IoT Based Remote Medical Diagnosis System Using NodeMCU

Fahim Faisal
Dept. of Computer Science and Engineering
Daffodil International University
Dhaka, Bangladesh.
embeddedfahim@gmail.com

Syed Akhter Hossain
Dept. of Computer Science and Engineering
Daffodil International University
Dhaka, Bangladesh.
aktarhossain@daffodilvarsity.edu.bd

***Abstract*—Internet of Things (IoT) can help us better our lives in many ways by rendering real time information over the internet collected from a smart network of devices. In this paper, we have discussed about Remote Medical Diagnosis System (RMDS), which can come to aid for humans in many life-threatening situations. Often, people fall sick in locations where there are no hospitals or healthcare facility nearby, in such cases people sometimes even die due to lack of proper treatment and diagnosis. In rural areas of third-world countries, this problem is even more intense. For demonstration purpose, heartrate and body temperature of a person is determined and rendered over the internet. Health data is uploaded in real-time and can be viewed through a web browser. The aim of RMDS is to remotely provide health information of a patient to a healthcare professional in life-threatening situations. It can also be used for remote patient monitoring of regular patients of a doctor.**

Keywords—IoT, Heartbeat Sensor, Temperature Sensor, Microcontroller, Photoplethysmography

I. Introduction

Internet of Things (IoT) has a very close relationship with embedded systems, where a smart system consisting of various sensors and modules is connected to the internet. The modules can either be connected to each other through the internet or they can be inter-connected through a local network and the local network can further be connected to the internet. IoT opens doors to numerous possibilities and smart solutions to complicated problems. From agriculture to home automation to smart cities to healthcare, application of IoT is beyond limits. The research in this field has gained a lot of popularity recently. Utilizing IoT in healthcare gives us a lot of benefits. Firstly, it reduces the cost of treatment as diagnosis is performed remotely. Also, it saves valuable time of both doctors and patients while maintaining descent accuracy. It also has the potential of reducing the burden of manual data collection. In life-threatening situations, it can provide image of the patient's health in real-time so that the doctor can make an informed and reasonably quick decision from a remote location. Keeping these facilities in mind, we proposed an intelligent system consisting of a microcontroller, a Wi-Fi module, a button, two sensors (heartbeat and temperature), a Wi-Fi router and a website. The raw sensor data is first collected and processed by the microcontroller and then it is passed on to the Wi-Fi module. The Wi-Fi module then pushes the data to a remote webserver through a Wi-Fi router. The server stores the data and displays it through the website. A doctor can easily view all the health information by visiting this website and logging in with his/her credentials, which will be provided by the system administrators.

II. Related Works

Several comprehensive reviews about the subject of health monitoring with wearable sensors have been previously presented in the literature form. Many such reviews focus on giving a global overview of the topic. Studies on health monitoring systems include wearable, mobile and remote systems.

Abasi Julius and Zhang Jian-Min proposed a remote patient monitoring system consisting of a microcontroller, three sensors and a radio module which can upload health information to a remote server [1]. Shivam Gupta *et al.* proposed a system consisting of three sensors, a microcontroller and a Wi-Fi module [2]. Moeen Hassanalieragh *et al.* have discussed about wearable wireless sensor networks and the effect of these networks in healthcare, in their paper [3]. Ahmed Abdelgawad *et al.*, in their paper, have discussed about a system that uses Raspberry Pi and iBeacon cloud services for respectively collecting/processing and relaying data [4]. Hemalatha Rallapalli and Pavani Bethelli have proposed a system with RFID security. The data is first stored in a local server and then transferred to a remote server [5]. Jorge Gómeza *et al.* proposed a system where an android device is used to process the data coming from the sensors and then the device sends the data to a remote server [6]. T.Saraswathi and S.Amutha proposed a system consisting of an Arduino microcontroller, temperature sensor, heartbeat Sensor, Wi-Fi module, Bluetooth module and Network Radio Frequency (NRF) module. It handles both inpatients and outpatients of a hospital. Inpatients' data is transmitted using NRF module to a local receiver and further processed and displayed on the website while outpatients' data is transmitted over Wi-Fi [7]. Mobyen Uddin Ahmed *et al.* presented a literature review that discusses the possibilities of wearable sensors and their connectivity through the Internet of Things for 360° health data collection [8]. Dalyah Y. Al-Jamal *et al.* discussed the possible connectivity of sensors and the use of the Internet of Things for monitoring one's heart conditions and recording those data for foreseeing any upcoming heart disease [9]. Mingzhe Jiang *et al.* proposed a system to monitor one's facial expression to obtain mental and physical health data and render that data over the internet [10].

III. Methodology

978-1-7281-2741-5/19/$31.00 ©2019 IEEE

Remote Medical Diagnosis System (RMDS) has the ability to measure two types of health data, one is heartrate (Heartbeats Per Minute) or BPM and the other one is body temperature in Celsius. The most important and sophisticated part of this system is the correct measurement of heartrate. For the calculation and determination of heartrate, accurate detection of heartbeats is crucial. In RMDS, Photoplethysmography (PPG) [11] has been used to detect heartbeats.

Photoplethysmography [11] is a low-cost, non-invasive heartbeat and oxygen level determining mechanism which is based on the reflection, refraction and scattering of light. In this mechanism, a diagnosis device monitors the perfusion of blood to the dermis and subcutaneous tissue of the skin by passing Infrared Ray (IR) through the skin. IR is a kind of light with wavelength greater than 800nm, which is located right after red light in the electromagnetic spectrum and is invisible to the naked eye. A portion of the emitted IR is absorbed by blood and the rest is reflected, refracted and scattered. By monitoring the amount of reflection or refraction and scattering, heartbeats can be detected. Thus, there are two modes of PPG; transmissive and reflective. IR of wavelength close to 940nm is best suited for the deepest penetration and yielding the best deep tissue blood flow measurement [12].

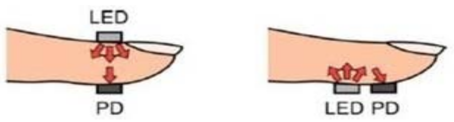

Fig. 1: Transmissive and Reflective PPG Modes.

At the beginning of a heartbeat, the capillaries beneath our skin fills with blood. At that moment, the volume of blood is the highest. With time, blood volume decreases and the amount of reflection, refraction and scattering increases accordingly, as less IR is absorbed by blood. The reflected/refracted/scattered IR is caught by an IR receiver (photodiode). Based on the amount of reflected/refracted/scattered IR, the voltage across the photodiode fluctuates and using these voltage levels, an Operational Amplifier (Op-Amp) generates different analog values which are then fed to a microcontroller for further processing.

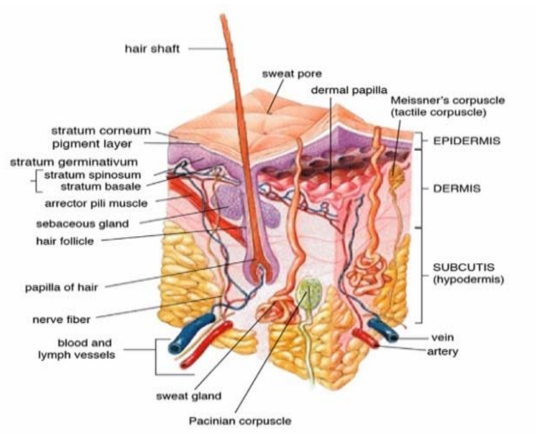

Fig. 2: Diagram of the Layers of Human Skin.

A PPG waveform comprises two main components; 'AC' arterial pulsatile changes in blood flow synchronized to heart beat and 'DC' elements attributed to venous blood, tissue, respiration, sympathetic nervous system activity and thermoregulation. It is the AC changes which are used to extract heartbeats. The signal is generally mixed with a lot of noise so a descent amount of filtering is needed for obtaining acceptable data.

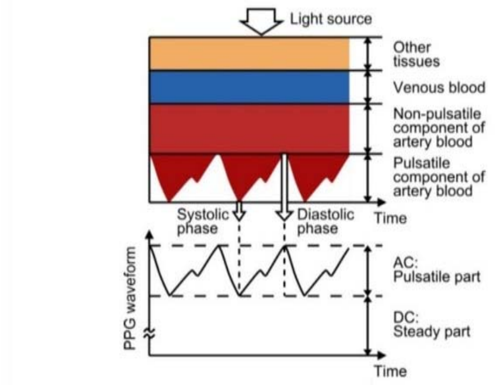

Fig. 3: Variation in Light Attenuation by Tissue.

IV. Electronic Components

We need a few hardware components to materialize the proposed system. First, we need a processor for data collection and processing. In other words, we need a brain for the system. The brain of RMDS is the microcontroller *Atmel ATMega328P*. An *Arduino Nano* [13] has been used for utilizing this microcontroller and programming it easily.

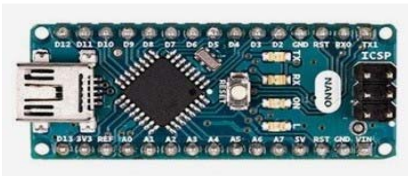

Fig. 4: Arduino Nano.

Two sensors are attached to the *Arduino Nano* [13]. One of them is the *Heartbeat Sensor*. The other one is the temperature sensor *LM35* [14].

The custom-built *Heartbeat Sensor* is a small device that utilizes PPG in reflective mode. It needs to be placed under the fingertip of a patient. It has an Infrared Ray (IR) emitter-receiver pair and an Operational Amplifier (Op-Amp) onboard. IR is emitted from the emitter (IR LED) and reflected onto the receiver (photodiode). Depending on the amount of reflected light, the voltage level across the IR receiver (photodiode) fluctuates. Output of the photodiode is connected to an Op-Amp for amplification as voltage levels across the photodiode are too delicate to be detected by a microcontroller. The Op-Amp acts as a non-inverting amplifier with an amplification factor of 1001. The output of the first Op-Amp is fed to the second Op-Amp, which acts as a voltage comparator. The output of the second Op-Amp triggers a N-P-N transistor and from there the signal is sent to the microcontroller. The Op-Amp used in this circuit is LM358, which has two Op-Amps on the same chip.

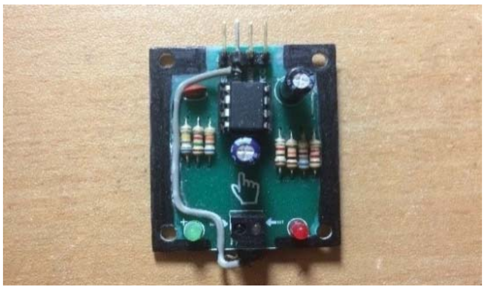

Fig. 5. Heartbeat Sensor.

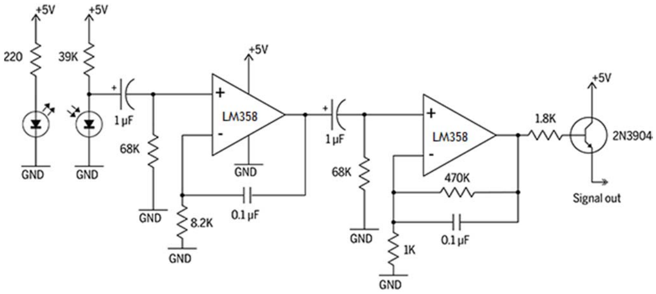

Fig. 6: Heartbeat Sensor Schematic.

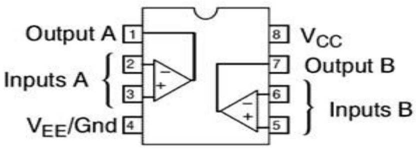

Fig. 7: LM358 Pinout.

Temperature sensor IC, *LM35* has been used to measure the human body temperature. It is placed adjacent to the *Heartbeat Sensor* in order to make the measurement process more convenient. It has a thermocouple placed inside which reacts to changes in temperature. Different temperatures generate different potential difference or voltages across the thermocouple. These changes in voltage level are further amplified through Op-Amps and supplied to the output pin and thus, we get different analog values on the output pin. *LM35* has an advantage over linear temperature sensors calibrated in Kelvin, as it is calibrated in Celsius and we don't need to subtract a large constant voltage from the output to obtain convenient Centigrade scaling. The low-output impedance, linear output, and precise inherent calibration of the *LM35* makes interfacing to readout or control circuitry very convenient. The sensor circuitry is sealed and therefore is not subjected to oxidation and other natural processes. With this sensor, temperature can be measured more accurately than with a thermistor. It is also low self-heating and does not cause more than 0.1°C temperature rise in still air. The operating temperature ranges from -55°C to 150°C.

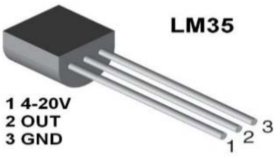

Fig. 8: LM35 Pinout.

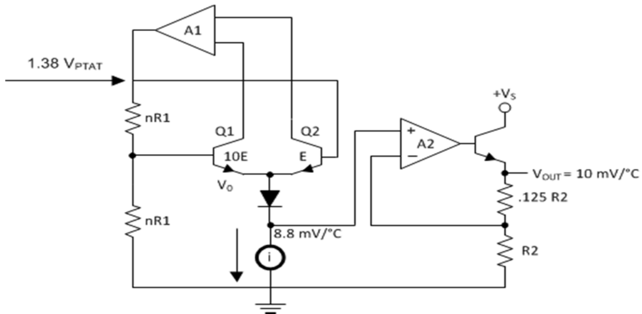

Fig. 9: Functional Block Diagram of LM35.

The microcontroller processes and filters these raw data with sophisticated algorithms and generates digital values which are then handed over to the Wi-Fi module, *ESP8266.* A *NodeMCU v2* [15] has been used to utilize this Wi-Fi module. The Wi-Fi module uploads the data to a webserver through a Wi-Fi network created by the Wi-Fi router [16]. The *NodeMCU* can also be used as a full-fledged microcontroller but due to its 3.3V logic level, connecting the sensors directly to it would require additional logic level shifting circuitry and also the *NodeMCU* has only one analog input pin while RMDS has two sensors which output analog values.

A custom-built 5V Lithium-Ion rechargeable power supply has been used to power the system as all of the used components require 5V to operate. Two 1000 mAh Lithium-Ion batteries have been used in parallel in the power supply. The 3.7V of the batteries is further boosted in the Battery Management System (BMS) [17]. It has 5V output and can deliver 4A of current at any instance through two channels, 2A per channel. One channel provides power to the *Arduino*, heartbeat and temperature sensor and the 16x2 LCD [18]. The *NodeMCU* is given power through a separate channel of the power supply as it draws a lot of current, specially while communicating with remote webservers. There is also a button which initiates the heartbeat and temperature measurement. It is directly connected to the *Arduino*.

The BMS has its own voltage regulators on each channel so additional voltage regulation wasn't required. In average, the total current consumption of the system is around 250mA. The used power supply can handle the load flawlessly. The system can run on batteries for 8 hours straight and then the BMS cuts the power off automatically to protect the batteries from over-discharging.

Fig. 10: NodeMCU Microcontroller.

V. Measurement of BPM and Temperature

The measurement of Beats Per Minute (BPM) requires a very carefully designed and sophisticated algorithm. We take the output values of the *Heartbeat Sensor* and then filter it inside the microcontroller. The microcontroller is set up to fetch data every time there is hardware interrupt created by the button. A button press is registered through the change in state of the variable linked to the button. Every time the button is pressed the button variable goes from HIGH to LOW as the button is connected through an emulated pull-up resistor [19] to the microcontroller. If a button press is registered, the time and heartbeat count is initiated. Every time there is a heartbeat, the *Heartbeat Sensor* generates HIGH logic (around 4.78V), which is read by the "digitalRead()" function [20] of the *Arduino* platform and a value of 1 is assigned to the "count" variable. The value of the "count" variable is incremented by 1 every time, upon detection of further heartbeats. We take a note of the time in which the first heartbeat took place, through the "millis()" function [21] of the *Arduino* platform, which is basically a timer that keeps track of how much time has passed since the microcontroller has last been reset, in milliseconds (1 second = 1000 milliseconds). After the first beat has taken place, we let the timer run for 10 seconds and take note of how many beats take place within those 10 seconds. If we multiply the number of beats within 10 seconds with 6, we get the total number of beats occurring within 60 seconds, i.e. 1 minute. Thus, we get our BPM.

```
void loop()
{
  while(digitalRead(btnPin) > 0);

  beginning = millis();
  ending = beginning + 10000;

  while(millis() <= ending)
  {
    if(digitalRead(sigPin))
    {
      count++;
```

Fig. 11: Partial Arduino Code (BPM Calculation).

The measurement of temperature is comparatively straight-forward as the output of the *LM35* temperature IC is linear, i.e. its voltage level increases with increase in temperature and decreases with decrease in temperature. We measure the output voltage of the temperature sensor IC through the "analogRead()" function [22] of the *Arduino* platform, which returns digital values as *Arduinos* have 8-bit Analog-to-Digital Converters (ADC) on the analog input pins. We need to convert these digital values back to analog values as the analog voltage levels are directly proportional to the temperature in Celsius. The conversion is accomplished through the following equation,

$$\frac{Resolution\ of\ the\ ADC}{System\ Voltage} = \frac{ADC\ Reading}{Analog\ Voltage\ Measured}$$

Once converted, we get the analog voltage value in millivolts, which needs to be divided by 10 in order to get the temperature value in Celsius as *LM35*'s scale factor is 0.01V/°C, i.e. the output voltage varies by 10mV in response to every °C rise/fall in ambient temperature.

```
tempValue = analogRead(tempPin);
millivolts = (tempValue / 1024.0) * 5000;
celsius = millivolts / 10;
```

Fig. 12: Partial Arduino Code (Temperature Calculation).

VI. Software and Communication

The *Arduino Nano* has been programmed in the *Arduino* language for ease of programming. The microcontroller code filters the raw sensor data and converts it into Beats Per Minute (BPM). The temperature values are also processed by this code.

The *NodeMCU* was programmed in C++ using the PlatformIO IDE [23]. Although *NodeMCU*'s official programming language is Lua, C++ was used for simplicity and efficiency. After booting up, the *NodeMCU* automatically starts collecting data from the *Arduino Nano* via serial communication. These data are further pushed to a remote webserver using a Wi-Fi network created by the Wi-Fi router. BPM and temperature values are collected from the *Arduino* and stored inside sperate variables on the *NodeMCU*. A URL string is formed using these values, which is accessed by utilizing the "GET" method [24] of HTTP. Here, we use Transmission Control Protocol (TCP) for secure and reliable data transmission. The webserver contains a PHP script that receives data from the *NodeMCU* in real-time and stores it into a MySQL database along with the timestamp and identification number. When the URL of RMDS is accessed another PHP script fetches the data from the database and displays it on the homepage. The web address of RMDS is rmds.embeddedfahim.com. It was written in HTML and beautification was done using CSS. The homepage is programmed to refresh automatically every 30 seconds. When first accessed, the website will ask for username and password, which can be collected from the system administrators. The administrators have a separate section in the website from which they can manage the system easily and efficiently.

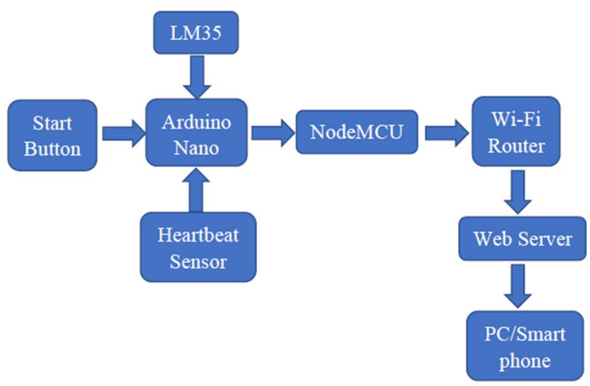

Fig. 13: Data Flow Path of RMDS.

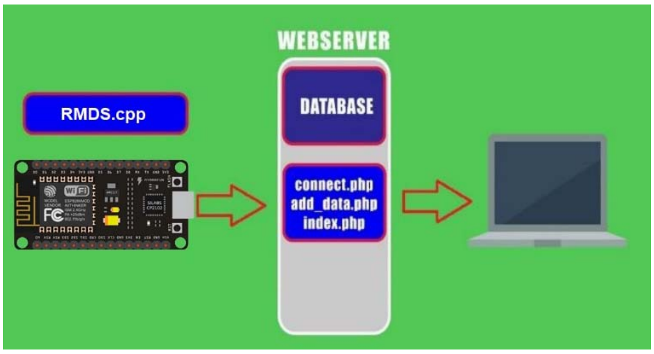

Fig. 14: Server-side Mechanism.

```
String link =  String("http://rmds.embeddedfahim.com/add_data.php?bpm=") + bpm + "&temp=" + temp;
http.begin(link);
int httpCode = http.GET();
Serial.println(httpCode);

if(httpCode == 200)
{
  Serial.println("Successfully uploaded data to the web server..");
  digitalWrite(16, HIGH);
}
else
{
  Serial.println("Error uploading data to the web server!!");
}

http.end();
```

Fig. 15: Partial NodeMCU Code (Uploading Data to Server).

```
$conn = mysqli_connect($servername, $username, $password, $dbname);

if(!$conn) {
        die("Connection failed: " . mysqli_connect_error());
}

date_default_timezone_set('Asia/Dhaka');
$date = date('Y-m-d h:i:s', time());
$bpm = $_GET['bpm'];
$temp = $_GET['temp'];

$sql = "INSERT INTO data (date, bpm, temperature) VALUES ('$date', $bpm, $temp)";

if(mysqli_query($conn, $sql)) {
        echo "New record created successfully!";
}
else {
        echo "Error: " . $sql . "<br>" . mysqli_error($conn);
}

mysqli_close($conn);
```

Fig. 16: Partial PHP Code (Adding Data to Database).

VII. Experiment and Observations

Several experiments were conducted to ensure proper functionality of the system. This helps to accurately determine the future complications that may occur in actual field. RMDS showed success scenarios when tested. Temperature and BPM values were being uploaded and displayed on the website in real time. Below are some results obtained, which are seen on the serial monitor on PC, 16x2 LCD and the RMDS website. All the health information and data are marked by unique ID numbers. If anyone needs to find information of a particular patient, then he/she just needs to remember ID number(s) of the data of the concerned patient. In fig. 17, data is being displayed on the serial monitor of the *Arduino* IDE on PC. In fig. 18, data is being almost spontaneously uploaded to the remote webserver which is then being displayed on the website of RMDS. In fig. 20, comparison between RMDS and a comparison device (Android) is shown.

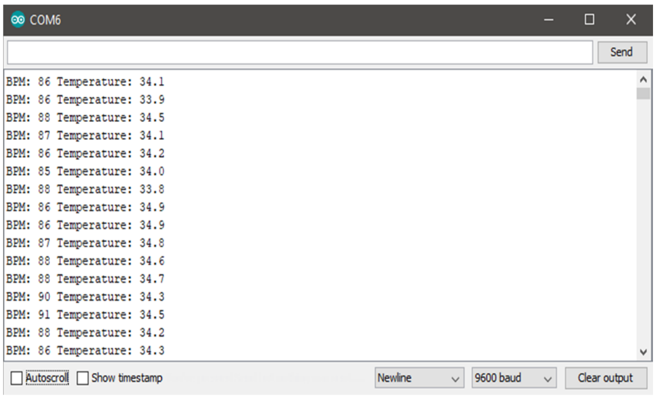

Fig. 17: Results on the Serial Monitor.

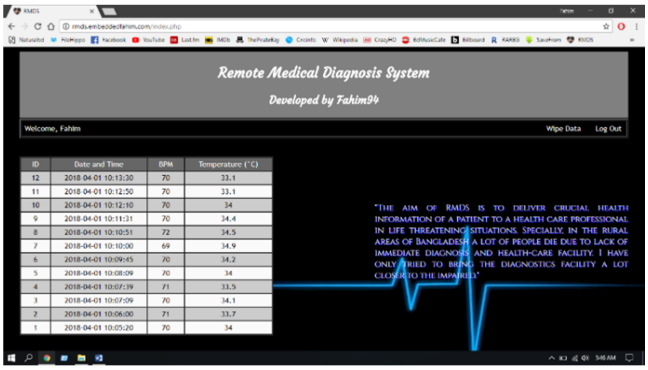

Fig. 18: Health Information Acquired and Uploaded in Real Time.

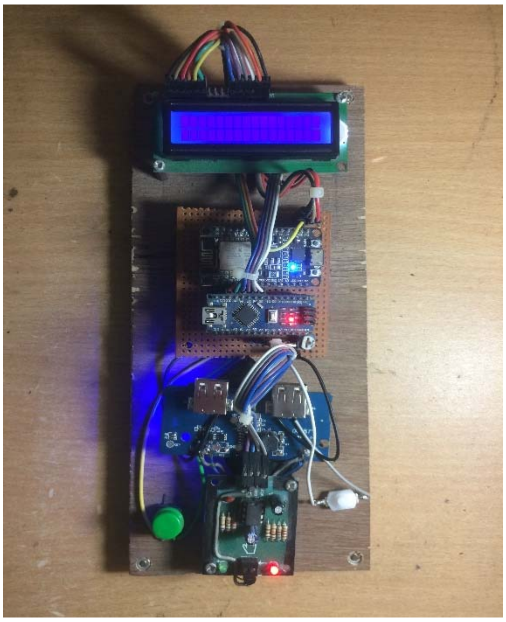

Fig. 19: RMDS Hardware.

We compared RMDS with other existing heartrate measuring devices to see the accuracy of our system. In particular, we used an android device with a heartrate program installed and utilized it as a comparison model. The program is called "Heart Rate Plus" [25], which can be obtained from the Google Play Store for free. The program also utilizes Photoplethysmography for heartbeat detection and BPM measurement.

Fig. 20: Comparison between RMDS and Android.

For simulation, we took two healthy people (both male) as test subjects respectively of age 25 and 55. We measured both of their heartbeats and temperature in different scenarios and physiological conditions, e.g. in agile and resting conditions. Each of their heartbeats were measured 5 times (every time in a different condition), which are shown in the tables below:

TABLE I. RESULTS OF PERSON 1

Serial No.	BPM Recorded by RMDS	BPM Recorded by Android	Temperature (°C)
01	77	77	34.9
02	78	78	35.1
03	80	81	34.8
04	87	85	35.2
05	91	94	35.3

TABLE II. RESULTS OF PERSON 2

Serial No.	BPM Recorded by RMDS	BPM Recorded by Android	Temperature (°C)
01	67	66	35.7
02	70	71	35.8
03	74	74	35.3
04	78	80	36.0
05	81	80	35.9

VIII. ADVANTAGES OF RMDS

The main advantage of RMDS is that it's extremely portable and lightweight. It generates descent BPM and temperature results compared to other similar systems. Also, it costs only around 30 USD (including web hosting & domain cost for 1 year) to build this system with the current facilities. The low power consumption and long battery life increases usability and overall efficiency.

Health data collection has always been manual thus prone to human-made mistakes and other errors. RMDS reduces the scope of those errors and simultaneously ensures an increased level of transparency and efficiency. The data upload is instant and a healthcare professional can see the data from the website anytime, from anywhere around the world.

IX. AREAS OF FURTHER DEVELOPMENT

As this is only the first prototype of the system, there's room for a lot of improvements. First, the Heartbeat Sensor can be upgraded to a better one. The IR emitter-receiver couple on our sensor is very small in size and thus getting accurate heartbeat data is a bit difficult. As the surface area of the IR emitter-receiver is small, we need to be very careful while placing our fingertip on top of it. Random movements of the finger can cause inaccurate BPMs to be generated, but if we're careful and the finger is firmly placed on top the sensor, the values are usually pretty accurate. So, there is room for improvement in this particular area. More diagnosis modules can be added for collecting more health information. For example, ECG module, Blood Pressure module, Oxygen Level module etc. Wi-Fi can be replaced with GSM/GPRS module so that the system becomes completely independent in terms of communication and the mobility of the system increases. A dedicated Android/iOS app can be developed for smartphones which will display the data within the app. It will deduct the burden of typing the URL of RMDS every time somebody wants to access patients' data. Specially, for doctors it would be helpful since their time is extremely precious. Also, the website of RMDS doesn't have the facility to store patients' names. So, there can be separate sessions for each patient, where patients' basic information will also be stored. With proper funding and logistics support, a lot more can be done in this horizon. We have only tried to develop a system that is portable enough to render health information over the internet from anywhere in the world. Our main goal was to reduce the number of deaths caused by lack of immediate diagnosis.

X. CONCLUSION

In this modern era, IoT has become one of the brightest fields by which human life has become easier, safe and efficient through variety of its applications. In every sector of our daily life, we realize the impact of this particular field. However, our effort was to develop a life saver. Heartrate (BPM) and temperature of a person was determined on-device and rendered over the internet through a Wi-Fi network. Uploaded data was stored in a remote webserver and displayed through the RMDS website. Comparison with another device was conducted in order to ensure precision and accuracy. Finally, advantages and areas of further development were discussed. Remote health diagnosis systems, particularly those equipped with IoT technology, offer access to increased frequency of patients' health data, help to reduce hospital stays and enable patient monitoring even after release. They can save lives through real-time interventions and support while reducing cost and diagnosis time. These systems have the potential of radically improving healthcare facilities around the globe. However, challenges in sensing, analytics, and visualization of health data requires further research and discussion. They need to be addressed before these systems can be designed for seamless integration into clinical practice.

XI. ACKNOWLEDGEMENT

The authors of this paper are very much grateful to the faculty members of the Department of Computer Science and Engineering, Daffodil International University, Dhaka, Bangladesh, and Department of Computer Science and Engineering, Bangladesh Army University of Engineering and Technology, Natore, Bangladesh, for their encouragement throughout the build of this project. Without their support, we could never pursue such a complicated project.

REFERENCES

[1] Abasi Julius and Zhang Jian-Min, "IoT Based Patient Health Monitoring System Using LabVIEW and Wireless Sensor Network", published on the International Journal of Science and Research (IJSR), ISSN (Online): 2319-7064, Volume 6, Issue 3, March 2017

[2] Shivam Gupta, Shivam Kashaudhan, Devesh Chandra Pandey, Prakhar Pratap Singh Gaur, "IoT based Patient Health Monitoring System", published on the International Research Journal of Engineering and Technology (IRJET), ISSN: 2395-0072, Volume 4, Issue 3, March 2017

[3] Moeen Hassanalieragh, Alex Page, Tolga Soyata, Gaurav Sharma, Mehmet Aktas, Gonzalo Mateos, Burak Kantarci, Silvana Andreescu, "Health Monitoring and Management Using Internet-of-Things (IoT) Sensing with Cloud-based Processing: Opportunities and Challenges", published on 2015 IEEE International Conference on Services Computing, 978-1-4673-7281-7/15 © 2015 IEEE, DOI 10.1109/SCC.2015.47

[4] Ahmed Abdelgawad, Kumar Yelamarthi and Ahmed Khattab, "IoT-Based Health Monitoring System for Active and Assisted Living", published by ICST Institute for Computer Sciences, Social Informatics and Telecommunications Engineering 2017, O. Gaggi et al. (Eds.): GOODTECHS 2016, LNICST 195, pp. 11–20, 2017, DOI: 10.1007/978-3-319-61949-1_2

[5] Hemalatha Rallapalli and Pavani Bethelli, "IoT Based Patient Monitoring System", published on Int'l Journal of Computing, Communications & Instrumentation Engg. (IJCCIE) Vol. 4, Issue 1 (2017) ISSN 2349-1469 EISSN 2349-1477

[6] Jorge Gómeza, Byron Oviedob and Emilio Zhumab, "Patient Monitoring System Based on Internet of Things", published on the 7th International Conference on Ambient Systems, Networks and Technologies (ANT 2016)

[7] T.Saraswathi and S.Amutha, "IoT Based Wireless Healthcare Monitoring", published on International Journal of Emerging Trends in Science and Technology (IJETST) Vol. 03, Issue 05, Pages 784-792, May 2016, ISSN 2348-9480

[8] Mobyen Uddin Ahmed, Mats Bj¨orkman, Aida ˇCauˇsevi´c, Hossein Fotouhi and Maria Lind´en, "An Overview on the Internet of Things for Health Monitoring Systems". In: Mandler B. et al. (eds) Internet of Things. IoT Infrastructures. IoT360 2015. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, vol 169. Springer, Cham

[9] Dalyah Y. Al-Jamal, Maryam H. Eshtaiwi and Liyakathunisa Syed, "IoT Based Process Model for Heart Monitoring Process", published on International Journal of Computer and Systems Engineering Vol. 11, No. 6, 2017

[10] Mingzhe Jiang, Tuan Nguyen Gia, Arman Anzanpour, Amir-Mohammad Rahmani, Tomi Westerlund, Sanna Salanter¨a, Pasi Liljeberg and Hannu Tenhunen, "IoT-based Remote Facial Expression Monitoring System with sEMG Signal", published on 2016 IEEE Sensors Applications Symposium (SAS), 978-1-4799-7250-0/16 ©2016 IEEE

[11] "Photoplethysmography," Last accessed on February 1, 2019, at 1:00:00PM [Online]. Available: https://www.news-medical.net/health/Photoplethysmography-(PPG).aspx

[12] "940mm IR for PPG" Last accessed on February 1, 2019, at 1:05:00PM [Online]. Available: https://www.youtube.com/watch?v=2v3rae-73jc

[13] "Arduino Nano," Last accessed on February 1, 2019, at 1:10:00PM [Online]. Available: https://www.arduino.cc/en/Guide/ArduinoNano

[14] "LM35," Last accessed on February 1, 2019, at 1:15:00PM [Online]. Available: https://wiki.eprolabs.com/index.php?title=Temperature_Sensor_LM35

[15] "NodeMCU v2," Last accessed on February 1, 2019, at 1:20:00PM [Online]. Available: https://www.seeedstudio.com/NodeMCU-v2-Lua-based-ESP8266-development-kit.html

[16] "Wi-Fi Router," Last accessed on February 1, 2019, at 1:25:00PM [Online]. Available: https://ryanscomputers.com/network/router/tp-link-tl-wr840n-v2-300mbps-router.html

[17] "Battery Management System," Last accessed on February 1, 2019, at 01:30:00PM [Online]. Available: https://www.allmartbd.com/5v-2a-power-bank-board.html

[18] "16x2 LCD," Last accessed on February 1, 2019, at 01:35:00PM [Online]. Available: https://www.allmartbd.com/lcd1602-with-backlight.html

[19] "Emulated Pull-Up Resistor," Last accessed on February 1, 2019, at 01:40:00PM [Online]. Available: https://www.arduino.cc/en/Tutorial/InputPullupSerial

[20] "digitalRead()," Last accessed on February 1, 2019, at 01:45:00PM [Online]. Available: https://www.arduino.cc/reference/en/language/functions/digital-io/digitalread/

[21] "millis()," Last accessed on February 1, 2019, at 01:50:00PM [Online]. Available: https://www.arduino.cc/reference/en/language/functions/time/millis/

[22] "analogRead()," Last accessed on February 1, 2019, at 01:55:00PM [Online]. Available: https://www.arduino.cc/reference/en/language/functions/analog-io/analogread/

[23] "PlatformIO," Last accessed on February 1, 2019, at 02:00:00PM [Online]. Available: https://docs.platformio.org/en/latest/

[24] "HTTP GET Method," Last accessed on February 1, 2019, at 02:05:00PM [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods