

People's Democratic Republic of Algeria
Ministry of Higher Education and Scientific Research
University M'Hamed BOUGARA – Boumerdes



Institute of Electrical and Electronic Engineering
Department of Electronics

Final Year Project Report Presented in Partial Fulfilment of
the Requirements for the Degree of

MASTER

In computer engineering
Option: computer engineering

Title:

**IoT Based Healthcare Monitoring
System**

Presented by:

- **Souhila REFFAA**
- **Ouiza BOUSSAMET**

Supervisor:

Mr. A. MOHAMMED-SAHNOUN

Registration Number 2019/2020

Acknowledgement

In The Name of ALLAH, The Most Gracious and The Most Merciful.

We gratefully acknowledge our parents for their tremendous support, encouragement; this project would never be completed without them.

We would like to express our gratitude to Mr. A. MOHAMMED-SAHNOUN for his supervision, advice and guidance throughout the development of this project.

We would like to thank our friends for their motivation and being by our sides in the rough times .

Finally, we want to express our deepest thank to Dr. Abdelkader ZITOUNI for his excellent role in helping us with advices and providing some electronic components.

DEDICATION

In The Name Of ALLAH, The Most Gracious and The Most Merciful.

I want to dedicate this humble work

To My mother

A strong and gentle soul who taught me to trust in Allah believed in hard work and that so much could be done with little

To my father

For earning and honest living for me and for supporting and encouraging me to believe in myself

To my two brothers

Thank you for annoying me all the time but I can't imagine my life without you two

To my all friends

Thank you for all the days that we spent together you were the most beautiful souls that I have never met

finally I dedicate to the strongest person I know: " Me "

Ouiza Boussamet

Dedicacation

I dedicate this humble work to :

To the memory of my grandparents, my uncle "Yekhlef", and my aunt "TaTa Nacira" May God have mercy on them.

To my beloved parents, thank you for all your hard efforts, there is no word for the gratitude you deserve .

To my sweet, loving sister "JIFI", thank you for everything, I'm such a lucky person to have you in my life. To my brother "MOHAMED".

To my friends "DIDI" ,"NOUNOU", "CICI", "LINDA", "NESRINE" thank you for being by my side all the time , thank you for all the wonderful and unforgettable moments we spent together.

To my friend "IMAN", you have been and still my best friend.

To a close person "ZINOU", thank you for your constant support and encouragement.

To everybody, with whom I shared a single second of joy in my life.

Souhila Reffaa

Abstract

With the development of the technologies and the miniaturization of sensors, there have been attempts to use the new technologies in different areas to ameliorate the quality of human life.

One of the last technologies used is the internet of things. IoT visualizes a future of anything anywhere by anyone at any time.

The principal domain that has seen the utilization of IoT is in the healthcare sector. Its concepts have been widely used to interconnect medical resources and offer smart healthcare service to the patients.

Therefore, this project is to solve a healthcare problem currently society is facing. The main idea is to monitor one's health wirelessly and keep track of the patient's condition or send medical parameters like Body temperature, Pulse rate, and ECG, wirelessly to a physician or doctor.

This system is comprised of three main parts. The first part being, detection and collection of patient's vitals using sensors and Arduino UNO microcontroller, the second part is for relaying and receiving automatically the data to an IoT platform. The IoT platform used in this project is ThingSpeak. And the last part was providing the detected data for remote viewing in order to enable the doctor or guardian to analyze the patient's health progress away from hospital premises. In addition to a warning push button that can be pressed in an emergency situation in order to send an Email alerts to the hospital services.

Abstract

With the development of the technologies and the miniaturization of sensors, there have been attempts to use the new technologies in different areas to ameliorate the quality of human life.

One of the last technologies used is the internet of things. IoT visualizes a future of anything anywhere by anyone at any time.

The principal domain that has seen the utilization of IoT is in the healthcare sector. Its concepts have been widely used to interconnect medical resources and offer smart healthcare service to the patients.

Therefore, this project is to solve a healthcare problem currently society is facing. The main idea is to monitor one's health wirelessly and keep track of the patient's condition or send medical parameters like Body temperature, Pulse rate, and ECG, wirelessly to a physician or doctor.

This system is comprised of three main parts. The first part being, detection and collection of patient's vitals using sensors and Arduino UNO microcontroller, the second part is for relaying and receiving automatically the data to an IoT platform. The IoT platform used in this project is ThingSpeak. And the last part was providing the detected data for remote viewing in order to enable the doctor or guardian to analyze the patient's health progress away from hospital premises. In addition to a warning push button that can be pressed in an emergency situation in order to send an Email alerts to the hospital services.

Contents

Acknowledgement.....	I
Dedication.....	II
Abstract.....	IV
Contents.....	V
List of figures.....	VIII
General introduction.....	1

Chapter I: Remote Healthcare Technology

Introduction.....	2
1.1 Medical Monitoring systems	2
1.2 Remote health monitoring systems (RHMS).....	2
1.3 Mobile health monitoring systems (MHMS).....	3
1.4 Wearable health monitoring systems (WHMS).....	3
1.5 Internet of Things	3
1.6 Applications of IoT	3
1.6.1 Traffic monitoring.....	3
1.6.2 Agriculture	4
1.6.3 Education.....	4
1.6.4 Government.....	4
1.6.5 Wearables.....	4
1.6.6 Health.....	5
1.7 Remote health monitoring systems	5
1.8 Advantages of remote monitoring system	6
1.8.1 Improve the quality of healthcare.....	6
1.8.2 Reduce costs.....	6
1.8.3 Reduce burden in health services.....	6
1.8.4 Patient engagement.....	6
1.8.5 Chronic condition management.....	6
1.9 Remote healthcare Challenges and issues.....	7
1.9.1 Security and privacy.....	7
1.9.2 Accuracy of data.....	7
1.9.3 Training and adaptability.....	7

1.9.4 Sensors and wearable devices.....	7
1.9.5 Connecting data to providers.....	7
1.10 Conclusion.....	8

Chapter II: IoT based healthcare system

Introduction.....	9
2.1 Project description	9
2.2 Data Collection Units	9
2.2.1 Heart Beat Sensor.....	9
2.2.2 LM35 Temperature Sensor.....	10
2.2.3 ECG (electrocardiogram) sensor (AD8232).....	10
2.2.4 DHT11 Sensor.....	11
2.3 Control Units.....	12
2.3.1 ARDUINO UNO.....	12
2.3.2 NodeMCU Esp-8266 Module (Wi-Fi Module).....	13
2.4 Control unit tasks	13
2.4.1 Monitoring Data Collection.....	14
2.4.2 Monitoring Data Upload to Server.....	14
2.4.3 Monitoring using ThingSpeak server.....	14
2.5 Conclusion.....	14

Chapter III: System design and configuration

Introduction.....	15
3.1 Overall design of the system	15
3.2 Data aquisition of systems	16
3.3 Data transfer of the system	16
3.3.2 What is the Internet of Things?	17
3.3.3 Web server vs Web client.....	17
3.3.4 Arduino Web server.....	17
3.3.5 Web requests.....	18
3.3.6 ESP8266 AT Commands.....	18
3.3.7 Setting the mode.....	18
3.3.8 Connecting to wifi.....	18
3.3.9 Enebling the connections.....	19

3.3.10 Sending and receiving data.....	19
3.3.11 Setting up a global server.....	19
3.4 Thingspeak for IoT	19
3.4.1 Configuring ThingSpeak to record Patient Data online.....	20
3.5 If This Then That ‘IFTTT’ for IoT.....	23
3.5.1 Configuring IFTTT for triggering Mail/SMS based on ThingSpeak Values.....	23
3.6 Conclusion.....	31

Chapter IV: System design and configuration

Introduction	33
4.1 Circuit diagram.....	33
4.2 Hardware description.....	33
4.2.1 Interfacing ESP8266 with Arduino Uno.....	35
4.2.2 Interfacing ECG sensor with Arduino Uno.....	36
4.2.2.1 Pin connections with Arduino.....	36
4.2.2.2 AD8232 ECG sensor placement on body.....	37
4.2.3 Pulse sensor interfacing with Arduino.....	37
4.2.3.1 Pinout.....	37
4.2.3.2 Sensor placement on body.....	38
4.2.4 DHT11 sensor interfacing with Arduino.....	38
4.2.5 LM35 sensor with Arduino.....	39
4.2.6 Push button interfacing.....	39
4.3 System setup.....	39
4.4 Arduino IDE Code/Program.....	40
4.4.1 Explanation of the important part of the program code.....	40
4.5 Results and Discussion.....	42
4.5.1 Results.....	42
4.5.2 Discussion.....	45
4.6 Future enhancements.....	46
4.7 Conclusion.....	46

General Conclusion

REFERENCES

Appendix –A-

Appendix –B-

List of figures

Chapter I: Remote Healthcare Technology

<i>Fig. 1.1: Some examples for IoT applications.....</i>	<i>3</i>
<i>Fig. 1.2: Overview of remote healthcare system.....</i>	<i>5</i>

Chapter II: IoT based healthcare system

<i>Fig. 2.1: Pulse sensor.....</i>	<i>9</i>
<i>Fig. 2.2: LM-35 temperature sensor.....</i>	<i>9</i>
<i>Fig. 2.3: ECG sensor.....</i>	<i>10</i>
<i>Fig. 2.4: DHT11 sensor.....</i>	<i>11</i>
<i>Fig. 2.5: Arduino UNO board</i>	<i>12</i>
<i>Fig. 2.6: ESP-12 pin configuration</i>	<i>13</i>
<i>Fig. 2.7: ESP-01 pin configuration.....</i>	<i>13</i>

Chapter III: System design and configuration

<i>Fig. 3.1 : Block diagram of the system.....</i>	<i>16</i>
<i>Fig. 3.2 : Home page of thingspeak.com</i>	<i>20</i>
<i>Fig. 3.3: Channels menu in thingspeak.com.....</i>	<i>21</i>
<i>Fig. 3.4: The fields of the channel in thingspeak.com.....</i>	<i>21</i>
<i>Fig. 3.5: Channel sharing settings.....</i>	<i>22</i>
<i>Fig. 3.6: The doctor's access to patient's channel.....</i>	<i>22</i>
<i>Fig. 3.7: ThingHTTP menu in thingspeak.com.....</i>	<i>23</i>
<i>Fig. 3.8: IFTTT service</i>	<i>24</i>
<i>Fig. 3.9: Webhooks service in IFTTT</i>	<i>24</i>

<i>Fig. 3.10: Webhooks documentation.....</i>	25
<i>Fig. 3.11: Webhooks applet in IFTTT</i>	25
<i>Fig. 3.12: Webhooks applet triggering in IFTTT</i>	26
<i>Fig. 3.13: Applet menu</i>	26
<i>Fig. 3.14: Google Sheets service in IFTTT</i>	27
<i>Fig. 3.15: Google Sheets service configuration in IFTTT.....</i>	27
<i>Fig. 3.16: The created applet</i>	28
<i>Fig. 3.17: Email service in IFTTT</i>	28
<i>Fig. 3.18: Email service configuration in IFTTT</i>	29
<i>Fig. 3.19: My applets</i>	29
<i>Fig. 3.20: ThingHTTP menu in thingspeak.com</i>	30
<i>Fig. 3.21: ThingHTTP configuration</i>	30
<i>Fig. 3.22: React configuration in thingspeak.com</i>	31
<i>Fig. 3.23: React configuration for panic in thingspeak.com</i>	31

Chapter IV: System design and configuration

<i>Fig. 4.1: Overall designed circuit</i>	33
<i>Fig. 4.2: ESP8266 pinout</i>	34
<i>Fig. 4.3: ECG probe placement</i>	35
<i>Fig. 4.4: ECG placement on forearm</i>	36
<i>Fig. 4.5: Backside of the pulse sensor</i>	36
<i>Fig. 4.6: DHT11 pinout.....</i>	37
<i>Fig. 4.7: LM35 pinout</i>	37
<i>Fig. 4.8: Pushbutton</i>	38

<i>Fig. 4.9: System setup</i>	38
<i>Fig. 4.10: Program code part 1</i>	39
<i>Fig. 4.11: Program code part 2</i>	39
<i>Fig. 4.12: Program code part 3</i>	40
<i>Fig. 4.13: Program code part 4</i>	40
<i>Fig. 4.14: Program code part 5</i>	41
<i>Fig. 4.15: Program code part 6</i>	41
<i>Fig. 4.16: Program code part 7</i>	42
<i>Fig. 4.17: The header of code</i>	42
<i>Fig. 4.18: Variable part of the code</i>	43
<i>Fig. 4.19: Setup part of the program code</i>	44
<i>Fig. 4.20: Loop part of the program</i>	44
<i>Fig. 4.21: Humidity and ECG function</i>	45
<i>Fig. 4.22: Pulse rate and temperature function</i>	45
<i>Fig. 4.23: Body temperature and panic button function</i>	45
<i>Fig. 4.24: Sample of patient's vital data</i>	46
<i>Fig. 4.25: ECG plot</i>	46
<i>Fig. 4.26: BPM result</i>	46
<i>Fig. 4.27: Temperature and humidity data</i>	47
<i>Fig. 4.28: Panic pulse</i>	47
<i>Fig. 4.29: Body temperature</i>	47
<i>Fig. 4.30: Google Sheet data arrival</i>	48
<i>Fig. 4.31: Arrival of emergency email</i>	48

General introduction

Recently, we are witnessing a huge technological improvement in all areas and services, technology has become a part of people's lives, affecting their daily activities, work, communications, and multiple services.

This advance in technology plays a crucial role in the healthcare field as well, we can notice that in hospitals, pharmaceutical industries, medical laboratories, and also in personal health tools like a thermometer or blood pressure monitor and so many medical devices that are still in development.

Digital technology is the application of technology on systems, devices, and tools that deal with processing data and digital pieces of information.

Using the combination of digital technology and healthcare services, a remote healthcare system could be developed, remote healthcare services are developed systems using digital technologies and electronic medical devices (sensors and microcontrollers), the main aim of these systems is to enable healthcare services remotely so that enhance the patient monitoring at home, especially for people that have chronic diseases and people with special healthcare needs.

In this project, an IoT based Patient Health Monitoring System using Arduino UNO device is proposed to collect the required parameters like temperature, heartbeat and ECG (electrocardiogram), then evaluate the data obtained from the IoT devices by doctors. To accomplish this, the system involves many sensors to screen these fundamental signs.

Data generated by the sensors are processed by Arduino UNO. ESP8266 provides ability to embed Wi-Fi capabilities within other systems. It offers a complete and self-contained Wi-Fi networking solution. It can be used to host the application or to offload Wi-Fi networking functions from another application processor. The data generated from Arduino is available in the IoT website ThinkSpeak.com with the use of Wi-Fi module. The IoT platform used in this project is ThingSpeak. It is an open-source Internet of Things (IoT) application to store and retrieve data from things using the HTTP protocol over the Internet.

In the first chapter we will introduce the remote healthcare technology and IoT applications.

The chapter two basically represents the electronic devices used in this system.

The chapter three consists of configurations steps of ThingSpeak web server and the needed web services.

Then chapter four represents detailed procedures of system realization and its performance in real-time within the reachable results.

Chapter one

Remote Healthcare Technology

Nowadays, advanced information technologies and communication devices facilitate complex medical problems and enable doctors and health providers to receive patient's health information on their smart phones, laptops or personal computers and perform appropriate actions, feedback to the patient on the basis of collected data.

These new concepts and terminologies are presented below.

1.1 Medical Monitoring systems:

Monitoring is the continuous medical proctoring of the patient, it is the frequent tasks performed by health service providers for patients in hospitals or health facilities , this performance could be in multiple patterns depending on patient illness and health difficulties, to achieve that accurately doctors should be enabled to examine their patient health status continuously and repetitively especially for patients with chronic diseases, this concern provoked researchers to develop remote monitoring using computer technologies and information science linked to medicine.

1.2 Remote health monitoring systems (RHMS):

Systems that allow the medical observation remotely, so that patient could be monitored at home; these systems should be able to transmit high-quality data to health providers and enable them to perform appropriate feedback, remote monitoring systems play an important role when patient and doctor are at a distance and such systems are capable of reducing costs and enhance the quality of healthcare delivery.

1.3 Mobile health monitoring systems (MHMS):

Based on smartphones, personal digital assistants and it is the application of mobile computing technologies for improving communication between patients and doctors; It enables the delivery of accurate medical information anytime anywhere by means of mobile devices.

1.4 Wearable health monitoring systems (WHMS):

Applying wearable smart devices or biosensors, it is used basically for essential signs monitoring, these systems use electronic devices that are worn on a person in order to accurately relay important medical and biological data to a database.

1.5 Internet of Things:

The Internet of Things (IoT) is a universal concept of being able to control physical objects from a distance and exchange data connecting them with other devices and systems over the internet.

The network tools are used to connect any object through sensors and receive data from the real world objects around us, and then use this data with aid of the Internet for processing, analyzing, and utilizing it properly. The following figure shows some examples for IoT applications:

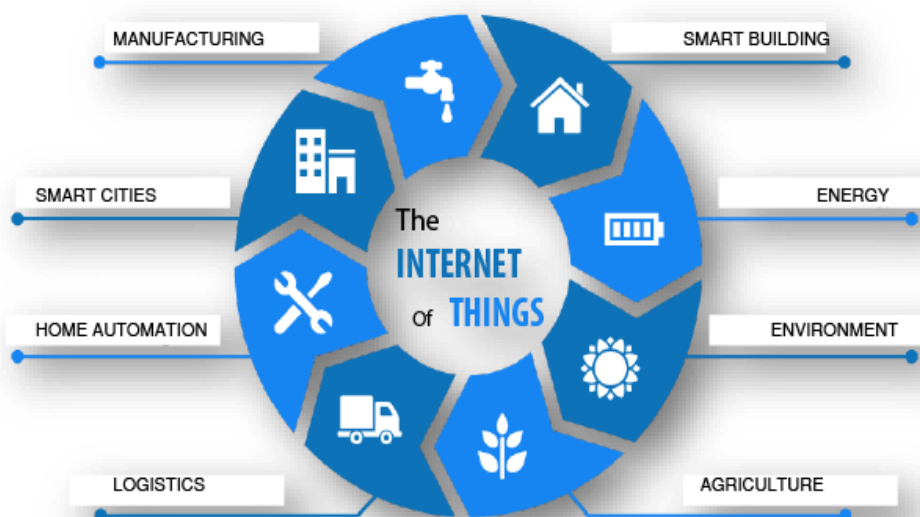


Fig.1.1: Some examples for IoT applications.[1]

1.6 Applications of IoT:

1.6.1 Traffic monitoring:

Recently, traffic monitoring becomes a difficulty in countries due to the population increase, the rise in the number of cars, and the cities expand.

Researchers nowadays design and develop systems that control vehicular traffic using sensing technology and the internet of things platforms; these systems enable users to verify all conditions of different routes, time estimations, and distances.

1.6.2 Agriculture:

As the world is progressing, agriculture needs to be developed as well and to be combined with technology; the internet of things perform multiple smart systems and platforms in this field , one of them, for example, is greenhouses that enhances the yield of crops by monitoring the ecological conditions.

1.6.3 Education:

The internet of things provides multiple applications and platforms for education, it gives education specialists new tools to optimize classwork, improve the efficiency of the learning method, and connect with learners better.

1.6.4 Government:

The government are responsible for providing services to individuals and protect their rights, and for the management and coordination in all areas; to perform that government nowadays uses the applications of the internet of things to better connect citizens and public entities, improve operational performance and maintenance, improve safety achievement, and simplifies several factors examination such as population growth, water supply , transportation patterns, social services, and land use.

1.6.5 Wearables:

Wearables are intelligent devices that can be worn as external accessories, they can be in used according to their function in health, daily life activities, sports, safety, and other areas; IoT enables wearables to collect and organize data and information about users in an efficient and accurate way and also helps them in making smarter decisions and make those devices more specialized and practical applications.

1.6.6 Health:

IoT develops for healthcare service multiple applications and systems, these applications expand professionalism, increase accuracy and quantity of medical data and also increase the potential of finding solutions and discovery of previously unknown issues.

1.7 Remote health monitoring systems:

Health is a word to describe a personal physical and mental situation, it is an essential interest in human daily lives and it is an important component of the population structure that affects the process of developing; and that prompt medical services and researchers to improve health population referred to as healthcare service.

Healthcare services have rapidly evolved with the advance of technology and have the potential to change the way healthcare is delivered; the core of today's healthcare deliveries is the smart monitoring systems using advanced information technologies and automatic devices, this project is highly focused on the smart and wireless monitoring system.

This project aims to design a remote healthcare monitoring system based on the internet of things interfaces to enhance patient health monitoring at a remote location (home) especially provided for patients with chronic diseases, this system should be able to collect data which is patient's essential parameters like temperature, blood pressure, heartbeats, and other data, then present these parameters in a valuable way to health providers and allow them to analyze, and perform appropriate solutions as a feedback to the patients using the suitable electronic sensors, microcontrollers, wifi modules, and internet platforms. The figure below shows an overview of remote healthcare system

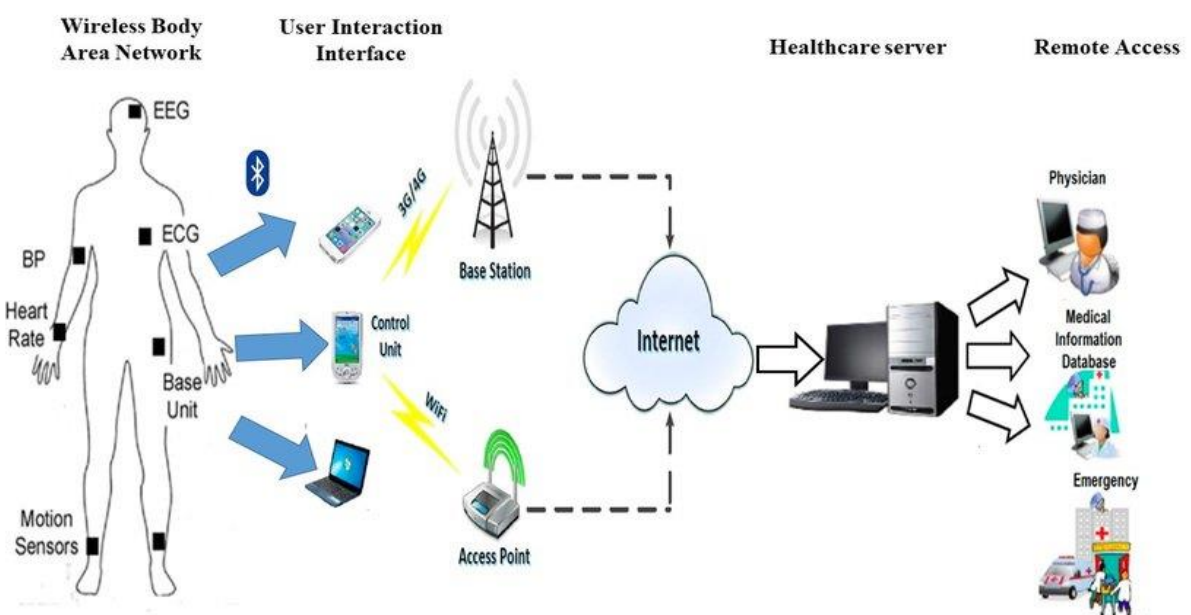


Fig 2.2: Overview of remote healthcare system.[2]

1.8 Advantages of remote monitoring system:**1.8.1 Improve the quality of healthcare:**

The physician's principle goal is to improve health outcomes for their patients, using RPM, (Remote Patient Monitoring), doctors can receive more data about their patients which offers them the opportunity to better analyze, understand individual patient's needs and provide proper solutions , RPM also makes care more convenient and accessible for patients which is a very important factor for improving healthcare quality.

1.8.2 Reduce costs:

RPM can limit or delay many costly events such as emergency situations and hospitalizations by helping patient keep measurements and maintain good health.

1.8.3 Reduce burden in health services:

Since RPM provide data collection and analysis remotely, patient can get care without going to clinics and that facilitates work to health providers and maintains clinical settings organized. If we are dealing with a contagious disease, RPM can make patients healthier and can also keep them away from hospitals to avoid any infections.

1.8.4 Patient engagement:

Since RPM enables patients to remain engaged with their care providers continuously and at a distance, they can get aware of their health states and also get informed advice and appropriate solutions for their issues so that they maintain balanced health status, it makes them feel supported along the way.

1.8.5 Chronic condition management:

Patients with chronic diseases such as diabetes and hypertension need frequent measurements and repetitive treatments, Remote patient monitoring can improve the management of chronic conditions by the continuous examination of parameters such as blood glucose in diabetes and blood pressure in hypertension to perform full control on the health situation.

1.9 Remote healthcare Challenges and issues:

1.9.1 Security and privacy:

Since RPM systems manage data over information technologies and networks, they are subject to security issues. The remote exchange of parameters makes the risk of hacking or stealing patients' data; RPM must ensure the confidentiality of information by the use of some protocols and applications which make data only accessed by people who have authorization.

1.9.2 Accuracy of data:

The most significant factor in the achievement of RPM systems is the accuracy of transferred data, this may need continuous affirmations and calibrations of the system to perform precision and avoid maximum errors.

1.9.3 Training and adaptability:

Since the RPM system uses advanced technologies and electronic devices, some of these devices and sensors are difficult to use, and also for some people internet platforms are hard to handle which require some training to the patients, to increase acceptance and flexibility of the system.

1.9.4 Sensors and wearable devices:

Regular transmission of the patient health parameters require the use of electronic devices, these devices should be comfortable and secure to use, suitable, convenient, and also must be examined from time to time.

1.9.5 Connecting data to providers:

RPM system depends heavily on the availability of network and connectivity of data, especially when a large data packet is exchanged over the network and since not all consumers have a continual and fast network, this makes an obstacle for this system, network providers should ensure a continuous network for uninterrupted communication.

1.10 Conclusion:

Remote monitoring systems are the advanced and technological manner of continuous and repeated health observation, home medical care are effortless, cost less, and make both doctors and patients comfortable. These systems are going to continue developing in terms of the advance of technology by developing new devices and applications, with increasing awareness and acceptability of patients.

Chapter two

IoT Based Healthcare System

Capturing and sharing of vital data of the network connected devices through secure service layer is what defines IoT. In simple terms, Internet of Things (IoT) can be defined as the wireless network of devices which are connected to each other to share information and data in order to communicate and produce new information so as to record and analyze it for future use.

The Internet of things in the field of healthcare plays a major role in providing ease to patients and doctors. It consists of a system that communicates between network connected systems, applications and devices that can help patients and doctors to monitor, track and record vital data and medical information. Some of the devices include smart meters, wearable health bands, fitness shoes; RFID (Radio Frequency Identification) based smart watches and smart video cameras. Also, applications for smart phones also help in keeping a medical record with real time alert and emergency services.

2.1 Data Collection Units :

For data collection, we are using the following sensors:

2.1.1 Heart Beat Sensor:

The heart beat sensor is a plug and play heart rate sensor for Arduino contains of 3 pins: (GND) Ground Pin, VCC pin (3V to 5V), and the signal pin S. It is an electronic device that is used to measure the heart rate i.e. speed of the heartbeat. Heart Rate can be monitored in two ways: one way is to manually check the pulse either at wrists or neck and the other way is to use a Heartbeat Sensor. It sips power with just 4mA current draws at 5V. For this, it is great for mobile application [3]. The figure 2.1 shows Pulse sensor:



Fig. 2.1: Pulse sensor. [3]

2.1.2 LM35 Temperature Sensor:

LM35 is a temperature measuring device having 3 pins VCC, GND, and an analog output voltage (OUT) proportional to the temperature. It provides output voltage in Centigrade (Celsius). It does not require any external calibration circuitry. The sensitivity of LM35 is 10 mV/degree Celsius. As temperature increases, output voltage also increases. E.g. 250 mV means 25°C. It is a 3-terminal sensor used to measure surrounding temperature ranging from -55 °C to 150 °C. LM35 gives temperature output which is more precise than thermistor output. The following figure 2.2 shows LM35 sensor:

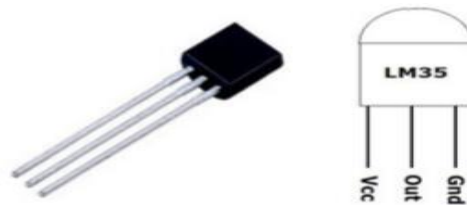
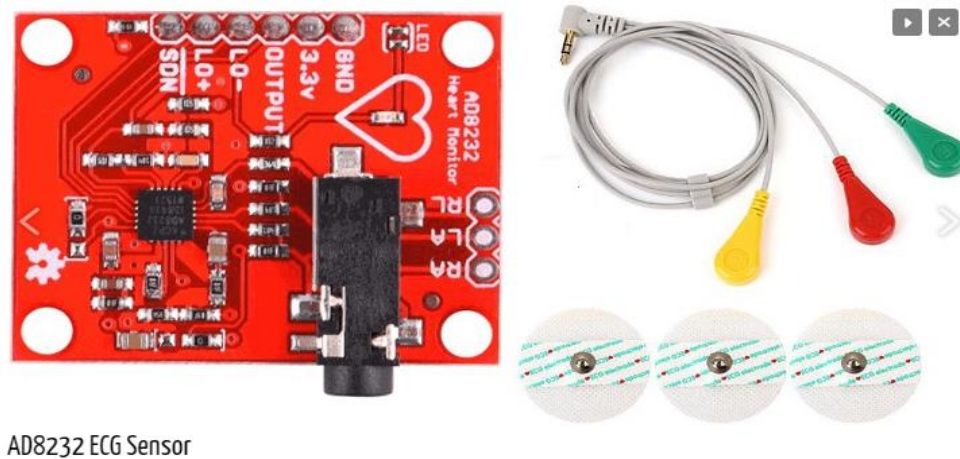


Fig. 2.2: LM-35 temperature sensor.[3]

2.1.3 ECG (electrocardiogram) sensor (AD8232):

ECG records the electrical activity generated by heart muscle depolarization, which propagate in pulsating electrical waves towards the skin. Although the electricity amount is in fact very small, it can be picked up reliably with ECG electrodes attached to the skin. The full ECG setup comprises at least four electrodes which are placed on the chest or at the four extremities according to standard nomenclature (RA = right arm; LA = left arm; RL = right leg; LL = left leg). Of course, variations of this setup exist to allow more flexible and less intrusive recordings, for example, by attaching the electrodes to the forearms and legs. ECG electrodes are typically wet sensors, requiring the use of a conductive gel to increase conductivity between skin and electrodes. It contains 6 pins: GND (Ground), VCC, OUTPUT (Analog output signal) , LO- (Leads-off Detect -) , LO+ (Leads-off Detect +), and SDN (Shutdown) . the figure 2.3 shows the ECG sensor:

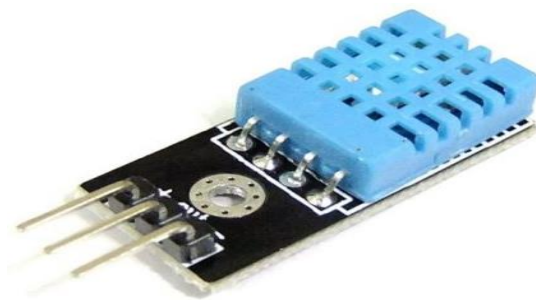


AD8232 ECG Sensor

Fig. 2.3 : ECG sensor[4].

2.1.4 DHT11 Sensor:

DHT11 is humidity and temperature sensor. It has 3 pins Ground, VCC and the data pin. It calculates both humidity and temperature is one sensor. This is the advantage of DHT11. It has the high reliability and excellent long term stability. The supply voltage of this sensor is 3 V – 5.5 V [5]. Advantages of this are it gives fast response, small size and low power consumption. Humidity describes the water content so it will use to measure the sweat of the body. The sweat of the body is measured in percentage. When the value of this sweat measured goes above the 70% then it will give alert to user. At the same time it will give notification to user on application as drink water alert. In this sensor it has 2 electrodes and moisture holding substrate between these electrodes. As the humidity changes, the conductivity of substrate changes and hence the resistance between the electrodes changes. Humidity accuracy is +/-5% and it works between 10%-50%.the the figure 2.4 below shows DHT11 sensor :

*Fig. 2.4: DHT11 sensor [6].*

2.2 Control Units :

2.2.1 ARDUINO UNO :

The Arduino based IoT (Internet of Things) device has the following circuit connections in figure 2.6. The Arduino UNO is ATmega328 based microcontroller board. It is one of the most popular prototyping boards. The board comes with built-in Arduino boot loader. It has 14 GPIO (General-Purpose Input Output) pins, 6 PWM (Pulse Width Modulation) pins, 6 Analog inputs and on board UART (Universal Serial Asynchronous Receiver Transmitter), SPI and TWI interfaces an onboard resonator, a reset button, and holes for mounting pin headers. While programming the board, it can be connected to the PC using USB port and the board can runs on USB power. The Arduino UNO has 32 Kb Flash memory, 1 Kb EEPROM and 2 Kb SRAM. The board can be connected to different Arduino Shields for connectivity with Ethernet, Bluetooth, Wi-Fi, Zigbee or Cellular network and it can be connected to most of the IoT platforms.

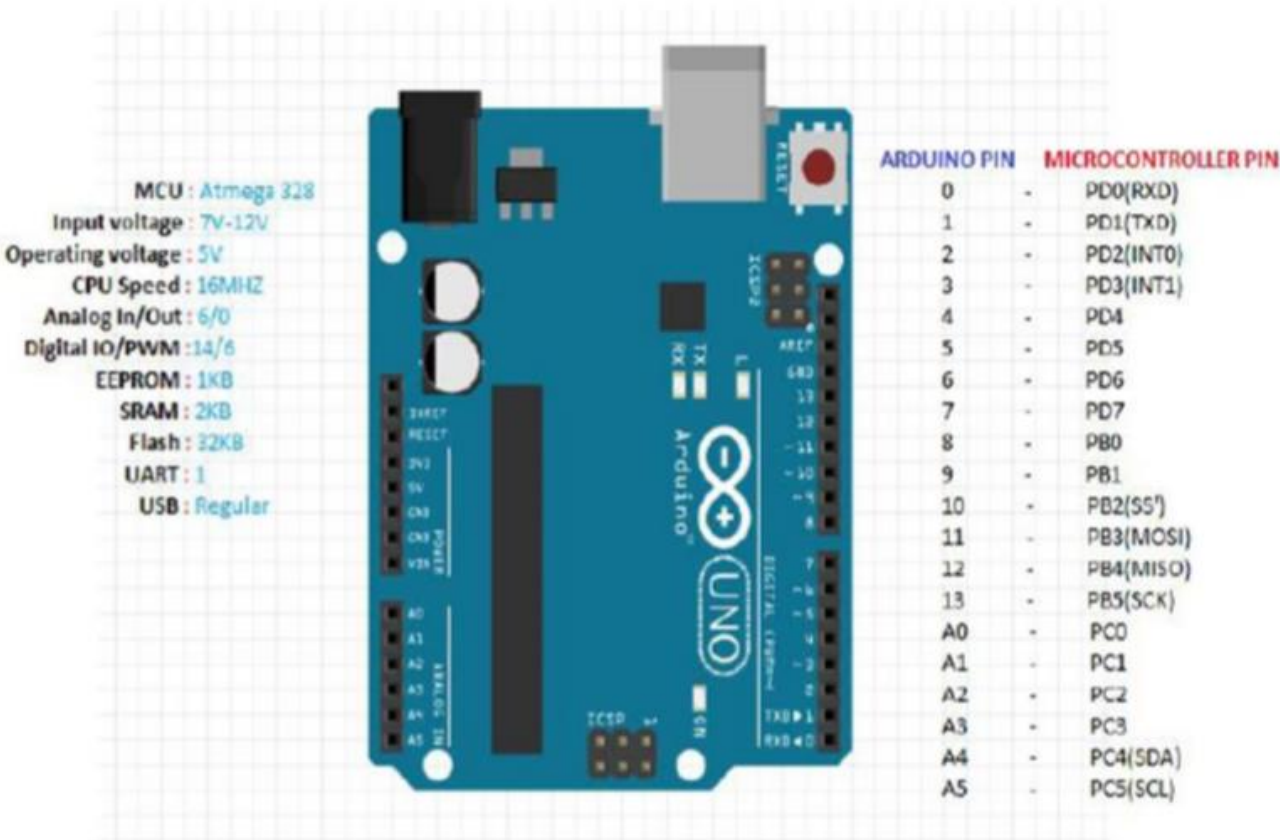


Fig. 2.5: Arduino UNO board. [3]

2.2.2 NodeMCU Esp-8266 Module (Wi-Fi Module):

NodeMCU is an open source IoT (internet of things) platform. It includes firmware which runs on the ESP8266 Wi-Fi SoC (System-on-Chip) from Espressif Systems, and hardware which is based on the ESP-12 module [4]. The term "NodeMCU" by default refers to the firmware rather than the development kits. The firmware uses the Lua scripting language. It is based on the eLua project, and built on the Espressif Non-OS SDK (Software Development Key) for ESP8266. It uses many open source projects, such as lua-cjson and SPIFFS.

ESP8266 Wi-Fi Module is used to connect the Arduino board with a Wi-Fi router, so it can access the cloud. It is a self contained SoC with integrated TCP/IP protocol stack that can access to a Wi-Fi network. The ESP8266 is capable of either hosting an application or off loading all Wi-Fi networking functions from another application processor. Each ESP8266 module comes preprogrammed with an AT command set firmware. The module comes available in two models - ESP-01 and ESP12. ESP-12 has 16 pins available for interfacing while ESP01 has only 8 pins available for use. The ESP-12 has the following pin configuration as shown in the Appendix-A-

In this project we are using the ESP-01 model of the chip it's pin configuration are presented in Appendix-A- It is developed by a third-party manufacturer called AI-Thinker. It has an onboard MCU (Microcontroller Unit), which allows users to control I/O digital pins directly via the Arduino IDE.

The RESET and VCC pins of the module are connected to the 3.3 V DC from Arduino while Ground pin is connected to the common ground. The Tx and Rx pins of the module are connected to the 2 and 3 pins of the Arduino UNO.

2.3 Control unit tasks :

The control unit performs three tasks:

2.3.1 Monitoring Data Collection:

For this task ESP8266-01 is being used, the values from the sensors are all being received on Analog pins and are controlled using GPIO pins.

2.3.2 Monitoring Data Upload to Server:

Again for this task ESP8266 Wi-Fi module is used. The values received from all the sensors are being uploaded to the server using the concept of IoT (Internet of Things) in this the device is connected to Wi-Fi with the help of esp8266 WiFi module and the program written in it.

2.3.3 Monitoring using ThingSpeak server:

The currently used server is ThingSpeak, which is an open-source Internet of Things (IoT) application and API to store and retrieve data from things using the HTTP protocol over the Internet or via a Local Area Network. ThingSpeak enables the creation of sensor logging applications, location tracking applications, and a social network of things with status updates. The data collected is uploaded to the server using internet on a public channel called 'HMS'. The data is represented in the form of charts and a complete database of all the charts (sensors) can be obtained [3].

2.4 CONCLUSION :

In general IoT based health care platform which connects with smart sensors attach with physical body for health monitoring for daily checkup, provides the necessary tools to monitor, analyze and control the system remotely. For this purpose modern acquisition systems are needed and due to the importance of observing medical patient, continuous remote monitoring is also important.

Chapter three

System Design and Configuration

IoT (Internet of things) now days became an emerging topic in technological field. It is based on interconnecting of devices with each other over the internet network, It facilitates the digitalization and automation of the manually operating devices [3]. One of the best applications of IoT technology are in fields that are intensely practical, and one of the major field is health care monitoring system. The lack of proper health monitoring cause a serious health issues for a lots of patients .that why health experts are taking advantage of IOT smart devices to keep an eye on their patients.

Here in this project, we will make an IoT based Health Monitoring System which records the patient Electrocardiogram (ECG), heartbeat rate (BPM) and body temperature, humidity, and temperature of the room, and also send an email/SMS alert whenever those readings go beyond critical values. Pulse rate, and body temperature readings are recorded over ThingSpeak and Google sheets therefore patient health can be monitored from anywhere in the world over the internet. A panic will also be attached so that patients can press it on an emergency to send emails/SMS to their relatives. This system can be used to promote basic nursing care in the hospital environment by improving the quality of medicare and patient safety. Rural area is lack behind the proper patient monitoring system. So, remote monitoring and guidance awareness by sharing information in an authenticated manner are the main objectives of our project.

3.1 Overall design of the system :

The system overall diagram is shown in figure 3.1. To achieve the function of the design and convenient man-machine interaction, the system is divided into two main parts: data acquisition and data transfer and visualization. The data acquisition system consists of Arduino UNO [2.4.1] and all the sensors: PULSE rate[chp2, sec1.1], LM35[chp2, 1.2], ECG (AD8232)[chp2, 1.3], DHT11[chp2, 1.4].

The data transfer to the web site, is performed through ESP-01 (ESP8266) [2.2.2], this Wi-Fi module is connected to the Arduino UNO to upload the same data to ThingSpeak Server as it finds the Wi-Fi Access Point. For displaying and monitoring the data are uploaded to ThingSpeak server to show the received information.

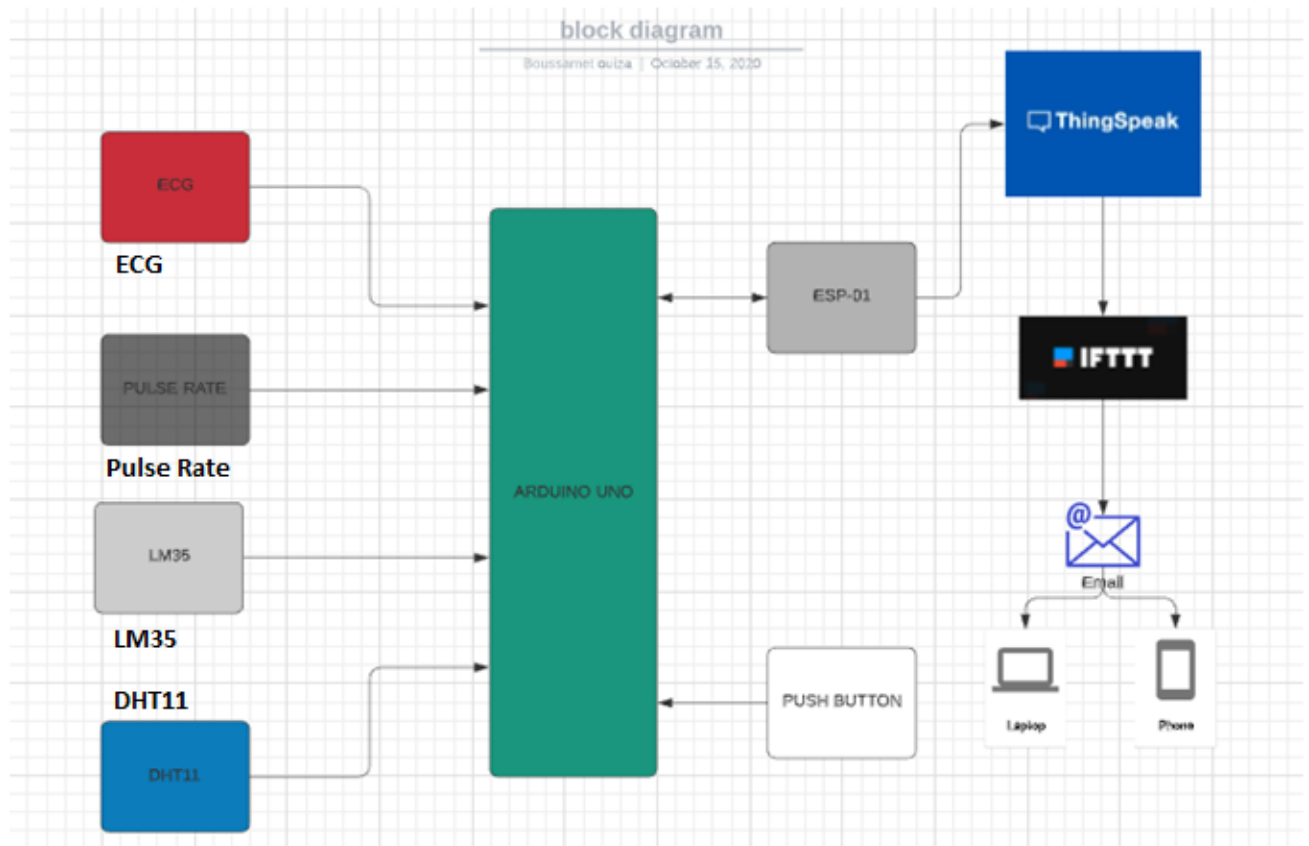


Fig. 3.1: Diagram of the system.

3.2 Data aquisition of systems :

Data acquisition tends to use sensors to convert physical parameters into digital signals to be accessible to software programs. For data acquisition of this system several sensors are being used such as body temperature sensor, pulse sensor, ECG, DHT11.

A push-button is connected to the system in order to send an emergency mail to the doctor through ThingSpeak and IFTTT (If This Then That) .The physiological parameters are obtained, processed using ARDUINO UNO microcontroller.

3.4 Data transfer of the system :

With the advent of smart systems powered by data and artificial intelligence, it seems like the predictions regarding the IoT industry are right after all[8]. Internet of Things has been successfully changing residential technology and manufacturing to the point of being called the fourth industrial revolution.

3.4.1 What is the Internet of Things?:

The Internet of Things, or IoT, refers to the billions of physical devices around the world that are now connected to the internet, all collecting and sharing data. Thanks to the arrival of super-cheap computer chips and the ubiquity of wireless networks, it's possible to turn anything, from something as small as a pill to something as big as an aeroplane, into a part of the IoT. Connecting up all these different objects and adding sensors to them adds a level of digital intelligence to devices that would be otherwise dumb, enabling them to communicate real-time data without involving a human being. The Internet of Things is making the fabric of the world around us more smarter and more responsive, merging the digital and physical universes.[9]

3.4.2 Web server vs. Web client:

A web server is either software, hardware, or a combination of both that contains files needed to process and deliver web pages. A web client is simply any device that can send an HTTP/web request to a web server. HTTP (Hypertext Transfer Protocol) is a unique protocol that a web server and web client use to communicate. Example: when visiting www.thingspeak.com. The website URL is entered to the web browser. After a few seconds, the thingspeak homepage appears, the computer should be connected to the internet. In this example, the computer is a web client. It sends a web request using a web browser application, i.e., Chrome or Firefox. The web browser sends the request to the web server that hosts thingspeak, which then returns the data needed to display. A web server that hosts a website is usually a purpose-built computer that stores a massive amount of data.

3.4.3 Arduino Web server :

An Arduino connected with an ESP-8266 module is enough for a simple web server. The Arduino web server gives the ability to store a web page and extends control over the sensors and other connected devices [8]. Things like reading sensor values and toggling relay switches can be done anywhere via a WiFi connection. Additionally, a web server that can be accessed anywhere via the internet is called a global server. Meanwhile, a web server that can only be visited in the Local Area Network (LAN) is called a local server.

3.4.4 Web requests :

To obtain data from web servers, web clients use HTTP like HTTP GET and HTTP POST :

- HTTP GET is a web request that retrieves data from a web browser. It does not change anything on the server. It just fetches the data from it.
- HTTP POST is a web request that transmits data to the server. It adds something new to the server.

A typical example of a GET request is the simple browsing of a website. On the other hand, POST requests are used in typing text into a web page, for instance, a username and password.

3.4.5 ESP8266 AT Commands :

The ESP8266 AT Commands allow users to perform operations like testing the connection, setting the mode of operation, connecting to WiFi, determining the IP address, etc.

3.4.6 Setting the mode :

The mode of operation is setting by typing the following AT command: AT+CWMODE=1

The ESP8266-01 has three modes of operation: (1) Station (STA); (2) Access Point (AP); and (3) Both :

- In the first mode, the WiFi module acts as a Station (STA). The module gains the ability to connect to an available WiFi network.
- In the second mode, the WiFi module acts as an Access Point (AP). The module acts as a WiFi network where devices like computers can connect to it.
- In the third mode, the WiFi module acts as both an AP and an STA.

To check what mode the ESP8266 is in, type in AT+CWMODE? . The response is going to be number 1, 2, or 3 which corresponds to the mode of operation.

3.4.7 Connecting to wifi :

To connect to a WiFi network, the following command is typed: AT+CWLAP= "SSID", "Password".

These are case sensitive, the exact WiFi Network's name and password must be typed. Also, there should be no spaces between the quotation marks and the comma. An OK response means that it is successfully connected. The verification of the connection is by sending this AT command: AT+CIFSR.

3.4.8 Enabling the connections :

In our project, the ESP-01 should be set to support multiple connections since it is used as a server. To do that, this command is entered in the serial monitor: AT+CIPMUX=1

Furthermore, the server starts using the command: AT+CIPSERVER=1,80.

The first number indicates the port status. A value of 0 means it is closed while a value of 1 means it is opened. On the other hand, the second number indicates the port number. Port 80 is the default port number for the HTTP protocol, which is also used for HTML pages.

3.4.9 Sending and receiving data :

To send a GET request, the ESP-01's IP address is sent to the computer's web browser. This is going to send a response to the serial monitor. The response contains several useful informations like the details of the file to be retrieved, the name of the browser used for the request, the operating system, and so on.

3.4.10 Setting up a global server :

In this section, ESP-01 should be connected to the internet in order to communicate with a global server that displays the data on a web page that can be accessed anywhere.

3.5 Thingspeak for IoT :

Sensors, or things, sense data and typically act locally. ThingSpeak enables sensors, instruments, and websites to send data to the cloud where it is stored in either a private or a public channel. ThingSpeak stores data in private channels by default, but public channels can be used to share data with others. Once data is in a ThingSpeak channel, they can be analyzed and visualized, calculate new data, or interact with social media, web services, and other devices. [10]

3.5.1 Configuring ThingSpeak to record Patient Data online :

To monitor the data and control the system over the internet ThingSpeak provides a very good tool for IoT based projects like channels and web-pages, To analyze and visualize the data and acts by triggering a reaction.

In this project ThingSpeak is used to monitor patient heart beat, ECG and body temperature, also the humidity and temperature of the room online using internet.

IFTTT platform is connected to ThingSpeak to email/message service so that an alert message can be sent whenever the patient is in a critical state. Configuration steps of the user interface:

Step 1: user which is the patient needs to create an account on ThingSpeak.com, then sign in and get start :

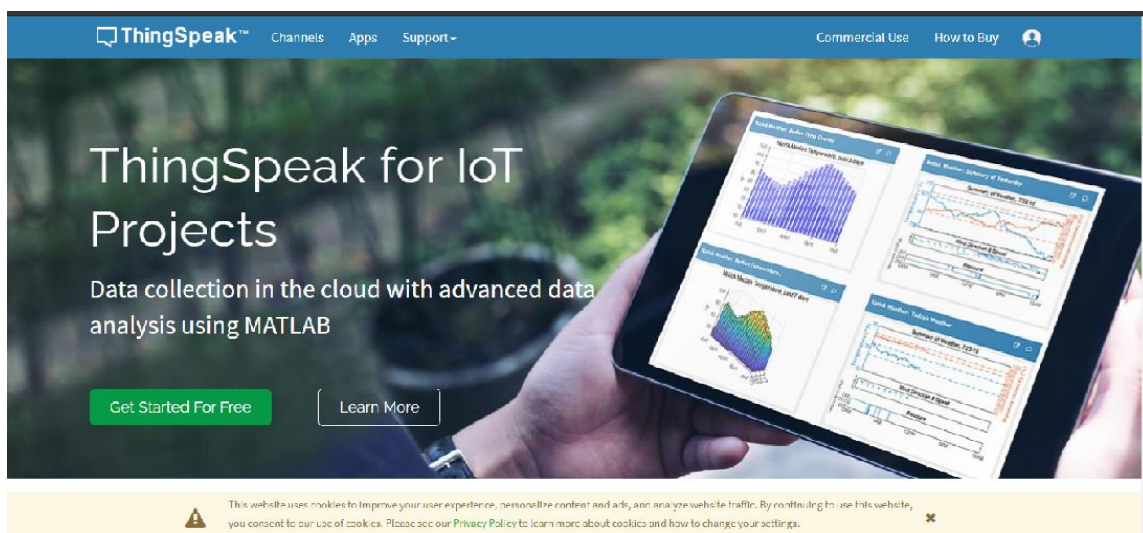


Fig. 3.2 : Home page of thingspeak.com.

Step 2:- In the channels menu a new channel is created:

My Channels

New Channel

Search by tag

Name	Created	Updated
iohelthcare	2020-06-07	2020-10-02 21:48
esp	2020-06-11	2020-06-11 01:30
Patient data monitoring	2020-10-03	2020-10-07 20:16

Help

Collect data in a ThingSpeak channel from a device, from another channel, or from the web.

Click **New Channel** to create a new ThingSpeak channel.

Click on the column headers of the table to sort by the entries in that column or click on a tag to show channels with that tag.

Learn to [create channels](#), explore and transform data.

Learn more about [ThingSpeak Channels](#).

Examples

- [Arduino](#)
- [Arduino MKR1000](#)
- [ESP8266](#)
- [Raspberry Pi](#)
- [Netduino Plus](#)

This website uses cookies to improve your user experience, personalize content and ads, and analyze website traffic. By continuing to use this website, you consent to our use of cookies. Please see our [Privacy Policy](#) to learn more about cookies and how to change your settings.

Fig. 3.3: Channels menu in thingspeak.com.

Step 3: for creating the channel, we have to fill the Name and then fill ‘Pulse Rate’, ‘Temperature’, ‘humidity’, ‘Pulse Rate’, ‘ECG’, ‘bodytemp’ and ‘Panic’ in Field 1, Field 2 to Field 6 labels, we tick the checkboxes for the Fields. Also we tick the check box for ‘Make Public’ option below in the form and finally we save the channel.

Channel Settings

Percentage complete: 30%

Channel ID: 1171782

Name: Patient data monitoring

Description:

Field 1: Temperature

Field 2: humidity

Field 3: pulse rate

Field 4: ECG

Field 5: bodytemp

Field 6: panic

Field 7:

Channel Settings

- **Percentage complete:** Calculated based on data entered into the various fields of a channel. Enter the name, description, location, URL, video, and tags to complete your channel.
- **Channel Name:** Enter a unique name for the ThingSpeak channel.
- **Description:** Enter a description of the ThingSpeak channel.
- **Field#:** Check the box to enable the field, and enter a field name. Each ThingSpeak channel can have up to 8 fields.
- **Metadata:** Enter information about channel data, including JSON, XML, or CSV data.
- **Tags:** Enter keywords that identify the channel. Separate tags with commas.
- **Link to External Site:** If you have a website that contains information about your ThingSpeak channel, specify the URL.
- **Show Channel Location:**
 - **Latitude:** Specify the latitude position in decimal degrees. For example, the latitude of the city of London is 51.5072.
 - **Longitude:** Specify the longitude position in decimal degrees. For example, the

This website uses cookies to improve your user experience, personalize content and ads, and analyze website traffic. By continuing to use this website, you consent to our use of cookies. Please see our [Privacy Policy](#) to learn more about cookies and how to change your settings.

Fig. 3.4 : The fields of the channel in thingspeak.com.

Step 4: the patient should record the doctor's email address in the channel sharing settings, in order to enable his doctor to check and monitor the parameters continuously, this can be provided for multiple doctors.

Channel ID: 1171782
Author: ouiza9
Access: Private

Private View Public View Channel Settings Sharing API Keys Data Import / Export

Channel Sharing Settings

Keep channel view private
 Share channel view with everyone
 Share channel view only with the following users:

Email Address

Email Address	Shared On	Delete
souhilareffaa@gmail.com	2020-11-12	<input type="button" value="X"/>

Help

ThingSpeak allows you to control who can view the data in your channel. Irrespective of the settings on this tab, reading data from or writing data to the fields of a channel requires the appropriate API key for the channel.

Channel Sharing Settings

- Keep channel view private:** Selecting this option keeps your channel private. Only you will be able to see the channel view.
- Share channel view with everyone:** Selecting this option makes the public view of your channel viewable by anyone browsing the ThingSpeak website.
- Share channel view only with the following users:** Selecting this option shares the private view of your channel only with specific ThingSpeak users.

Fig.3.5: Channel sharing settings.

Channels Shared With Me

Patient data monitoring

Channel ID: 1171782
Author: ouiza9

Fig.3.6: The doctor's access to patient's channel.

Step 5: we will use ThingHTTP application of the server to trigger the IFTTT applet for data entry to Google sheets and send email. ThingHTTP enables communication among devices, websites, and web services without having to implement the protocol on the device level. we specify actions in ThingHTTP, and then trigger it using other ThingSpeak apps such as React. To make New ThingHTTP, we will need URL for triggering which we will get from IFTTT.

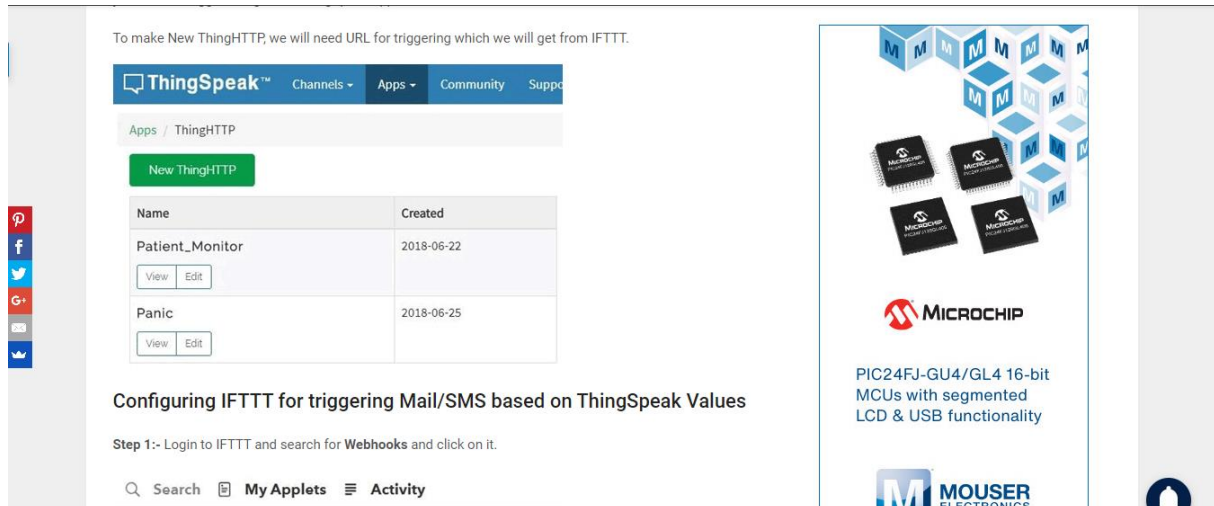


Fig. 3.7: ThingHTTP menu in thingspeak.com.

3.5.2 If This Then That ‘IFTTT’ for IoT :

If This Then That is a web-based service that allows users to create chains of conditional statements triggered by changes that occur within other web services such as Gmail, Facebook, Telegram, Instagram, or Pinterest. The service is offered in freeware, subscription, and enterprise versions.

The conditional chains are called applets. An applet may for example send an e-mail message if the user tweets using a hashtag, or copy a photo on Facebook to a user's archive if someone tags a user in a photo.

In addition to the web-based application, the service runs on iOS and Android. In February 2015, IFTTT renamed its original application to IF, and released a new suite of apps called Do, with which users can create shortcut applications and actions. As of 2015, IFTTT users created about 20 million recipes each day. All of the functionalities of the Do suite of apps have since been integrated into a redesigned IFTTT app. [11].

3.5.3 Configuring IFTTT for triggering Mail/SMS based on ThingSpeak Values:

Step 1: Login to IFTTT and search for Webhooks and click on it:

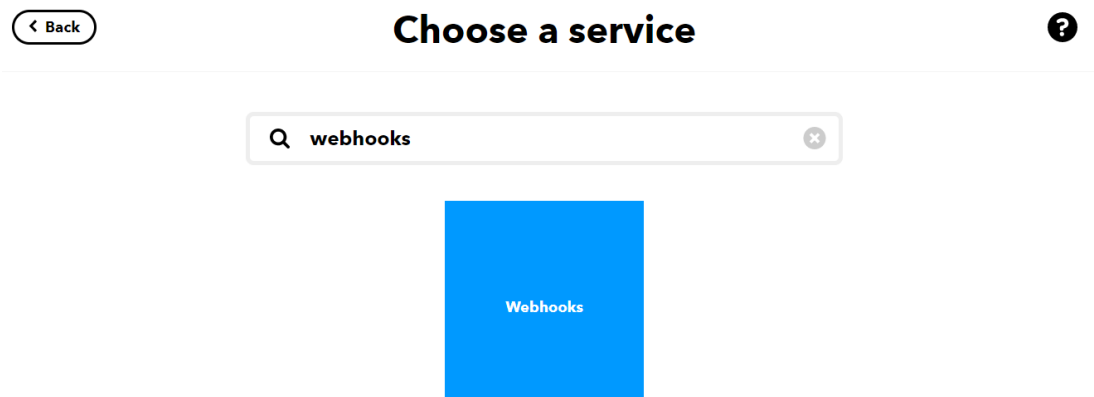


Fig. 3.8: IFTTT service.

Step 2:- Click on Documentation:

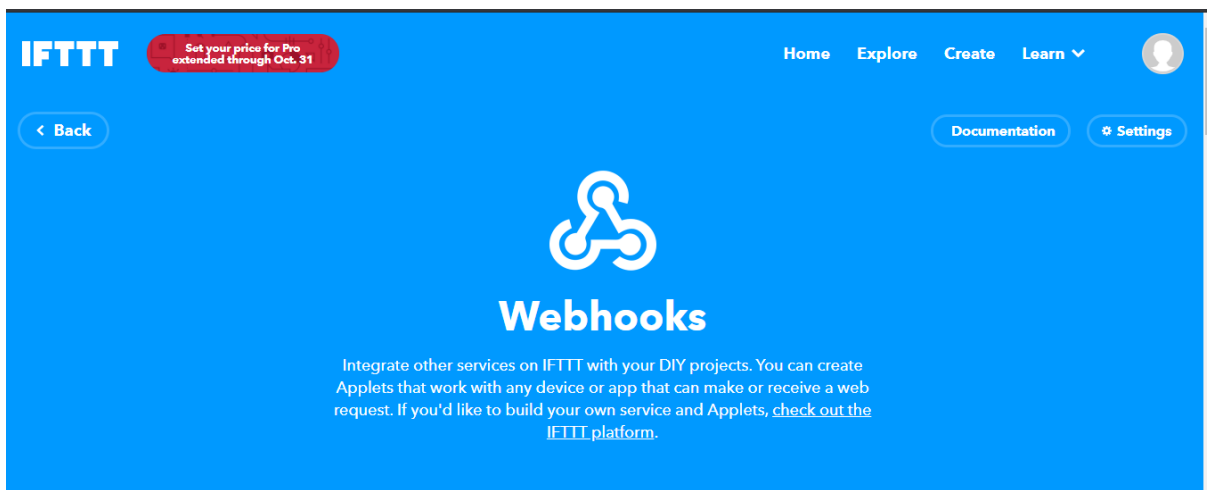


Fig. 3.9: Webhooks service in IFTTT.

Step 3: We typed “Patient_Info” in the event box and copy the URL. We will use this URL in ThingHTTP.



Fig. 3.10: Webhooks documentation.

Now let's make Applet to link ThingHTTP to Google sheet and to send email/sms. After that we will jump to complete our ThingHTTP.

Step 4: Click on New Applete in My Applet option.

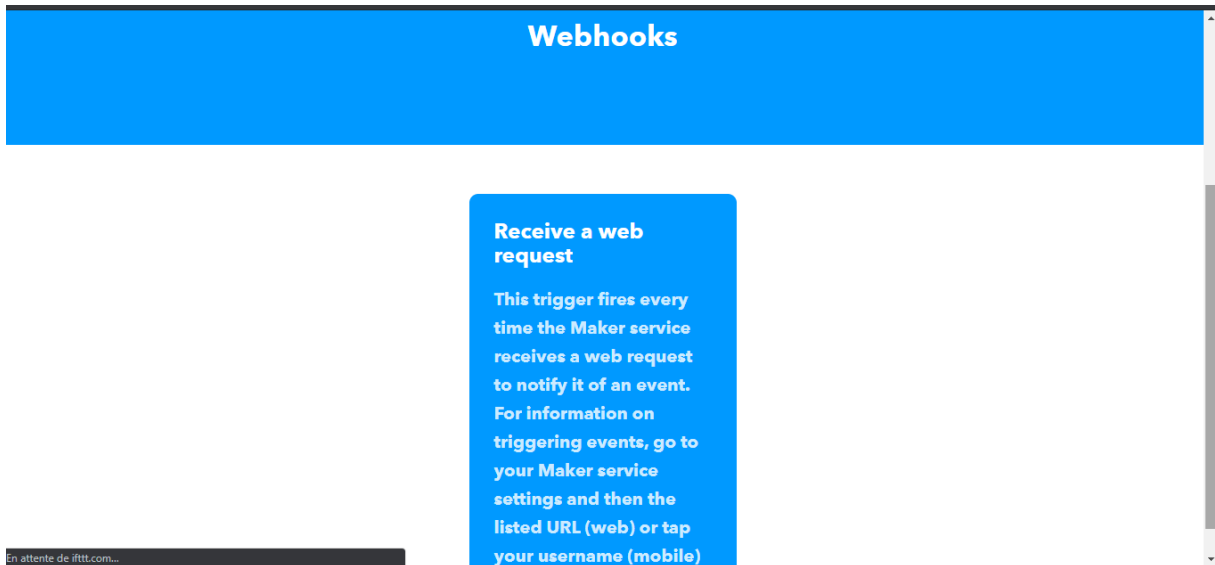


Fig. 3.11: Webhooks applet in IFTTT.

Step 5: Type the Event Name which is same as you write in the event box in webhooks URL. Click on Create Trigger.

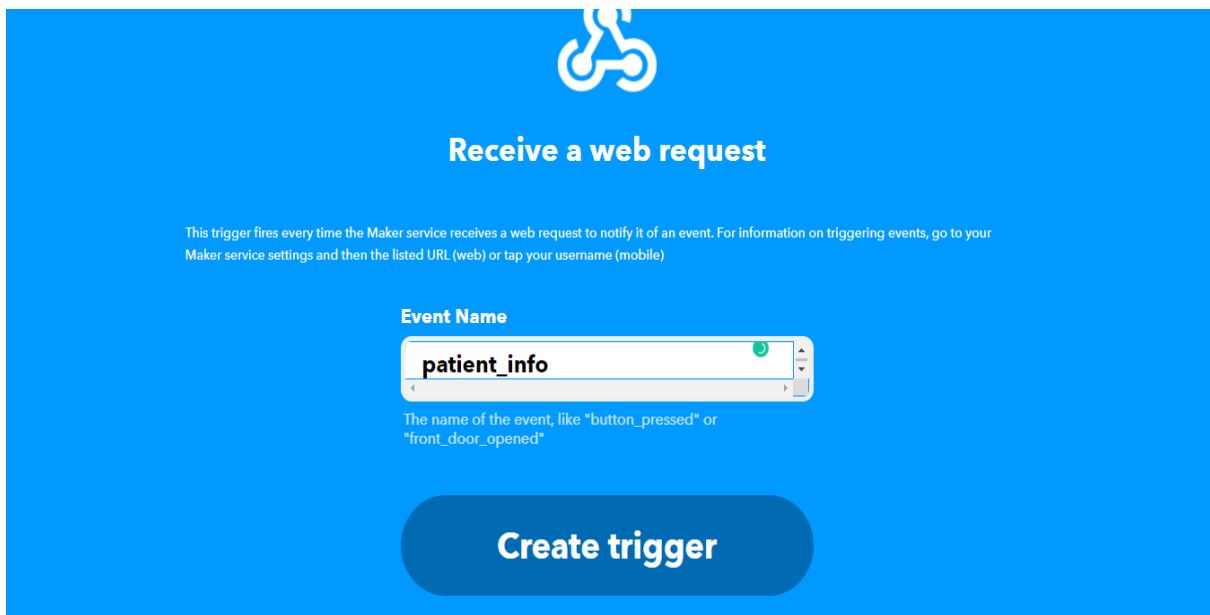


Fig. 3.12: Webhooks applet triggering in IFTTT.

Step 6: Click on “+that” and search for Google Sheets and click on it.

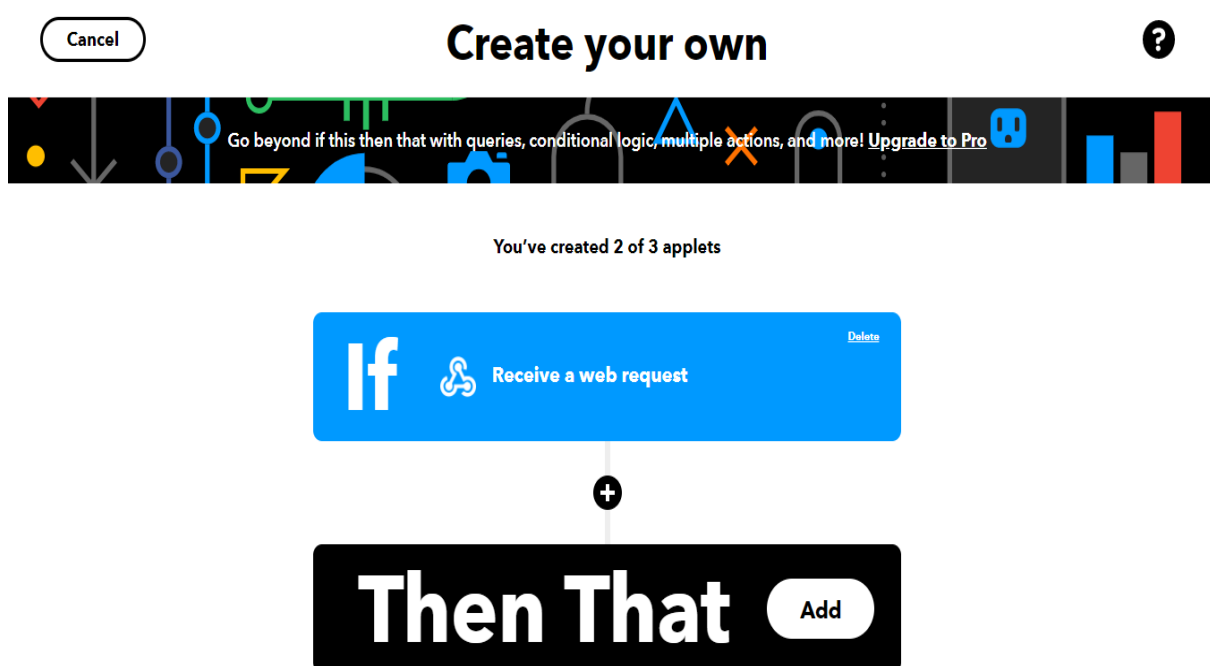


Fig. 3.13: Applet menu.

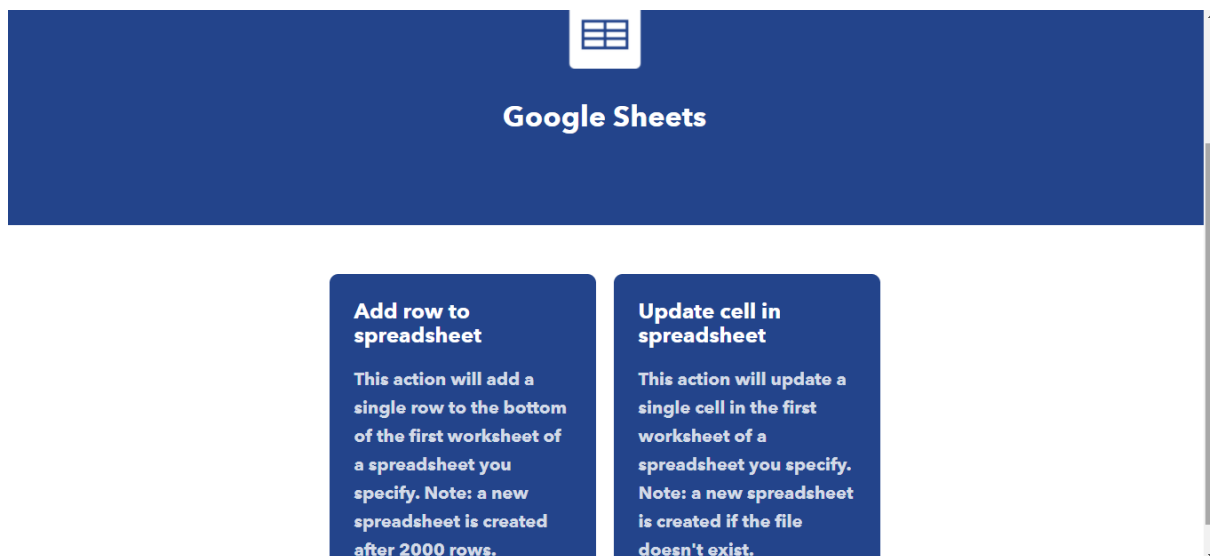


Fig. 3.14: Google Sheets service in IFTTT.

Step 7: In formatted row box, we have date and time, event name, BPM value and body temperature which will be written as shown.



Fig. 3.15: Google Sheets service configuration in IFTTT.

Step 8:- Review the applet and click on finish.

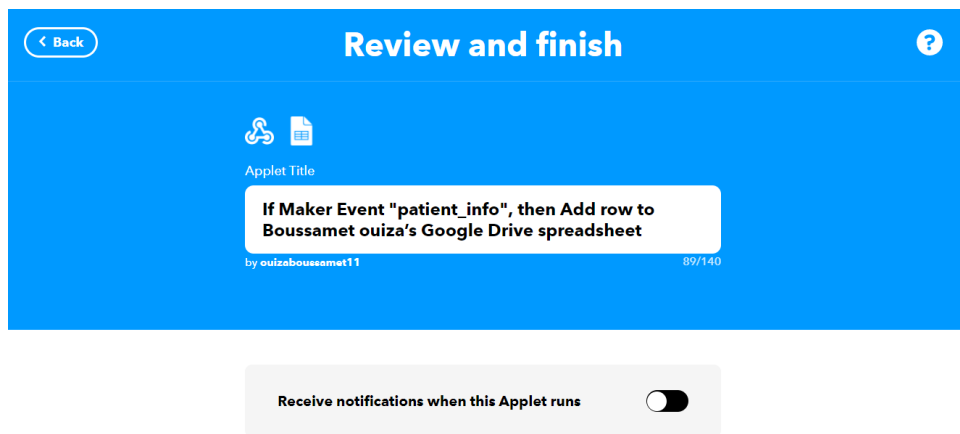


Fig. 3.16 : The created applet.

In the same way, we have to make applet for sending email when Panic event is occurred. So again click on “+this” and select Webhooks, then in event name enter “Panic”. In “+that” search for Gmail and click on it.

Now, click on Send an email.

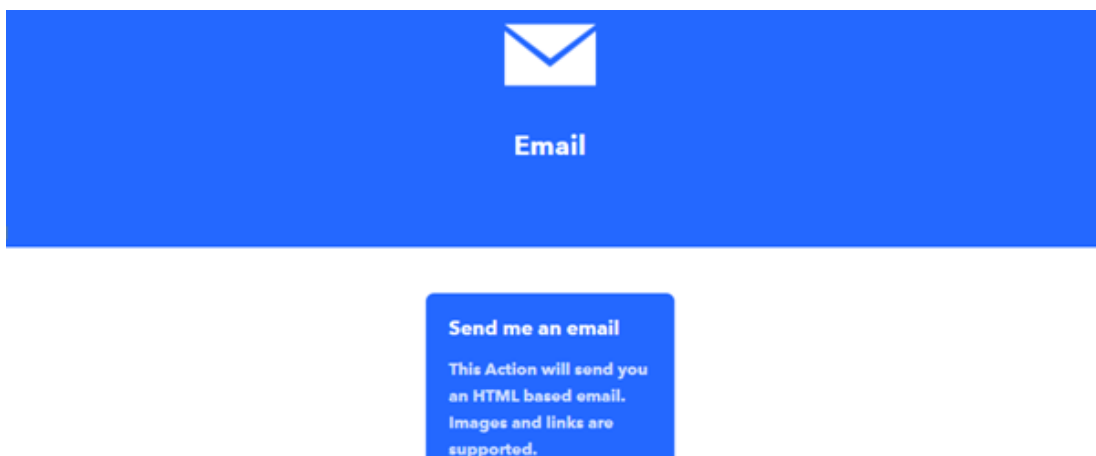


Fig. 3.17: Email service in IFTTT.

We type the email addresses on which we wish to receive email when there is a panic to the patient here we can select the doctor email or the hospital centre.

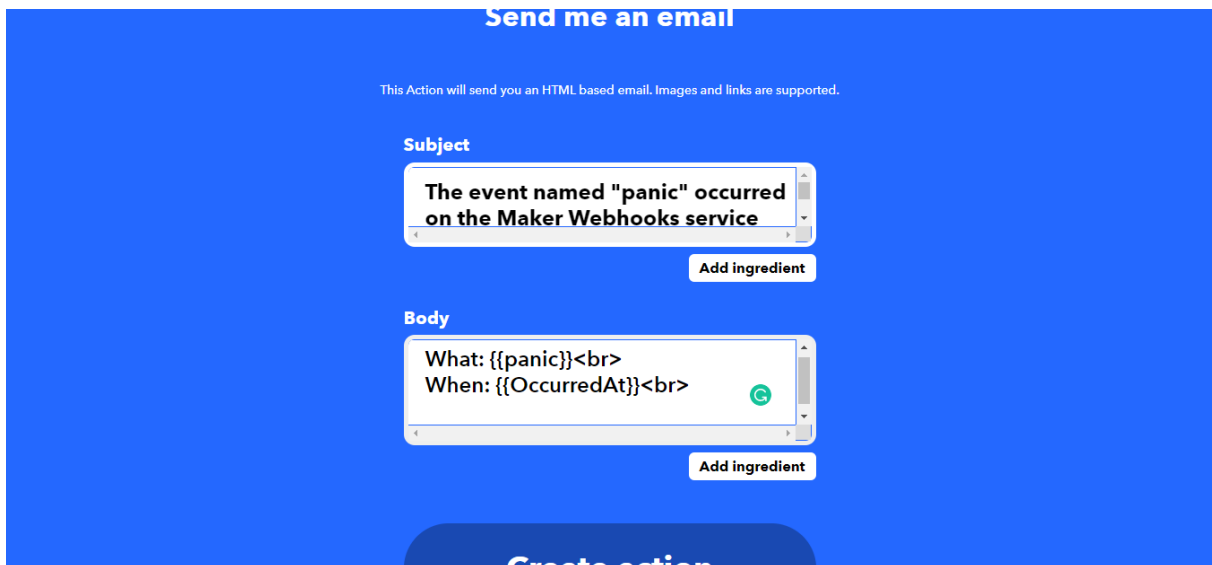


Fig. 3.18: Email service configuration in IFTTT.

Then we type the body content to send in the email and click on create action. Review it and finish.

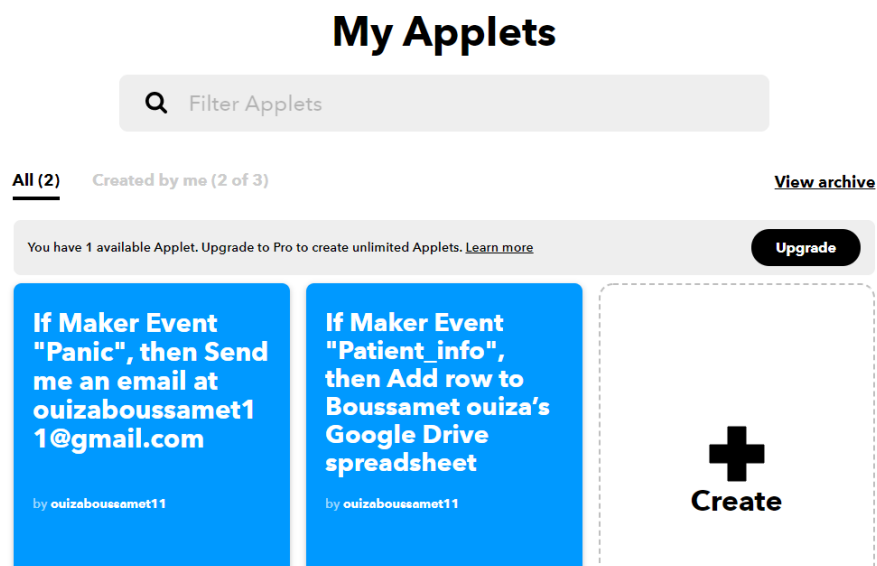


Fig. 3.19 : My applets.

- Configuring ThingHTTP to connecte ThingSpeak with IFTTT :

Step 1: We click on new ThingHTTP. Give any name and Paste the URL that you copied from the webbooks documentation. Fill Remaining information as shown below.

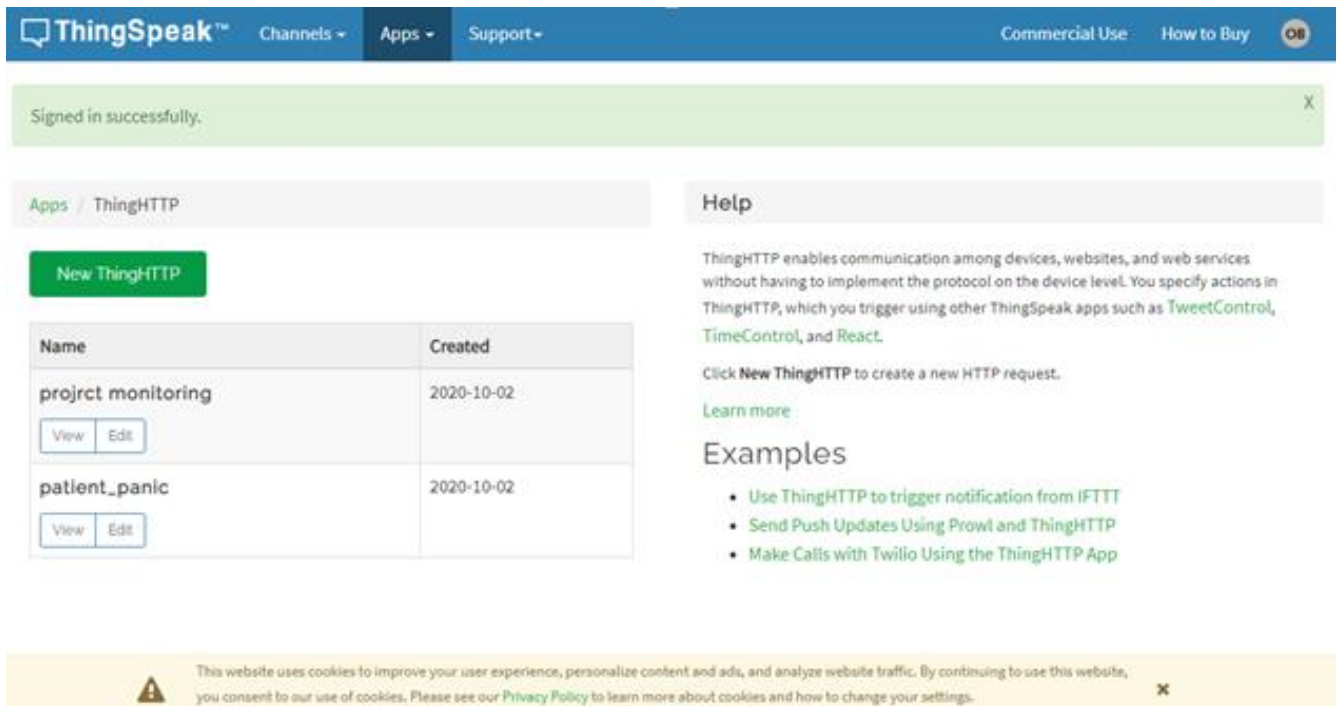


Fig. 3.20: ThingHTTP menu in thingspeak.com.

In Body, we have to write the information that we want to send to the IFTTT applet. Here we are sending patient Pulse reading and temperature.

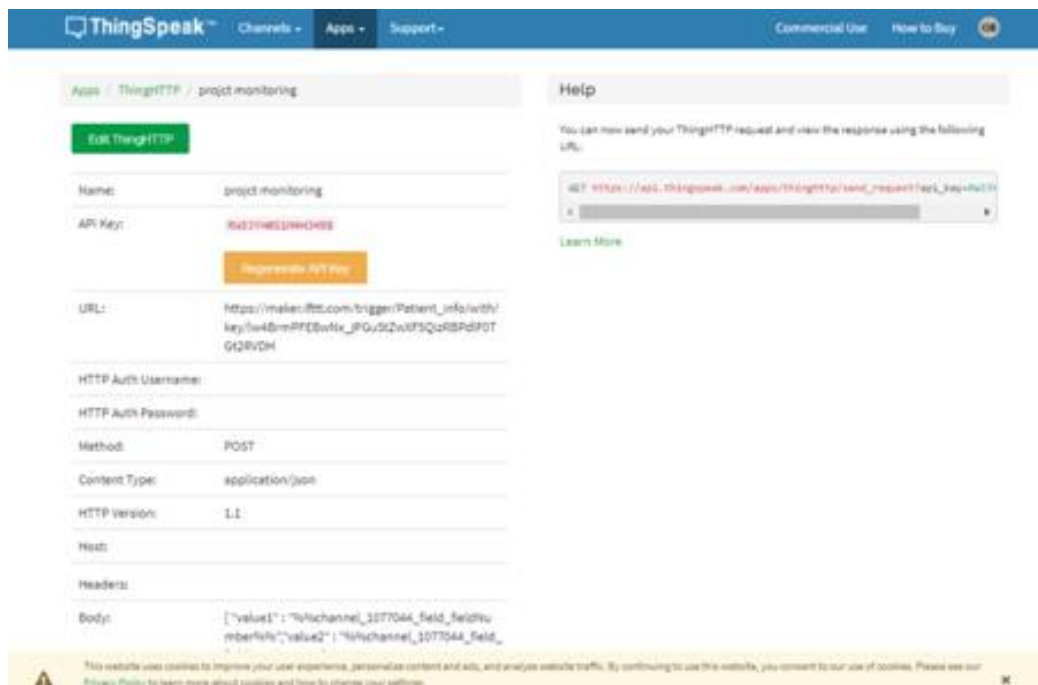


Fig. 3.21: ThingHTTP configuration.

After filling the information, then we save the created *ThingHTTP*. In the same manner, we have to make ThingHTTP for “Panic”.

In URL, we write **Panic** in place of **Patient_Info**. Body remains empty and All other information are same as in previous ThingHTTP.

- Now, we have to make **React** to trigger the URL :

React works with ThingHTTP app to perform actions when channel data meets a certain condition. To make React, we click on Apps then React. We click on **New React**.

We give a name to our React. Condition we define it as Numeric and Test Frequency as on Data Insertion.

We choose the Condition on which to trigger the URL. We select our channel from the *If Channel* drop down menu. Choose field 1 i.e Pulse rate and make condition of *greater than* any value for example 60.

We choose ThingHTTP from Action drop down menu and select the ThingHTTP, then “Run action each time condition is met” and we save React.

The screenshot shows the 'React configuration' page in ThingSpeak. The breadcrumb trail is 'Apps / React / Pulse rate / Edit'. The form fields are as follows:

- React Name:** Pulse rate
- Condition Type:** Numeric
- Test Frequency:** On Data Insertion
- Condition:** If channel: Patient Monitoring (523997)
- field:** 1 (Pulse Rate)
- Operator:** is greater than
- Value:** 60

Fig. 3.22: React configuration in thingspeak.com.

In the same way, we make React for the Panic as shown.

The screenshot shows the 'React configuration' page in ThingSpeak for a 'Panic' react. The breadcrumb trail is 'Apps / Support'. The form fields are as follows:

- React Name:** Panic rate
- Condition Type:** Numeric
- Test Frequency:** On Data Insertion
- Condition:** If channel: Patient data monitoring (1171782)
- field:** 6 (panic)
- Operator:** is equal to
- Value:** 1
- Action:** ThingHTTP
- then perform ThingHTTP:** patient_panic

On the right side, there is a 'React Settings' section with the following instructions:

- React Name:** Enter a unique name for your React.
- Condition Type:** Select a condition type corresponding with your data. A channel can hold numeric sensor data, text, strings, status updates, or geographic location information.
- Test Frequency:** Choose whether to test your condition every time data enters the channel or on a periodic basis.
- Condition:** Select a channel, a field and the condition for your React.
- Action:** Select ThingTweet, ThingHTTP, or MATLAB Analysis to run when the condition is met.
- Options:** Select when the React runs.

A 'Learn More' link is also present.

Fig. 3.23: React configuration for panic in thingspeak.com.

Now, we have done with Web based work.

3.7 Conclusion :

Home based health monitoring systems are being proposed as a low cost solution. Such a system consists of physiological data that stores, process and communicate through a local manner such as smart phones, personal computers. Such systems should satisfy strict safety, security, reliability, and long-term real-time operation requirements. This system is expected to monitor patient under critical care more conveniently and accurately for diagnosing and also the doctors will be able to monitor the patient's condition sitting in his own office without being physically present near to the patient's bed.

Chapter four

Practical

Implementation and

Results

This chapter represents detailed procedures of system realization with its performance in real-time ,it gives also design functionalities, methodology behind the system, IoT(internet of things) platforms and services used to link between patient and health providers, including the program code with explication.

It shows also the implementation process of the circuit by indicating connections through microcontroller, Wifi module, and all electronic sensors.

Finally, it provides summaries and major aspects examined in this thesis and it is also intended to place these findings in a wider environment.

4.1 Hardware description:

For this system we required two types of power supply 5V or 3.3 V because of some components are operated in 3.3 V however other components are operated in 5V.

All the sensors, which generates the analog output is connected to an analog pin of Arduino's analog pins. And Digital sensors like DHT11 are connected to digital pins.

The analog/digital data are processed by Arduino and with the help of internet connectivity IoT all the monitored data can be sent to the cloud (Here we have used ThingSpeak).

4.2 Circuit diagram:

In the following figure 4.1, it is shown the complete design of the whole project circuit which includes Arduino Uno microcontroller board connected with ESP8266_01 wifi module, pulse sensor, electrocardiogram sensor, temperature sensor, DHT11 which used for room's temperature and humidity detection, and a panic push button so that patients can press it on emergency states to send email to their doctors.

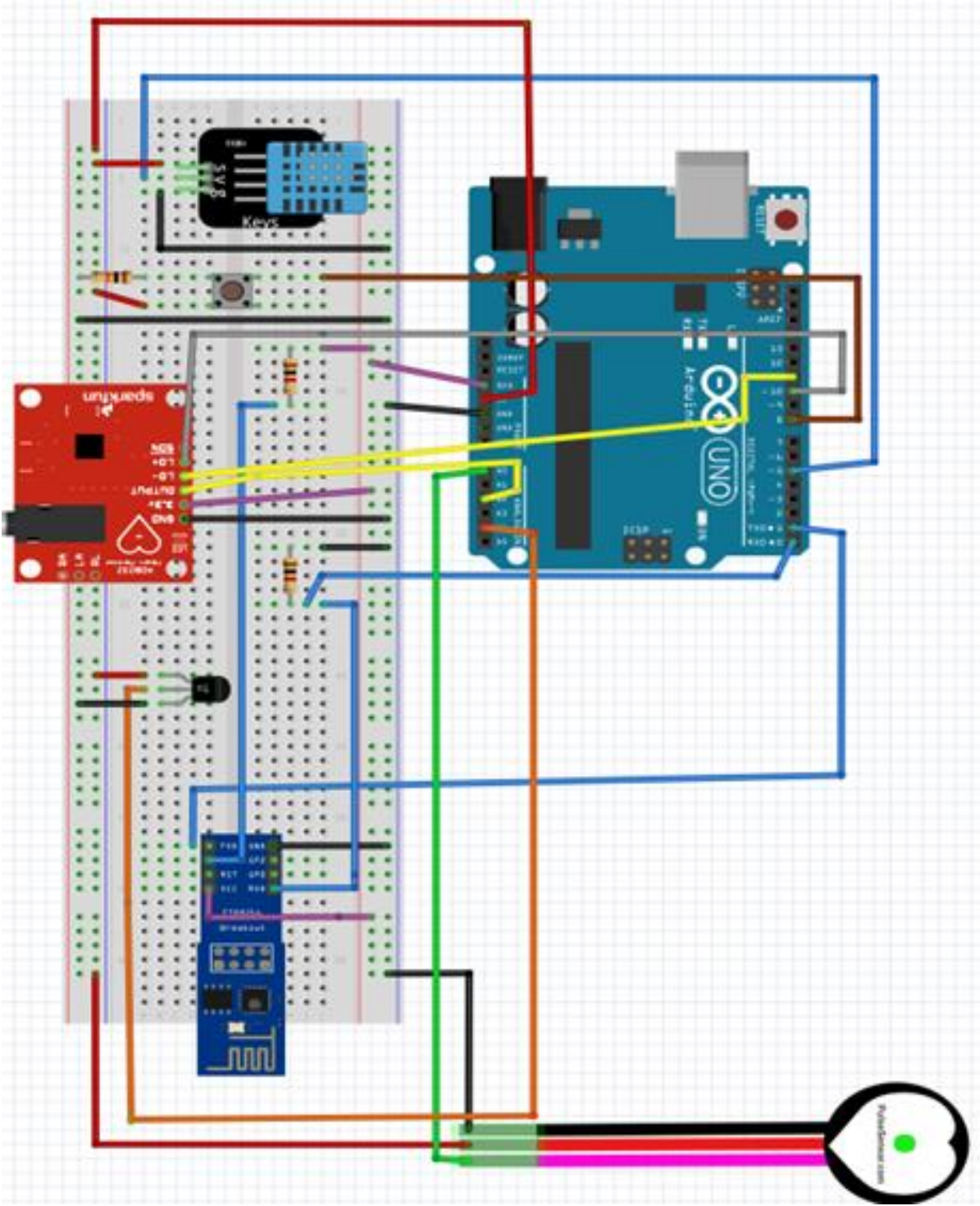


Fig.4.1: Overall designed circuit.

System component inter-connection table

ARDUINO	ECG sensor				Pulse sensor			Body Temp sensor			DHT11 sensor				ESP 01				
	OUT	LO +	LO-	3.3v	GN	VCC	SIGN	Vs	Vout	GN	VCC	GN	DATA	VCC	RX	TX	GN	CH-PD	
VCC(3.3v)	-	-	-	*	-	-	-	-	-	-	-	-	-	*	-	-	-	*	
GND	-	-	-	-	*	-	-	-	-	*	-	*	-	-	-	-	*	-	
VCC(5v)	-	-	-	-	-	*	-	*	-	-	*	-	-	-	-	-	-	-	
A4	*	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	
A0	-	-	-	-	-	-	*	-	-	-	-	-	-	-	-	-	-	-	
A5	-	-	-	-	-	-	-	-	*	-	-	-	-	-	-	-	-	-	
RX	-	-	-	-	-	-	-	-	-	-	-	-	-	-	*	-	-	-	
TX	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	*	-	-	
D5	-	-	-	-	-	-	-	-	-	-	-	-	*	-	-	-	-	-	

Notes:

- * : component pins connected
- - : component pins not connected

4.2.1 Interfacing ESP8266 with Arduino Uno:

ESP8266-01 can be interfaced with Arduino Uno, it consists of total 8 pins such as TX, RX, VCC, GPIO0, GPIO2, RST, GND, CH_PD as shown in the following figure 4.2. ESP8266 runs on 3.3V and it gets power from Arduino Uno so that its VCC shall be connected to the 3.3V power supply, TX goes to Arduino TX, RX goes to Arduino RX, CH_PD which is the chip enable pin should be connected to 3.3V power supply, and ground to ground, both 1k resistors act as pull up and pull down resistors for the CH_PD and RX pins, respectively.

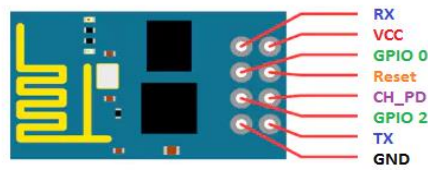


Fig .4.2: ESP8266 pinout[11].

1. **GND** → Ground
2. **GPIO2** → General purpose I/O pin No.2
3. **GPIO0** → General purpose I/O pin No.1
4. **RX** → Data Receiving pin
5. **VCC** → 3.3v
6. **REST** → Reset Pin, Active LOW
7. **CH_PD** → Chip Enable Pin. Active HIGH
8. **TX** → Data Transmitting pin

4.2.2 Interfacing ECG sensor with Arduino Uno :

The AD8232 is a chip used to measure the electrical activity of the heart. This electrical activity can be charted as an ECG or Electrocardiogram.

ECG sensor consists of 9 pins SDN, LO+, LO-, OUTPUT, 3.3V, GND these are essential connectors for operating this monitor with an Arduino board, also there are RA (Right Arm), LA (Left Arm), and RL (Right Leg) pins. Moreover, there is an LED indicator light that will pulsate to the rhythm of a heart beat.

4.2.2.1 Pin connections with Arduino:

1. GND (Ground) is connected to Arduino ground.
2. 3.3V connected to 3.3V power supply of Arduino.
3. OUTPUT (output signal) connected with Analog pin 2.
4. LO- (Leads-off Detect -) is connected to Arduino pin 11.
5. LO+ (Leads-off Detect +) is connected to Arduino pin10.
6. SDN (Shutdown) is not used.

4.2.2.2 AD8232 ECG sensor placement on body:

It is recommended to snap the sensor pads on the leads before application to the body. The closer to the heart the pads are, the better the measurement. The cables are color-coded to help identify proper placement.

The electrode pads can be placed on the forearms and leg as shown on the diagram on the left. Or they can be placed on the chest near the arms and above the right, lower abdomen (i.e. just above the right hip) as shown on the diagram on the right of the figure 4.3 .

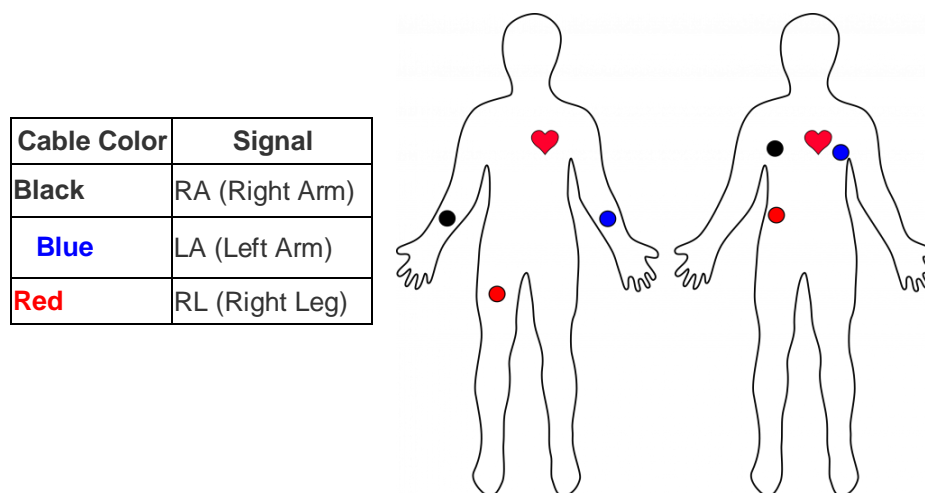


Fig.4.3: ECG probe placement.[12]

There are many other methods for placement of pads, they can be placed also on one forearm only as shown in the figure bellow.

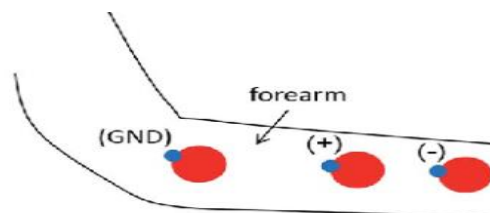


Fig .4.4: ECG probe placement on forearm.[13]

4.2.3 Pulse sensor interfacing with Arduino:

Pulse sensor detects heart rate data of patient, it is directly connected to Arduino and is quite easy to use, it has 3 pins described and showed (in the figure 4.5) in bellow:

4.2.3.1 Pinout:

- - (GND): Ground Pin.
- + (VCC): 3V to 5V Pin.

- S : signal connected to an Analog Pin.



Fig .4.5: Backside of the pulse sensor.[14]

The ground is connected to the Arduino ground, VCC to 5V power supply and the signal pin to the Analog pin 0 of Arduino.

4.2.3.2 Sensor placement on body:

The sensor has two sides, on one side the LED is placed along with an ambient light sensor and on the other side we have some circuitry. The LED on the front side of the sensor should be placed over a vein of a human body. This can either be Finger tip or the wrist but it should be placed directly on top of a vein.

The LED emits light which will fall on the vein directly. The veins will have blood flow inside them only when the heart is pumping, so if we monitor the flow of blood we can monitor the heart beats as well.

4.2.4 DHT11 sensor interfacing with Arduino:

DHT11 sensor is used to measure the temperature and humidity of the room , it consist of 3 pins Ground, VCC and the data pin shown in the figure 4.6.

VCC should be connected to 5V power supply of Arduino,Ground to Arduino ground and data pin to digital pin 5 of the Arduino.

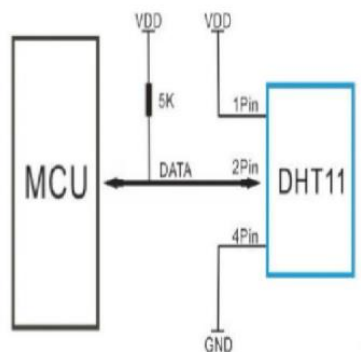


Fig .4.6:DHT11 pinout.[15]

4.2.5 LM35 sensor with Arduino:

LM35 is an analog, linear temperature sensor whose output voltage varies linearly with change in temperature. LM35 is three terminal linear temperature sensor its pinout is shown in the following figure. It can measure temperature from -55 degree celsius to +150 degree celsius. LM35 can be operated from a 5V supply and the stand by current is less than 60uA.

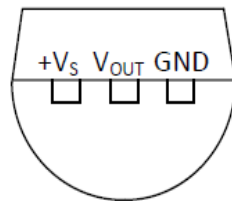


Fig .4.7: LM35 pinout.[16]

+V_s is connected to 5V power supply of Arduino, GND to Arduino ground and V_{out} to the Analog pin 5 of the Arduino.

LM35 should be placed on the finger of the patient to detect his body temperature.

4.2.6 Push button interfacing :

Pushbutton connects two points of a circuit when it is pressed , it has four legs (or four pins) as shown in the figure bellow. Two opposite pins connect when the button is pushed; otherwise, they are disconnected. In one side, we connect one leg to Arduino pin 8. In other side, we supply 5V electricity to one leg. The other leg is connected to a 10k ohm resistor before going to ground.

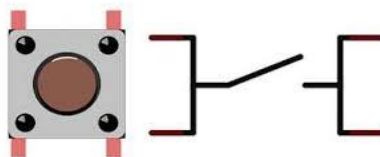


Fig. 4.8: Pushbutton.[17]

4.3 System setup:

In the following figure it is shown the complete device setup which includes Arduino uno microcontroller board connected with all sensors and wifi module which is used to send all the monitored data to the ThingSpeak cloud, it is also connected to a pushbutton which is a

2- In second part of the code we have declared all the variables that we need, we also make instance for timer, SoftwareSerial, dht11 sensor and pulse sensor. We set our WiFi name, password, IP(Internet Protocol) of ThingSpeak.com, and the port number .

3- In the setup function shown in the Appendix-B- fig.B.2, we start by setting the baud rate for the serial communication between Arduino serial monitor and esp8266. after that, we send AT command so that the ESP starts the communication process. And then we send the AT+CWMODE=1 command to make ESP8266 work in station mode. In addition to AT+CWJAP=, this command is sent in order to connect to our Access Point (Wi-Fi router).

4- In the loop section as shown in the Appendix-B- fig.B.3, we declared a string to update information on ThingSpeak channel for that we need the API key that we copied from ThingSpeak platform after that we attach the reading with the GET URL using "&field=";after the readings the information we use two main command that are :

- “AT+CIPSTART=0 \TCP\” this command will establish TCP (Transmission Control Protocol) command over port 80.
- “AT+CIPSEND=” this command will send the information or the data that we get from the sensors.

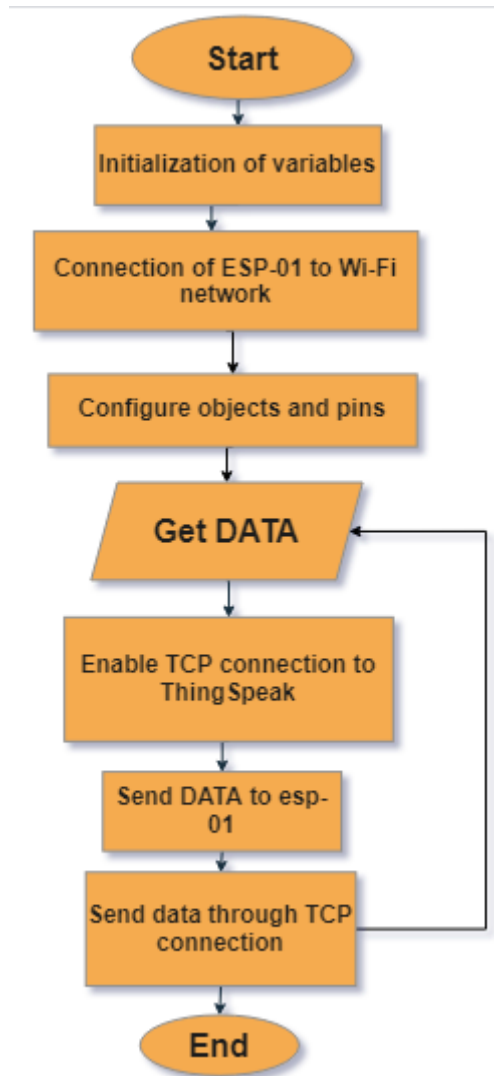
5- The next part in the loop shown at Appendix-B- figures from fig.B.3 to fig.B.6 is the functions that we have created to return the sensors values in form of string and also we made a function for panic-button .

4.4.2 Flowchart of the system:

We start by defining all libraries and declaration of variables. We enable connection of ESP-01 to WiFi network using At commands .Then we do some configuration of objects and pins.

In the loop section we use String Get data that reads data from sensors and allow connection to the Cloud using API key, after that TCP connection should be enabled using

AT commands. We send the collected data to ESP01, then, these data will be sent through TCP connection.



4.5 Results and Discussion:

4.5.1 Results:

This system can be used to transmit the patient's parameters in real-time to remote location. These parameters and values are sent wirelessly to the Arduino Uno board receiver which is connected to the central station personal computer. The sensed data can be seen over the serial monitor/ plotter of Arduino IDE and also on ThingSpeak Cloud service using a unique username and password. The following are some results from the system.

In the following figure 2.24 the patient's parameters are shown on the serial monitor of arduino IDE . The Arduino Serial Monitor function can display serial data sent from Arduino. To start the Serial Monitor we must click the Serial Monitor toolbar icon.

```

5. at command => AT+CIPMUX=1 Ok
6. at command => AT+CIPSTART=0,"TCP","api.thingspeak.com",80 Fail
0. at command => AT+CIPSEND=0,93 Ok
2. at command => AT+CIPCLOSE=0 Ok
  bodytemp = 36.62°C
7
  BPM= 40
  Humidity in %= 53
  Temperature(C)= 26
3. at command => AT+CIPMUX=1 Ok
4. at command => AT+CIPSTART=0,"TCP","api.thingspeak.com",80 Ok
5. at command => AT+CIPSEND=0,92 Ok
7. at command => AT+CIPCLOSE=0 Ok
  bodytemp = 36.62°C
348
  BPM= 42
  Humidity in %= 53
  Temperature(C)= 26

```

Fig 4.24 : Sample of patient's vital data

In the figure4.25 bellow we have the ecg graph shown on the Serial Plotter of Arduino. Serial Plotter is one of the tools in Arduino IDE. Serial Plotter receives data from Arduino and visualizes data as waveforms. Serial Plotter can visualize not only single but also multiple sensor data in the same graph.

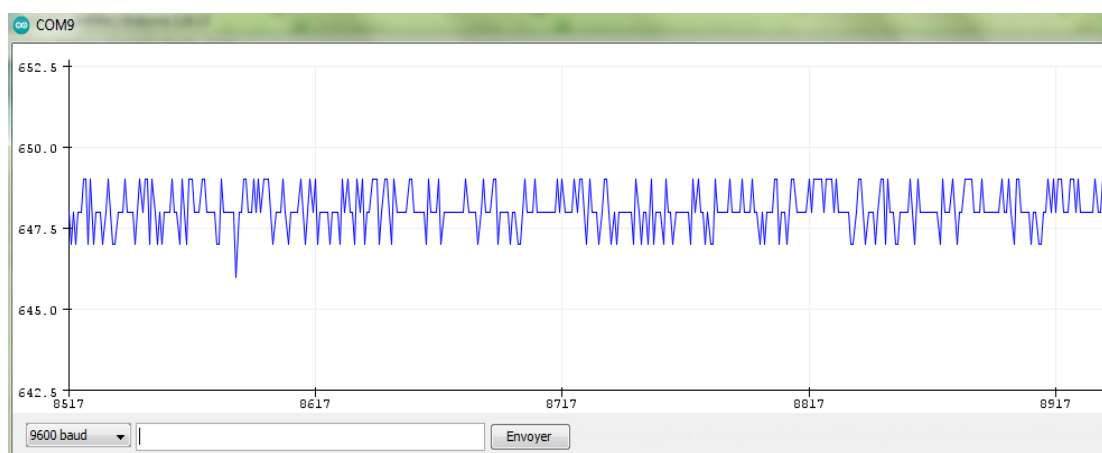


Fig 4.25: ECG plot

The following figures show the data uploaded to ThingSpeak cloud . the doctor can check patient's parameters whenever he opens the channel , and it is shown as follow:

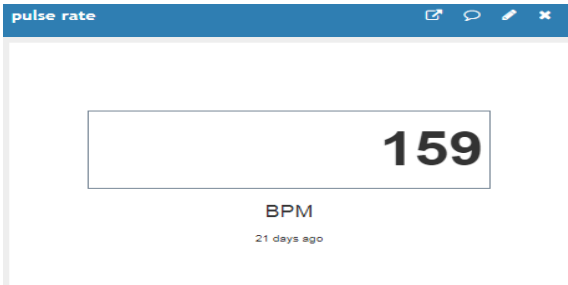


Fig 4.26: BPM result

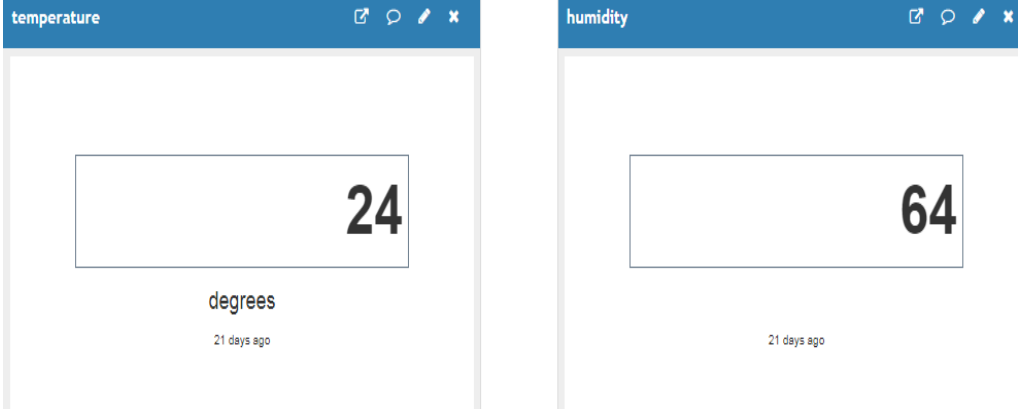


Fig 4.27: Temperature and humidity data

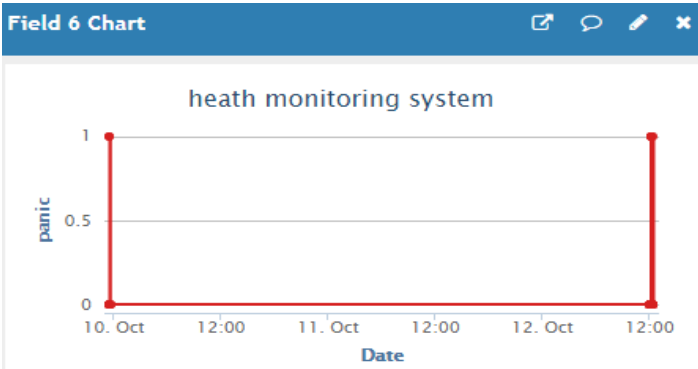


Fig 4.28: Panic pulse

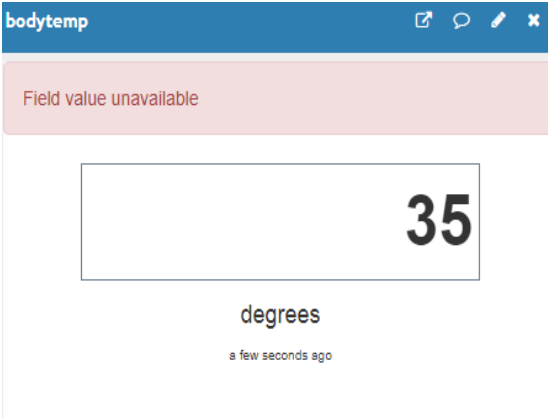
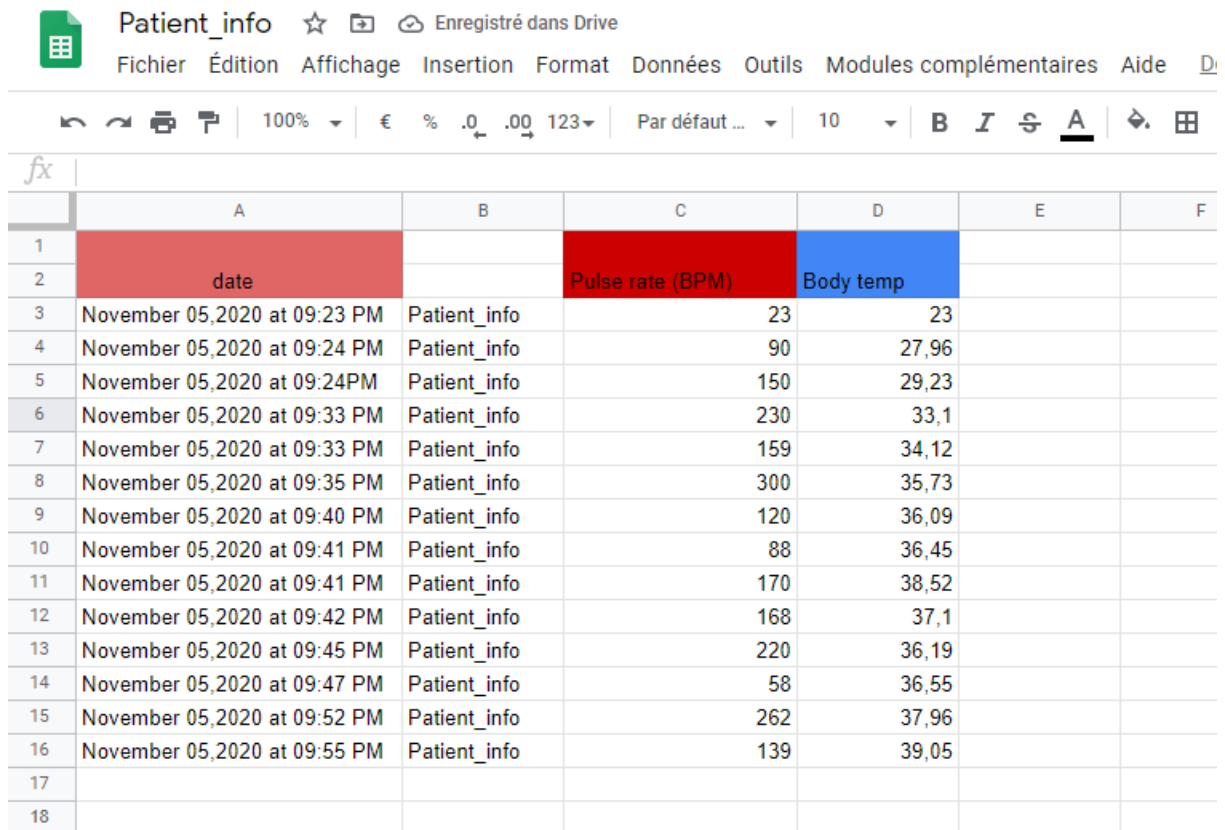


Fig 4.29: Body temperature



	A	B	C	D	E	F
1						
2	date		Pulse rate (BPM)	Body temp		
3	November 05,2020 at 09:23 PM	Patient_info	23	23		
4	November 05,2020 at 09:24 PM	Patient_info	90	27,96		
5	November 05,2020 at 09:24PM	Patient_info	150	29,23		
6	November 05,2020 at 09:33 PM	Patient_info	230	33,1		
7	November 05,2020 at 09:33 PM	Patient_info	159	34,12		
8	November 05,2020 at 09:35 PM	Patient_info	300	35,73		
9	November 05,2020 at 09:40 PM	Patient_info	120	36,09		
10	November 05,2020 at 09:41 PM	Patient_info	88	36,45		
11	November 05,2020 at 09:41 PM	Patient_info	170	38,52		
12	November 05,2020 at 09:42 PM	Patient_info	168	37,1		
13	November 05,2020 at 09:45 PM	Patient_info	220	36,19		
14	November 05,2020 at 09:47 PM	Patient_info	58	36,55		
15	November 05,2020 at 09:52 PM	Patient_info	262	37,96		
16	November 05,2020 at 09:55 PM	Patient_info	139	39,05		
17						
18						

Fig 4.30: Google sheet data arrival.



Fig 4.31: Arrival of emergency email.

4.5.2 Discussion:

The practical designed system was implemented successfully, data was collected accurately, sent and displayed on internet platform so that they can be reachable from any location, this system realize the concept of the remote healthcare monitoring system based on the internet of things.

However, in the realization of this project we have faced some issues, the first problem was that electronic devices are so sensitive that can effect data accuracy and efficiency, the second one is the disturbances of the ECG plot due to the circuit noise.

Finally, sometimes internet problems occur on our system connectivity which increases the transmission time of data and create an unwanted delay on the emergency email arrival.

4.6 Future enhancements:

In future this work can be extended by adding more medical sensors to the existing setup for exemple glucose sensor, organic blood oxygen sensor, body falling sensor and the Blood pressure sensor . This work is done based on single person's data collection and in future this can be extended to multiple people.

This system also can be devlopped by creating a special web site using HTML, CSS, JavaScript and MySQL rather than using an IoT platform, it will make the system more accessible and available for users.

The whole system, which we have designed can be integrated into a small wearable device as small as a cell phone or a wrist watch. This will help the patients to easily carry this device with them wherever they go.

4.7 Conclusion:

The concept of our work was to provide better life to patients by improving the efficiency of health care services.

Monitoring system is built which allows the doctor to view the patient's parameters remotely and dynamically in an internet platform in real time; all he needs is an Internet access.

This system enables the patient to be free in their daily life without taking travelling long distances to hospitals and clinics as well monitoring their health continuously.

General conclusion

This project work bettered our understanding of the nature and working principles of the several electronic devices that we have utilized in the construction of our project.

The objective of this project was to built a low power, low cost, reliable monitoring system that would accurately collect the patient's parameters then transmit them to medical services at a remote location.

A real-time health monitoring system has been successfully implemented and the main objective of the experiment was achieved.

The resulting system was indeed low in power and cost, accurate, reliable and also provided real-time monitoring.

Moreover, an alert system was provided which helps the patient to have the first aide when an emergency situation occurs.

The designed system modules can further be enhanced and produced to an integrated single system. The important matter that came up during project design is that all the circuit components used in the remote system are available.

This work is specifically performed for one patient and it may be expanded in the future to include multiple people.

References

- [1] : <http://www.innovaker.com/Python-IOT-Smart-Home-and-Smart-City> date : 3 octobre 2020
- [2] : <https://www.intechopen.com/online-first/innovative-wearable-sensors-based-on-hybrid-materials-for-real-time-breath-monitoring> date : 10 octobre 2020
- [3] : Aditya Sharma, Anuj Kumar Singh, Khushi Saxena, Mr. Abhinav Bansal, “ Smart Health Monitoring System using IoT”, International Journal for Research in Applied Science & Engineering Technology (IJRASET), Volume 8 Issue V May 2020- Available at www.ijraset.com.
- [4] : <https://how2electronics.com/wp-content/uploads/2019/03/AD8232-ECG-Measurement-Pulse-Monitoring-Sensor-Monitor-Module.jpg> date : 19 octobre 2020
- [5] : <https://www.mouser.com/datasheet/2/758/DHT11-Technical-Data-Sheet-Translated-Version-1143054.pdf> date : 20 octobre 2020
- [6] : <https://fr.dhgate.com/product/dht11-temperature-and-humidity-sensor-module/405147119.html> date : 20 octobre 2020
- [7] : Prof(Dr).Jayant Shekhar, Mr.Desalegn Abebaw , Dr. Mesfin Abebe Haile, Md.Ahmed Mehamed, Mr.Yohannis Kifle ,” Temperature and Heart Attack Detection using IOT(Arduino and ThingSpeak)”, International Journal of Advances in Computer Science and Technology, 7(11), November, Volume 7 No.11, November 2018.
- [8] : <https://www.circuitbasics.com/how-to-set-up-a-web-server-using-arduino-and-esp8266-01/> date : 24 octobre 2020.
- [9] : <https://www.zdnet.com/article/what-is-the-internet-of-things-everything-you-need-to-know-about-the-iot-right-now/> date : 24 octobre 2020.
- [10] : https://thingspeak.com/pages/how_to date : 28 octobre 2020.
- [11] : <https://components101.com/wireless/esp8266-pinout-configuration-features-datasheet> date : 01 novembre 2020.

[12] : <https://learn.sparkfun.com/tutorials/ad8232-heart-rate-monitor-hookup-guide/all>. Date : 01 novembre 2020.

[13] : <https://pubs.rsc.org/en/content/articlelanding/2016/ra/c6ra00079g/unauth#!divAbstract>. Date : 01 novembre 2020.

[14] : <https://www.indiamart.com/proddetail/pulse-sensor-12392893291.html> date :02 novembre 2020.

[15] : https://www.researchgate.net/figure/DHT11-Sensor-Humidity-sensor-is-used-for-sensing-the-vapours-in-the-air-The-change-in-RH_fig4_317338046 date :02 novembre 2020.

[16] : <https://blog.podkalicki.com/attiny13-temperature-logger-using-lm35-and-software-uart>. date : 04 novembre 2020

[17] : <http://archive.fabacademy.org/fabacademy2017/fablabvigyanashram/students/53/assignments/Assignment13/switch.html>. date : 04 novembre 2020.

[18] : <https://github.com/WorldFamousElectronics/PulseSensorPlayground>. date :02 octobre 2020

[19] : <https://github.com/JChristensen/Timer> date :02 octobre 2020

[20]: <https://github.com/adafruit/DHT-sensor-library> date :02 octobre 2020.

Appendix -A-

Pin Number	Pin Name	Pin Function
1	RESET	Active Low External Reset Signal
2	ADC(TOUT)	ADC Pin Analog Input
3	CH_PD	Active High Chip Enable
4	GPIO16	General purpose IO
5	GPIO14	General purpose IO
6	GPIO12	General purpose IO
7	GPIO13	General purpose IO
8	VCC	Power Supply
9	Ground	Ground
10	GPIO15	General purpose IO, should be connected to ground for booting from internal flash
11	GPIO1	General purpose IO, Serial Tx1
12	GPIO0	General purpose IO, Launch Serial Programming Mode if Low while Reset or Power ON
13	GPIO4	General purpose IO
14	GPIO5	General purpose IO
15	GPIO3	General purpose IO, Serial Rx
16	GPIO1	General purpose IO, Serial Tx

Fig. A.1: ESP-12 pin configuration[7].

Pin Number	Pin Name	Pin Function
1	Ground	Ground
2	GPIO1	General purpose IO, Serial Tx1
3	GPIO2	General purpose IO
4	CH_PD	Active High Chip Enable
5	GPIO0	General purpose IO, Launch Serial Programming Mode if Low while Reset or Power ON
6	RESET	Active Low External Reset Signal
7	GPIO3	General purpose IO, Serial Rx
8	VCC	Power Supply

Fig. A.2: ESP-01 pin configuration[7].

Appendix -B-

```
remotehealthcare $
#define USE_ARDUINO_INTERRUPTS true
#define DEBUG true;
#include <SoftwareSerial.h>
#include <dht11.h>
#define RX 2
#define TX 3
#define dht_apin 5 // Analog Pin sensor is connected to
#include <PulseSensorPlayground.h> // Includes the PulseSensorPlayground Library.
#include "Timer.h"

// Variables
Timer t;
const int PulseWire = 0; // PulseSensor PURPLE WIRE connected to ANALOG PIN 0
const int LED13 = 13; // The on-board Arduino LED, close to PIN 13.
int Threshold = 550; // Determine which Signal to "count as a beat" and which to ignore.
// The "Gettting Started Project" to fine-tune Threshold Value beyond default setting otherwisethe default "550" value.

int val;
int error;
int panic;

PulseSensorPlayground pulseSensor; // Creates an instance of the PulseSensorPlayground object called "pulseSensor"

dht11 dhtObject;
String AP = "mywifi"; // AP NAME
String PASS = "wiz2019"; // AP PASSWORD
String API = "SE4UDAB9CY0081X6"; // API KEY
String HOST = "api.thingspeak.com";
String PORT = "80";
<
```

Fig. B.1 : Program code part 1.

```
remotehealthcare $
String HOST = "api.thingspeak.com";
String PORT = "80";
int countTrueCommand;
int countTimeCommand;
boolean found = false;
int valSensor = 1;

SoftwareSerial esp8266 (RX, TX);

void setup() {
  Serial.begin(115200);
  esp8266.begin(115200);
  sendCommand("AT", 5, "OK");
  sendCommand("AT+CWMODE=1", 5, "OK");
  sendCommand("AT+CWJAP=\"" + AP + "\",\"" + PASS + "\"", 20, "OK");

  // Configure the PulseSensor object, by assigning our variables to it.
  pulseSensor.analogInput(PulseWire);
  pulseSensor.blinkOnPulse(LED13); //auto-magically blink Arduino's LED with heartbeat.
  pulseSensor.setThreshold(Threshold);

  // Double-check the "pulseSensor" object was created and "began" seeing a signal.
  if (pulseSensor.begin()) {
    Serial.println("We created a pulseSensor Object !");} //This prints one time at Arduino power-up, or on Arduino reset.

  pinMode(10, INPUT); // Setup for leads off detection LO +
  pinMode(11, INPUT); // Setup for leads off detection LO -
```

Fig. B.2 : Program code part 2.

```
remotehealthcare $
```

```
pinMode(10, INPUT); // Setup for leads off detection LO +
pinMode(11, INPUT); // Setup for leads off detection LO -
}

void loop() {

String getData = "GET /update?api_key="+ API+"&field1="+getTemperatureValue()+"&field2="+getHumidityValue()+"&field3="
                +getPulserateValue()+"&field4="+getECG()+"&field5="+getbodytemp()+"&field6="+getpanicbutton();
sendCommand("AT+CIPMUX=1",5,"OK");
sendCommand("AT+CIPSTART=0,\"TCP\", \"\"+ HOST +\"\", "+ PORT,15,"OK");
sendCommand("AT+CIPSEND=0," +String(getData.length()+7),7,">");
esp8266.println(getData);delay(2000);countTrueCommand++;
sendCommand("AT+CIPCLOSE=0",5,"OK");}

String getPulserateValue() {
int myBPM = pulseSensor.getBeatsPerMinute(); // Calls function on our pulseSensor object that returns BPM as an "int".
                                              // "myBPM" hold this BPM value now.
Serial.print(" BPM= ");                      // Print phrase "BPM: "
Serial.println(myBPM);                       // Print the value inside of myBPM.

delay(5);
return String (myBPM);
}
}
```

Fig. B.3: Program code part 3.

```
remotehealthcare $
```

```
String getTemperatureValue() {

dhtObject.read(dht_apin);
Serial.print(" Temperature (C)= ");
int temp = dhtObject.temperature;
Serial.println(temp);
delay(5);
return String(temp);

}

String getHumidityValue() {

dhtObject.read(dht_apin);
Serial.print(" Humidity in %= ");
int humidity = dhtObject.humidity;
Serial.println(humidity);
delay(5);
return String(humidity);

}
}
```

Fig. B.4: Program code part 4.

```
remotehealthcare $
```

```
}  
String getECG () {  
  
    if((digitalRead(10) == 1) || (digitalRead(11) == 1)) {  
        Serial.println('!');  
    }  
    else {  
        // send the value of analog input 0:  
        Serial.println(analogRead(A2));  
        Serial.print( " " );  
    }  
  
    //Wait for a bit to keep serial data from saturating  
    delay(5);  
    return String (analogRead(A2));  
}  
  
String getbodytemp() {  
  
    val = analogRead(A4);  
    float mv = ( val/1024.0)*5000;  
    float cel = mv/10;  
    float farh = (cel*9)/5 + 32;  
    Serial.print(" bodytemp = ");  
    Serial.print(cel);  
    Serial.println("*C");  
    delay (5);  
    return String (cel);  
}
```

Fig. B.5: Program code part 5.

```
remotehealthcare $
```

```
String getpanicbutton(){  
    panic = digitalRead(8);  
    if(panic == HIGH){  
        Serial.println(panic);// "patient is in problem!"  
    }  
    delay (1);  
    return String (panic);  
}  
  
void sendCommand(String command, int maxTime, char readReplay[]){  
    Serial.print(countTrueCommand);  
    Serial.print(". at command => ");  
    Serial.print(command);  
    Serial.print(" ");  
    while(countTimeCommand < (maxTime*1))  
    {  
        esp8266.println(command);//at+cipsend  
        if(esp8266.find(readReplay)//ok  
        {  
            found = true;  
            break;  
        }  
    }  
  
    countTimeCommand++;  
}
```

Fig. B.6 : Program code part 6.

```
remotehealthcare $ [REDACTED]
esp8266.println(command);//at+cipsend
if(esp8266.find(readReply))//ok
{
    found = true;
    break;
}

countTimeCommand++;
}

if(found == true)
{
    Serial.println("Ok");
    countTrueCommand++;
    countTimeCommand = 0;
}

if(found == false)
{
    Serial.println("Fail");
    countTrueCommand = 0;
    countTimeCommand = 0;
}

found = false;
```

Fig. B.7: Program code part 7.