the distance from a measuring tape and the sensor. This value is then tabulated and graphed to analyse and compare the variations visually. From table 5, it can be observed that there

are definitely variations between the actual distance and the sensor distance. The highest variance between the actual and sensor distance is at 5cm and 10cm. The error is substantially lower and in between 15cm to 40cm. From Figure 16, it can be observed that the sensor distance is always lower that the actual distance by 2cm. It can also be noted that the sensor distance is very close to actual distance, thus the value can be considered to be reliable. Since the threshold distance used in the parking sensor network is 5cm, the lower sensor distance can work in favour to the parking network. For instance, if the actual car is parked 6cm from the sensor, the program will recognise the parking spot as occupied. This will improve the overall efficiency of the parking sensor network.

Table 5: Comparison between the Actual Distance and Sensor Distance.

Actual Distance (cm)	Sensor Distance (cm)	Percentage Error (%)
5	3	40.00
10	8	25.00
15	14	6.67
20	19	5.00
25	23	8.00
30	27	10.00
35	33	5.71
40	37	7.50

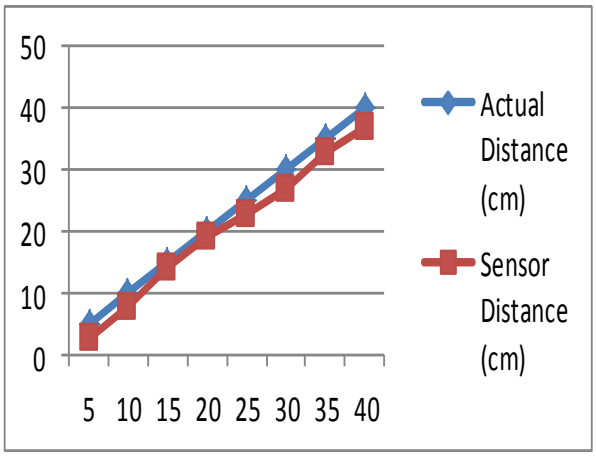

Fig 16: Graphical Comparison between the Actual Distance and Sensor Distance

4.1 Initial Parking Sensor Network Results

As discussed and shown in Section III, the initial parking sensor network was designed using two ultrasonic sensors and one NodeMCU microcontroller. The network was tested by placing an object in front the sensor within the threshold distance range and observing the changes on the database and mobile application. The obtained results is shown and discussed in Figure 17 and the Firebase DB results when the parking space is vacant as shown in Figure 18. From Figure 17, it can be observed that when the parking spaces are vacant or empty the colour of the parking spots in the mobile app GUI does not change and remains at the initial colour of Green. From Figure 18, it can be observed that the occupancy variable of parking1 and 2 is 0 when it is vacant. It is also can be seen from Figure 19 that when the parking spaces are occupied the colour of the parking spots in the mobile app GUI changes colour from the initial colour of Green to Red and based on Figure 20, it can be observed that the occupancy variable of parking1 and 2 changes from an initial value of 0 to 1 to represent the occupied parking spaces. Thus, it can be concluded that the parking network is functioning accurately and as intended. This network can then be further expanded to include more parking spaces to simulate a real world parking lot.

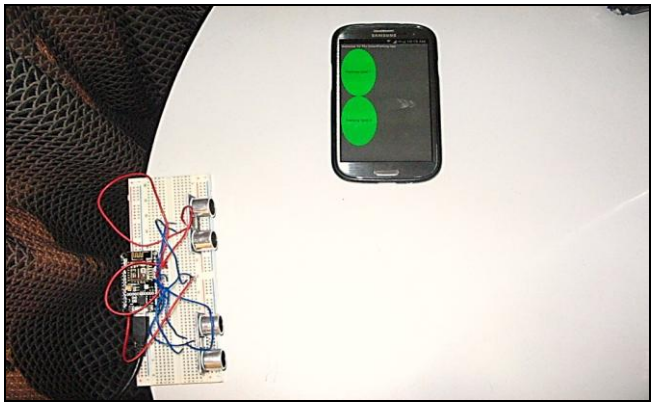

Fig 17: Results of Mobile Application when Parking1 and Parking2 is Vacant

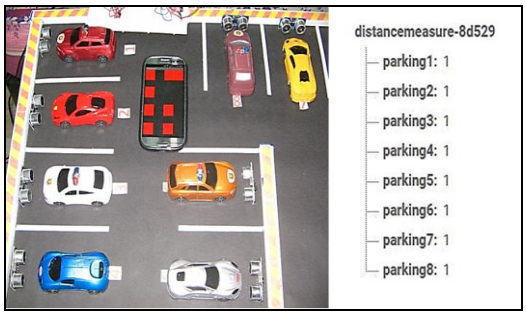

Fig 18: Results of Firebase Database when Parking1 and Parking2 is Vacant

Fig 19: Results of Mobile Application when Parking1 and Parking2 is Occupied

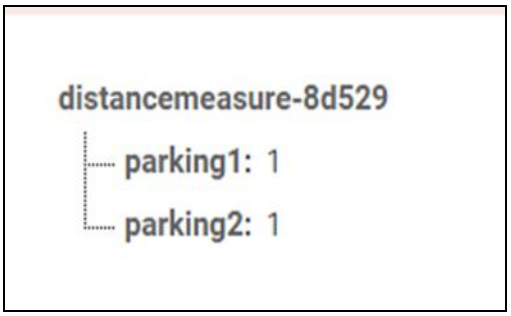

Fig 20: Results of Firebase Database when Parking1 and Parking2 is Occupied

4.2 Final Parking Sensor Network Results

As discussed in Section III, the final parking sensor network consists of eight ultrasonic sensors and two NodeMCU microcontrollers. The hardware is connected as shown previously in Section III using the fritzing software. The final parking network is tested using a toy vehicle to resemble an actual vehicle in a real car park. The results of the Firebase database and the mobile application is observed and recorded.

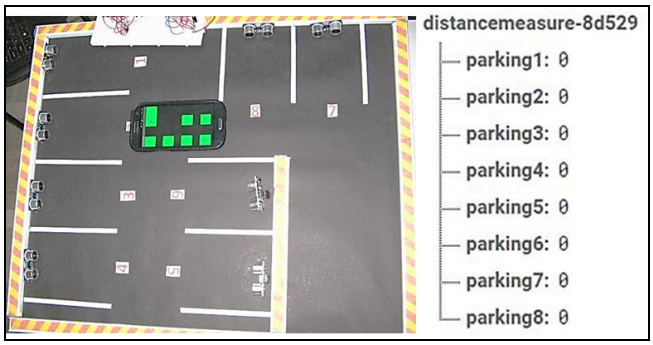

Fig 21: The Mobile App and Firebase Results when all the Parking Spaces are Vacant

Fig 22: The Mobile App and Firebase results when all the parking spaces are occupied

From the results of the parking sensor network based on Figures 21 and 22, it is evident that the network is working accordingly. The occupancy value is updated accurately when a car is parked in a parking spot and the value remains unchanged until the car leaves the spot. The occupancy value is also displayed accurately when the parking is vacant. It is also evident that the mobile application is functioning properly as intended. The colour of the parking spot GUI immediately changes to Red when the occupancy value is registered as 1. When the value of the occupancy is 0, the parking GUI immediately changes colour to Green. The intended outcome showed that the project is successful in its implementation and desired outcomes. The accuracy of the mobile app results shows that it can be implemented in a smart campus environment effectively.

5. Conclusion

The parking sensor network works by implementing the basic concepts of the Internet of Things. Such concept has been utilized in this project to achieve the desired outcomes. The parking sensor network in this project was able to display real-time data on a GUI feed which in turn able to improve the convenience and efficiency of finding a parking space in a smart campus environment. This was established through the use of an online database and a mobile application. A fully functional mobile application was also designed in this project for the sole purpose of communicating with the data shared to the internet by the sensors. The mobile application was tested and shown to be working as intended. The mobile application was able to effectively display the availability of parking spaces at all times. Thus, the project is deemed to be successful in its implementation. The initial objectives of the project were all met and fulfilled effectively using the concept of Internet of Things. For future improvements, this project can be enhanced by implementing the parking sensor network in an actual car park. The inductive sensor can be used along with the ultrasonic sensor to improve the overall accuracy of the occupancy values. The mobile application can be improved by implementing GPS feature in the application. The GPS feature will help users to find any nearby vacant parking spots based on the users' actual location. The mobile application can also be improved by including a reserve function. While viewing the availability of parking spaces may be convenient to users, but a parking space may have already been occupied by the time the person arrives at the sport. The reserve function will be a very convenient feature which can further improve the efficiency of finding parking spaces for users. Upon implementing the reserve function, a payment system can also be designed in the app which charges users for reserving a parking space. This is important as it will deter users from reserving parking spaces excessively and only use it when needed.

Acknowledgement

This study was supported in part by UNITEN Seed Fund with the project code J510050692.

References

[1] H.M.A.P.K. , B, J.D.C. , J, A.R.S.P. , R, A.U., , Z. , M & R.G.l, 2016, 'Proceedings of the 1st Manufacturing & Industrial Engineering Symposium', Smart Campus Phase One: Smart Parking Sensor, no. 16.

[2] Chidella, KK, Asaduzzaman, A & Mridha, MF 2015, 'Proceedings of the IEEE SoutheastCon', A Time and Energy Efficient Parking System Using ZigBee Communication Protocol, no. 15.

[3] Ramaswamy, 2016, 'FIFTH INTERNATIONAL CONFERENCE ON RECENT TRENDS IN INFORMATION TECHNOLOGY', IoT Smart Parking System for Reducing Green House Gas Emission , no. 16.

[4] Adhatarao, , Alfandi, , Bochem, & Hogrefe, 2014, 'IEEE Vehicular Networking Conference (VNC)', Smart Parking System for Vehicles, no. 14.

[5] M.P. Jones, "HC-SR04 User Guide," Data Sheet. [Online]. https://www.mpja.com/download/hc-sr04_ultrasonic_module_user_guidejohn.pdf

[6] Anish A. Naik, Deeksha A. Naik, Shruti S. Naik & Shubham D. Naik 2016, 'IJSTE - International Journal of Science Technology & Engineering', Inductive Proximity Sensor Interfaced with Arduino, vol 2, no. 09.

[7] Texas Instruments 2013, 'CC3200 SimpleLink™ Wi-Fi® and Internet-of-Things Solution, a Single-Chip Wireless MCU', SWAS032F, Texas Instruments, Dallas

[8] Handson Technology, "ESP8266 NodeMCU WiFi Devkit User Manual V1.2," Leipzig, Datasheet.

[9] N. Singh, "International Journal of Innovative Research in Computer and Communication Engineering," Study of Google Firebase API for Android, vol. 4, no. 9, pp. 16738-16743, 2016.

[10] F. Turbak, "MIT App Inventor Getting Started Guide," UserGuide 2012.