

**A
PROJECT REPORT
On
“HEART RATE MONITORING SYSTEM”**

**By
INDERJEET SINGH SAINI
(ROLL NO. B150244617)**

**BARI NIKHIL NILESH
(ROLL NO. B150244604)**

Under the Guidance of

Dr.V.S. RANE

**In partial fulfillment of the requirement for the degree of
B.E (INSTRUMENTATION & CONTROL ENGINEERING)**

Awarded by

SAVITRIBAI PHULE PUNE UNIVERSITY



**Department of Instrumentation Engineering
Dr.D.Y.Patil Institute of Technology Pimpri, Pune-411018
YEAR 2021-2022**



Dr. D. Y. Patil Institute of
Technology, Pimpri, Pune-411018

Department of Instrumentation Engineering.

CERTIFICATE

This is to certify that the project titled
HEART RATE MONITORING SYSTEM

submitted by

INDERJEET SINGH SAINI (B150244617)

BARI NIKHIL NILESH (B150244604)

*is record of bonafide work carried out by them, under my guidance, in partial
fulfillment of the Degree of Bachelors of Engineering*

Instrumentation & Control

of

Savitribai Phule Pune University

Date: / /2022

Place: D.I.T., Pimpri, Pune-411018

DR.V.S. RANE
**Project guide &
Project incharge**

Prof.A.D. SONAR
H.O.D.

DR.PRAMOD PATIL
Principal

Acknowledgement

We would sincerely like to thank all the people without whose unstinted support our project would not have been successful. We take this opportunity to express our profound gratitude and deep regards to our guide Dr. V.S.Rane for his exemplary guidance, monitoring and constant encouragement throughout the course of this thesis. The blessing, help and guidance given by him time to time shall carry us a long way in the journey of life on which we are about to embark upon. Also, we are highly indebted to our teacher and alumni for his guidance as well as for providing necessary information regarding the project and helping us in completing the project. We also take this opportunity to express a deep sense of gratitude to our Principal Dr.PRAMOD PATIL, for his cordial support and encouragement towards the completion of the project. We also extend our appreciation to our laboratory in-charge for providing us with necessary equipment's whenever needed. Lastly, we thank our parents, friends and every person who helped us directly or indirectly for their support, co-operation and encouragement which helped us in the completion of this project.

INDERJEET SINGH SAINI (B150244617)

BARI NIKHIL NILESH (B150244604)

Abstract

In this project, we presented the design and development of an integrated device for measuring heart rate using Pulse sensor to improve estimating the heart rate. As heart related diseases are increasing day by day, the need for an accurate and affordable heart rate measuring device or heart monitor is essential to ensure quality of health.

However, most heart rate measuring tools and environments are expensive and do not follow ergonomics. Our proposed Heart Rate Measuring device is economical and user friendly based on the principle of photoplethysmography. It measures the change in volume of blood through any organ of the body which causes a change in the light intensity through that organ.

It offers the advantage of portability over tape-based recording systems. The project explains how a Arduino uno microcontroller can be used to analyze heart beat rate signals in real-time. The Hardware and software design are oriented towards Arduino uno-based system, hence minimizing the size.

Contents

Acknowledgement	i
Abstract	ii
1 INTRODUCTION	1
2 LITERATURE REVIEW	3
2.1 Introduction.	3
2.2 Heart rate measurement	4
2.3 Heart attack.	6
3 HARDWARE DESIGN	10
3.1 Electronic Hardware Design	10
3.1.1 Electronic Components Used	10
4 SOFTWARE DESIGN	22
4.1 Software Requirements.	22
5 WORKING	29
5.1 General Layout of System	29
5.2 code for Arduino uno	30
6 RESULTS AND DISSCUSION	44
7 CONCLUSION	45
Bibliography	46

List of Figures

2.1	Blood flow through the heart	4
2.2	manual method	5
2.3	monitor method	6
2.4	Heart attack.	7
2.5	Heart attack symptoms.	8
3.1	Arduino uno.....	11
3.2	Lcd display	12
3.3	pulse sensor.....	14
3.4	pulse sensor circuit	15
3.5	LM 35	16
3.6	BREADBOARD	17
3.7	ESP 8266-01	18
3.8	led diode	18
3.9	USB A to B cable.....	19
3.10	Resistors	20
3.11	jumper wires.....	20
4.1	STEP 1 for setting thingspeak	24
4.2	STEP 2 for setting thingspeak	25
4.3	STEP 3 for setting thingspeak	25
4.4	STEP 4 for setting thingspeak	26
4.5	STEP 5 for setting thingspeak	26
4.6	STEP 6 for setting thingspeak	27
5.1	GENERAL LAYOUT OF SYSTEM	29

5.2	code page 1	30
5.3	code page 2	31
5.4	code page 3	32
5.5	code page 4	32
5.6	code page 5	33
5.7	code page 6	34
5.8	code page 7	34
5.9	code page 8	35
5.10	code page 9	36
5.11	code page 10.....	37
6.1	working model of project.....	38
6.2	Project readings 1	39
6.3	Project readings 2	40
6.4	Project readings 3	41
6.5	Project readings 4	42
6.6	Thingspeak	43

List of Tables

3.1	Electronic Components Used.....	10
3.2	Pin Description of LCD	13

Chapter 1

INTRODUCTION

This chapter describes an overall of this project including a background, a problem statement and the objectives of the project.

* BACKGROUND

Cardiovascular disease is one of the main causes of death in the many countries, it accounted for over 15 million deaths worldwide. In addition, several million people are disabled by cardiovascular disease. The delay between the first symptom of any cardiac ailment and the call for medical assistance has a large variation among different patients and can have fatal consequences.

One critical inference drawn from epidemiological data is that deployment of resources for early detection and treatment of heart disease has a higher potential of reducing fatality associated with cardiac disease than improved care after hospitalization. Hence new strategies are needed in order to reduce time before treatment. Monitoring of patients is one possible solution.

* PROBLEM STATEMENT

Some severe diseases and disorders e.g. heart failure needs close and continual monitoring procedure after diagnosis, in order to prevent mor-

tality or further damage as secondary to the mentioned diseases or disorders. Monitoring these types of patients, usually, occur at hospitals or healthcare centers. Heart arrhythmias for instance, in many cases, need continual long-term monitoring. However, the patients are often too early released, owing to need of hospital bed for another patient on the waiting list, who needs to be hospitalized immediately.

***OBJECTIVES**

The aim of our project is to provide a heart attack detection and heart beat monitoring device by sensing the heart rate of the patient using the wearable device (heart rate band) that will then send data to the Thingspeak application through Wifi / CVS file / mqtt. The Thingspeak application will process the data, and plot the graph for the heart beats vs time.

Chapter 2

LITERATURE REVIEW

2.1 Introduction

The heart is the strongest muscle that is responsible for pumping blood and delivering oxygen all around the body. Regardless it's just the size of a person's fist; its composed of four main chambers (two atria and two ventricles).

The following figure shows blood flow through the heart starting from the right atrium which receives oxygen-poor blood from the body and pumps it to the right ventricle; the right ventricle pumps the oxygen-poor blood to the lungs. Then the left atrium receives oxygen-rich blood from the lungs and pumps it to the left ventricle. Finally, the left ventricle pumps the oxygen-rich blood to the body. The heart rate is the number of heart beats per minute based on the number of contractions of the ventricles. The heart rate differ from one person to another due to several factors such as age, body mass, athleticism, obesity, medication, alcohol use, smoking, gender and physical activity level.

For individuals who have certain heart conditions, the heart may not efficiently push blood through the body with each contraction. These individuals have a pulse that is lower than their heart rate. The heart rate is often synchronized with the pulse. However, heart rate and pulse rate are technically different because a heart rate measures the rate

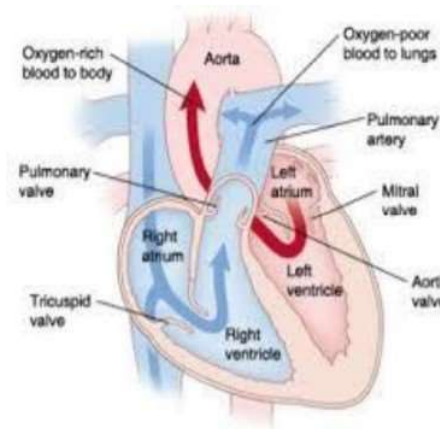


Figure 2-1: Blood flow through the heart

Figure 2.1: Blood flow through the heart

of (heart beats) of the heart, whereas a pulse rate measures the rate of palpable blood pressure increases throughout the body. Normally, healthy adults who are reasonably fit and overweight, and do not smoke or drink heavily, will have resting heart rates between 60 and 100 beats per minute (bpm).

Average, healthy teenager heart rates are the same as those for adults, while children under 10 years of age experience higher heart rates and pulses:

- *Newborns (70-190 bpm).
- * Infants (80-160 bpm).
- * Toddlers (80-130 bpm).
- * Preschoolers (80-120bpm).
- * Elementary Age (70-115 bpm).

These ranges help in determining the normality of the heartbeat.

2.2 Heart rate measurement

The heart rate measurement is essential for providing a person's cardiovascular fitness. There are several spots on the body that the heart rate

can be taken from, at which an artery is close to the surface and a pulse can be felt. The most common places to measure heart rate is at the wrist (radial artery) and the neck (carotid artery) using the palpation method. Other places sometimes used are the elbow (brachial artery) and the groin (femoral artery).

There are two methods are used to measure the heart rate:

1. Manual Method

In this method there are two places where the beats can be measured using fingers:

* Wrist (radial artery): here the index and middle fingers are used together where they are placed on the opposite wrist, about $\frac{1}{2}$ inch on the inside of the joint, in line with the index finger. Once beats are found, then the number of beats can be counted within a one-minute period.

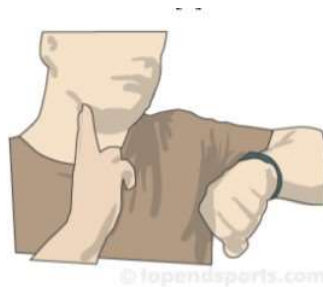


Figure 2.2: manual method

*Neck (carotid pulse): to measure the heart rate at the neck, the first two fingers placed on either side of the neck until the beats are felt.

2. Monitor Method

To get more accurate heart rate measurement, a heart rate monitor (ECG/EEG) can be used. There are many devices and phone Applications that can be used to measure the heart rate such as wristbands, chest straps and armbands, with most delivering digital readouts which designed primarily for tracking the pulses.

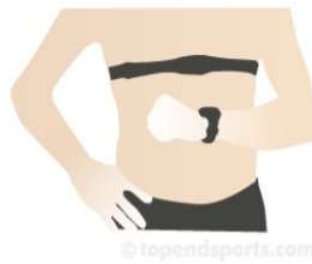


Figure 2.3: monitor method

This is particularly important during exercise where the motion of exercise often makes it hard to get a clear measurement using the manual method. Using a heart rate monitor is also useful when you wish to record heart rate changes over short time periods, where the heart rate may be changing. Many heart rate monitors require at least a little body perspiration between the chest strap and the skin for best conduction of the signal.

Using those devices may also help individuals with abnormal heart rate conditions to determine what causes them.

2.3 Heart attack

The heart muscle requires a constant supply of oxygen-rich blood to nourish it. This critical blood supply provided by the coronary arteries. If there is a coronary artery disease, those arteries become narrow and blood cannot flow as well as they should. The accumulation of fat, cholesterol and other substances that form deposits in the arteries that feed the heart causes stop of the flow of the blood.

The plaque deposits are hard on the outside and soft and mushy on the inside. When the plaque is hard, the outer shell cracks, platelets come to the area, and blood clots from around the plaque. If the artery totally

blocked by a blood clot, the heart muscle becomes starved for oxygen. Within a short time, death of heart muscle cells occurs, causing permanent damage. This is called a “heart attack”.

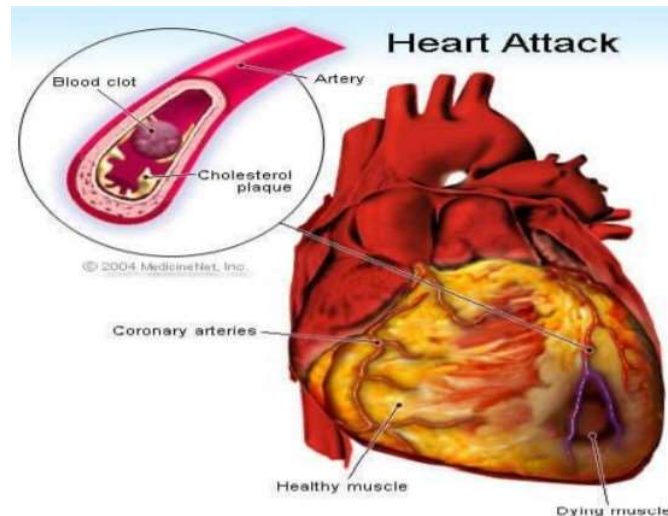


Figure 2.4: Heart attack

The amount of damage to the heart muscle depends on the size of the area supplied by the blocked artery and the time between injury and treatment. In the past, heart attacks often ended in death. But today, most of people that have heart attacks are still alive and this is due to increased awareness of signs and symptoms of heart attacks and the development of treatment methods and detection. In addition, the general lifestyle, types of food, stress and tension, all of those have an important role in recovering from heart attacks.

Heart attack symptoms

Heart attack has several signs and symptoms which includes chest pain, jaw pain, toothache, headache, shortness of breath, nausea (less common but possible symptom), vomiting, general discomfort, sweating, heart-burn and indigestion, arm pain (more commonly the left arm), upper back pain, general malaise (vague feeling of ill ness).

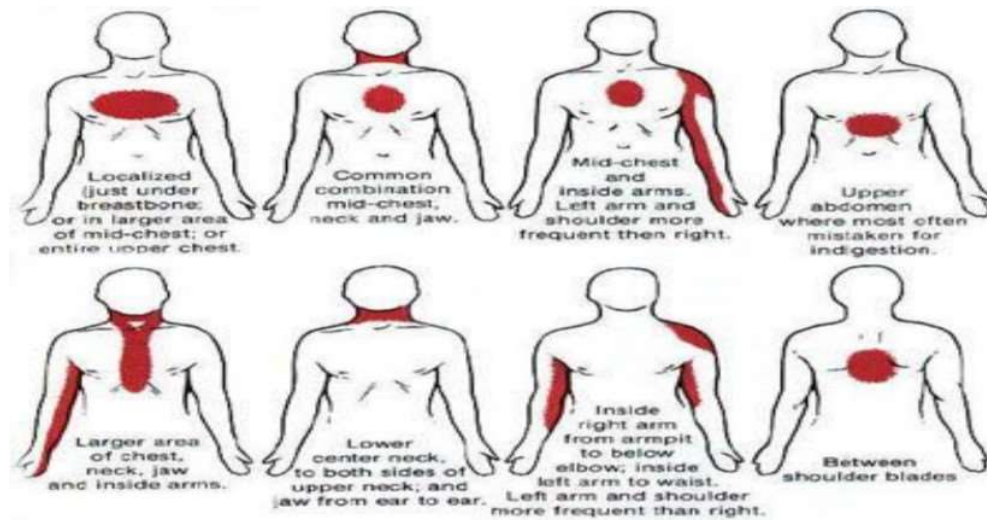


Figure 2.5: Heart attack symptoms

Sometimes there are no symptoms, this case called “silent heart attack” which is occurring with mild symptoms and may seem unrelated to the heart. A heart attack can occur anytime, anywhere, but there are many warning signs occur before heart attack, therefore, you should note and not underestimate any of the previous symptoms, which may help in early detection and treatment of heart attack.

Heart attack detection

Heart attack detection based on patients’ signs and symptoms, medical and family histories, and test results. There are several diagnostic tests used to detect the heart attack such as:

1. EKG (Electrocardiogram)

An EKG is a simple, painless test that detects and records the heart’s electrical activity. The test shows how fast the heart is beating and its rhythm (steady or irregular). An EKG also records the strength and timing of electrical signals as they pass through each part of the heart. An EKG can show signs of heart damage due to coronary heart disease (CHD) and signs of a previous or current heart attack.

2. Blood Tests

During a heart attack, heart muscle cells die and release proteins into the bloodstream. Blood tests can measure the amount of these proteins in the bloodstream. Higher than normal levels of these proteins suggest a heart attack. Commonly used blood tests include troponin tests, CK or CK–MB tests, and serum myoglobin tests. Blood tests often are repeated to check for changes over time.

3. Coronary Angiography

Coronary angiography (an-jee-OG-ra-fee) is a test that uses dye and special x rays to show the insides of your coronary arteries. This test often is done during a heart attack to help find blockages in the coronary arteries. A thin, flexible tube called a catheter is put into a blood vessel in your arm, groin (upper thigh), or neck. The tube is threaded into your coronary arteries, and the dye is released into your bloodstream.

Special x rays are taken while the dye is flowing through the coronary arteries. The dye lets your doctor study the flow of blood through the heart and blood vessels. If your doctor finds a blockage, he may recommend a procedure called percutaneous (per-kuTA-ne-us) coronary intervention (PCI), sometimes referred to as coronary angioplasty (AN-jeeoh-plas-tee). This procedure can help restore blood flow through a blocked artery.

Sometimes a small mesh tube called a stent is placed in the artery to help prevent blockages after the procedure.

Chapter 3

HARDWARE DESIGN

3.1 Electronic Hardware Design

3.1.1 Electronic Components Used

Sr.No.	Component Name	Quantity Required
1	Arduino uno	1
2	LCD Display 16×2	1
3	pulse Sensor	1
4	Lm 35 temperature sensor	1
5	Breadboard	1
6	ESP8266-01 wifi module	1
7	LED	1
8	USB A TO B cable	1
9	resistor 1 kilohm	1
10	resistor 2 kilohm	1
11	Jumper wires(male to male)	15
12	Jumper wires (male to female)	10

Table 3.1: Electronic Components Used

1.Arduino uno :

Arduino/Genuino Uno is a microcontroller board based on the AT-mega328P . It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz quartz crystal, a USB

connection, a power jack, an ICSP header and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started. You can tinker with your UNO without worrying too much about doing something wrong, worst case scenario you can replace the chip for a few dollars and start over again.



Figure 3.1: Arduino uno

Technical specs

Microcontroller ATmega328P

Operating Voltage - 5V

Input Voltage (recommended)- 7-12V

Input Voltage (limit)- 6-20V

Digital I/O Pins- 14 (of which 6 provide PWM output)

PWM Digital I/O Pins- 6

Analog Input Pins - 6

DC Current per I/O Pin- 20 mA

DC Current for 3.3V Pin- 50 mA

Flash Memory- 32 KB (ATmega328P) of which 0.5 KB used by boot-loader

SRAM- 2 KB (ATmega328P)

EEPROM- 1 KB (ATmega328P)

Clock Speed- 16 MHz

LED BUILTIN - 13

Length- 68.6 mm

Width- 53.4 mm

Weight- 25 g

2.LCD:

JHD16×2 LCD (Liquid Crystal Display) screen is an electronic display module and find a wide range of applications. A 16×2 LCD display is very basic module and is very commonly used in various devices and circuits. These modules are preferred over seven segments and other multi segment LEDs. The reasons being: LCDs are economical; easily programmable; have no limitation of displaying special and even custom characters (unlike in seven segments), animations and so on. A 16x2 LCD means it can display 16 characters per line and there are 2 such lines. In this LCD each character is displayed in 5×7 pixel matrix. This LCD has two registers, namely, Command and Data. This module has 14 Pin. Out of which; 8 are data lines, 3 are control lines and 3 power lines. Table shows pin out of LCD.



Figure 3.2: lcd display

Pin No.	Description
1	GND(Ground)
2	VCC(Supply Voltage)
3	VEE(Contrast Voltage)
4	R/S(Instruction Register)
5	R/W(Read/Write)
6	E(Enable)
7	D0(Data0)
8	D1(Data1)
9	D2(Data2)
10	D3(Data3)
11	D4(Data4)
12	D5(Data5)
13	D6(Data6)
14	D7(Data7)

Table 3.2: Pin Description of LCD

3.pulse sensor:

The pulse sensor is an IR sensor having a photocell which is used to emit and detect the infrared light. It is essentially a photoplethysmography, which is a well-known medical device used for non-invasive heart rate monitoring.

Principle of Heartbeat Sensor:

The pulse sensor is based on the principle of photo plethysmography. It measures the change in volume of blood through any organ of the body which causes a change in the light intensity through that organ (a vascular region). In case of applications where heart pulse rate is to be monitored, the timing of the pulses is more important. The flow of blood volume is decided by the rate of heart pulses and since light is absorbed by blood, the signal pulses are equivalent to the heart beat pulses.



Figure 3.3: pulse sensor

There are two types of photoplethysmography:

Transmission: Light emitted from the light emitting device is transmitted through any vascular region of the body like earlobe and received by the detector.

Reflection: Light emitted from the light emitting device is reflected by the regions.

Working of a Heartbeat Sensor: The basic heartbeat sensor consists of a light emitting diode and a detector like a light detecting resistor or a photodiode.

The heart beat pulses cause a variation in the flow of blood to different regions of the body. When a tissue is illuminated with the light source, it either reflects (a finger tissue) or transmits the light (earlobe). Some of the light is absorbed by the blood and the transmitted or the reflected light is received by the light detector. The amount of light absorbed depends on the blood volume in that tissue. The detector output is in form of electrical signal and is proportional to the heart beat rate.

This signal is actually a DC signal relating to the tissues and the blood volume and the AC component synchronous with the heart beat and caused by pulsatile changes in arterial blood volume is superimposed on the DC signal. Thus, the major requirement is to isolate that AC component as it is of prime importance.

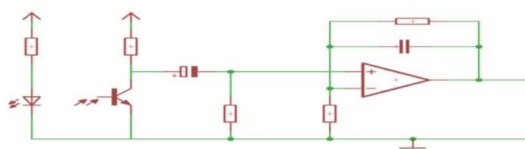


Figure 3.4: pulse sensor circuit

To achieve the task of getting the AC signal, the output from the detector is first filtered using a 2 stage HP-LP circuit and is then converted to digital pulses using a comparator circuit or using simple ADC. The digital pulses are given to a microcontroller for calculating the heart beat rate, given by the formula

$$\text{BPM (Beats per minute)} = 60 * f \text{ Where } f \text{ is the pulse frequency.}$$

4.LM 35 temperature sensor:

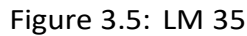
Introduction:

In general, a temperature sensor is a device which is designed specifically to measure the hotness or coldness of an object. LM35 is a precision IC temperature sensor with its output proportional to the temperature (in °C). With LM35, the temperature can be measured more accurately than with a thermistor.

It also possesses low self-heating and does not cause more than 0.1 °C temperature rise in still air. The operating temperature range is from -55°C to 150°C. The LM35 low output impedance, linear output, and precise inherent calibration make interfacing to readout or control circuitry especially easy.

Working principle:

There are two transistors in the center of the drawing. One has ten



The two resistors are calibrated in the factory to produce a highly accurate temperature sensor. The integrated circuit has many transistors in it – two in the middle, some in each amplifier, some in the constant current source, and some in the curvature compensation circuit. All of

that is fit into the tiny package with three leads.

5. Breadboard:



Figure 3.6: BREADBOARD

Very simply a Breadboard contains rows of holes in which you can stick components and create a circuit. Each row (usually of about 5 holes) is connected, so two different components can be connected together by just one row - no extra wires involved!

6.ESP8266-01:

The ESP8266 ESP-01 is a Wi-Fi module that allows microcontrollers access to a Wi-Fi network. This module is a self-contained SOC (System On a Chip) that doesn't necessarily need a microcontroller to manipulate inputs and outputs as you would normally do with an Arduino.

for example, because the ESP-01 acts as a small computer. Depending on the version of the ESP8266, it is possible to have up to 9 GPIOs (General Purpose Input Output).

Thus, we can give a microcontroller internet access like the Wi-Fi shield does to the Arduino, or we can simply program the ESP8266 to not only have access to a Wi-Fi network, but to act as a microcontroller as well. This makes the ESP8266 very versatile, and it can save you some money and space in your projects.

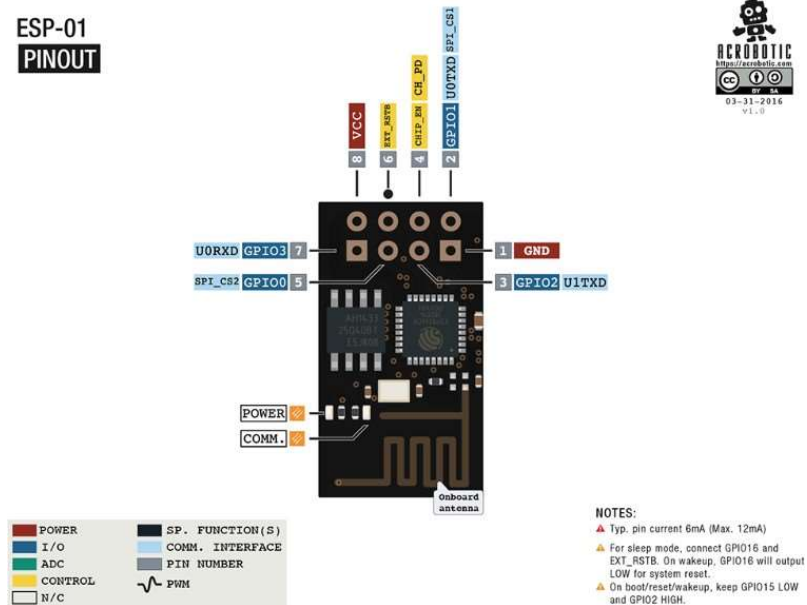


Figure 3.7: ESP 8266-01

7.LED



Figure 3.8: led diode

A light-emitting diode (LED) is a semiconductor light source that emits light when current flows through it. Electrons in the semiconductor recombine with electron holes, releasing energy in the form of photons.

The color of the light (corresponding to the energy of the photons) is determined by the energy required for electrons to cross the band gap of the semiconductor.

8. USB A TO B CABLE

The initial original designed USB for the rectangular and flat USB port is USB-A. It is also known as standard-A. The USB type A is supported by almost every USB system. In contrast, when you are working with ultra-speed USB applications or devices, USB type-B connector is used to carry power and data.

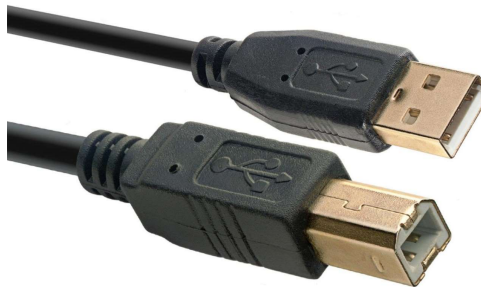


Figure 3.9: usb A to B cable

USB type A is rectangular whereas USB type B is available in square shape or multiple designs, however, none of them is reversible. The USB Type B is backward compatible with other hardware versions while USB type A is not. USB Type-A connectors are always compatible with the type-A socket present on the host device.

9. RESISTORS

A passive electrical component with two terminals that are used for either limiting or regulating the flow of electric current in electrical circuits.

The main purpose of resistor is to reduce the current flow and to lower the voltage in any particular portion of the circuit. It is made of copper wires which is coiled around a ceramic rod and the outer part of the resistor is coated with an insulating paint.



Figure 3.10: Resistors

Applications of Resistor

Following are the applications of resistors:

Wire wound resistors find application where balanced current control, high sensitivity, and accurate measurement are required like in shunt with ampere meter.

Photo resistors find application in flame detectors, burglar alarm, in photographic devices, etc.

Resistors are used for controlling temperature and voltmeter.

Resistors are used in digital multi-meter, amplifiers, telecommunication, and oscillators.

They are also used in modulators, demodulators, and transmitters.

10. JUMPERS WIRE

A jump wire (also known as jumper, jumper wire, DuPont wire) is an electrical wire, or group of them in a cable, with a connector or pin at each end (or sometimes without them – simply “tinned”), which is normally used to interconnect the components of a breadboard or other prototype or test circuit, internally or with other equipment or components, without soldering. Individual jump wires are fitted by insert-



Figure 3.11: jumper wires

ing their” end connectors” into the slots provided in a breadboard, the header connector of a circuit board, or a piece of test equipment.

Chapter 4

SOFTWARE DESIGN

4.1 Software Requirements

1.Arduino IDE: Arduino is a both an open-source software library and an open-source breakout board for the popular AVR micro-controllers. The Arduino IDE (Integrated Development Environment) is the program used to write code, and comes in the form of a downloadable file on the Arduino website. The Arduino board is the physical board that stores and performs the code uploaded to it. Both the software package and the board are referred to as 0Arduino.

Connecting the Arduino Connecting an Arduino board to your PC is quite simple. On Windows:

1. Plug in the USB cable - one end to the PC, and one end to the Arduino board.
2. When prompted, select” Browse my computer for driver” and then select the folder to which you extracted your original Arduino IDE download.
3. You may receive an error that the board is not a Microsoft certified device - select “Install anyway.”
4. Your board should now be ready for programming.

Preparing the Board Before loading any code to your Arduino board, you must first open the IDE. Double click the Arduino .exe file that you downloaded earlier. A blank program, or” sketch,” should open.

The Blink example is the easiest way to test any Arduino board. Within the Arduino window, it can be found under File-Examples- Basics-Blink.

Before the code can be uploaded to your board, two important steps are required.

1. Select your Arduino from the list under Tools-Board. The standard board used in RBE 1001, 2001, and 2002 is the Arduino Mega 2560, so select the "Arduino Mega 2560 or Mega ADK" option in the dropdown.

2. Select the communication port, or COM port, by going to Tools-Serial Port.

If you noted the COM port your Arduino board is using, it should be listed in the dropdown menu. If not, your board has not finished installing or needs to be reconnected.

Loading Code

The upper left of the Arduino window has two buttons: A checkmark to Verify your code, and a right-facing arrow to Upload it. Press the right arrow button to compile and upload the Blink example to your Arduino board.

The black bar at the bottom of the Arduino window is reserved for messages indicating the success or failure of code uploading. A "Completed Successfully" message should appear once the code is done uploading to your board. If an error message appears instead, check that you selected the correct board and COM port in the Tools menu, and check your physical connections.

If uploaded successfully, the LED on your board should blink on/off once every second. Most Arduino boards have an LED prewired to pin 13.

It is very important that you do not use pins 0 or 1 while loading code. It is recommended that you do not use those pins ever.

2.Thingspeak:

ThingSpeak is an open-source Internet of Things (IoT) application and API to store and retrieve data from things using the HTTP and MQTT 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. ThingSpeak is an open cloud data platform where you can store and retrieve data.

SETTING UP THINGSPEAK STEP 1

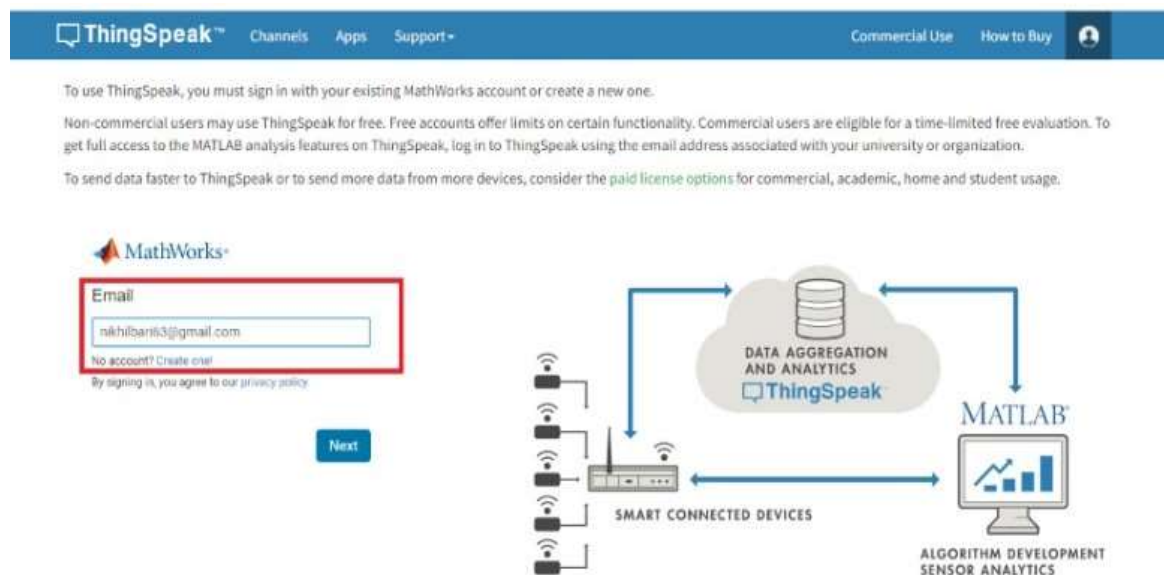


Figure 4.1: STEP 1 for setting thingspeak

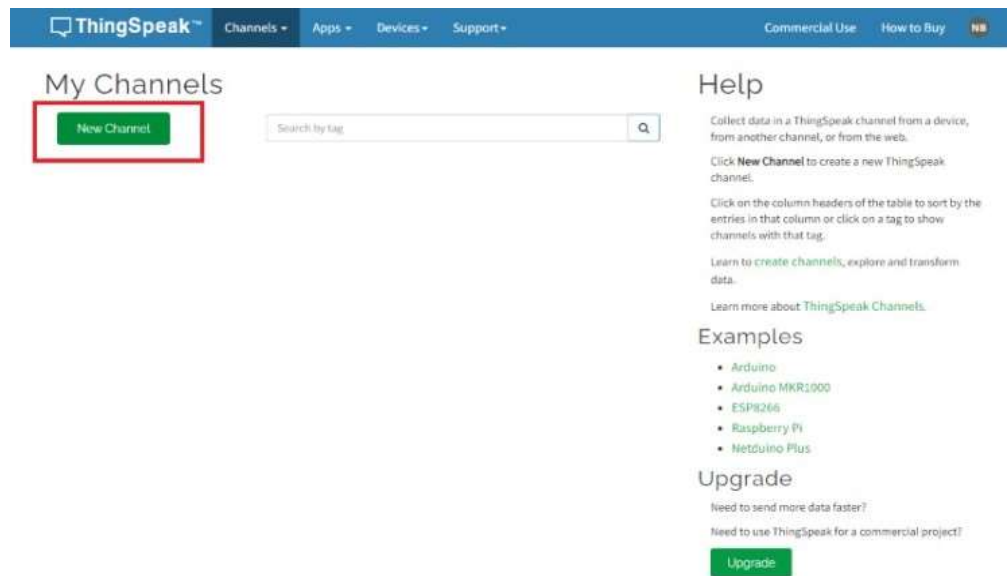


Figure 4.2: STEP 2 for setting thingspeak

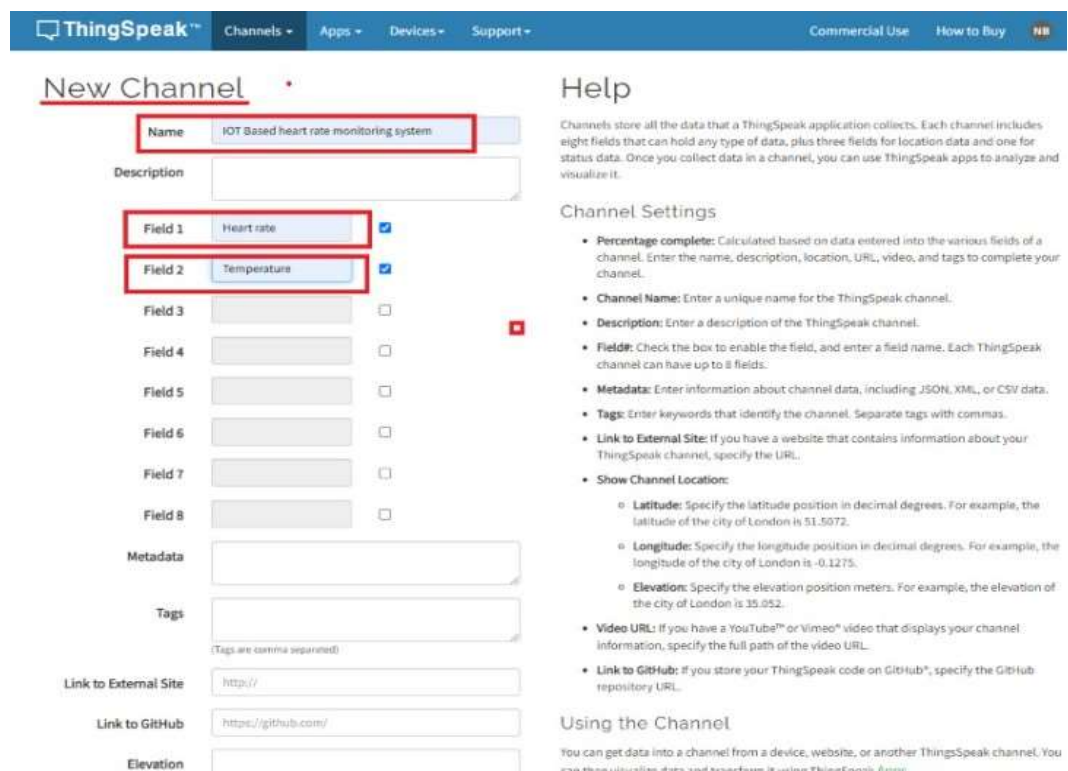


Figure 4.3: STEP 3 for setting thingspeak

ThingSpeak™ Channels Apps Devices Support Commercial Use How to Buy

Field #

Metadata

Tags
(Tags are comma separated)

Link to External Site

Link to GitHub

Elevation

Show Channel Location ☐

Latitude

Longitude

Show Video ☐

☒ YouTube ☐ Vimeo

Video URL

Show Status ☐

Save Channel

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 longitude of the city of London is -0.1275.

Elevation: Specify the elevation position in meters. For example, the elevation of the city of London is 35.052.

Video URL: If you have a YouTube™ or Vimeo® video that displays your channel information, specify the full path of the video URL.

Link to GitHub: If you store your ThingSpeak code on GitHub®, specify the GitHub repository URL.

Using the Channel

You can get data into a channel from a device, website, or another ThingSpeak channel. You can then visualize data and transform it using ThingSpeak Apps.

See [Get Started with ThingSpeak™](#) for an example of measuring dew point from a weather station that acquires data from an Arduino® device.

[Learn More](#)

Figure 4.4: STEP 4 for setting thingspeak

ThingSpeak™ Channels Apps Devices Support Commercial Use How to Buy

IOT Based heart rate monitoring system

Channel ID: 1737251
Author: mwa0000017612001
Access: Private

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

Write API Key

Key **VYHTG087YGJEN8UK**

Generate New Write API Key

Read API Keys

Key **SKIB8LM1YKLE65C8**

Note

Save Note **Delete API Key**

Add New Read API Key

Help

API keys enable you to write data to a channel or read data from a private channel. API keys are auto-generated when you create a new channel.

API Keys Settings

- Write API Key:** Use this key to write data to a channel. If you feel your key has been compromised, click **Generate New Write API Key**.
- Read API Keys:** Use this key to allow other people to view your private channel feeds and charts. Click **Generate New Read API Key** to generate an additional read key for the channel.
- Note:** Use this field to enter information about channel read keys. For example, add notes to keep track of users with access to your channel.

API Requests

Write a Channel Feed

```
GET https://api.thingspeak.com/update?api_key=VYHTG087YGJEN8UK&field=
```

Read a Channel Feed

```
GET https://api.thingspeak.com/channels/1737251/feeds.json?api_key=
```

Read a Channel Field

```
GET https://api.thingspeak.com/channels/1737251/fields/1.json?api_key=
```

Figure 4.5: STEP 5 for setting thingspeak

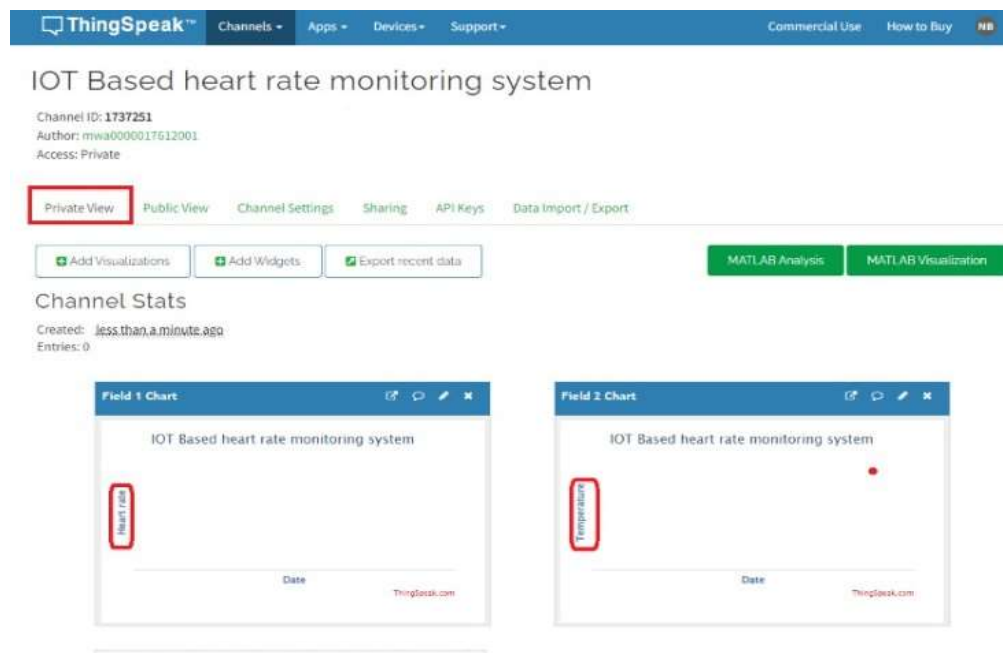


Figure 4.6: STEP 6 for setting thingspeak

AT COMMANDS AND THEIR FUNCTIONALITY

* AT: This type of command is used to test the startup function of WI-FI module. The response would be ok, against this command if everything is ok.

*AT+RST: This type of command is used for reset the WI-FI module when it is in working condition. The response would be ok, when reset the module.

*AT+GMR: This type of AT command is used to check the version of AT command and we used SDK version of AT command in this type of WI-FI module.

* AT+GSLP: This type of AT command is used to invoke the module from sleep mode, in other words this command is used for wake -up the module.

* ATE: This type of command is used for echo sound, means that, when

this command is entered then the echo send back to the sender.

*AT+RESTORE: This type of command is used for restore factory setting means, when this command is entered then all the parameters are reset automatically.

* AT+UART: This type of command is used for set the UART configuration and writes the new configuration in flash means, it is used for setting the default parameters.

* AT+SLEEP: This type of command is used to the set the module in sleep mode.

* AT+RFPOWER: This type of command is used for set the maximumvalue of power for ESP 8266 RXTF, but it is not precise.

* AT+RFVDD: This type of command is used for set the maximum valueof power for ESP 8266 RXTX according to the VDD33.

Chapter 5

WORKING

5.1 General Layout of System

Following figure shows us the general layout of the device, we will be using for measuring the pulse and the room temperature.

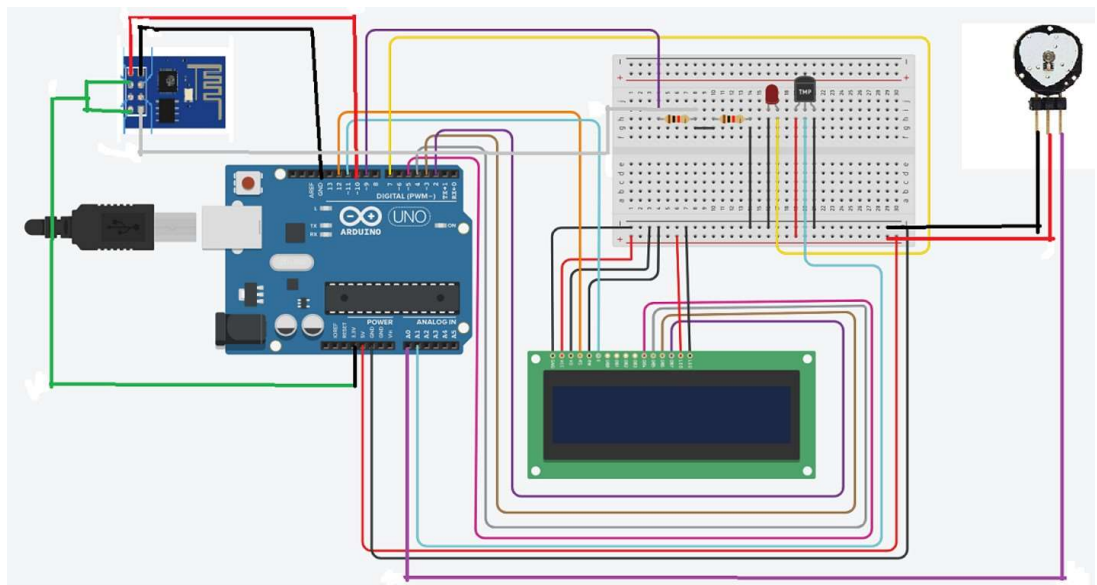


Figure 5.1: GENERAL LAYOUT OF SYSTEM

Here in this project, we have a IR based pulse sensor which will measure the heart pulse rate of the sensor when we placed then finger on the sensor also we have this LM35 temperature sensor which will measure

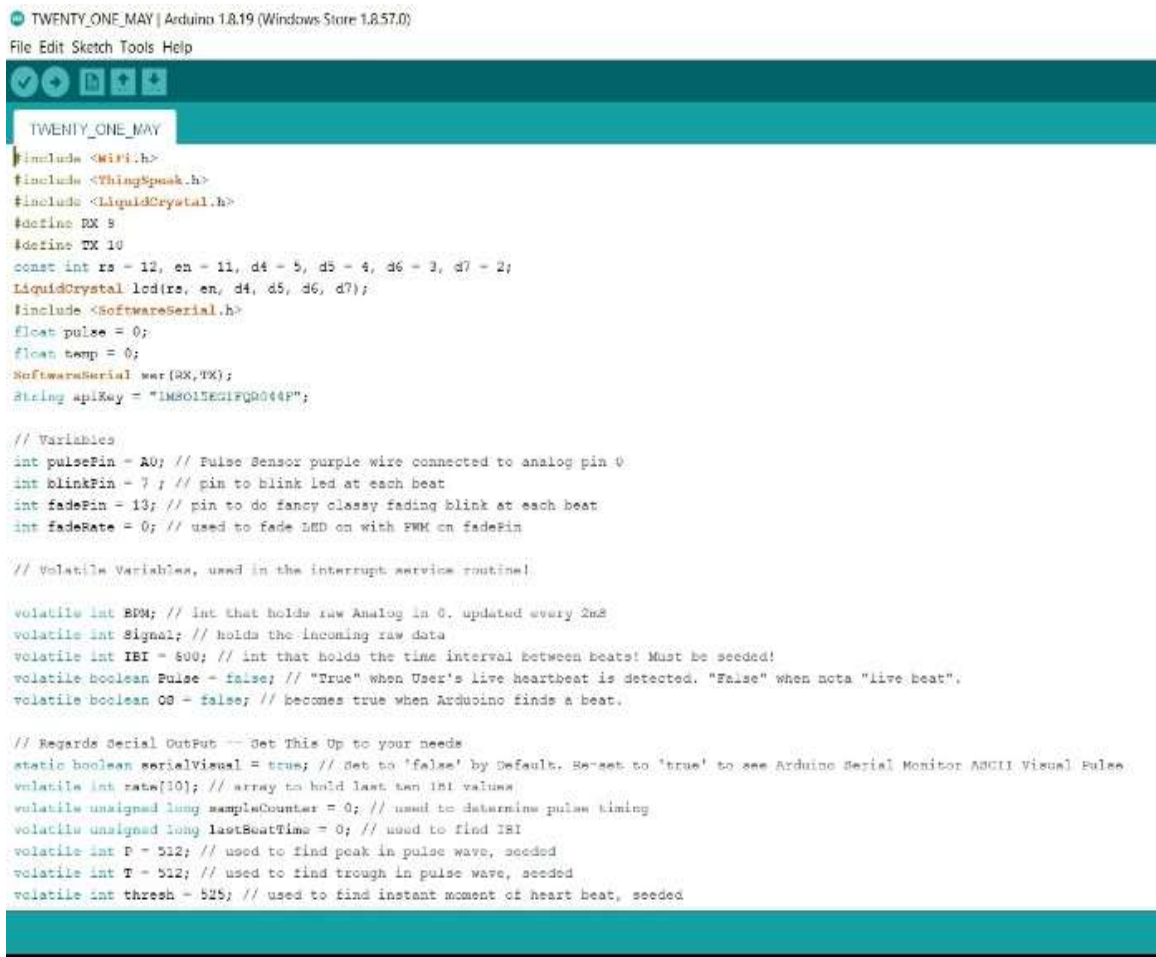
the room temperature.

This device will detect the pulse rate and temperature and will show the readings in BPM (Beats Per Minute) and TEMPERATURE in degree Fahrenheit on the LCD display.

Not only this It will send the readings to the ThingSpeak server via the Wi-Fi module ESP8266-01 or using CVS file or manually feeding the reading using the write key . This will help us to monitor the heartbeat and room temperature via the thingspeak from any part of the world.

ThingSpeak 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.

5.2 CODE FOR ARDUINO IDE



```

TWENTY_ONE_MAY | Arduino 1.8.19 (Windows Store 1.8.57.0)
File Edit Sketch Tools Help

TWENTY_ONE_MAY

#include <WiFi.h>
#include <ThingSpeak.h>
#include <LiquidCrystal.h>
#define RX 9
#define TX 10
const int rs = 12, en = 11, d4 = 5, d5 = 4, d6 = 3, d7 = 2;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);
#include <SoftwareSerial.h>
float pulse = 0;
float temp = 0;
SoftwareSerial ser(RX, TX);
String apiKey = "1M8015E61FG0044P";

// Variables
int pulsePin = A0; // Pulse Sensor purple wire connected to analog pin 0
int blinkPin = 7; // pin to blink led at each beat
int fadePin = 13; // pin to do fancy classay fading blink at each beat
int fadeRate = 0; // used to fade LED on with PWM on fadePin

// Volatile Variables, used in the interrupt service routine!

volatile int BPM; // int that holds raw Analog in 0. updated every 2ms
volatile int Signal; // holds the incoming raw data
volatile int IBI = 600; // int that holds the time interval between beats! Must be seeded!
volatile boolean Pulse = false; // "True" when User's live heartbeat is detected. "False" when not a "live beat".
volatile boolean QS = false; // becomes true when Arduino finds a beat.

// Regards Serial Output -- Get This Up to your needs
static boolean serialVisual = true; // Set to 'false' by Default. Re-set to 'true' to see Arduino Serial Monitor ASCII Visual Pulse
volatile int rate[10]; // array to hold last ten IBI values
volatile unsigned long sampleCounter = 0; // used to determine pulse timing
volatile unsigned long lastBeatTime = 0; // used to find IBI
volatile int P = 512; // used to find peak in pulse wave, seeded
volatile int T = 512; // used to find trough in pulse wave, seeded
volatile int thresh = 525; // used to find instant moment of heart beat, seeded

```

```
TWENTY_ONE_MAY

volatile int thresh = 525; // used to find instant moment of heart beat, seeded
volatile int amp = 100; // used to hold amplitude of pulse waveform, seeded
volatile boolean firstBeat = true; // used to seed rate array so we startup with reasonable BPM
volatile boolean secondBeat = false; // used to seed rate array so we startup with reasonable BPM

void setup()
{
  lcd.begin(16, 2);
  pinMode(blinkPin, OUTPUT); // pin that will blink to your heartbeat!
  pinMode(fadePin, OUTPUT); // pin that will fade to your heartbeat!
  Serial.begin(9600); // we agree to talk fast!
  interruptSetup(); // sets up to read Pulse Sensor signal every 2mS

  // IF YOU ARE POWERING The Pulse Sensor AT VOLTAGE LESS THAN THE BOARD VOLTAGE,

  // UN-COMMENT THE NEXT LINE AND APPLY THAT VOLTAGE TO THE A-REF PIN

  // analogReference(EXTERNAL);

  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print(" Patient Health");
  lcd.setCursor(0,1);
  lcd.print(" Monitoring ");
  delay(4000);
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("Initializing....");
  delay(5000);
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("Getting Data....");
  ser.begin(115200);
  ser.println("AT");
  delay(1000);
```

Figure 5.3: code page 2

```

delay(1000);
ser.println("AT+GMR");
delay(1000);
ser.println("AT+CWMODE=3");
delay(1000);
ser.println("AT+RST");
delay(5000);
ser.println("AT+CIPMUX=1");
delay(1000);

String cmd="AT+CWJAP=\"honor 7\", \"19992000\"";
ser.println(cmd);
delay(1000);
ser.println("AT+CIFSR");
delay(1000);
}

// Where the Magic Happens
void loop()
{
  serialOutput();
  if (QS == true) // A Heartbeat Was Found
  {

    // BPM and IBI have been Determined
    // Quantified Self "QS" true when arduino finds a heartbeat
    fadeRate = 255; // Makes the LED Fade Effect Happen, Set 'fadeRate' Variable to 255 to fade LED with pulse
    serialOutputWhenBeatHappens(); // A Beat Happened, Output that to serial.
    QS = false; // reset the Quantified Self flag for next time
  }
  ledFadeToBeat(); // Makes the LED Fade Effect Happen
  delay(20); // take a break
  read_temp();
  esp_8266();
}

```

Figure 5.4: code page 3

```

esp_8266();
}
void ledFadeToBeat()
{
  fadeRate -= 15; // set LED fade value
  fadeRate = constrain(fadeRate,0,255); // keep LED fade value from going into negative numbers!
  analogWrite(fadePin,fadeRate); // fade LED
}
void interruptSetup()
{
  // Initializes Timer2 to throw an interrupt every 2mS.
  TCCR2A = 0x02; // DISABLE PWM ON DIGITAL PINS 3 AND 11, AND GO INTO CTC MODE
  TCCR2B = 0x06; // DON'T FORCE COMPARE, 256 PRESCALER
  OCR2A = 0x7C; // SET THE TOP OF THE COUNT TO 124 FOR 500Hz SAMPLE RATE
  TIMSK2 = 0x02; // ENABLE INTERRUPT ON MATCH BETWEEN TIMER2 AND OCR2A
  sei(); // MAKE SURE GLOBAL INTERRUPTS ARE ENABLED
}
void serialOutput()
{ // Decide How To Output Serial.
  if (serialVisual == true)
  {
    arduinoSerialMonitorVisual('-', Signal); // goes to function that makes Serial Monitor Visualizer
  }
  else
  {
    sendDataToSerial('S', Signal); // goes to sendDataToSerial function
  }
}
void serialOutputWhenBeatHappens()
{
  if (serialVisual == true) // Code to Make the Serial Monitor Visualizer Work
  {
    Serial.print("*** Heart-Beat Happened *** "); //ASCII Art Madness
    Serial.print("BPM: ");
    Serial.println(BPM);
  }
}

```

Figure 5.5: code page 4


```
Serial.println(BPM);  
}  
else  
{  
  sendDataToSerial('B',BPM); // send heart rate with a 'B' prefix  
  sendDataToSerial('Q',IBI); // send time between beats with a 'Q' prefix  
}  
}  
void arduinoSerialMonitorVisual(char symbol, int data )  
{  
  const int sensorMin = 0; // sensor minimum, discovered through experiment  
  const int sensorMax = 1024; // sensor maximum, discovered through experiment  
  int sensorReading = data; // map the sensor range to a range of 12 options:  
  int range = map(sensorReading, sensorMin, sensorMax, 0, 11);  
  // do something different depending on the  
  // range value:  
  switch (range)  
  {  
    case 0:  
      Serial.println(""); //ASCII Art Madness  
      break;  
    case 1:  
      Serial.println("---");  
      break;  
    case 2:  
      Serial.println("-----");  
      break;  
    case 3:  
      Serial.println("-----");  
      break;  
    case 4:  
      Serial.println("-----");  
      break;  
    case 5:  
      Serial.println("-----|-");  
  }
```

Figure 5.6: code page 5

```

case 5:
  Serial.println("-----|-");
  break;
case 6:
  Serial.println("-----|---");
  break;
case 7:
  Serial.println("-----|-----");
  break;
case 8:
  Serial.println("-----|-----");
  break;
case 9:
  Serial.println("-----|-----");
  break;
case 10:
  Serial.println("-----|-----");
  break;
case 11:
  Serial.println("-----|-----");
  break;
}

void sendDataToSerial(char symbol, int data )
{
  Serial.print(symbol);
  Serial.println(data);
}
ISR(TIMER2_COMPA_vect) //triggered when Timer2 counts to 124
{
  cli(); // disable interrupts while we do this
  Signal = analogRead(pulsePin); // read the Pulse Sensor
  sampleCounter += 2; // keep track of the time in mS with this variable
  int N = sampleCounter - lastBeatTime; // monitor the time since the last beat to avoid noise

```

Figure 5.7: code page 6

```

int N = sampleCounter - lastBeatTime; // monitor the time since the last beat to avoid noise
// find the peak and trough of the pulse wave

if(Signal < thresh && N > (IBI/5)*3) // avoid dichrotic noise by waiting 3/5 of last IBI
{
  if (Signal < T) // T is the trough
  {
    T = Signal; // keep track of lowest point in pulse wave
  }
}
if(Signal > thresh && Signal > P)
{ // thresh condition helps avoid noise
  P = Signal; // P is the peak
} // keep track of highest point in pulse wave
// NOW IT'S TIME TO LOOK FOR THE HEART BEAT
// signal surges up in value every time there is a pulse
if (N > 250)
{ // avoid high frequency noise
  if ( (Signal > thresh) && (Pulse == false) && (N > (IBI/5)*3) )
  {
    Pulse = true; // set the Pulse flag when we think there is a pulse
    digitalWrite(blinkPin,HIGH); // turn on pin 13 LED
    IBI = sampleCounter - lastBeatTime; // measure time between beats in mS
    lastBeatTime = sampleCounter; // keep track of time for next pulse

    if(secondBeat)
    { // if this is the second beat, if secondBeat == TRUE
      secondBeat = false; // clear secondBeat flag
      for(int i=0; i<=9; i++) // seed the running total to get a realistic BPM at startup
      {
        rate[i] = IBI;
      }
    }
    if(firstBeat) // if it's the first time we found a beat, if firstBeat == TRUE
    {

```

Figure 5.8: code page 7

```

,
}
if(firstBeat) // if it's the first time we found a beat, if firstBeat == TRUE
{
    firstBeat = false; // clear firstBeat flag
    secondBeat = true; // set the second beat flag
    sei(); // enable interrupts again
    return; // IBI value is unreliable so discard it
}
// keep a running total of the last 10 IBI values
word runningTotal = 0; // clear the runningTotal variable
for(int i=0; i<=8; i++)
{ // shift data in the rate array
    rate[i] = rate[i+1]; // and drop the oldest IBI value
    runningTotal += rate[i]; // add up the 9 oldest IBI values
}
rate[9] = IBI; // add the latest IBI to the rate array
runningTotal += rate[9]; // add the latest IBI to runningTotal
runningTotal /= 10; // average the last 10 IBI values
BPM = 60000/runningTotal; // how many beats can fit into a minute? that's BPM!
QS = true; // set Quantified Self flag
// QS FLAG IS NOT CLEARED INSIDE THIS ISR
pulse = BPM;
}
}

if (Signal < thresh && Pulse == true)
{ // when the values are going down, the beat is over
    digitalWrite(blinkPin,LOW); // turn off pin 13 LED
    Pulse = false; // reset the Pulse flag so we can do it again
    amp = P - T; // get amplitude of the pulse wave
    thresh = amp/2 + T; // set thresh at 50% of the amplitude
    P = thresh; // reset these for next time
    T = thresh;
}
if (N > 2500)
{ // if 2.5 seconds go by without a beat

```

Figure 5.9: code page 8

```

},
if (N > 2500)
{ // if 2.5 seconds go by without a beat
  thresh = 512; // set thresh default
  P = 512; // set P default
  T = 512; // set T default
  lastBeatTime = sampleCounter; // bring the lastBeatTime up to date
  firstBeat = true; // set these to avoid noise
  secondBeat = false; // when we get the heartbeat back
}
sei(); // enable interrupts when youre done!
} // end isr
void esp_8266()
{
  // TCP connection AT+CIPSTART=4,"TCP","192.168.4313",80
  String cmd = "AT+CIPSTART=4,\"TCP\"184.106.153.149\"";
  cmd += "\"; // api.thingspeak.com
  cmd += "\",80";
  ser.println(cmd);
  Serial.println(cmd);
  if(ser.find((char*)"Error"))
  {
    Serial.println("AT+CIPSTART error");
    return;
  }
  String getStr = "GET /update?api_key=";
  getStr += apiKey;
  getStr += "&field1=";
  getStr += String(temp);
  getStr += "&field2=";
  getStr += String(pulse);
  getStr += "\r\n\r\n";
  // send data length
  cmd = "AT+CIPSEND=4,";
  cmd += String(getStr.length());
  ser.println(cmd);

```

Figure 5.10: code page 9

```

getStr += "&field1=";
getStr += String(temp);
getStr += "&field2=";
getStr += String(pulse);
getStr += "\r\n\r\n";
// send data length
cmd = "AT+CIPSEND=4,";
cmd += String(getStr.length());
ser.println(cmd);
Serial.println(cmd);
delay(1000);
ser.print(getStr);
Serial.println(getStr); //thingspeak needs 15 sec delay between updates
delay(8000);
}

void read_temp()
{
int temp_val = analogRead(A1);
float mv = (temp_val/1024.0)*5000;
float cel = mv/10;
temp = (cel*9)/5 + 32;
Serial.print("Temperature:");
Serial.println(temp);
lcd.clear();
lcd.setCursor(0,0);
lcd.print("BPM :");
lcd.setCursor(7,0);
lcd.print(BPM);
lcd.setCursor(0,1);
lcd.print("Temp.:");
lcd.setCursor(7,1);
lcd.print(temp);
lcd.setCursor(13,1);
lcd.print("F");
}

```

Figure 5.11: code page 10

Chapter 6

RESULTS AND DISSCUSION

The implementation has been done by dividing the whole project into small parts (hardware and software) to achieve the project objectives. All hardware components have been integrated into one circuit as shown below: After we have connected all the components in the circuit, we

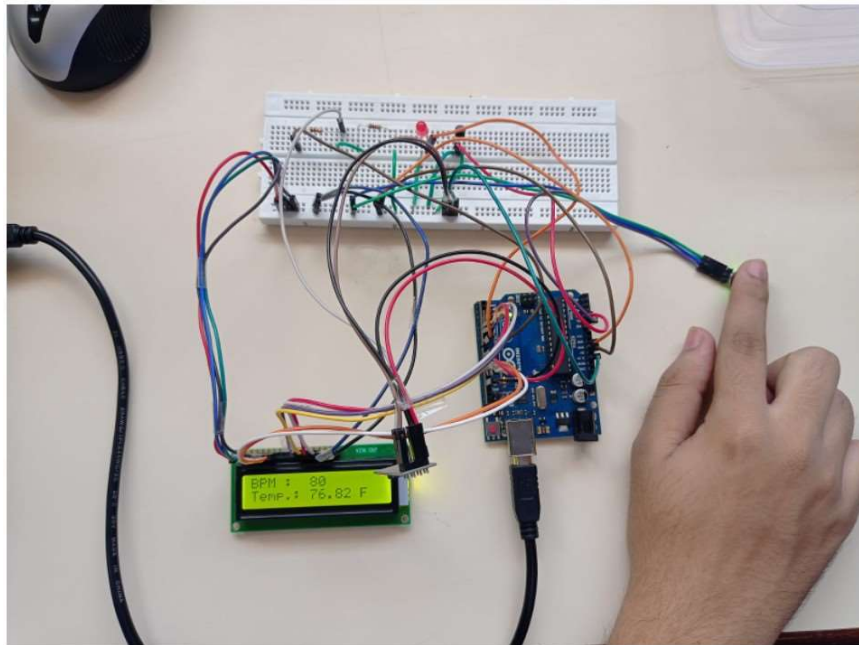


Figure 6.1: working model of project

connected this circuit with the computer and we loaded the code on the Arduino uno. This code contains many algorithms that run all the components.

When we opened the serial monitor of the Arduino programmer we obtain the following results:

```
*PROJECT READINGS 1 - Notepad
File Edit Format View Help
-----|
*** Heart-Beat Happened *** BPM: 83
Temperature:84.73
"AT+CIPSTART=4,""TCP""184.106.153.149""",80"
"AT+CIPSEND=4,67"
GET /update?api_key=1MSO15EGIFQR044F&field1=84.73&field2=189.00
""
""

---
*** Heart-Beat Happened *** BPM: 158
Temperature:90.01
"AT+CIPSTART=4,""TCP""184.106.153.149""",80"
"AT+CIPSEND=4,67"
GET /update?api_key=1MSO15EGIFQR044F&field1=90.01&field2=236.00
""
""

-----|---
*** Heart-Beat Happened *** BPM: 72
Temperature:90.01
"AT+CIPSTART=4,""TCP""184.106.153.149""",80"
"AT+CIPSEND=4,67"
GET /update?api_key=1MSO15EGIFQR044F&field1=90.01&field2=233.00
""
""

-----|---
*** Heart-Beat Happened *** BPM: 86
Temperature:93.52
"AT+CIPSTART=4,""TCP""184.106.153.149""",80"
"AT+CIPSEND=4,67"
GET /update?api_key=1MSO15EGIFQR044F&field1=93.52&field2=227.00
""
""

-----|---
*** Heart-Beat Happened *** BPM: 80
Temperature:76.82
"AT+CIPSTART=4,""TCP""184.106.153.149""",80"
"AT+CIPSEND=4,67"
GET /update?api_key=1MSO15EGIFQR044F&field1=76.82&field2=104.00
""
""

-----|---
*** Heart-Beat Happened *** BPM: 110
Temperature:79.46
"AT+CIPSTART=4,""TCP""184.106.153.149""",80"
"AT+CIPSEND=4,67"
GET /update?api_key=1MSO15EGIFQR044F&field1=79.46&field2=109.00
""
""
```

Figure 6.2: project readings 1

```

*PROJECT READINGS 1 - Notepad
File Edit Format View Help
***
*** Heart-Beat Happened *** BPM: 229
Temperature:94.40
"AT+CIPSTART=4,""TCP""184.106.153.149""",80"
"AT+CIPSEND=4,67"
GET /update?api_key=1MSO15EGIFQRo44F&field1=94.40&field2=230.00
***
***
-----|-----
*** Heart-Beat Happened *** BPM: 78
Temperature:82.98
"AT+CIPSTART=4,""TCP""184.106.153.149""",80"
"AT+CIPSEND=4,67"
GET /update?api_key=1MSO15EGIFQRo44F&field1=82.98&field2=226.00
***
***
-----
*** Heart-Beat Happened *** BPM: 83
Temperature:97.04
"AT+CIPSTART=4,""TCP""184.106.153.149""",80"
"AT+CIPSEND=4,67"
GET /update?api_key=1MSO15EGIFQRo44F&field1=97.04&field2=105.00
***
***
-----
*** Heart-Beat Happened *** BPM: 102
Temperature:90.01
"AT+CIPSTART=4,""TCP""184.106.153.149""",80"
"AT+CIPSEND=4,67"
GET /update?api_key=1MSO15EGIFQRo44F&field1=90.01&field2=180.00
***
***
-----|-----
*** Heart-Beat Happened *** BPM: 87
Temperature:144.50
"AT+CIPSTART=4,""TCP""184.106.153.149""",80"
"AT+CIPSEND=4,68"
GET /update?api_key=1MSO15EGIFQRo44F&field1=144.50&field2=201.00
***
***
-----|-----
*** Heart-Beat Happened *** BPM: 79
Temperature:141.86
"AT+CIPSTART=4,""TCP""184.106.153.149""",80"
"AT+CIPSEND=4,68"
GET /update?api_key=1MSO15EGIFQRo44F&field1=141.86&field2=223.00
***
***

```

Figure 6.2: project readings 2


```

*PROJECT READINGS 1 - Notepad
File Edit Format View Help

-----|---
*** Heart-Beat Happened *** BPM: 72
Temperature:90.01
"AT+CIPSTART=4,""TCP""184.106.153.149""",80"
"AT+CIPSEND=4,67"
GET /update?api_key=1MSO15EGIFQR044F&field1=90.01&field2=136.00
""
""

-----|---
*** Heart-Beat Happened *** BPM: 76
Temperature:97.92
"AT+CIPSTART=4,""TCP""184.106.153.149""",80"
"AT+CIPSEND=4,67"
GET /update?api_key=1MSO15EGIFQR044F&field1=97.92&field2=229.00
""
""

-----|---
*** Heart-Beat Happened *** BPM: 81
Temperature:87.37
"AT+CIPSTART=4,""TCP""184.106.153.149""",80"
"AT+CIPSEND=4,67"
GET /update?api_key=1MSO15EGIFQR044F&field1=87.37&field2=223.00
""
""

-----|---
*** Heart-Beat Happened *** BPM: 180
Temperature:93.52
"AT+CIPSTART=4,""TCP""184.106.153.149""",80"
"AT+CIPSEND=4,67"
GET /update?api_key=1MSO15EGIFQR044F&field1=93.52&field2=229.00
""
""

-----|---
*** Heart-Beat Happened *** BPM: 118
Temperature:77.7
"AT+CIPSTART=4,""TCP""184.106.153.149""",80"
"AT+CIPSEND=4,67"
GET /update?api_key=1MSO15EGIFQR044F&field1=77.70&field2=215.00
""
""

-----|---
*** Heart-Beat Happened *** BPM: 90
Temperature:89.13
"AT+CIPSTART=4,""TCP""184.106.153.149""",80"
"AT+CIPSEND=4,67"
GET /update?api_key=1MSO15EGIFQR044F&field1=89.13&field2=228.00
""
""

```

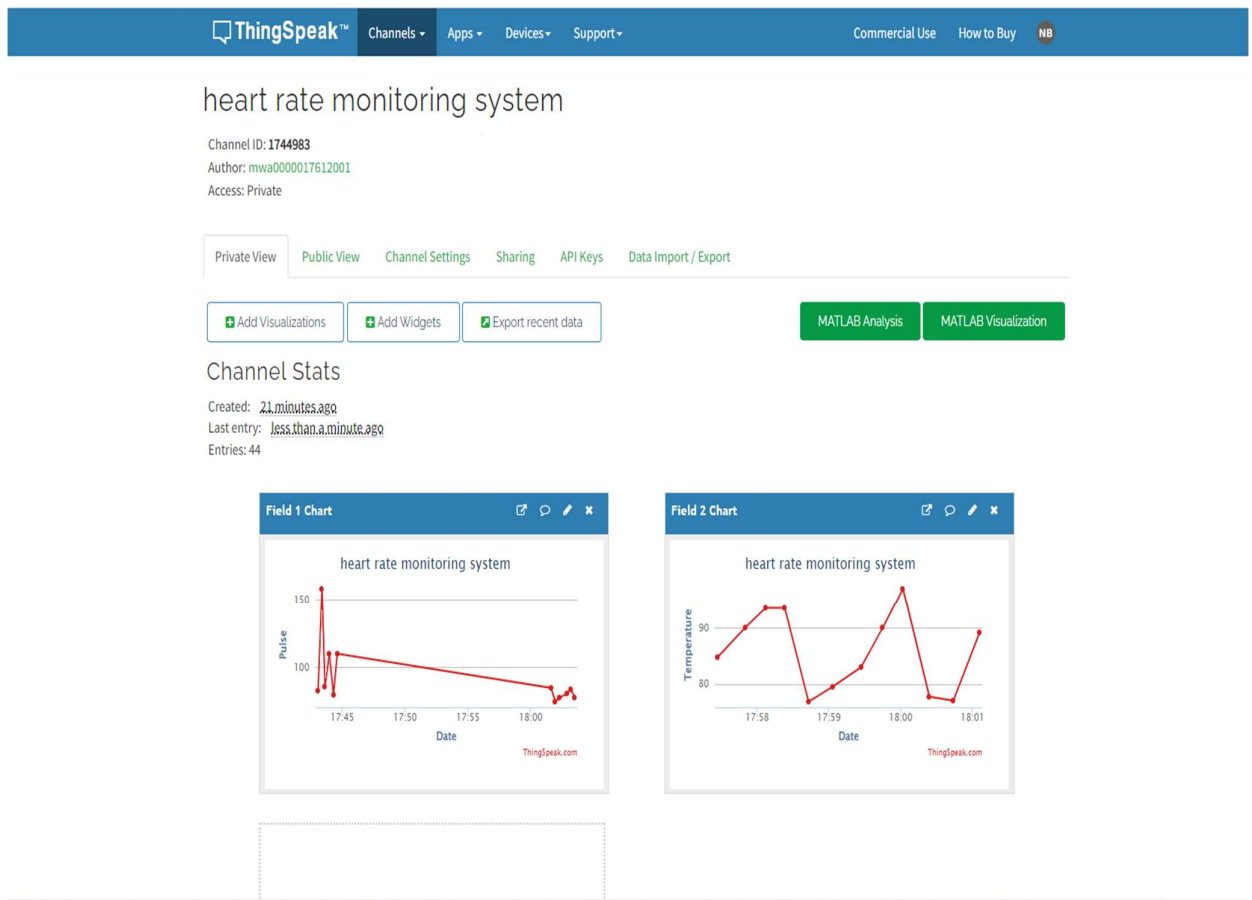
Figure 6.2: project readings 3

```

*PROJECT READINGS 1 - Notepad
File Edit Format View Help
*** Heart-Beat Happened *** BPM: 75
Temperature:95.28
"AT+CIPSTART=4,""TCP""184.106.153.149""",80"
"AT+CIPSEND=4,67"
GET /update?api_key=1MSO15EGIFQR044F&field1=95.28&field2=202.00
""
""
-----|
*** Heart-Beat Happened *** BPM: 74
Temperature:123.41
"AT+CIPSTART=4,""TCP""184.106.153.149""",80"
"AT+CIPSEND=4,68"
GET /update?api_key=1MSO15EGIFQR044F&field1=123.41&field2=229.00
""
""
-----
*** Heart-Beat Happened *** BPM: 62
Temperature:91.77
"AT+CIPSTART=4,""TCP""184.106.153.149""",80"
"AT+CIPSEND=4,67"
GET /update?api_key=1MSO15EGIFQR044F&field1=91.77&field2=198.00
""
""
-----|---
*** Heart-Beat Happened *** BPM: 72
Temperature:98.80
"AT+CIPSTART=4,""TCP""184.106.153.149""",80"
"AT+CIPSEND=4,67"
GET /update?api_key=1MSO15EGIFQR044F&field1=98.80&field2=233.00
""
""
-----|-----
*** Heart-Beat Happened *** BPM: 85
Temperature:77.70
"AT+CIPSTART=4,""TCP""184.106.153.149""",80"
"AT+CIPSEND=4,67"
GET /update?api_key=1MSO15EGIFQR044F&field1=77.70&field2=238.00
""
""
-----|---
*** Heart-Beat Happened *** BPM: 74|
Temperature:90.01
"AT+CIPSTART=4,""TCP""184.106.153.149""",80"
"AT+CIPSEND=4,67"
GET /update?api_key=1MSO15EGIFQR044F&field1=90.01&field2=217.00
""
""

```

Figure 6.2: project readings



The above chart represent the values obtain form the serial monitor on the thingspeak.

From results we can found that:

- The temperature sensor (LM35) has good accuracy sensor. We took readings from the serial monitor, compared them with a device that containing temperature sensor and the results were identical.
- We found that the output of the pulse sensor that displayed on the plotter in the shape of ECG waves includes some noise signals, this noise signals is due to the non-ideal nature of the components. Some filtrations are needed to obtain the pure signal from the plotter to calculate its accuracy.
- All activities and interfaces run in a correct way, and you can move between activities in a smooth way without facing any error.

- When the results obtained from sensors, it means that all the Arduino codes are correct and does not contain any kinds of error.

Accuracy can be calculated using the formula-

*Accuracy = 1- error ratio

*Error ratio = $(| \text{actual value} - \text{measured value} |) / (\text{actual value})$

TEMPERATURE CALCULATIONS =

- Accuracy = 1- error ratio
- Here in case error ratio = $| 82.4 - 94.91 | / 82.4$
= 0.1518
- So, accuracy = $1 - 0.1518$
= **84.82**
- The above readings were taken in Fahrenheit, and for conversion purposes we use the formula $\{f-32\} * 5/9 = c$.
- Measured value is calculated as an average of 25 values obtain form serial monitor.

PULSE SENSOR CALCULATIONS =

- Accuracy = 1- error ratio
- Here in case error ratio = $| 75 - 96 | / 75$
= 0.28
- So, accuracy = $1 - 0.28$
= **72**
- In the above reading the actual value is taken as 75 BPM {beats per minutes}.
- The measured readings were taken as 96, which is average of 25 reading taken In serial monitor.

CHAPTER 7

CONCLUSION

- In this project, the design and development of an Arduino based heart rate monitoring system has been presented. The device is portable, durable, flexible, reliable and cost effective. Also, it is efficient, easily estimable data and easy-to-use for the end user.
- Experimental results have shown acceptable range with actual heartbeat rates. Finally, this handheld system has proven to be a useful heart rate counting system for the end user. However, further improvement is expected with the upgraded module to improve and simplify the system for the users.

- **Limitation –**

The main limitation of this study is that: • The hardware implementation does not give the accurate results of the monitoring system due to the non-ideal nature of the components.

- **Future work -**

Furthermore, some enhancements and improvements could be done:

- Enhance the performance of the heart rate algorithm to provide more accuracy results.
- Make and send the data from the patient account to the doctor account for continuous monitoring.
- Send a message to the doctor and ambulance in case of abnormal rise or fall in the readings obtained from the sensors, this message includes the patient location and abnormal readings of the patient.

Bibliography

- [1] Braunwald e. (editor), "heart disease: a textbook of cardiovascular medicine, fifth edition", Philadelphia, w.b. saunders co., 1997, p. 108
- [2] Muhammad Ali Mazidi, Rolin McKinlay, Janice Gillespie Mazidi, "The 8051 Microcontroller and Embedded Systems Using Assembly and C", Second Edition(2007).
- [3] Arduino UNO Datasheet(<http://avrchip.com/arduino-uno-datasheet-and-tutorial/>)
- [4] Edwards S., "Heart rate Monitor Book", Leisure systems international, 1(3), 122-134 (1993) 4
- [5] LM35 Data sheet(<https://www.engineersgarage.com/electronic-components/lm35-sensordatasheet>).

