International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 08 Issue: 02 | Feb 2021 www.irjet.net p-ISSN: 2395-0072

Internet of Things based Real-Time Elevator Monitoring System

Dilpreet Luthra[1], Vivek Sharma[2], Utkarsh Kant Khare[3], Honey Nalwaya[4]

[1]Student, VIT University, T.N, dilpreetluthra5@gmail.com
[2]Student, VIT University, T.N, vivek.sharma99133@gmail.com
[3]Student, VIT University, T.N, utkarshkantkhare@gmail.com
[4]Student, VIT University, T.N, honeynalwaya99@gmail.com

Abstract - *This paper focuses on the development of an application based on the Internet of Things that enables the maintenance company and the technician to real-time monitor the status of the elevator and to receive notifications about emergency or scheduled maintenance period. The application comprises of the elevator's communication module, the IoT platform, and the front-end android application. With the help of the communication module, data gathered by the sensors on the elevator is sent to the IoT platform where the data is stored, visualized, and analyzed. From the IoT platform, the data is then sent to the mobile application. The front-end mobile application is responsible for notifying the technician and the maintenance company about any emergency or scheduled maintenance period. This application also provides a map feature that shows the shortest path from the technician's location to the elevator. The main goal of this paper is to make the elevator ride safe for the passenger by ensuring timely maintenance and prompt response to an emergency.*

Key Words: Android, Elevator, Emergency, Internet of Things, Monitor, Real-time.

1. INTRODUCTION

Elevators are the vertical moving transportation machines that carry freight or passengers between the levels of the building. Because of the elevator, it become possible for the engineers to carry out the construction of high rising buildings. Today, elevators are a vital part of our daily life. With just one press of a button, we can go to any floor in a building. But being so easy-to-operate and time-saving, how much safe these elevators are? Many times because of an unattended problem, the elevator malfunctions and suffers an accident. Passengers get seriously injured and sometimes even die in such accidents. Also, because of delays in maintenance, elevators go out-of-service, which causes great inconvenience to the one who wanted to go to a higher floor and especially to those who are not abled. So, what can be the possible solutions to ensure passenger safety and to keep the elevator fully functioning all the time?

One possible solution is to frequently monitor the parameters that play a vital role in the proper functioning of the elevator. An elevator has lots of sensors installed on it, for example, a door sensor, speed measuring sensor, temperature sensor, governor to regulate the elevator's speed, weight sensor, level sensor, etc. We need to make the data obtained by these sensors available to the maintenance company and to the technician so that they can track the elevator's working condition. But how can we do that?

Internet of things (IoT) is a network of "things" that incorporates sensors, software, and other technologies. Devices in this network can communicate with each other by exchanging data, can decide the working of each other, and can also establish new connections. Using IoT, it is possible for us to connect sensors on the elevator with the network and send the data obtained by the sensors to the IoT platform, where the data will be processed and analyzed to prevent any accidents, to avoid maintenance delays, and to respond to any emergency promptly.

We implemented a front-end mobile application in the technician's and maintenance company's mobile phone, which will receive the data from the cloud continuously, notify them about unusual changes in data or schedule maintenance period, and provide visualization of the data received from the cloud. Using this application users can also access the past data of the elevator. We also included a map service in this application, which will give the user the shortest path from user's location to the elevator.

2. BACKGROUND

Alejandro Duarte Suárez, Octavio José Salcedo Parra, and Jhon Hernán Díaz Forero, in 2018, proposed a design in their article, Design of an Elevator Monitoring Application using Internet of Things, in which they implemented an application on the cloud that is responsible for receiving the data and processing it. Based on the data processing results, the application notifies the technician. The backend of the application is implemented in Microsoft Azure and the frontend in the technician's mobile phone. Whenever a new data is received, its ID is consulted in the database initiated by Azure, and if necessary notification with all the appropriate information like building in which the elevator is installed, type of elevator, etc. is built-up and is sent to the frontend application on the technician's mobile phone. They verified their application by making a series of fault reports and observed the time it takes to notify the frontend. Concerning the application, it takes about 10 seconds to start the connections and make the necessary queries for the operation of the application [1].

In 2016, Xiaohang Pan designed an elevator safety monitoring system based on the internet of things in the article, The Design and Reliability Analysis of Elevator Monitoring System Based on the Internet of Things. He monitored the environmental parameters like temperature and humidity. He has used CC2530 processor and communication module, DHT11 for getting the temperature and humidity, ADXL345 module for acceleration monitoring, and LM2576S for supplying 3-3.3V. The results showed that his easy-to-operate design could achieve real-time monitoring of environmental parameters with high precision on the running state of the elevator. [2].

In the wake of increasing elevator accidents in China, Zihan Ming, Shaoyi Han, Zhanbin Zhang, and Shuang Xia in the year 2018 authored an article, Elevator Safety Monitoring System Based on Internet of Things, in which after a complete analyzation and evaluation of elevator safety monitoring system based on its functionality, performance, and feasibility, they presented a design scheme which comprises of Browser/Server and Client/Server architectures. They used Da Vinci software architecture and IETF's proposed real-time transmission protocol (RTP) for video and audio transmission. Their command and control system's front-end interface has frames, where each frame shows the video surveillance of an elevator. It also shows the real-time operation status of elevators. The result of this study has shown the value and significance of the development of such systems [3].

In 2015, in the article, Design of Elevator Monitoring and Alarm System Based on WiMAX, authored by Hong Jiang, Yong Shi, and Lei Qi describes a WiMAX(Worldwide Interoperability for Microwave Access) based elevator monitoring and alarm system. The system includes fault detection, trapped people communication, and information management for informing the maintenance company. Their study shows that WiMAX could be a good choice over Wi-Fi since it covers a large geographical range. Their research work is another example of how internet of things can transform the elevator safety [4].

The cloud platform is the most crucial part of an IoT application. Therefore, selecting the right cloud platform based on the requirements of the IoT project is very important. In 2017, an article titled, A survey of IoT cloud platforms, authored by Partha Pratim Ray, surveys some popular IoT cloud platforms based on various service domains like application development, device management, system management, heterogeneity management, data management, tools for analysis, deployment, monitoring, visualization, and research. At the end of the survey, 26 different cloud platforms were compared based on the aforementioned service domains [5].

In 2016, Sharmad Pasha worked on a project in the article, Thingspeak Based Sensing and Monitoring System for IoT with MATLAB Analysis, that uses the Thingspeak IoT platform for monitoring the sensed data at the cloud level. Thingspeak allows us to make different channels and specify their fields based on our project requirements. The project uses the Arduino UNO board and ESP8266 Wi-Fi module for processing and transferring the data to the Thingspeak. Thingspeak's API services provide a feature of porting the sensed data to the MATLAB, using channel ID and API Key, for further analysis. In this project, values for different environmental parameters like temperature, humidity, rain, heat, light, air quality, barometric pressure, and sea level pressure were sensed and sent to Thingspeak using the Wi-Fi module. In the end, different sensed parameters were visualized on Thingspeak and analysed on MATLAB R2016a. This project has shown that Thingspeak is a captivating web service and an easy-to-operate tool to track the sensed data [6].

In 2018, Arun V and Dr. M. Poongothai authored an article, Implementation of IoT Based Intelligent Transportation System. Intelligent Transportation System (ITS) is a well-known method to minimize traffic problems. This article proposed IoT based ITS which aims to locate and navigate the bus of the author's college. The article uses a GPS module to acquire the geographical coordinates of the bus at a regular time interval and update the same on the Thingspeak IoT platform using the NodeMCU. It also incorporates the Google Map API and a user-friendly mobile application developed using MIT APP Inventor. The users of this application can track the location of the bus anytime, irrespective of their own location. The complete system was implemented successfully using the Arduino IDE and MIT APP Inventor [7].

In the year 2013, Shaileen Crawford Pokress and José Juan Dominguez Veiga wrote an article, MIT App Inventor Enabling personal mobile computing, in which they have given a brief introduction about MIT App Inventor and described how it promotes a new era of personal mobile computing where people without having any prior coding experience, can create, design, and use their android application. MIT App Inventor allows makers to focus on the logic of the application program rather than its syntax [8].

3. METHODOLOGY

To demonstrate the working of proposed application, we selected six parameters: temperature, level, speed, governor, voltage, and emergency state. The first five parameters will be sensed by sensors and the last parameter will be determined during the program execution on the basis of the temperature and speed parameter. We wrote a python program in which for each sensor, we defined a random function with a range somewhat higher than the sensor's range indispensable for the precise functioning of the elevator. These high ranges are to set the emergency state to high or 1 so that we can demonstrate the functioning of proposed application during an emergency and the elevator malfunction. The random functions can be treated as the sensors on elevator. The program also involves API KEYs and IDs for updating the data on the IoT platform.

Out of 26 IoT platforms, which were discussed in [5], we find Thingspeak to be the most suitable platform for our application. It has got what exactly we want. It is an open-source platform licensed under GPL version 3 and written in Ruby. It has different pricing for academic and personal use purposes. In our application, Thingspeak is like an in-charge whose work is to receive the data, visualize it, and send it to the front-end mobile application.

The front-end application was developed using an open-source application development software, MIT App Inventor, and was implemented in an android mobile phone. The front end mobile application work is to receive the data from the Thingspeak and to notify the user about high emergency state in data and scheduled maintenance period. In case of an emergency, technicians and maintenance companies can use a map feature of the application, which shows the shortest path from their location to the location of the elevator in an emergency.

4. THINGSPEAK

Thingspeak is an open-source Internet of Things platform developed by MathWorks. It is written in Ruby programming language and licensed under GPL version 3. It is very useful and easy-to-use platform. Thingspeak offers the services for gathering the data from different devices, visualizing the gathered data, to allow other applications to access the data using the API key, and also incorporate MATLAB support for further analyses. The data on the Thingspeak get stored in the channels. Each channel has unique read and write API keys using which users can access and modify the data. The channels by default are in private view and users can make it public by changing the channel sharing settings. Users can also download the data using the read API key in the CSV or JSON format.

To get started with Thingspeak, we need to create an account. Thingspeak provides free accounts also. Once we have logged in, the new channels can be created by clicking on the "New Channel" option. After clicking on it, we have to give the channel a name and need to specify its fields. The metadata for the channel can be entered in the form of a JSON file, XML, file, or CSV file. The channel can also be linked to a website or GitHub account. We can also specify the location of the channel. This is helpful in tracking a moving object (refer to figure 1). Once all the required fields and options are filled click on the save button. In the

channel stats, we can see that four empty graphs are created with heading as the channel name, date in the x-axis, and specified field in the y-axis (Field Label 1). The received data will be visualized here. Each graph corresponds to a field in a channel (refer to figure 2).

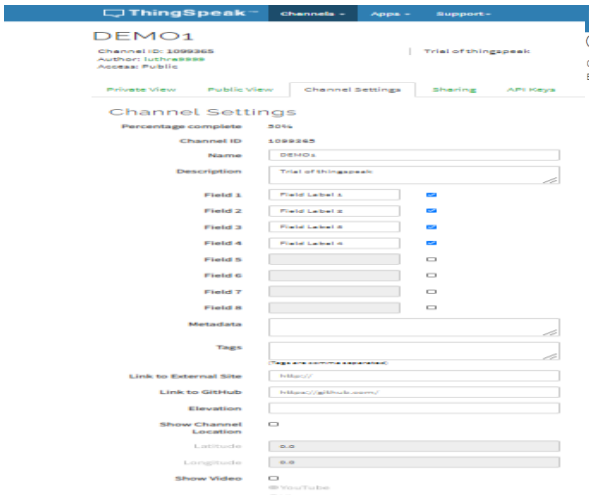


Fig – 1: Required fields and other options

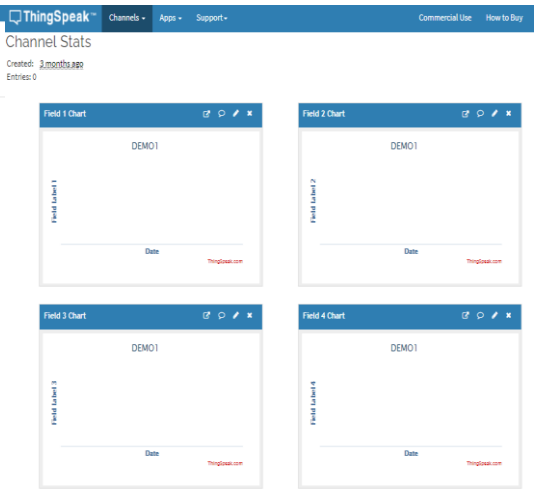


Fig - 2: Empty graph for each field

Now, how to send the data to the channel that we created above? We can do so by using the API keys provided under the option "API Keys" (refer to figure 3). The API keys and requests are auto-generated when the account is created and is unique for a user and a channel. Users can generate new API keys whenever there is a need. Now suppose we need to write the data in field 1 of Demo 1 channel created above. We need to copy the URL "under API Requests" and paste it into the browser. Then specify the field we want to change in the URL and assign it a value. For example, the field is 1 and the value is 10, then the URL will be: https://api.thingspeak.com/update?api_key=API_KEY&field1=0. This will update the value on the Thingspeak channel Demo 1(refer to figure 4). In the same way, read URLs need to be changed for reading the data.

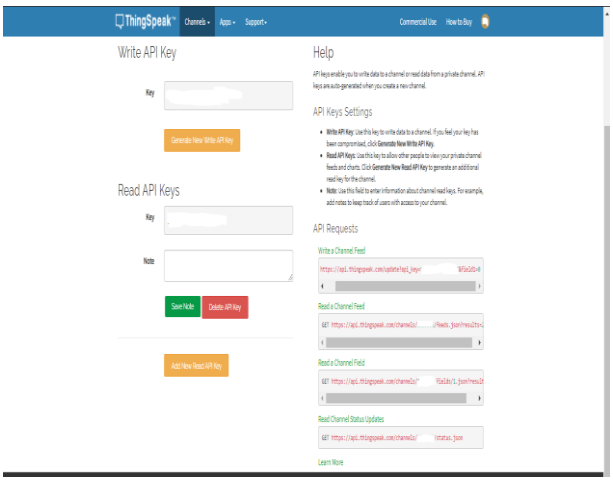


Fig - 3: Read and Write API Keys and URLs

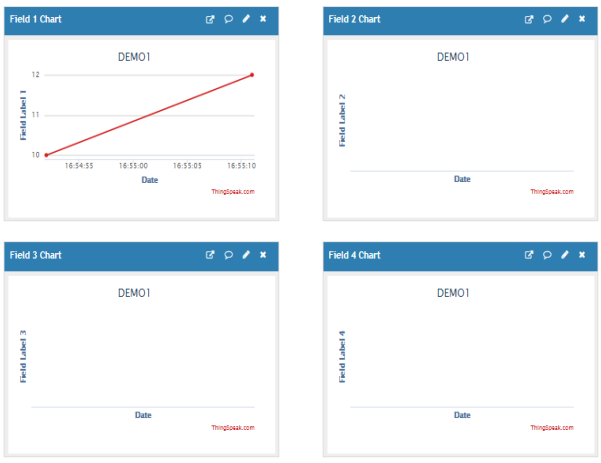
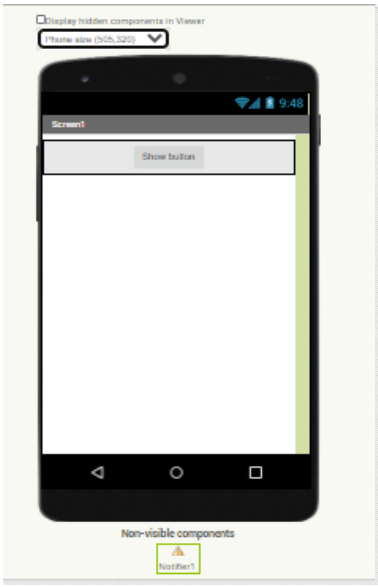


Fig - 4: Updated visualization

5. MIT APP INVENTOR

MIT App Inventor is a free cloud-based open-source software that allows to build applications on the web browser itself. It was originally developed by Google but now maintained by the Massachusetts Institute of Technology (MIT). It allows newbies to computer programming to create application software for Android phones. It provides blocks for different functions which work very similar to the functions of any programming language. The biggest advantage of using MIT App Inventor is that it allows users to drag and drop visual objects to create an application that can run on android devices. Users need to design the application interface in the design window and implement logical blocks for it in the block window. Once the user is done with the design and block implementation, the application can be tested using an AI companion, emulator, or USB. Once the user is sure that the application is working fine, he can download the .apk file and implement it on any android phone. There is no subscription needed for using MIT App Inventor, anyone with a Google account can use it.

The first step in creating an application in App Inventor is to design the screen and add components to it. App Inventor provides three types of screen for designing: phone size screen, tablet size screen, and computer size screen. Users can choose any of the screens based on their requirements. Suppose we want to make an application that shows some alert dialog box when the show button is pressed. For doing so, we have to drag the vertical alignment from the palette section to the screen and then insert the button inside this vertical alignment. The vertical alignment is used when the components are to be placed below one another. For placing the components side-by-side horizontal alignment should be used. We can edit the properties of the used components in the properties section. Now, for the dialog box, we need to add one non-visible component, Notifier. Whenever a logic calls notifier, a dialog box will appear on the screen (refer to figure 5)



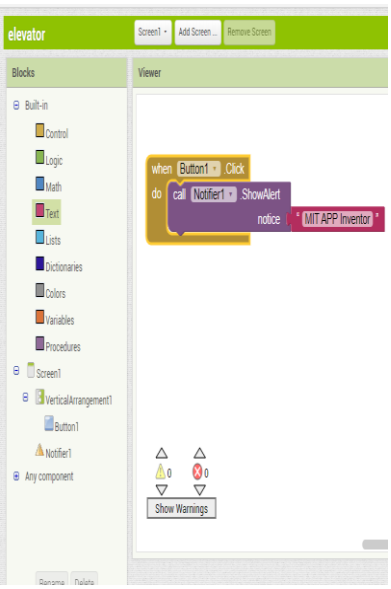


Fig - 5: Design screen **Fig - 6:** Block screen

Now the second step is to implement the logic for our design. For this, we need to switch to the block window. The Block window has control, logic, math, text, list, dictionaries, colors, variable, and procedure blocks. We can also find blocks corresponding to the components that we added to the design screen. For implementing the logic for the screen shown in figure 5, we need to drag the required functions and arrange them in way that satisfy our logic (refer to figure 6).

Now once the application is created we just need to download its .apk file and install it on the android mobile phone (refer to figure 7 and figure 8).

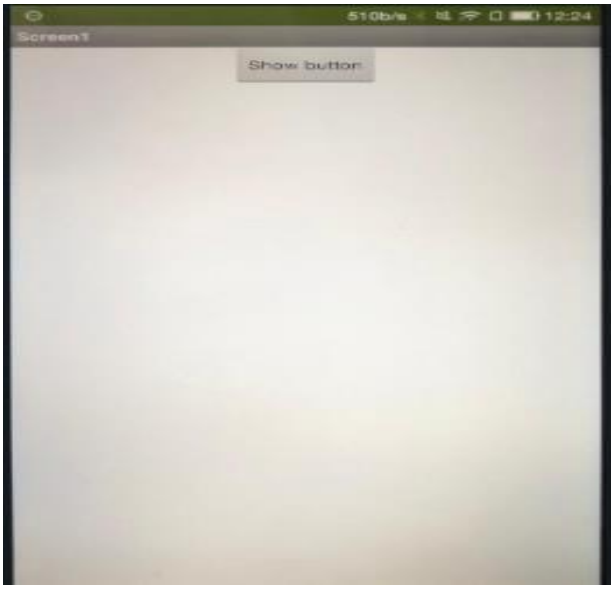


Fig - 7: Application running on phone

Fig - 8: Dialog box appeared when show button is pressed

6. WORKFLOW

As mentioned above, we have selected six parameters that we would be sending to the Thingspeak. There are many other parameter that is important for elevator working, but for the sake of demonstration we selected only six parameters. Below we have mentioned those parameter with the range required for correct functioning of elevator and with the range that we have provide to the random function.

- Temperature - The temperature of the elevator's motor room should be between 60 to 90 degrees Fahrenheit. The random function is defined with a range of 55 to 110 degrees.
- Level - The level sensor estimate the elevator's position with respect to floor level. It can value between -1 to 1. A value near -1 means the elevator is somewhat below the floor level, near 1 means somewhat above the floor level, and near 0 means at the same floor level. The random is defined with the same range.
- Speed - The safe speed for the elevator should be between 2.2 m/s to 9.8 m/s. The random function for the speed will have a range of 2.2 m/s to 14 m/s.
- Governor - It is a speed regulator. Whenever speed crosses its safe range the governor is set to high and the brakes are applied. It can only two values, either 0 or 1.
- Voltage - The voltage value should be between 200 - 230V. The random function is declared with the same range,
- Emergency State - This is a very important parameter based on which alert messages will be sent to the technician or maintenance company. It can have only two values, either 0 or 1. Here, 0 represents no emergency, and 1 represents an emergency. Its value will become 1 when the temperature and speed are not in their safe range and when the passenger presses the emergency button on the elevator panel.

The parameters: temperature, level, speed, governor, and voltage are sensed by sensors. But it is very difficult to obtain exactly the same readings of the sensors using a prototype that we will get when sensors are installed on the elevators. Therefore, we decided to define random function for each of the sensor. As mentioned, the range for these random functions are higher than the required one, therefore there will be occasions when the generated values are high enough to satisfy our program logic and set emergency state to high or 1. This is done only to demonstrate the working of application during emergency and the elevator malfunction. Once the value is generated it is sent to Thingspeak. On Thingspeak, three channels are created each with

six fields. The first five fields correspond to sensors and the last field to the emergency state. Whenever new data is received it gets stored on Thingspeak and the graphs for each field get updated on the channel stats (refer to figures 9 and 10).

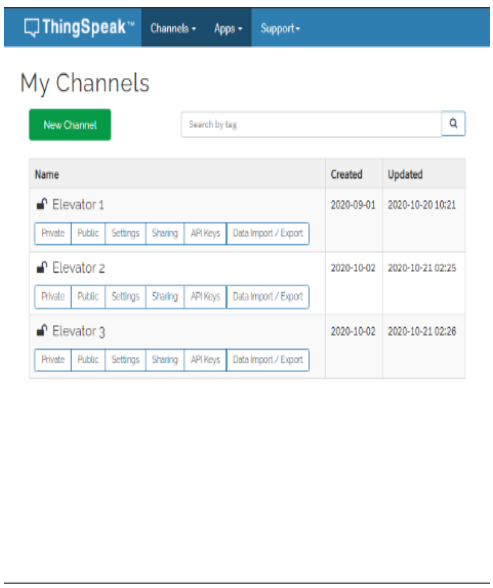


Fig - 9: Channels for each elevator

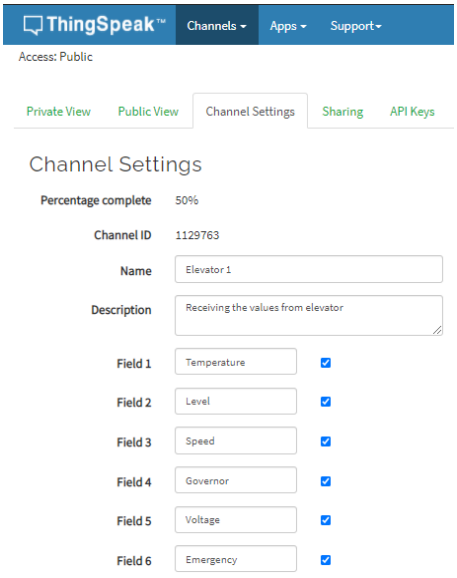


Fig - 10: Five fields of sensors and one of Emergency

Once the data has been stored and visualized on Thingspeak, now we can retrieve this data on the front-end application implemented on the mobile phone of the technician and the maintenance company. For data retrieval, we have to use the read API Request URL. Since our main goal is to notify the technician and the maintenance company about an emergency, therefore we have only retrieved the current value Emergency State. The URL: https://api.thingspeak.com/channels/1129763/fields/6/last, will give us the last value of the Emergency State (see figures 11 and 12). If the emergency state is 1, the application builds a message and sends it to the user as a push notification. The status option in the application will be updated with an emergency warning and will provide visualizations for all the parameters using which technicians can determine the type of emergency. The complete information about elevator like address of the building, contact number etc. can be viewed in status option. The map feature on front-end application already has locations of three elevators and the office. The map on the application will automatically shows the shortest path to the elevator which technicians and maintenance companies can follow to reach the emergency location as soon as possible. This application keeps the track of maintenance dates. The maintenance company needs to set the date of maintenance and this application will notify them about maintenance date. In this way, an emergency can be responded promptly and maintenance can be done timely.

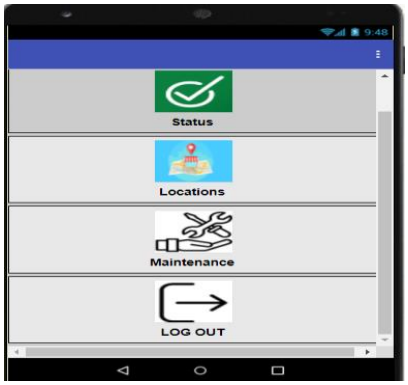


Fig - 11: Options available on front-end

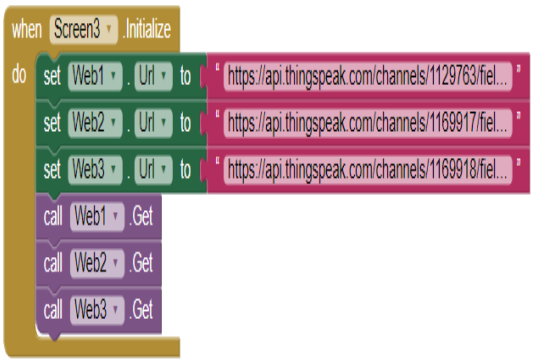


Fig - 12: Using Read API Request URL for retrieving data

Figure 13 describes complete flow of our proposed application. First the data is generated by sensors (here random function can be treated as sensors). The program logic sets the emergency state to high or low. Then the data is transferred to the Thingspeak channel using write API. Then the data is retrieved and lastly the application checks the value of emergency state. If emergency state is 1 or high then notification is sent. Parallel to this process, the application keeps on comparing the dates. If the current date matches the maintenance date set by the user then the maintenance notifications are sent.

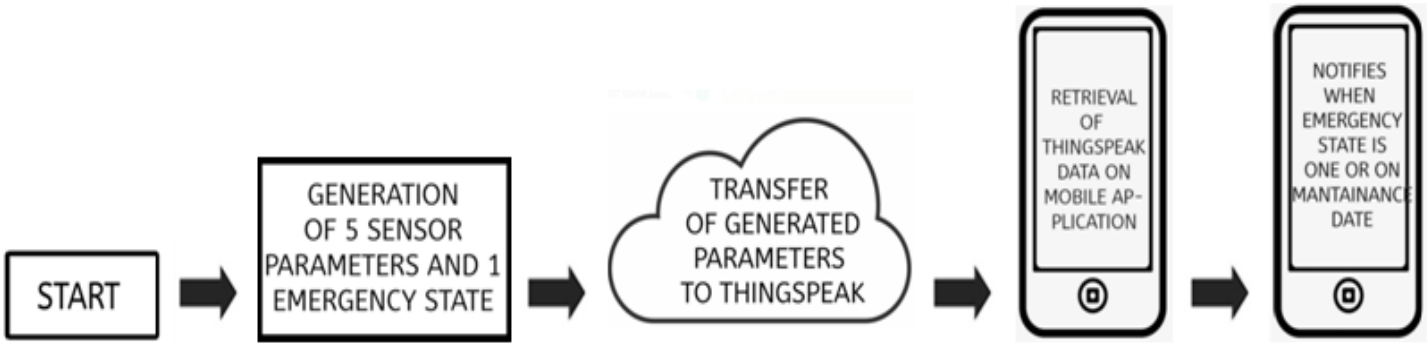


Fig - 13: Flow diagram of application

7. RESULTS

The proposed application involves three parts: First part is to send the data to Thingspeak. The sensor data generated by random function is sent to the Thingspeak. In program, we declared three variables that stores the write URL of three channels and defined a function under which generated data is being sent to the Thingspeak. We set an interval of 10 secs for updating the data on the Thingspeak. When we run the python program we found that the data was successfully uploaded on the Thingspeak but the time it took to upload data of each elevator was more than 10 secs. This happened because the same program was responsible for updating the data for the other two elevators. The total time taken to update the data of each elevator can be expressed as:

Assume Elevator 3 data is to be uploaded and the program is continuously running.

Formula: **Total time taken = 10 secs interval + time take to upload data of Elevator 1 + time taken to upload data of Elevator 2**

After every successful upload, a message “Uploaded Successfully” gets printed on the terminal. Figure 14 and 15 shows the result of first part of application.

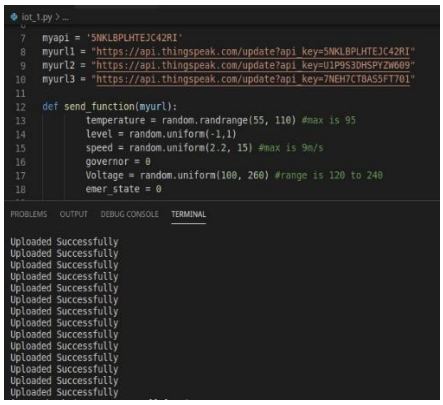


Fig - 14: Terminal showing message. **Fig - 15:** Thingspeak visualization

The second part of our application is to retrieve the data from the Thingspeak channel and make it available on the front-end application created using MIT App Inventor. For doing so we added two non-visible components Web and WebView. The Web is used to connect the application to the provided URL and get its content. Whereas, the WebView is used only to display the content present in the provided URL.

URL provided to the Web - https://api.thingspeak.com/channels/1129763/fields/6/last

The above URL consists of the last value of field 6 or emergency state. The GET function of the Web components makes this value available to the application. This value is then checked and if found equal to 1, a push notification will be sent. Also, the status option will get updated on the application.

URL provided to the WebView - https://thingspeak.com/channels/1129763/charts/1?bgcolor=%23ffffff&color=%23d62020&dynamic=true&results=60&type=line&update=15

The above URL consists of visualizations of field 1. WebView only displays these visuals in the status option. The data of these visuals are not made available to the application.

When we implemented the front-end application on the mobile phone, we found that there was no delay in updating the data on the front-end. Once the data is stored on Thingspeak, it get reflected immediately on the front-end application. When the user opens the front end application he has to enter user id and password. After logging in, he will see four options: Status, Locations, Maintenance, and Log out. On clicking on the status he see the elevator status as shown in figure 16. Then if he wants to see further information he can click the button of respective elevator. He would find visuals of all the parameters and other information about elevator as shown in figure 17.

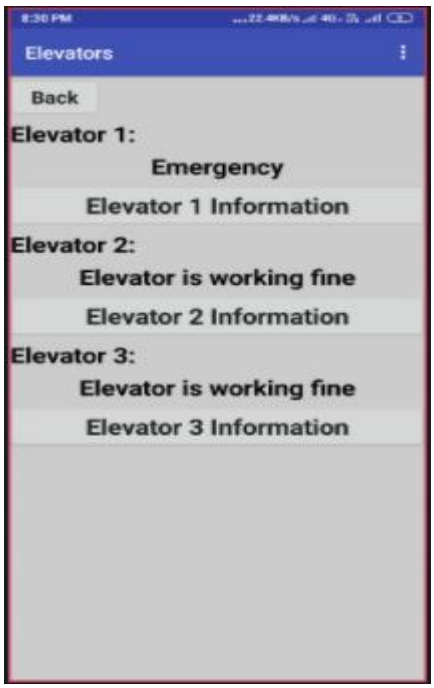


Fig - 16: Status of elevators

Fig - 17: Thingspeak visuals of all parameter

The third part of our application is to build the notifications for emergency and maintenance and also to show the shortest path to the elevator from the user's current location. For generating the notification we added a third party extension. The logic that is implemented in the block window of the front-end application keeps on checking the emergency state's value. Whenever the emergency state becomes 1, this third party extension generates a message and sends it in a form of push notification. Above in figure 18 we can see that elevator 1 is in emergency. As a result the front-end application sent a push notification (shown in

figure 18). When we opened the application and selected Locations option, we saw that the shortest path to elevator 1 is drawn successfully on the map (as shown in figure 19).

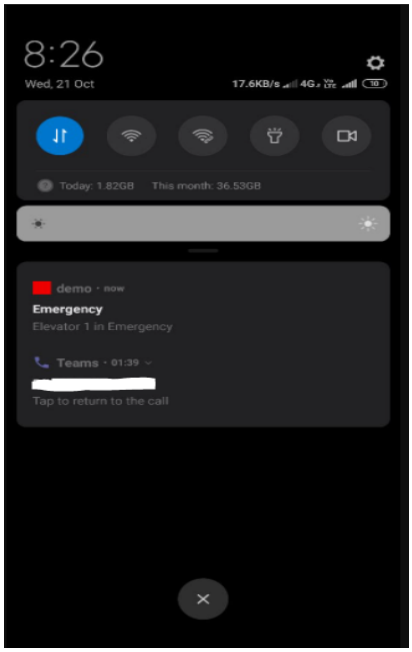


Fig - 18: Push notification received for elevator 1

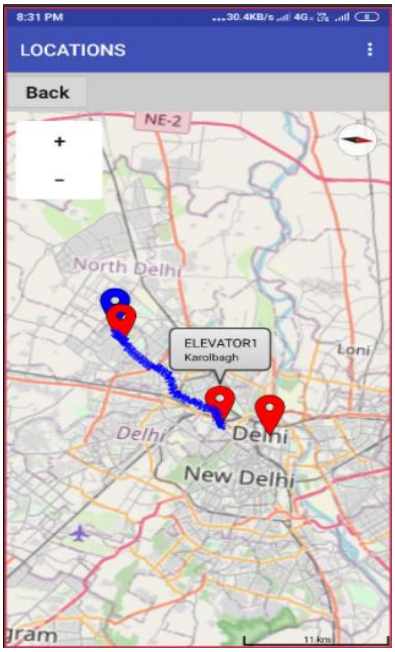


Fig - 19: Shortest path to elevator1

The front end application keeps checking the maintenance dates entered by the user. If the current data matches the date of the maintenance then a notification will be sent to the user. We tested this feature on 21st October 2020 and set the maintenance date of elevator 2 as 21st October 2020. As a result we received a notification for maintenance (shown in figure 20 and 21). The last option is the logout. There is one limitation of using application developed in MIT App Inventor that whenever user clears the background processes, the application stops working. The user should not clear the tab of the front-end application from the background at any case. Once the user has press log out, the application goes into the background process and keep on sending the notification about any emergency. If in case user clears the background, he just have re-login and press the logout.

Fig - 20: Maintenance date set by user

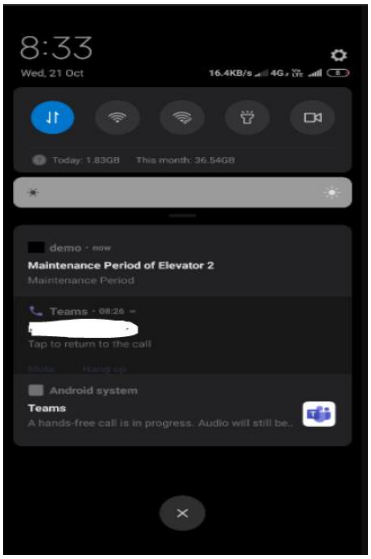


Fig - 21: Notification of maintenance

8. CONCLUSION AND FUTURE WORK

The Internet of Things has unimaginable benefits. It has the capabilities to completely transform the current remote monitoring systems. Using it we successfully implemented an application that can alert the maintenance company and the technicians about any emergency or scheduled maintenance work and help them to respond to any emergency promptly. Here, for the sake of demonstration purpose we generated reading from random functions. But our proposed application will work well with the sensors also. Instead of random function we can directly collect the data from the sensors and send it IoT platform using Wi-Fi modules and microcontrollers. The proposed application is very cost-effective because the sensors are already installed on the elevator and we just need to send it to the IoT platform.

In future we can work on making the application available languages other than English because in countries like India, technicians and the maintenance company might be more comfortable in their native language. Also, we can work on including the feature for fault type detection using machine learning techniques. We can use machine learning model and train it on sensor data and make prediction about future problem that elevator can face.

REFERENCES

[1] Alejandro Duarte Suárez, Octavio José Salcedo Parra, and Jhon Hernán Díaz Forero, Design of an Elevator Monitoring Application using Internet of Things,IJAER, Vol 13, 2018
[2] Xiaohang Pan, The Design and Reliability Analysis of Elevator Monitoring System Based on the Internet of Things, International Journal of Smart Home, Vol. 10, 2016
[3] Zihan Ming, Shaoyi Han, Zhanbin Zhang, and Shuang Xia, Elevator Safety Monitoring System Based on Internet of Things, China, 2018
[4] Hong Jiang, Yong Shi, and Lei Qi, Design of Elevator Monitoring and Alarm System Based on WiMAX, Design of Elevator Monitoring and Alarm System Based on WiMAX
[5] Partha Pratim Ray, A survey of IoT cloud platforms, Future Computing and Informatics Journal 1, 2017
[6] Sharmad Pasha, Thingspeak Based Sensing and Monitoring System for IoT with Matlab Analysis, International Journal of New Technology and Research, 2016
[7] Arun.V and Dr.M.Poongothai, Implementation of IoT Based Intelligent Transportation System, Asian Journal of Applied Science and Technology (AJAST), 2018
[8] Shaileen Crawford Pokress and Jos´e Juan Dominguez Veiga, MIT App Inventor Enabling personal mobile computing, 2013