Low Cost, Easy-to-Use, IoT and Cloud-Based Real-Time Environment Monitoring System Using ESP8266 Microcontroller

NUBA SHITTAIN MITU [a, b], VASSIL T. VASSILEV [a], MYASAR TABANY [a]
Khm0225@my.londonmet.ac.uk, V.Vassilev@londonmet.ac.uk, M.Tabany@londonmet.ac.uk
[a] School of Computing and Digital Media, London Metropolitan University, *166-220 Holloway Road*, London, N7 8DB, UK.
[b] Bangladesh Atomic Energy Research Establishment,Ganakbari, Savar, Dhaka, 1349, BANGLADESH.

Abstract— This paper proposes a low-cost, Easy-to-use, IoT and cloud-based system solution for environmental data monitoring in real-time through the combination of Internet of Things (IoT) and Cloud Computing technology via Arduino IDE. This paper is present a low implementation cost Data Collection Circuit (DCC) using AT-Arduino commands-based microcontroller ESP8266 and custom IoT device for environment data collection from any physical circumstances. This paper has the scope to introduce the NoSQL, scalable, serverless, real-time database that is Google's firebase, to store the sensor data for real-time monitoring and management of the database. This paper is giving access to environmental information about a UK university library during busy exam weeks includes the major atmosphere parameters: temperature and humidity, which has a massive effect on the Earth's weather, human physical and mental health. The data can be monitored in the firebase database through terminal devices like Laptop, Smart Phone and Tablet which is endowed with the internet facility. This paper is also present a designed and developed Android App using the MIT App Inventor tool for synchronizing the firebase cloud data on it and monitoring these data in real-time from anywhere in the world in no time.

Keywords— IoT; Cloud Computing; ESP8266; Firebase; Cloud Integration; Android App.

1. Introduction

Now we are entering an era of the "Internet of Things" (abbreviated as IoT). This term has been defined by different authors in many different ways. Let us look at the most popular definition of IoT. The Internet of Things is simply an interaction between the physical and digital worlds and the 'digital world interacts with the physical world using a plethora of sensors and actuators [1]. The emerging concept of IoT plays a vital role to help in attaining this mission and policy decisions can be made based on real data 'and more importantly, their impact can be monitored almost in real-time. The increasing growth of mobile devices and development in the area of communication, cloud computing, embedded systems have made the IoT concept more relevant [2]. Internet of Things (IoT) [3] refers to objects that are connected through Internet, which means using modern information technology such as intelligent sensing, identification technology and wireless communication to realize the interconnection between objects. It is known as the third wave of information industry development after computer and the Internet. Cloud Computing [4] refers to an emerging development platform that addresses on-demand-shared resources and has the potential to handle several sources, for example, servers, networks, applications, etc. Cloud computing allows centralized usage and management of services in a flexible and optimized manner. The computing capabilities and storage facilities of the Cloud allow IoT to handle, store and compute such vast data. The most commonly utilized cloud for IoT is Amazon, Microsoft Azure, Google Cloud, Open IoT, etc.

Nowadays, most IoT researchers are dealing with this subject. One of them presenting real applications other presenting their research advancements on IoT inhabitants. Our IoT and cloud-based system target the temperature and humidity data collection using a designed and developed low-cost, Data Collection Circuit (DCC) of the system. In practice, we want to demonstrate that a relatively cheap, easy-to-use, time-optimised IoT and cloud-based real-time environmental data monitoring system solution. The Data Collection Circuit (DCC) is designed using wi-fi enabled microcontroller ESP8266, DHT11 sensor devices, etc. and must be well-configured within the Arduino IDE to communicate with the rest of the components of the system architecture.

Schwartz [5] demonstrates how to configure and upload code to the ESP8266 chip and for that Schwartz [5] using the Arduino IDE. This makes using the ESP8266 much easier, and it'll be using as a well-known interface and language to configure the chip with most of the existing Arduino Libraries [5].

A fundamental aspect of the Internet of Things is the integration with the Cloud infrastructure, which hosts interfaces and web-based applications that enable communication with sensors and external systems. Therefore, the Cloud computing infrastructure might provide data access and management features, to collect and manage data made available by smart objects. Firebase-driven approaches use protocols that are specifically designed for the communication between different devices, such as REST API over HTTP/HTTPS. IoT objects generate a significant amount of information, because of the extensive coverage of sensors and the continuous sensing of data. Gartner report1 estimated that 2.9 billion connected things were in use in the consumer sector in 2015 and will reach over 13 billion in 2020, while by 2019 there will be 780 million wearable devices and 2.2 billion smartphones. As a result, real-time data processing has become a new challenge, since one of the fundamental aspects of the IoT is the ability to respond to a trigger in real-time. In the IoT context, the best approach for data storage and retrieval is the use of a NoSQL or a Time Series (TS) database, which is optimized for managing arrays of numbers indexed by time and all the data is provided in the JSON format and can be retrieved and plotted [6]. The widespread of IoT devices make it possible to collect an enormous amount of data. The traditional SQL (Structured Query Language) database management systems are not suitable for storing this huge amount of data. For this task, distributed NoSQL database management systems are the most appropriate. However, for professionals developing such applications, the effective design and practical implementation of live sensor data collection from extreme condition site and cloud storage is a major challenge because of the huge amount of data [7,8]. Traditionally, real-time systems manage their data (e.g. chamber temperature, aircraft locations) in application dependent structures. As real-time systems evolve, their applications become more complex and require access to more data. It thus becomes necessary to manage the data in a systematic and organized fashion. Database management systems provide tools for such organization, so in recent years there has been interested in "merging" database and real-time technology. The resulting integrated system, which provides database operations with real-time constraints is generally called a real-time database system (RTDBS) [9, 10].

During our research, one of the main aims is to set up a distributed, NoSQL, scalable, serverless database system in the cloud to store and manage the sensor data. We select firebase as our data storage component under Google's original server, which is NoSQL and use REST API for database management. At a time, it will minimise the server maintenance cost and can save time in the context of making complex SQL query. Real-time monitoring of the collected sensor data is possible from firebase console using terminal devices like laptop, smartphone, etc. The firebase console data is more useful for data analytics, future prediction, real-time system behaviour monitoring and statistical analysis and more especially data will remain safe without any malicious infection. People must have to concern about their vast sensor IoT data security, which is produced every day from smart home, smart health system, etc. Getting back those sensitive data from the cloud to own hand might jeopardize the insecurity feeling of the smart IoT application system owner. Firebase is a good example of the IaaS (Infrastructure as a Service) service model of cloud computing.

Firebase is a framework that is useful for building portable and web applications for businesses that require real-time database which implies when one user updates a record in the database, the update should be conveyed to every single user instantly. A real-time Database is a cloud-hosted database. Data is stored as JSON and synchronized continuously to each associated client. When any cross-platform application is developed with iOS, Android, and JavaScript SDKs, the greater part of the user's demand is based on one Real-time database instance and this instance gets updated with each new data. This feature allows developers to skip the step of developing a database, and Firebase handles most of the backend for the applications [11].

With the advent of new mobile technologies, the mobile application industry is advancing rapidly. Consisting of several operating systems like Symbian

OS, iOS, blackberry, etc., Android OS is recognized as the most widely used, popular and user-friendly mobile platform. This open-source Linux kernel-based operating system offers high flexibility due to its customization properties making it a dominant mobile operating system. Android applications are programmed in java language. Google Android SDK delivers a special software stack that provides developers with an easy platform to develop android applications. Moreover, developers can make use of existing java IDEs which provides flexibility to the developers. Java libraries are predominant in the process of third-party application development [12].

This paper also targets the enormous flexibility for cloud data accessing and monitored in real-time from anywhere. In this point of view, we designed and developed an Android App using MIT App Inventor for seamless cloud data accessing in no time. The MIT build App of the system can monitor the live firebase data in real-time which fulfilled the goal set of the authors. 'Firebase Integration' is a term which must be successfully done within the firebase and Android Application of the system to get synchronized with the firebase database for real-time data monitoring on the App and it is also presented and discussed in the proposed system.

In forthcomings, the paper includes section 2 which presents a detailed system architecture description and considers all the software and hardware components of the system. Section 3 and 4 contains operating results and discussions, economic advantages of the system respectively. Finally, section 5 concludes the research, future works of this IoT and cloud-based real-time environment monitoring system.

2. System Architecture

System architecture presented in Figure 1, shows three major components of the proposed system. The initial part is represented a Data Collection Circuit (DCC) which is designed and developed using wi-fi enabled microcontroller ESP8266 12-E version and custom IoT device-DHT11 for collecting live environmental data of the system. The DCC can collect the live data via Arduino IDE from any physical circumstances after a successful configuration within the IDE. On top of that, the Arduino Design and Implementation on IDE must be successful for the seamless data collection and to communicate with the data storage component of the proposed system and is discussed well in section 2.3. The data collection made using DCC of the system print on IDE serial monitor before push to the cloud according to the Arduino design too. The design and development of the Data Collection Circuit (DCC) of the system clarifying it as a low-cost, easy-to-use and time-optimised which is discussed and justified well in section 2.1 and 3.

The data storage component of this IoT and Cloud-based system is a NoSQL, serverless, real-time database under the Google's cloud which is Firebase to store the data updates and monitoring the data in real-time. Firebase also has a built-in database management facility that can perform the database actions: Read, Write, Delete, retrieve data through REST API used. The data transferred and or pushed to the cloud can be used for more complex analysis that will predict and prioritise more precisely the environmental conditions, optimise the time needed to reach the system implemented locations and provide timely information about the environmental conditions. Through the accessible devices admin or user of the system can monitor the data from firebase console in real-time over http request made. The firebase setup must be successful using the google credentials to get access of its real-time database is discussed in section 2.2. The data loading delay to the firebase cloud according to the Arduino design and implementation of the system is discussed in section 2.3 and visualized in section 3. The data storage component selection of the proposed system introduced and clarifying as a low-cost, easy-to-use and time-optimised and is discussed and justified well in section 2.2 and 3.

The Android Application (App) of the system is suitable to any android compatible devices and is designed and developed using MIT App Inventor. The App of the system can monitor the air data of the implemented area in real-time from anywhere in no time. Towards a very handy and flexible real-time monitoring the 'Firebase integration' within the developed App must be successful to get synchronized with the firebase database for the real updates on the App which is discussed in section 2.4. The Android Application development for real-time monitoring of the system addressed and clarifying the low-cost, easy-to-use, time-optimised parameters well through the discussions and justifications in the section 2.4 and 3. The App can be installed on user's android devices if the authors willingly share it to any media like Google play, etc. also discussed in section 2.4.

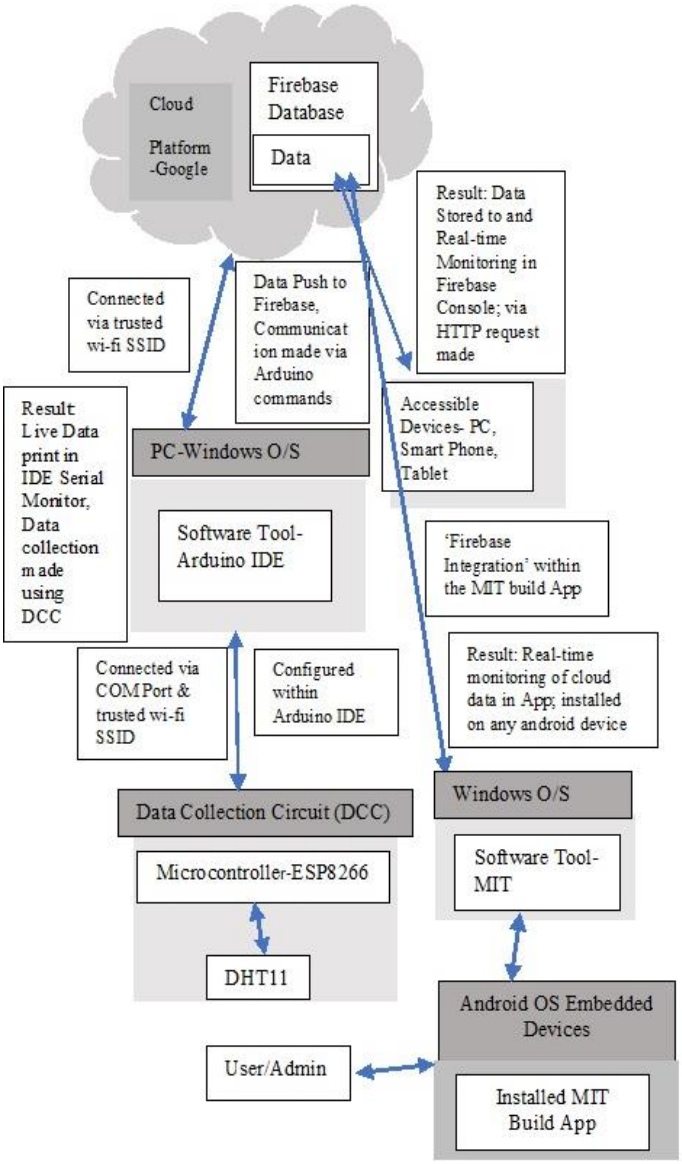

Fig. 1 System Architecture: includes the software and hardware components of the system for data collection, data storage and monitoring from firebase console and App.

2.1 Data Collection Circuit (DCC) Design, Development and Configuration with Arduino IDE

The Data Collection Circuit (DCC) component of the system architecture is designed and developed using wi-fi enabled ESP8266 microcontroller, DHT11 humidity and temperature sensor and fewer jumper wires, is low cost, low power consumptions and easy-to-use. It saved the programming space, use of many wires and chipsets at a time.

DHT11: The DHT11 Temperature & Humidity sensor is an economical peripheral manufactured by D-Robotics UK (www.droboticsonline.com). DHT11 features a temperature & humidity sensor complex with a calibrated digital signal output which includes a resistive-type humidity and an NTC temperature measurement component and connects to a high-performance 8-bit microcontroller, offering excellent quality, fast response, anti-interference ability and cost-effectiveness [13] These sensors are small, economical, user-friendly and have low energy consumption. They have four pins, one of which is enabled for data transmission is up to 100 meters [14]. It is capable of measuring relative humidity between 20 and 90% RH within the operating temperature range of 0 to 50°C with an accuracy of ±5% RH. Temperature is also measured in the range of 0 to 50°C with an accuracy of ±2°C. Both values are returned with an 8-bit resolution [15].

ESP8266: The NodeMCU ESP8266 is a microcontroller with integrated Wi-Fi, which means that there is no need for an additional Wi-Fi chipset. The design of the SoC allows communication through the GPIOs by connecting to the Internet and transmitting data over the Internet. This is a perfect connection for the Internet of Things (IoT). It has a price of about $2.50, with a physical size of 49 ×24.5 ×13 mm, consumes 0.00026–0.56 W of power, current consumptions during flashing 10uA-170mA to 800mA and the operating temperature is 40ºC to +125 ºC. It allows maximum 5 TCP connections and it's work through the several Network Protocol's IPv4, TCP / UDP / FTP / HTTP. This is the best hardware around in terms of cost and this chip is the future of the IoT [14][16]. ESP8266 on-board processing and storage capabilities allow it to be integrated with the sensors and other application-specific devices through its GPIOs with minimal development up-front and minimal loading during runtime. With its high degree of on-chip integration, which includes the antenna switch balloon, power management converters, it requires minimal external circuitry, and the entire solution, including the front-end module, is designed to occupy minimal PCB area. ESP8266 offers a complete and self-contained Wi-Fi networking solution, allowing it to either host the application or to offload all Wi-Fi networking functions from another application processor [17]. The ESP8266 12-E microcontroller is used for data collection in this IoT based system is the latest version and has a Cp2102 driver installed on it. market and is used for establishing a wireless network connection for a microcontroller or ESP8266

12-E: The ESP-12E or NodeMCU version 12E is a miniature Wi-Fi module present in the processor. The core of ESP-12E is ESP8266EX, which is a high integration wireless SoC (System on Chip). It is a low-cost solution for developing IoT applications [18]. The ESP8266 12E module has 30 pins including 17 GPIO pins. The GPIO port connections of the ESP8266 module are summarised in **Table 1** found in [18]; which is partially edited by the authors.

Table 1: Summary of the GPIO port connections of the ESP8266 module

Microcontroller-ESP8266 Pin	GPIO	Purpose
3v3		+3.3V Power input
GND		Ground pin
Vin		Power (+5
4	16	General Purpose Input
20	5	General Purpose I/O
19	4	General Purpose I/O
18	0	General Purpose I/O
17	2	General Purpose I/O
5	14	General Purpose I/O
6	12	General Purpose I/O
7	13	General Purpose I/O
16	15	General Purpose I/O
14	6	Clock Pin of SPI
10	7	MISO Pin of SPI
9	11	Chip selection Pin of

Source: **[Datasheet-ESP8266]** https://components101.com/wireless/esp12e-pinout-datasheet;https://www.make-it.ca/nodemcu-arduino/nodemcu-details-specifications/

Figure 2 present the IoT diagram and design snap of the Data Collection Circuit (DCC) component of the system.

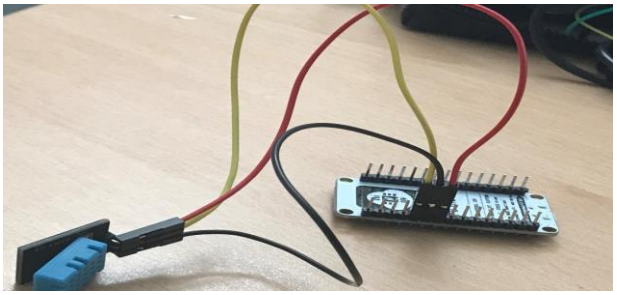

Fig. 2 Data Collection Circuit (DCC) component of the system

The Data Collection Circuit (DCC) can collect the live temperature and humidity data from any physical circumstances seamlessly and following is the specification.

Data Collection Circuit (DCC) Specification: The DCC can be power up through the PC and certainly there is no access power plug is needed. The DCC will consume a maximum of +10 volt to operate the circuit but +5volt is recommended [17] because the NodeMCU device of this DCC is only required a voltage +5 to +10v to power up. There is no power converter, and or battery, are required to operate this circuit. This circuit is straightforward to use and can power up through the PC using a USB cable. One thing noted here, the PC must be connected to any trusted wi-fi network and the DCC must belong to the same wi-fi network.

DCC Connectivity- 1. DHT11 GND (Ground) pin to NodeMCU ESP8266 V12E-1 GND pin. 2. DHT11 power (+ve) pin to ESP8266's 3V3 logic (power) pin. 3. DHT11 data pin (Pin No.-2) to ESP8266's data pin D4. It can be get connected to any data pin D0-D8 of ESP8266.

DCC Configuration with Arduino IDE: The NodeMCU ESP8266 offers a variety of development environments, including compatibility with the Arduino IDE (Integrated Development Environment). In our system context, the initial and one of the vital tasks is the complete and successful configuration of the DCC within the software tool- 'Arduino IDE' for seamless sensor data collection.

Arduino IDE: The Arduino is a microcontroller development board series - Uno, Mega, Nano, Mini etc. are the examples. Arduino IDE (Integrated Development Environment) is open-source software and that enables better and assisted code editing, compiling, and debugging. It runs on Windows, Mac OS X, and Linux. The environment is written in Java and based on processing and other open-source software. Arduino IDE has inbuilt functions and commands that though work on the Java platform, is customised to run on the Arduino board. Thus, Arduino IDE serves for code editing, its compilation, debugging and then burning the code into the Arduino board [19]. In [20], the Arduino software is found with necessary installation guidelines and steps which is runs on the Java Platform. Our IoT based system used the Arduino software version 1.8.9. In [21], ESP8266 wifi module board manager installation steps are found to get it configured within the Arduino IDE. In [22]-[26], the required Arduino libraries for the communications are found which are

installed in Arduino IDE and updated with its every recent release, whereas some of them talk properly with its older version.

ESP8266 configuration testing within the Arduino IDE: A small circuit is designed and developed using 1 (one) LED and 1 (one) Resistor and some jumper wires with the ESP8266 and uploaded to the built-in Arduino command 'BLINK' in IDE to test the successful configuration of the esp8266 module within the Arduino IDE. After a successful configuration, the ESP8266 module will show its connection status on Arduino IDE via COM port.

2.2 Cloud Selection and Setup for the Data Storage Component of the System

Sensor data storage and management in a preferred cloud platform is an inevitable task for most IoT based real-time monitoring system. In our system, data collected from Data Collection Circuit are stored in Firebase, a NoSQL Backend-as-a-Service(BaaS)that handles a wide variety of data types, including JSON files. Some of the exciting features of firebase such as NoSQL data management, real-time database, cloud integration within the several applications, etc., help us to fix the right cloud selection. An example of Infrastructure as a Service model is provided by Firebase, which uses a REST API approach for the communication between IoT devices and the Cloud.

Firebase: It is a Real-Time and cloud-hosted database developed by Google. Data in Firebase is stored in the JSON (JavaScript Object Notation) format and is updated in real-time as soon as one of the clients connected to that database changes the data [27]. It provides one of the most stable and fastest real-time databases and offers so many useful features like NoSQL, real-time data synchronization to every connected client, firebase analytics, hosting, storage and many more.

The real-time databases like firebase are used to interface some controller which can be connected to the internet and can be able to exchange data with cloud server. The data structure stored for the DHT11 sensor of this IoT based system contains live temperature and humidity values.

Firebase Setup: To own the project space under the firebase in google cloud requires some necessary and major steps like set up a project, writing real-time database security rules for data security, database secret key and database host address generation, etc., are found in [28]. The auto-generated firebase database 'secret key' and 'host address' for this system will have several major uses like, in 'Arduino design and implementation' -to communicate with the cloud storage and in 'Cloud Integration'- to synchronise the firebase database with a designed and developed Android APP for user communication. The database 'secret key' and 'host address' will be defined as 'FIREBASE_HOST' and 'FIREBASE_AUTH' respectively where needed. The following figure 5 shows the data stored under the firebase project of the google cloud. The real-time database of our IoT and the cloud-based system is available at [29] through any client HTTPS request made. The database data can be retrieved as a JSON document will be available at [30]. Appending. JSON after the URL address of this IOT and cloud-based system in every HTTP request made must be needed.

Firebase is well featured to manage the real-time database using its RESTful API (Application Programming Interface). The state-of-the-art topic 'Real-time data monitoring (RTDM)' is playing through the NoSQL data storage component selection for our IoT and cloud-based system.

Firebase Database REST API: Firebase is used the REST API to manage the database and allow the database actions include: PUT, GET, POST, PATCH, DELETE etc., are found in [31]. In our IoT based system, the firebase database is managed by the admin following the above-mentioned REST API actions if required. Firebase only responds to encrypted traffic so that all data remains safe and HTTPS is required to respond.

2.3 Arduino Design and Implementattion

The Arduino design and implementation are considered as the paramount task for communicating with the Data Collection Circuit (DCC) and data storage component of the system. In figure 3, the Arduino Commands are shown with necessary Arduino libraries via a trusted wi-fi SSID.

```
#include <DHT.h>#include <ArduinoJson.h>
#include <ESP8266WiFi.h> #include <FirebaseArduino.h> #include <Adafruit_Sensor.h> #include <ESP8266HTTPClient.h>
#defineFIREBASE_HOST ""#define FIREBASE_AUTH ""
#define WIFI_SSID " " ; #define WIFI_PASSWORD " "
#define DHTPIN D4;  #define DHTTYPE DHT11;  DHT dht("", "" );
```

```
void setup() Serial.begin(115200); delay(1000);
WiFi.begin(WIFI_SSID, WIFI_PASSWORD);

while (WiFi.status() != WL_CONNECTED) {
Serial.print("."); delay(500); }

Firebase.Begin(FIREBASE_HOST, FIREBASE_AUTH);
dht.begin(); //Start reading dht sensor } void loop() {
…… return; } //string conversion

String fireHumid = String(h) + String("%");
Serial.print("Humidity: "); Serial.print(h);

String fireTemp = String(t) + String("°C");

Serial.print("% Temperature: "); Serial.print(t);
delay(4000);//after 4 seconds delay values will send to
cloud

Firebase.pushString(" ", variable) Firebase.pushString("
", variable); }
```

Fig. 3 Arduino Commands for enviromental data collection and store to the firebase database

The Arduino commands in Figure 3, needs to be uploaded on IDE after a successful connection of the Data Collection Circuit (DCC) via COM port of the PC.

After 100% uploading of Arduino commands on IDE, live temperature and humidity data will be collected and at the same time displays in the IDE Serial Monitor. The collected data then can be pushed to the cloud after a predefined delay of approx. 4 seconds.

2.4 Android Application (APP) Development using MIT App Inventor to get access and monitoring of Firebase real-time data through 'Cloud Integration'

Google cloud is collaborating with MIT to allows IoT builders to integrate their IoT application system with popular platforms like Android and IOS through its Application Program Interface (API) and client libraries and also allows Android Applications to get real-time updates from their database through cloud integration. This paper also targets the real-time data monitoring in a very flexible way, and it is implemented by developed an android app using the MIT App Inventor and installed that app on an android device.

MIT: MIT is the most advanced cloud-based app building tool maintained by the Massachusetts Institute of Technology (MIT). The MIT App Inventor development environment is supported for macOS, GNU/Linux, and Windows operating systems, and several popular Android phone models and the applications created with App Inventor can be installed on any Android devices.

MIT Platform Setup: To set up the MIT Platform for this system, we choose MIT App Inventor 2 found in [32] and it includes a list of considerable works such create a project on MIT using google credentials, firebase database integration with MIT, design and develop the android App and finally the installation of that app on any android device. The created project on MIT for our system is available at [33].

Cloud Integration: Firebase Integration with an MIT build Application (APP) is treated as one of the major tasks of our IoT and a cloud-based system for making the App accessible and to getting access to the cloud data from anywhere. The following figure 4 shows the firebase cloud integration with MIT.

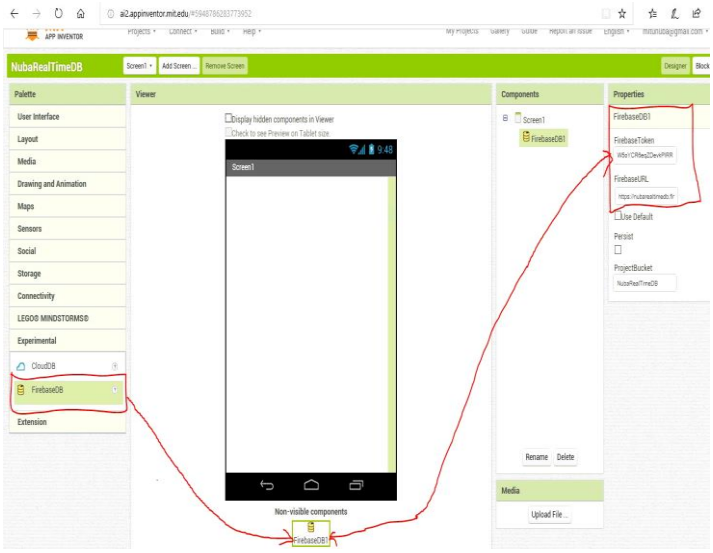

Fig. 4 Firebase Cloud Integration on MIT.

'Drag and Drop' the Firebase Database and property value set on the MIT workspace are the considerable works towards firebase cloud integration. Drag the Firebase Database from the 'Experimental' menu and drop it to the viewer window and the properties 'Firebase Token' and 'Firebase URL' must be filled up with values which are same as the 'Firebase Secret Key' and 'Firebase URL' of the firebase database and are indicated in above figure 4 using the red mark. After successful cloud integration, the MIT workspace is ready to design and develop the App.

App Design: The App design of this system includes front end and back-end design using built-in MIT

App Inventor's user interfaces and functions respectively. The front-end design is the screen view of the app for our system. The front-end design of the app occupied some of the built-in user interfaces components such as button, label, horizontal and vertical arrangement, etc. And it required various property settings for each component. The back-end design of the App for our system includes logic development using built-in functions such as control, logic, math, text, etc. The logic design is a major task to talk accurately with the designed front-end of the app. However, these logics are mainly responsible to read the firebase data and monitor these data in real-time on the developed app. The following figures 5 and 6 shows the Front-end design view and Back-end logic design of the APP on MIT.

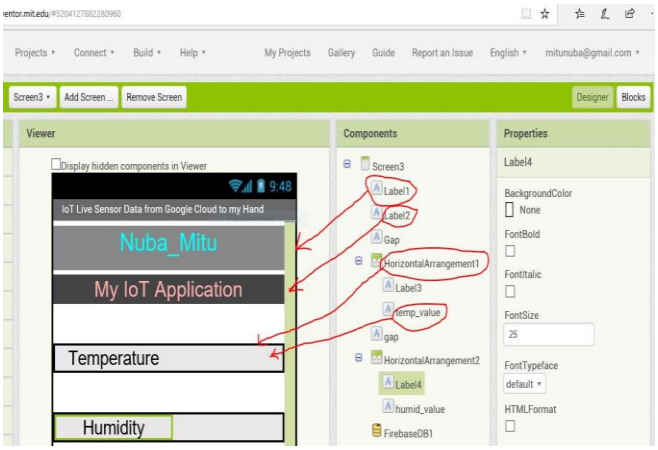

Fig. 5 Front-end design view of the APP on MIT.

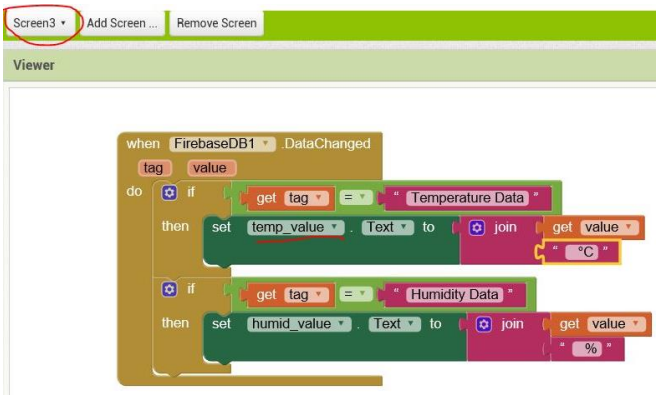

Fig. 6 Back-end logic design of the APP on MIT.

App Development: App development is surely important to make it usable on any android device and it includes a list of necessary steps such build an executable file, app installation to the android devices, export and import the MIT Project file, etc. To build an APK or executable (Android Package Kit) file is basically to make the app usable on any android devices and it includes some steps using MIT. The following figure 7 shows the App development progress on MIT.

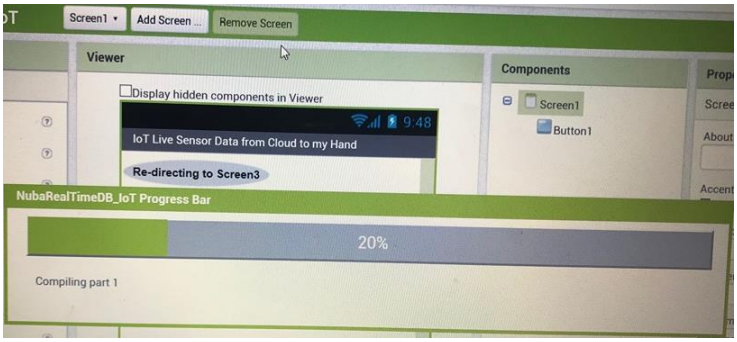

Fig. 7 APP building progress on MIT.

Supported File Types of MIT App Inventor and Android OS:

AIA File: Source code file created and used by App Inventor; web program used by beginning developers to create applications for Android devices; contains the source code blocks of an application project in development. App Inventor is a blocks-based programming tool that uses the source code in the AIA file to create applications.

APK File: An Android Package Kit (APK for short) is the package file format used by the Android operating system for the distribution and installation of mobile apps. Just like Windows (PC) systems use a .exe file for installing software, Android does the same [34].

Export and Import the MIT Project file: These are the important steps to get this instance save to any desired location of the PC for further use. After saving this MIT project it'll get an extension *.aia. This is not only useful for admin of this App, but anyone could use this designed App if it is shared to any media like Google Play, etc. In other words, it can be usable by anyone just clicking on download from shared places and import it to their own desired location using MIT platform. The following figure 8 shows the export and import steps of the MIT Project file.

Sharing and Packaging the App: The source code (.aia) files are not executable Android programs -- those are .apk files. The source code (.aia) files is also not Java SDK code -- it can only be loaded into App Inventor.

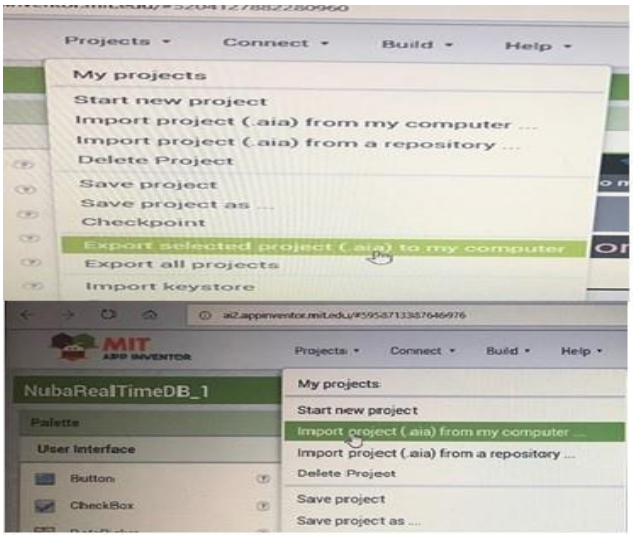

Fig. 8 Export and Import the MIT Project file.

App Installation: After building the executable file it must need the implementation on any android devices. And it requires a list of primary setups in any android operating systems such as security setup, downloading the APK file and installation of that file, etc. After all the steps successfully done, the App will be ready to monitor the cloud data in real-time from anywhere in no time. The following figure 9 to 11 shows the APP Installation progress and installed ready-to-use App on an Android Device of the system to monitor the firebase database data in real-time.

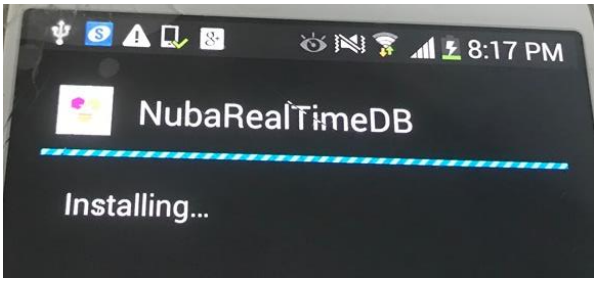

Fig. 9 APP Installation progress on an Android Device.

Fig. 10 Installed App 'NubaRealTimeDB' of the system on an Android Device.

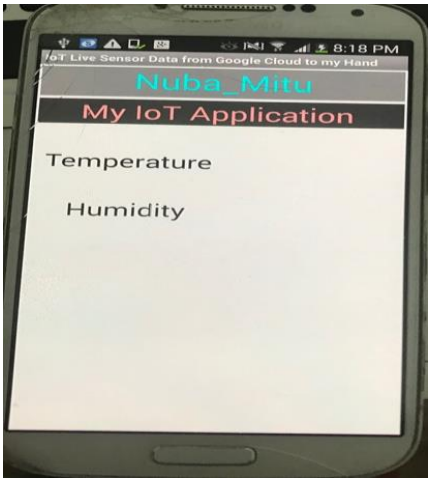

Fig. 11 Ready-to-Use Android Application for cloud data monitoring in real-time.

App Installation Pre-requisites on Android Devices: The installation instructions are very simple, but some prerequisites have to be attended before. Firstly, the android devices have the built-in wi-fi facilities. Operating System need to be upgraded to at least 1.4 or more. Finally, the App needs to be minimum 4-megabyte of space on android device to install.

3. Results and Discussions

The presented system architecture of this IoT and cloud-based system were tested, and they are well-suited for single-point measurement. Sensors and sensor nodes can be added to this system to have multiple measurement points. The system architecture of the system is horizontally scalable. All the software components of the system architecture are developed in a unified way considering the various IoT objects and microcontroller, tools, platforms, and an android application (APP) development. The system performance is adequate, and it is capable to perform its desired roles.

Usually, most of the university library remains full during each semester's early exam preparation hours. The results have the access to the major environmental information of a UK university central library facility deploying the system during the very busy exam preparation hours of the autumn semester in 2019.

The following figures 12 to 17 shows the results of the system are, data collection using developed Data Collection Circuit (DCC), live sensor data store and real-time monitoring in firebase console, environmental information monitoring graphs and

real-time cloud data monitoring on a developed Android Application from anywhere.

The Data Collection Circuit (DCC) of the system can constantly collect temperature and humidity data and print it to the IDE serial monitor according to the Arduino design, once it is plugged into the PC via Arduino IDE. The DCC can certainly collect massive amounts of data and the cloud storage of the system can store and manage the enormous amount of data too. We strongly believe that our IoT and cloud-based system addressed the emerging 'Big Data' concept from a data collection point of view.

The authors demand the proposed system can satisfy a list of parameters on its each component are Low-cost, Easy-to-use, Time-optimised in design, development, implementation point of view. The developed Data Collection Circuit (DCC) of the proposed system architecture are low-cost, Easy-to-use, time-optimised which are justified and discussed well in section 2.1. The initial and total implementation cost of this IoT and Cloud-based system will be less than 20 GBP (Great British Pound) and which is the development cost of the Data Collection Circuit (DCC). The system implementation will cost nothing for the data storage component setup and no further cost for the real-time data monitoring as well. The platform setup for the android application development of the system for real-time monitoring also free-of-cost. No doubt, the implementation of the proposed system for collecting live sensor data from any physical circumstances, store data and monitoring in real-time will be completely low-cost. As this system is horizontally scalable so the system implementation cost might be rise after adding nodes for collecting various types of data. As this proposed system introduces 'big data' and the further storage implementation might add costs in the context to the data storage selection, migration to the payment plan according to the storage required, etc.

The design and development of the Data Collection Circuit (DCC) of the system possess simple circuitry, less wiring and user can handle it easily. At a time, it required less time to develop the circuit and simple Arduino commands to talk with the Data Collection Circuit (DCC) towards the data collection of the system can clarify the time-optimised parameters. The data collection using the DCC of the system from the test bed was so speedy. The authors were realizing the speediness of the live sensor data collection and calculating the data updates into graphs results. The authors also have concern to put time stamp against the data updates in future research also.

NoSQL database selection and setup as the data storage component of the system for storing and monitoring of the live environmental data also clarify the low-cost, easy-to-use, and time-optimisation parameters which is justified and discussed well in section 2.2. Traditional database required complex query to manage the database which involves much time needed and in most cases the handling of the database is not so easy. On top of that, database setup and maintenance also costly in most cases. In our proposed system the database setup and maintenance costs us nothing.

Low-cost, easy-to-use, and time-optimisation parameters are also playing through our proposed system in the context of the flexible data monitoring in real-time from anywhere in no time using the developed Android Application (APP) of the system. We select the MIT App Inventor 2 for design and develop an APP which costs us nothing for create and maintain the App. There are no additional costs to setup the MIT platform as well. No doubt, the platform selection for design and develop the App of the system meet up the effective use of time. The cloud data monitoring on the App from anywhere and without the physical presence of that experimented and or implemented area is successfully present in our case. The parameters are addressed well for cloud data monitoring in real-time using a developed app of the proposed system in section 2.4. In figure 12, shows the collected live humidity and temperature data print on the Arduino IDE serial monitor.

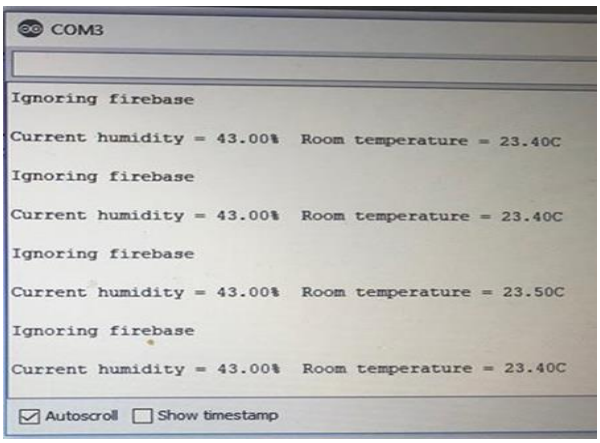

Fig. 12 Sensor Data printed on Arduino IDE Serial Monitor.

If the amount of data collected exceeds datastore capacity, storage increase, and migration is possible in firebase according to the payment plan set up. Figure 13 shows the live humidity and temperature

data updates under the firebase project which got a rapid change from its initial count after few gusts.

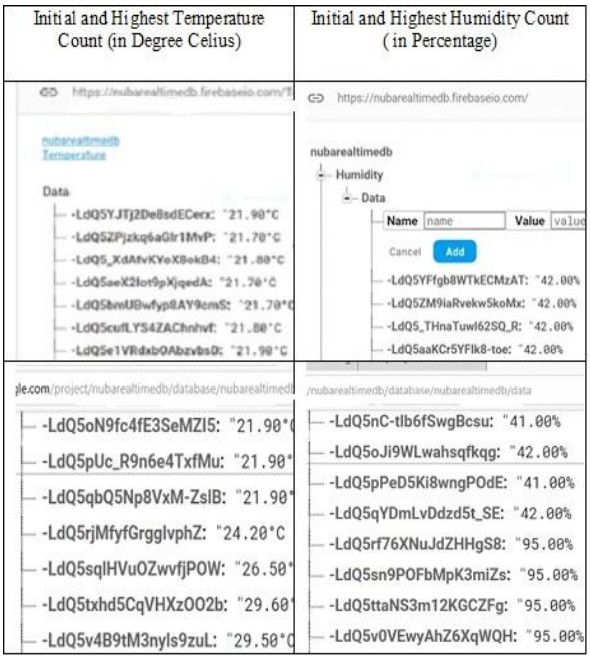

Fig. 13 Sensor Data Updates under the firebase project in google cloud

It is very important that the temperature and humidity of the environment monitored for a high-quality environment and has massive effects on human physical and mental health [35]. Following figures, 14 to 16 are shown the environmental information monitoring graphs according to the sensor data updates under the firebase console and are drawn using MS-XL Sheet. Firebase is not visualizing the data like other popular IoT platform Things speak, etc. instead it can show the real-time data updates in the console and allows the database management actions.

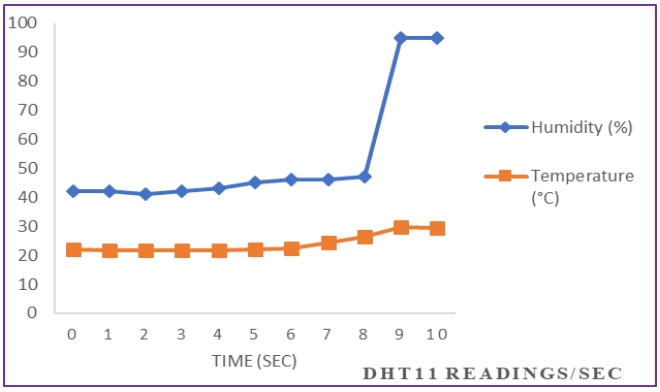

Fig. 14 Air humidity and temperature counts per second using Data Collection Circuit (DCC) on IDE Serial Monitor.

In figure 14 shows the humidity and temperature monitoring only for 10 seconds and the air humidity was exponentially increased after few gusts outside whereas the temperature was remained mostly flat. To start with the per second air humidity and temperature count and monitoring at the library helps us to collect a countable amount of environmental data and which are the primary and inevitable tasks of our IoT and cloud-based system.

In every 250 milliseconds, one temperature or humidity data can be read by the DHT11 device, which is found in the DHT tester example of Arduino 1.8.9 [20].

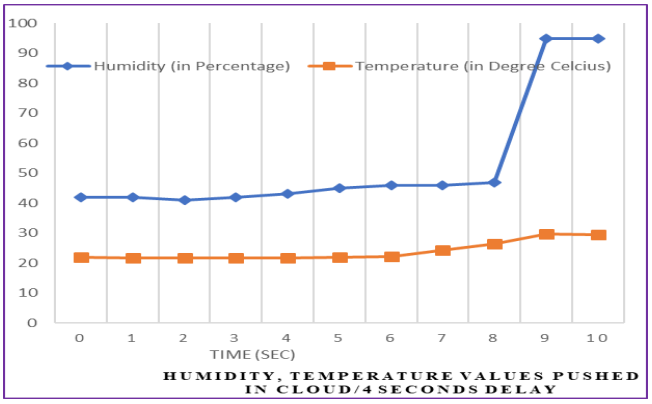

Fig. 15 Air humidity and temperature data loading delay to the firebase.

In figure 15 shows, the sensor data loading delay in the firebase database according to the Arduino design and implementation which is discussed in section 2.3. To be more exact, collected sensor data using a data collection circuit (DCC) of the system will take only 4 seconds to push and save in the firebase cloud.

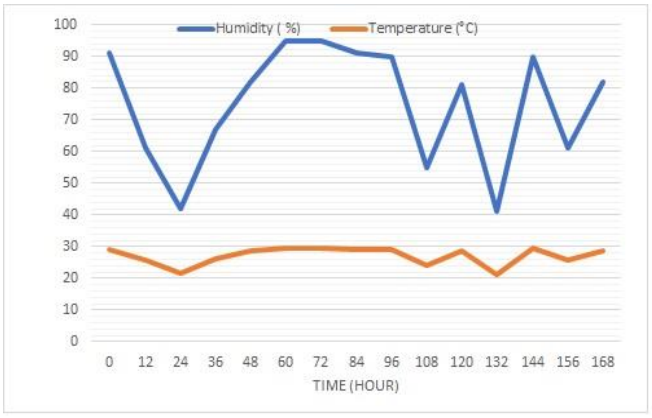

Fig. 16 Hourly Variation of the environmental data during a week.

In figure 16 shows the air humidity is mostly fluctuated in 40% - 90% and got the highest 95% in the middle of the week, whereas air temperature is mostly flat within 20°C - 29°C in every 24 hours monitoring count during a very busy autumn semester exam preparation week.

Humidity is the amount of water vapour in the air. It has a large effect on human mental and physical health. Humidity must be in 40-50%. Our experimented library service is equipped with the central humidifier to provide humidity control throughout the entire library but sometimes the unpredicted autumn gusts can increase the air humidity which is monitored in real-time by deploying our system. The authors are not concerned to establish these little findings to their research work instead put efforts to prove the low-cost, easy-to-use, time optimized parameters to the embedded IoT system.

Finally, the authors can present the targeted flexible data monitoring in real-time through the designed and developed android app of the system. We experiment with this application on an Android compatible smartphone with an updated version of the operating system, and it must be wi-fi enabled. This App will take at least 4-megabyte of free storage to install. Following figure 17 shows the environmental data monitoring on the app from anywhere in no time.

Fig. 17 Cloud data monitoring in real-time

In conclusion of this section, the authors are strongly commented that the proposed IoT and Cloud-based system is a Low-cost, Easy-to-use, Time-optimised solution for real-time environment monitoring towards the organizations with the similar needs.

4. Economic Advantages

Economic Advantages of this IoT and cloud-based system includes satisfying a list of criteria. Firstly, it mitigates the implementation costs compared with other IoT systems which are designed with popular microcomputer like the raspberry pi. The initial and total implementation cost of this system will be less than 20 GBP (Great British Pound). Secondly, this system must be useful for collecting physical data from any extreme condition site where a human cannot reach easily, or it is difficult to visit there frequently. Thirdly, real-time data monitoring from anywhere in the world without any physical presence at that place in no time using a developed android APP proves the system is flexible and time optimized. The traditional monitoring system has many more shortcomings, such as hard-wiring, high transmission bit error rate, high costs and small coverage compare with this kind of IoT and cloud-based system. The cloud data of our system will be more useful for further data analytics, monitoring of the real-time system behaviour, statistical analysis and processing, and interpretation for the future use case.

The authors put concentration on these issues basically for future use cases of the system. No doubt, this system can introduce the 'Big Data' concept. As this proposed system architecture is 'horizontally scalable' more sensor nodes can be added for various types of data collection and will resulting huge IoT data. Data and 'Big Data' both need to be analysed to find trends and draw conclusions about the information contained of the data. However, now a days, 'Big Data Analytics' is a trend for IoT and Smart IoT systems with the various technologies fog, cloud computing, internet of things (IoT), etc to make the SMART IoT data more usable.

There also arise a future use case of the proposed system in the context of the statistical analysis and processing of the data. It might be performed to collecting and analysing the data Big data to identify patterns and trends of the massive data and which might introduce the integration of the AI technologies with the system.

Any real-time system behaves in a predictable way. Now a days, IoT developers are intended to do more research based on Traffic Control Systems or Process control-based applications in industries such Driver behaviour monitoring and in health care for patient's abnormal behaviour monitoring, workers behaviour analysis on a large construction site, etc. with the essence of various technologies. Behaviour is a term

which might be suited with the environment and or weather behaviour as well. In our proposed real-time system, two major atmospheric parameters and or weather component is collected and monitored in real-time which must address the weather behaviour monitoring. The weather data has a major acceptance to predict the weather condition. In other way around, the proposed system can be expanded into a SMART weather system and will introduce the real-time monitoring of the huge SMART weather data and its behaviour. It might help to predict the weather condition and take an urgent decision according to the weather abnormalities found.

However, our IoT and cloud-based system also associated with fewer limitations. This system can monitor the data in real-time using a designed and developed android compatible APP of the system, and it might introduce security vulnerabilities. Android is more prone to security vulnerabilities that most users do not consider. This system is not gone through rigorous security check which is the concern of the author's future study.

5. Conclusions and Future Scopes

In this paper, a low-cost, easy-to-use, and time-optimised IoT and cloud-based system for the real-time environment monitoring has been present and experimented and that could be used by organizations who are searching the solutions with these similar needs. A simple, low-cost Data Collection Circuit (DCC) has been presented for major environmental data collection and which minimizes the initial and total implementation cost of the system compare than other IoT system which are using popular microcontroller like Raspberry pi, etc. The Data Collection Circuit (DCC) has been tested and well-configured within the Arduino IDE and according to the Arduino design it was collected huge amounts of environmental data from the experimented field, and it is believed to collect huge amounts of IoT data from any physical circumstances too. Google's firebase cloud selection for the data storage component of the system ensures 'flexible data management', 'the effective use of time', and 'no maintenance cost' according of its REST API, NO SQL and serverless features respectively compare than other IoT systems those who are configured and managed the traditional database for the similar needs. Based on IoT principles, a complete communication of the Data Collection Circuit (DCC) with the Google's firebase cloud platform is established.

The environmental information data of a UK university library during a busy semester exam preparation week has been experimented with, huge live data stored and monitored in real-time in the firebase console, which is believed to has a massive effect on human physical and mental well-being. The collected huge environmental data during a week has been calculated and visualized in a graph format which is discussed and justified well in section 3.

To chasing enormous flexibility of cloud data monitoring, an android application (APP) is designed and developed using the MIT App Inventor tool and has been successfully installed and or experimented on an android device which is capable to monitor the data from anywhere in no time. Firebase integration is a new launched facility of the Google's cloud which allows the synchronization of its database with any MIT build application and has taken place successfully in the paper. And it results the live updates monitoring in real-time using the App which also declares the tactical platforms and tools selection towards flexibility of any proposed system. The live data updates have been monitored in real-time using the developed APP of the system during a week, which is also presented in the section 3.

Every research can contribute towards different findings which sometimes are not the goal set. In this context, our research has a contribution on a little finding and is discussed in section 3.

Our future works intents to implement this system in extreme condition site to measure the major atmosphere parameters where human cannot reach easily or visit there frequently. We also intend to design and develop the low-cost SMART IoT system on Environment and healthcare and will put concentration to perform the Big Data Analytics on huge amounts of IoT data. This system will be very useful and helpful that if other organizations hope to use some solutions with low cost, easy-to-use, time-optimised and risks for similar needs.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests and or personal relationships that would have appeared to influence the work reported in this paper.

Acknowledgement

This work is inspired and accredited by the School of Computing and Digital Media of London Metropolitan University, UK. Kh. Nuba Shittain Mitu would like to thank Cyber Security Research Centre, School of Computing and Digital Media, London Metropolitan University, UK for their unconditional support.

References

[1] Vermesan, O., Friess, P., *Internet of things: global technological and societal trends*, Aalnorg: River Publishers, 2011.

[2] Zanella, A., Bui, N., Castellani, A., Vangelista, L. and Zorzi, M., Internet of Things for Smart Cities, *IEEE Internet of Things Journal*, 1(1), 2014, pp. 22–32, doi: 10.1109/JIOT.2014.2306328

[3] C. Perera, C. H. Liu and S. Jayawardena, The Emerging Internet of Things Marketplace From an Industrial Perspective: A Survey, *IEEE Transactions on Emerging Topics in Computing,* vol. 3, no. 4, Dec. 2015, pp. 585-598, doi: 10.1109/TETC.2015.2390034.

[4] Mobasshir Mahbub, M. Mofazzal Hossain, Md. Shamrat Apu Gazi, IoT-Cognizant cloud-assisted energy efficient embedded system for indoor intelligent lighting, air quality monitoring, and ventilation, *Elsevier Internet of Things Journal*, Volume 11, 2020, 100266, ISSN 2542-6605, https://doi.org/10.1016/j.iot.2020.100266, Available at-(https://www.sciencedirect.com/science/article/pii/S2542660520301001).

[5] Schwartz, M., *Internet of things with ESP8266: build amazing internet of things projects using the ESP8266 Wi-Fi chip,* ISBN 978-1-78646-802-4, Birmingham Mumbai: Packt, 2016.

[6] International Forum on Research and Technologies for Society and Industry and Engineers, I. of E. and E., *2016 IEEE 2nd International Forum on Research and Technologies for Society and Industry Leveraging a better tomorrow (RTSI): 7-9 Sept, 2016.* Available at: http://ieeexplore.ieee.org/servlet/opac?punumber=7704484 (Accessed: 14 January 2020).

[7] Ferencz, K. and Domokos, J., IoT Sensor Data Acquisition and Storage System Using Raspberry Pi and Apache Cassandra, in 2018 International IEEE Conference and Workshop in Óbuda on Electrical and Power Engineering (CANDO-EPE). *2018 International IEEE Conference and workshop in Óbuda on Electrical and Power Engineering (CANDO-EPE), Budapest: IEEE*, 2018, pp. 000143–000146. doi: 10.1109/CANDO-EPE.2018.8601139.

[8] Mehmood, N. Q., Culmone, R. and Mostarda, L., Modeling temporal aspects of sensor data for MongoDB NoSQL database, *Journal of Big Data*, 4(1), 2017, p. 8. doi: 10.1186/s40537-017-0068-5.

[9] R. Abbott, H. Garcia-Molina, What is a Real-Time Database System?, *Abstracts of the Fourth Workshop on Real-Time Operating systems, IEEE*, July 1987, 134–138.

[10] Kao B., Garcia-Molina H., An Overview of Real-Time Database Systems. *In: Halang W.A., Stoyenko A.D. (eds) Real Time Computing. NATO ASI Series (Series F: Computer and Systems Sciences)*, vol 127, 1994, Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-88049-0_13.

[11] Volume 6, Issue 4, April 2018, *International Journal of Advance Research in Computer Science and Management Studies,* Research Article / Survey Paper / Case Study, Available online at: www.ijarcsms.com Real-time Communication Application Based on Android Using Google Firebase ISSN: 2321-7782 (Online) e-ISJN: A4372-3114 Impact Factor: 7.327 Nilanjan Chatterjee1 Department of Computer Science St. Xavier's College (Autonomous) Kolkata, India.

[12] A. Sarkar, A. Goyal, D. Hicks, D. Sarkar and S. Hazra, Android Application Development: A Brief Overview of Android Platforms and Evolution of Security Systems, *Third International conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC),* 2019, pp. 73-79, doi: 10.1109/I-SMAC47947.2019.9032440.

[13] [*Datasheet*] D-Robotics, (2010), DHT Manual, Available at- www.droboticsonline.com/

[14] Bajrami, X. and Murturi, I., An efficient approach to monitoring environmental conditions using a wireless sensor network and NodeMCU, *e & i Elektrotechnik und Informationstechnik*, 135(3), 2018, pp. 294–301. doi: 10.1007/s00502-018-0612-9.

[15] Gay, W., DHT11 Sensor, *in Gay, W., Advanced Raspberry Pi. Berkeley*, CA: Apress, 2018, pp. 399–418. doi: 10.1007/978-1-4842-3948-3_22.

[16] Jaffe, S. R., Design of Inexpensive and Easy To Use DIY Internet of Things Platform, *California Polytechnic State University*, 2016, doi: 10.15368/theses.2016.55.

[17] *[Datasheet*] Espressif Systems, *Espressif Smart Connectivity Platform: ESP8266*,2013. Available at- https://www.electroschematics.com/wp-content/uploads/2015/02/esp8266-datasheet.pdf (Accessed: 5 August 2019).

[18] *[Datasheet]* ESP 12E PIN OUT, Available At- https://components101.com/wireless/esp12e-pinout-datasheet; https://www.make-it.ca/nodemcu-arduino/nodemcu-details specifications/.

[19] Arduino software-Available at https://www.arduino.cc/

[20] Arduino IDE online version, Available: https://www.arduino.cc/en/Main/Software (Accessed March 2019)

[21] NodeMCU ES8266 board manager installation steps : available at-

[22] Arduino firebase Master- Available at- https://github.com/FirebaseExtended/firebase-arduino

[23] ArduninoJson Available at- https://github.com/bblanchon/ArduinoJson/tree/5.x

[24] DHT by adafruit Available at- https://github.com/adafruit/DHT-sensor-library

[25] Adafruit universal sensor Available at https://github.com/adafruit/Adafruit_Sensor

[26] ESP8266 wi-fi module Available at- https://github.com/esp8266/Arduino

[27] A. Alsalemi et al., Real-Time Communication Network Using Firebase Cloud IoT Platform for ECMO Simulation, *IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, Exeter, 2017, pp. 178-182, doi: 10.1109/iThings-GreenCom-CPSCom-SmartData.2017.31

[28] Firebase Setup Available At- https://console.firebase.google.com/?pli=1;

[29] This system- IoT and Cloud based environment monitoring system- realtime Database, Available At- https://console.firebase.google.com/project/nubarealtimedb/database/nubarealtimedb/data.

[30] This system- IoT and Cloud based system Database Data Retreived as Json Document Available At- https://nubarealtimedb.firebaseio.com/.json

[31] Firebase Rest API Usage, Available At- https://firebase.google.com/docs/reference/rest/database#section-cond-ifmatch

[32] MIT Tool Available At- "http://appinventor.mit.edu/explore/"

[33] MIT project space for the app development of this IoT and Cloud based system available at-" http://ai2.appinventor.mit.edu/#5948786283777 3952."

[34] Montegriffo, N., What is an APK file and how to install APKs on Android, 2020. Available at - "https://www.nextpit.com/android-for-beginners-what-is-an-apk-file".

[35] Brenner, L., How Does the Weather Affect Us, 2018. Available At-https://sciencing.com/ather-affect-us