Turkish Journal of Computer and Mathematics Education *Vol.12 No.10 (2021), 4500-4506*
Research Article

IoT Enabled Smart Hospital Management System for Covid-19 Patients

Abhishek Singh[a], Papiya Dutta[b], Anil Kumar Sahu*[c], Sanjay Kumar Suman[d], L. Bhagyalakshmi[e]

[a]Variable Energy Cyclotron Centre, Department of Atomic Energy, Govt. of India, Kolkata, INDIA,
singhabhishek@vecc.gov.in
[b,c,d]Bharat Institute of Engineering and Technology, Hyderabad, INDIA
[e]Rajalakshmi Engineering College, Chennai, INDIA
[b]papiyadutta@biet.ac.in,[c]anilsahu82@gmail.com, [d]prof.dr.sanjaykumarsuman@gmail.com,
[e]Prof.Dr.L.Bhagyalakshmi@gmail.com
Corresponding Authour:anilsahu82@gmail.com

Article History: Received: 10 January 2021; Revised: 12 February 2021; Accepted: 27 March 2021; Published online: 28 April 2021

Abstract: The hospital management system in rural areas lacks the proper treatment due to demand of efficient doctors and health care persons. Also, in this situation of COVID-19 pandemic, common people are facing problems in health check up facilities. As per the latest report India has the doctor to patient ratio which is much below the recommended by the WHO. As per WHO guidelines, there should be one doctor for every 1000 patients, in health care environments. India has a ratio of 1:1445 as per the latest records. Also, as per rules PPE kits are essential for the health care persons to handle the corona patients. India still faces the shortage of these PPE kits, which are needed to be manufactured by the Indian ordnance factories. To address this issue, an IoT based system has been developed, which could aid in overcoming the doctor shortage in health care environments. The IoT system designed is a wearable device to be weared by the patient, which could monitor the pulse rate, temperature and SpO2 levels of the concerned patient. The data can be sent to the cloud to be stored on any IoT server like Thingspeak or any other servers like Adafruit.

Keywords: COVID-19, IoT, 2019-nCOV, PPE kits, SpO2 level, ThingSpeak server, MIT App Inventor, ESP8266 Node MCU.

1. Introduction

In the current situation of COVID-19 pandemic, there is a constant risk of health and life for the doctors and nurses in the hospitals. The corona virus was first identified in Wuhan, China in the month of December, 2019. Soon after its discovery, WHO in January, 2020 announced the new virus as 2019-nCOV [1]. Some main symptoms of the virus are fever, dry cough, tiredness, nasal congestion, difficulty in breathing, diarrhoea etc. The gathering in hospital environments, using the common utilities like toilets, lifts and corridors make it very vulnerable to spread the viruses easily from one person to another. The IoT system is an effort to overcome the shortage of health care persons and PPE kits in the hospitals. Even the countries with better doctor to patient ratio such as US, China, Spain, Italy, UK and France are facing difficulties in managing the COVID-19 spread among the doctors and health care persons. In India there is still a demand for good PPE kits, and health care persons in the hospitals for COVID patient management.

2. Methodology

A. Architecture

The wearable IoT (Internet of Things) developed is based on ESP8266 NodeMCU Wi-Fi board [2], and is interfaced to other bio-metric sensors for the user data collected along with LCD display. ESP-01 module is developed by AIthinker team, which has ESP8266 core processor by M/s Tensilica. It has an integrated on-board WiFi antenna to support IEEE802.11 b/g/n wireless TCP-IP protocol stack. The 32-bit MCU system runs on clock of 80 MHz and supports RTOS. In this project, we have established the connection of the ESP8266 Node MCU with a local WiFi AP, in which the Node MCU acts as a station mode (STA), while connecting to a local Wi-Fi access point (AP), giving it the internet connectivity. The ESP8266 Wi-Fi module is chosen due to its low cost, small size and is well efficient platform for this application.

The ESP8266 nodeMCU module collects the data from the patients at regular intervals of 120 seconds. It then sends the data to an IoT ThingSpeak server, which can collect the data corresponding to each patient in a separate group referred to as channels in ThingSpeak. The data can be analysed visually or it could be collected on request by the concerned doctors after authentication through the mobile apps or web requests.

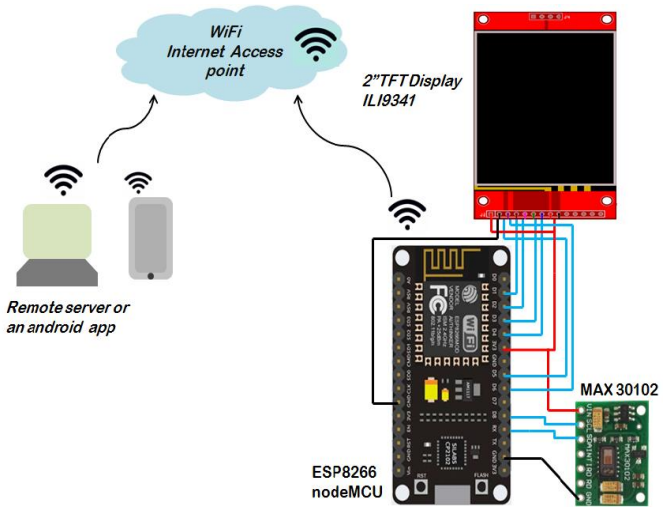

Fig. 1. IoT system architecture for patient monitoring

As shown in Fig. 1, the WiFi module ESP8266 acts as the controller as well as it serves to communicate to the internet server via internal hotspot connection just like in today's smartphones. The data collected from the sensor gets collected by module and sent immediately to the ThingSpeak server. This process repeats again and again with a regular interval of 120 seconds.

The medical parameters of the person like temperature, heart rate and SPO_2 level are collected using MAX30102 sensor modules from the nodeMCU module. The MAX30102 IC communicates to the nodeMCU module by I2C protocol and is powered by a 3.3 V supply only, which is derived from ESP8266 module itself. These three parameters are fetched from the sensor, and immediately send to the remote ThingSpeak server so as to be collected by the doctors using a mobile app which is also developed by us.

The android app developed is based on MIT app inventor's block based app development tool [3]. This app communicates with the ThingSpeak server to fetch the patient's data and display it on the app. A 2-inch TFT based display system named ILI9341communicates to the nodeMCU module by a 4-wire SPI interfacing protocol [4].

3. MIT App Inventor android app building tool

The MIT App inventor is basically a block based visual programming tool that everyone can use to build creative apps for any application. Using the MIT app inventor tool, one can build complex and big apps in a short time compared to the traditional android programming tools. The tool has two parts, one is app inventor design and the other is the app inventor block editor. The ready to use blocks are arranged in a manner so as to build the programming logic. There are various colored blocks in the tool like initializing global variable, screen initialization task, if-else block, callback functions block for each button press, blocks for assigning and reading the text box widget etc. There are plenty of other blocks available in the MIT App Inventor tool which makes the programming interactive and interesting.

As shown in Fig. 2, two blocks are shown above, a button press callback block, which upon clicking a button, executes the task in the 'do' block [4]. The other block shown in Fig. 2, shows an if-then-else block inside the checkbox callback function.

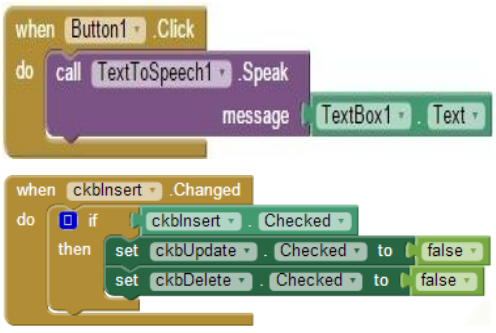

Fig. 2. Button click and checkbox callback function AI blocks

4. Introduction to ThingSpeak.com IoT server

As is clear from the block diagram of Fig. 5, ThingSpeak server collects the data from the sensor, and sends it to the doctor's mobile app or remote PC anywhere in the world upon getting a request. ThingSpeak is an IoT analytics platform server service that allows you to aggregate, visualize and analyse live data streams in the cloud. Also, MATLAB code and commands can be executed on the ThingSpeak platform to provide an additional support for data analysis. After ThingSpeak registering and logging, we can see it can have many channels as shown in Fig. 3, and each channel having 8 fields for any type of data, 3 location fields and one status field. These channels can be set as private channel or it can be set as public for sharing data publicly. Each of the channels has different read/write API keys for reading and writing the data in those channels. To read or write from the channels the device makes request to ThingSpeak API using HTTP requests. The basic objective is to collect information from the local sensors and send it to the internet writing them to the specific ThingSpeak server channels which corresponds to each patient.

Each of the channels corresponding to each patient has three fields, namely temperature, Pulse rate and SpO_2. These three fields can be updated from an ESP8266 WiFi module using read/write API keys as shown in Fig. 4. In our project, these fields are updated at regular intervals of 120 seconds.

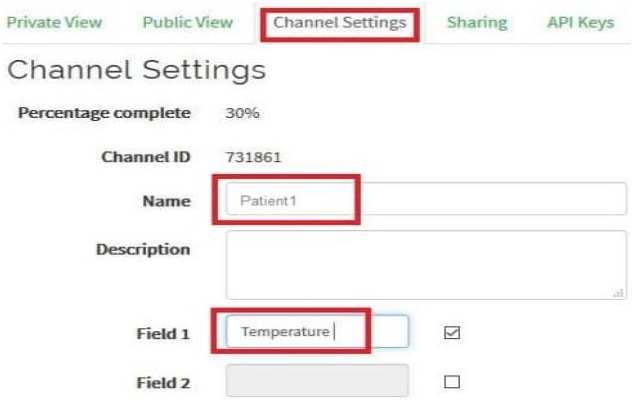

Fig. 3. ThingSpeak server Channel Settings window

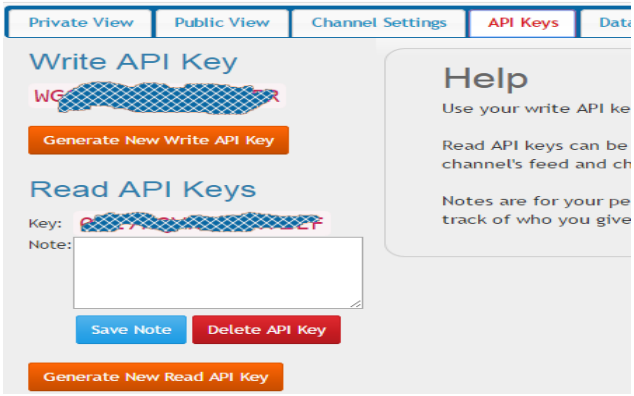

Fig. 4. ThingSpeak server Read / Write API key window

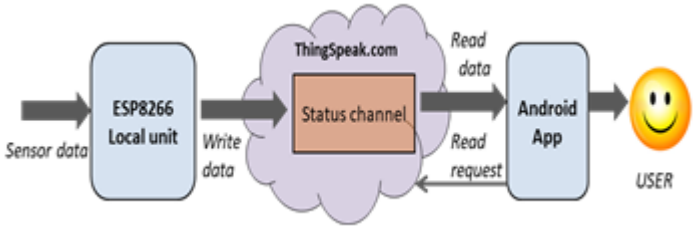

Fig. 5. Block diagram description of IoT system

5. Implementation description of the IoT system

A. ThingSpeak data reading from ESP8266 module

The system is implemented using the ESP8266 WiFi module. MAX30102 sensor attached to the module captures the temperature, pulse rate and SpO_2 levels of the patient wearing the device. It then sends this data to the ThingSpeak server over the internet. ESP8266 module needs to be configured in station mode for it to be connected to the local internet service. Following AT commands are issued to the ESP8266 WiFi module over its USB to serial debugging interface through PC [5, 6].

AT + UART_DEF= 9600,8,1,0,0 // serial baud settings

AT + CWMODE = 1 // Configure ESP8266 in Station mode

AT + CWJAP = "network name", "password" // Command to Join the network

AT + CIFSR // To get the IP address of WiFi module

Where "*network name*" is the name of the network to be connected to internet communication, "*password*" is the password of the network if any, otherwise left blank. Also last command is to get IP address of module for further use.

After configuring the module, it is programmed to write to the ThingSpeak server by the following piece of code.

ThingSpeak. Begin (client) // client is the name of WiFi client

ThingSpeak. Write field (channel-1, 1, Temp, writeAPIkey) // Function to write to the field-1 of channel-1

Also, we need to include "WiFi client. H" header in the Arduino code along with "ThingSpeak. H" header file [7]. The 'client' is a variable of data type WiFi Client. 'Temp' is value of temperature read from sensor MAX30102. Same command can be used for 'Pulse Rate' and 'SpO_2 value' in their corresponding channels. Fig. 6 and Fig. 7 show the plot of temperature and pulse rate data in ThingSpeak server.

Once the program is loaded into the ESP8266 module, it gets booted automatically upon giving the external power to the module. These commands are executed in a loop, which gets iterated after every 120 seconds using the internal timer. The figure below shows a snapshot of the channel-1 which denotes temperature data as well as pulse rate data in channel-2 captured at every 120 second interval.

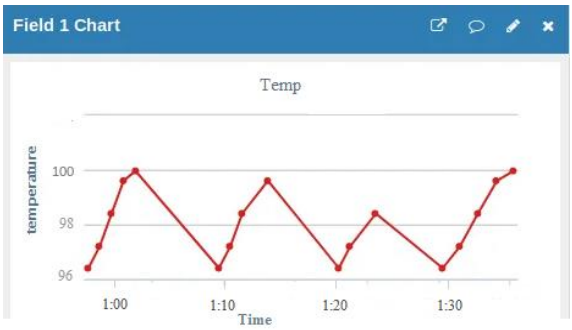

Fig. 6. Temperature plot in ThingSpeak.com

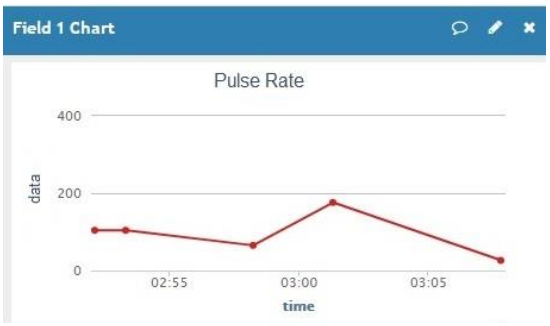

Fig. 7. Pulse Rate plot in ThingSpeak.com

B. MIT App Inventor design for patient monitoring

The patients upload their data to the ThingSpeak server, which could be fetched by the doctors using android app developed by us. The app is built using MIT app inventor development tool. Upon getting a request in the app, it sends a request to the ThingSpeak API using the patient's channel id and its corresponding read API key [7, 8]. This key must be known in advance to enable the app development easier.

The app front-end is shown in the figure below, which shows the app front widgets and appearance which will be presented to the user. Doctors need to connect their mobile to the internet first before invoking this app. Without the internet connection, the app gives an error and closes automatically. The app requires the patient number, which is nothing but the channel number of the ThingSpeak server. Upon pressing the OK button against channel number, the app tries to connect to the ThingSpeak API using the API keys, and display the temperature, pulse rate and SpO_2 levels on the screen as shown below.

The block based description of the App Inventor code is depicted in the figure below, which shows how to invoke the GET request for reading data from ThingSpeak server using channel number and its read API key.

The Fig. 8 below shows the front user GUI of the Android App developed and tested successfully to get the latest data from the ThingSpeak server, upon request from the OK button callback. The user needs to enter only the channel number corresponding to the patient ID.

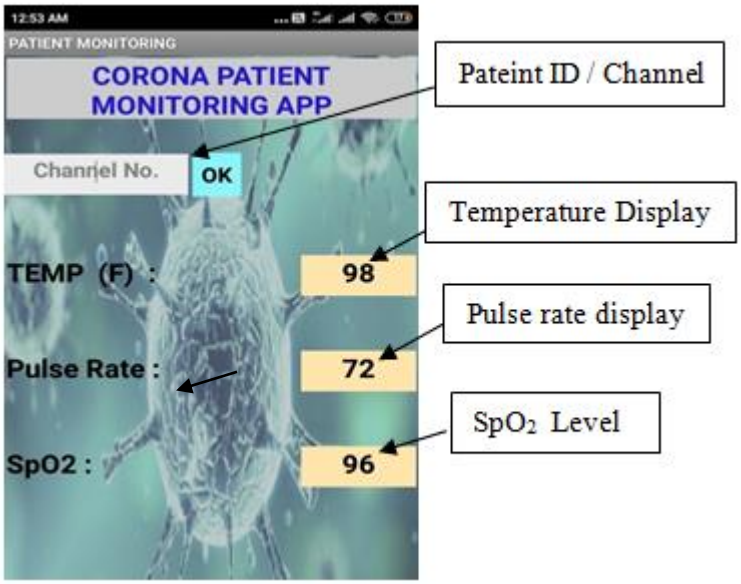

Fig. 8. Android App developed for patient monitoring

As shown in Fig. 8, the App is developed by the name "*Corona patient monitoring app*". In this app, first we need a good internet connection. The latest data regarding the patient has been retrieved using the app successfully in our lab.

Block based programming consists of an initialization part, button press callback GET request for retrieving the data, the read API feed link is initialized as shown in block Fig. 9 below, followed by the secret reads API key.

Fig. 9. The initialization part of ThingSpeak API Read Request

A web request extension is declared in the block programming which is named "FarmbotstatusCh" as shown in the figure below. This extension is used in block programming to call a web request upon clicking the OK button on GUI app. The procedural block called *FarmbotstatusCh. Url* along with *FarmbotstatusCh. Get* is used to complete the read API request, provided internet connection is there [8, 9, 10].

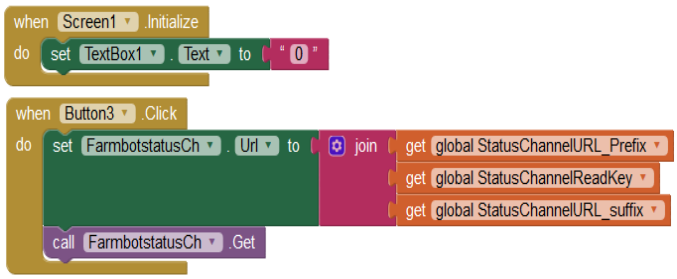

Fig. 10. The button click callback for ThingSpeak API Read request

Upon clicking the button, ThingSpeak server reads API is called, and it returns the read data in json format back to the App. To fetch this returned data from the ThingSpeak server, *FarmbotstatusCh. Got Text* procedure is used to extract the temperature, pulse rate and SpO_2 level corresponding to the fields declared in the ThingSpeak server [8, 9, 10].

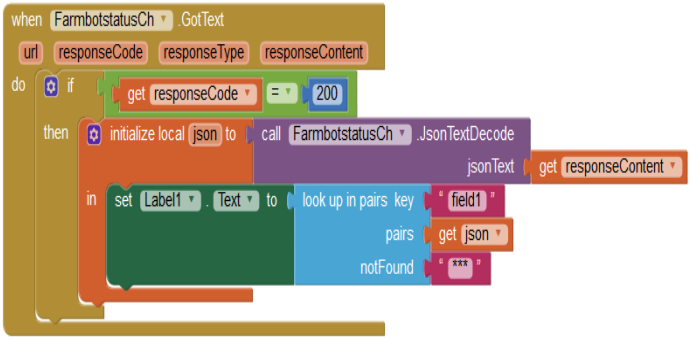

Fig. 11. App Inventor Block to get json data from ThingSpeak server

6. Conclusion and Future Scope

In this paper, patient monitoring job is relaxed by developing a useful IoT system, which can be handled easily by the patients while having an internet connection, so as to enable the doctors and other health workers remotely get the patients' data from anywhere in the world. This is useful in areas where there is a demand for health workers to monitor the patients on regularly basis. In the future work, the IoT system can be modified to get the patients and doctors interact via internet or some type of messaging service, so that a proper acknowledgement could be established between them.

References

1. Abdulrazaq, Assoc. Prof. Dr. Mohammed & Zuhriyah, Halimatuz & Al-Zubaidi, Salah & Karim, Sairah & Ramli, Rusyaizila & Yusuf, Eddy. (2020). NOVEL COVID-19 DETECTION AND DIAGNOSIS

SYSTEM USING IOT BASED SMART HELMET. International Journal of Psychosocial Rehabilitation. 24. 2296-2303. 10.37200/IJPR/V24I7/PR270221
2. Marco Schwartz, "Internet Of Things with ESP8266", PACKT Publishing.
3. Wen Xi, Evan W. Patton, "Block-Based approaches to Internet Of Things in MIT App Inventor".
4. https://simple-circuit.com/interfacing-arduino-ili9341-tft-display/
5. https://www.instructables.com/ESP8266-Communication-With-Server-and-ESP8266/
6. https://core.ac.uk/download/pdf/55305294.pdf
7. https://www.factoryforward.com/upload-sensor-data-thingspeak-using-nodemcu/
8. https://www.instructables.com/IoT-Made-Easy-With-UNO-ESP-01-ThingSpeak-and-MIT-A/
9. Sharmad Pasha, " ThingSpeak based sensing and Monitoring system for IoT with MATLAB analysis", International Journal of New Technology and Research (IJNTR), ISSN: 2454-4116, Volume-2, Issue-6, June 2016 Pages 19-23.
10. Nerella Ome, G. Someswara Rao, "Internet of Things based sensors to cloud system using ESP8266 and Arduino Due", International Journal of Advanced Research in Computer and Communication Engineering, Vol.5, Issue 10, October 2016