“HEALTHCARE MONITORING SYSTEM USING INTERNET OF THINGS”

TABLE OF CONTENTS

ABSTRACT

The recent development of, the Internet of Things (IoT) makes all objects interconnected and it has been recognized as the next technical revolution. Some of the applications of the Internet of Things are smart parking, smart homes, smart cities, smart environments, industrial places, agriculture fields, and health monitoring processes. One such application is in healthcare to monitor the patient health status Internet of Things makes medical equipment more efficient by allowing real time monitoring of patient health, in which sensor acquire data of patient and reduces human error. In the Internet of Things, patient's parameters get transmitted through medical devices via a gateway, where it is stored and analyzed. The significant challenge in the implementation of the Internet of Things for healthcare applications is monitoring all patients from various places.

Thus Internet of Things in the medical field brings out the solution for effective patient monitoring at reduced cost and also reduces the trade-off between patient outcome and disease management. In this project I discuss, monitoring a patient's body temperature and heartbeat using different development boards like Raspberry Pi and Arduino Uno.

CHAPTER 1: INTERNET OF THINGS (IOT)

1.1 INTRODUCTION

The **Internet of Things** (**IoT**) is the network of physical devices, vehicles, home appliances, and other items embedded with electronics, software, sensors, actuators, and network connectivity which enable these objects to connect and exchange data. Each thing is uniquely identifiable through its embedded computing system but is able to inter-operate within the existing Internet infrastructure. Experts estimate that the IoT will consist of about 30 billion objects by 2020. It is also estimated that the global market value of IoT will reach $7.1 trillion by 2020.

The IoT allows objects to be sensed or controlled remotely across existing network infrastructure, creating opportunities for more direct integration of the physical world into computer-based systems, and resulting in improved efficiency, accuracy, and economic benefit in addition to reduced human intervention. When IoT is augmented with sensors and actuators, the technology becomes an instance of the more general class of cyber-physical systems, which also encompasses technologies such as smart grids, virtual power plants, smart homes, intelligent transportation, and smart cities.

"Things", in the IoT sense, can refer to a wide variety of devices such as heart monitoring implants, biochip transponders on farm animals, cameras streaming live feeds of wild animals in coastal waters, automobiles with built-in sensors, DNA analysis devices for environmental/food/pathogen monitoring, or field operation devices that assist firefighters in search and rescue operations. Legal scholars suggest regarding "things" as an "inextricable mixture of hardware, software, data, and service".

These devices collect useful data with the help of various existing technologies and then autonomously flow the data between other devices.

The term "the Internet of Things" was coined by Kevin Ashton of Procter & Gamble, later MIT's Auto-ID Center, in 1999.

1.2: Components of healthcare monitoring system

Various components were used in this project. These are the essential parts, with the help of which various parameters of the human body can be measured. Basic parameters play a vital role in our healthcare system and irregularity in their normal behavior indicate some health issues.

ARDUINO BOARD:

.

PULSE SENSOR:

TEMPERATURE AND HUMIDITY SENSOR:

ESP8266 Wi-Fi module

RASPBERRY PI 3

ADC (MCP3008)

CHAPTER: 2 LITERATURE SURVEY

2.1: INTRODUCTION

The Internet of Things has numerous applications in healthcare, from remote monitoring to smart sensors and medical device integration. It has the potential to not only keep patients safe and healthy but to improve how physicians deliver care as well. Healthcare IoT can also boost patient engagement and satisfaction by allowing patients to spend more time interacting with their doctors.

But healthcare IoT isn't without its obstacles. The number of connected devices and the tremendous amount of data they collect can be a challenge for hospital IT to manage. There is also the question of how to keep all of that data secure, especially if it is being exchanged with other devices.

This essential guide will look at some of the current applications of healthcare IoT, including how it's being used in one Boston hospital to keep track of newborns in the NICU. Next, the guide explores some of the challenges of IoT in healthcare, such as the need to manage multiple connected devices and a lack of interoperability with EHR systems. Finally, this guide will posit the future of healthcare IoT, including how physicians can turn IoT data into actions.

2.2: SYSTEM OVERVIEW

In project, a Pulse sensor, temperature and humidity sensor, wi-fi module, and different development boards like Arduino Uno or Raspberry Pi 3 are connected to the circuitry. Programs or algorithms are stored in the microcontroller of the board. The sensed data is sent through an internet connection to the cloud which is then accessed using a Thingspeak account. Graphical representation of pulse graphs and numerical counts of temperature and humidity are analyzed by the health expert or the doctor. Suggestions and emergency services are provided accordingly. Messages can be sent in case of emergency to the family of the person using the device and also to the healthcare center nearby so that ambulance and emergency service can be provided at the right time.

CHAPTER 3:- SENSORS

3.1 : INTRODUCTION

In the broadest definition, a **sensor** is a device, module, or subsystem whose purpose is to detect events or changes in its environment and send the information to other electronics, frequently a computer processor. A sensor is always used with other electronics, whether as simple as a light or as complex as a computer.

Sensors are used in everyday objects such as touch-sensitive elevator buttons (tactile sensors) and lamps that dim or brighten by touching the base, besides innumerable applications of which most people are never aware. With advances in micromachinery and easy-to-use microcontroller platforms, the uses of sensors have expanded beyond the traditional fields of temperature, pressure, or flow measurement, for example into MARG sensors. Moreover, analog sensors such as potentiometers and force-sensing resistors are still widely used. Applications include manufacturing and machinery, airplanes and aerospace, cars, medicine, robotics, and many other aspects of our day-to-day lives.

A sensor's sensitivity indicates how much the sensor's output changes when the input quantity being measured changes. For instance, if the mercury in a thermometer moves 1 cm when the temperature changes by 1 °C, the sensitivity is 1 cm/°C. Sensors are usually designed to have a small effect on what is measured; making the sensor smaller often improves this and may introduce other advantages. Technological progress allows more and more sensors to be manufactured on a microscopic scale.

3.2 :PULSE SENSOR

INTRODUCTION

Do not connect the Pulse Sensor to your body while your computer or Arduino is being powered from the main AC line. That goes for charging laptops and DC power supplies. Please be safe and isolate yourself from the power grid, or work under battery power.

Connect the Pulse Sensor to: +V (red), Ground (black), and Analog Pin 0 (purple) on your favorite Arduino, or

Arduino-compatible device, and upload the ‘PulseSensoAmped_Arduino-xx’ sketch.

note: If you want to power the Pulse Sensor Amped

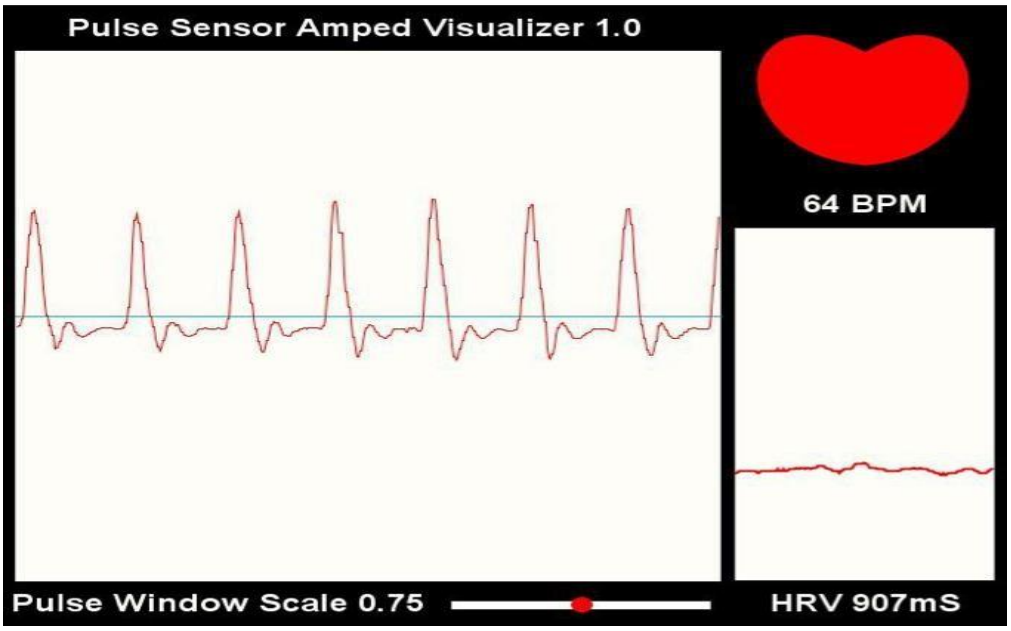
with low voltage (3.3V for example), make sure

you have this line of code in the setup()

analogReference(EXTERNAL);

Also, make sure that you apply the lower voltage to the Arduino Aref pin (next to pin 13). After it’s done uploading, you should see Arduino pin 13 blink in time with your heartbeat when you hold the sensor on your fingertip. If you grip the sensor too hard, you will squeeze all the blood out of your fingertip and there will be no signal! If you hold it too lightly, you will invite noise from movement and ambient light. Sweet Spot pressure on the Pulse Sensor will give a nice clean signal. You may need to play around and try different parts of your body and pressures. To view the heartbeat waveform

and check your heart rate, you can use the Processing sketch that we made. Start up Processing on your computer and run the Pulse Sensor Processing sketch. The large main window shows a graph of raw sensor data over time. The Pulse Sensor Data Window can be scaled using the scrollbar at the bottom if you have a very large or very small signal. At the right of the screen, a smaller data window graphs heart rate over time. This graph advances every pulse, and the Beats Per Minute is updated every pulse as a running average of the last ten pulses. The big red heart in the upper right also pulses to the time of your heartbeat. When you hold the Pulse Sensor to your fingertip or earlobe or (fill in body part here) you should see a nice heartbeat.

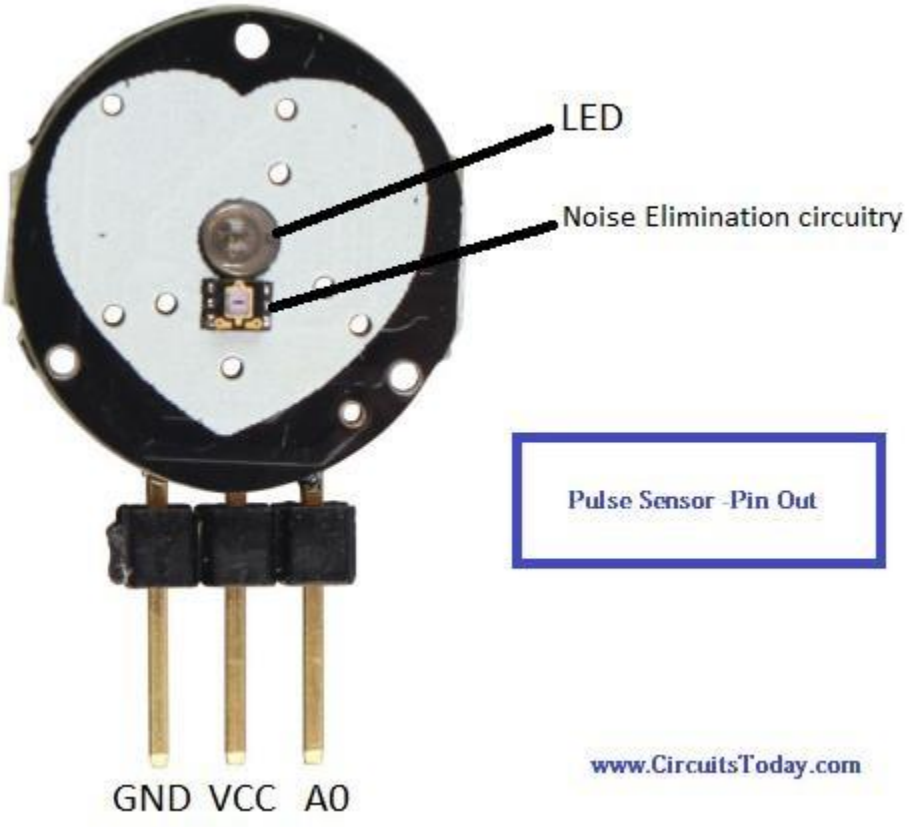


PIN DIAGRAM OF PULSE SENSOR

The pulse sensor has three pins which are described below:

- **GND:** Ground Pin
- **VCC:** 5V or 3V Pin
- **A0:** Analog Pin

There is also an LED in the center of this sensor module which helps in detecting the heartbeat. Below the LED, there is a noise elimination circuit which is supposed to keep away the noise from affecting the readings.



A.3 Working – Pulse Sensor

Principle of Heartbeat Sensor

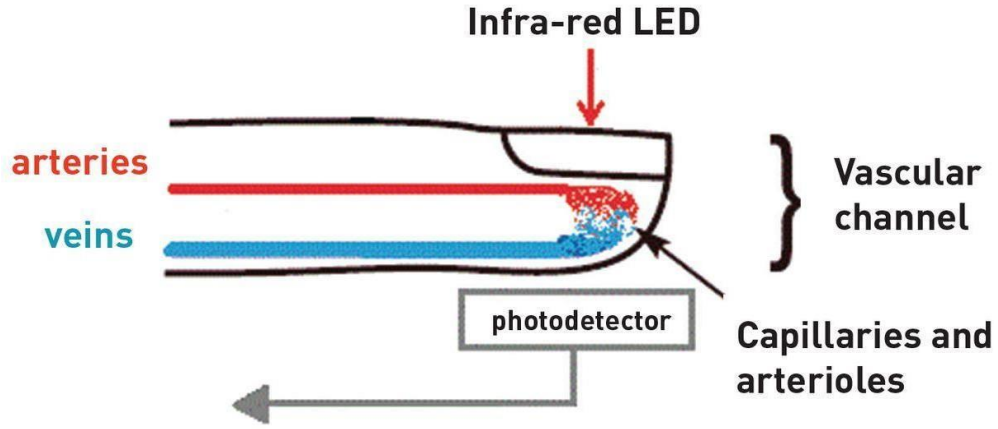
The heartbeat sensor is based on the principle of photoplethysmography. It measures the change in volume of blood through any organ of the body which causes a change in the light intensity through that organ (a vascular region). In the case of applications where heart pulse rate is to be monitored, the 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 heartbeat pulses.
There are two types of photoplethysmography.

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

When a heartbeat occurs blood is pumped through the human body and gets squeezed into the capillary tissues. The volume of these capillary tissues increases as a result of the heartbeat. But in between the heartbeats (the time between two consecutive heartbeats,), this volume inside capillary tissues decreases. This change in volume between the heartbeats affects the amount of light that will transmit through these tissues. This change is very small but we can measure it with the help of Arduino.

The pulse sensor module has a light that helps in measuring the pulse rate. When we place the finger on the pulse sensor, the light reflected will change based on the volume of blood inside the capillary blood vessels. During a heartbeat, the volume inside the capillary blood vessels will be high. This affects the reflection of light and the light reflected at the time of a heartbeat will be less compared to

that of the time during which there is no heartbeat (during the period of time when there is no heartbeat or the time period in between heartbeats, the volume inside the capillary vessels will be lesser. This will lead to higher reflection of light). This variation in light transmission and reflection can be obtained as a pulse from the output of a pulse sensor. This pulse can be then conditioned to measure heartbeat and then programmed accordingly to read as heartbeat count.



3.3: DHT11(TEMPERATURE AND HUMIDITY SENSOR)

INTRODUCTION-

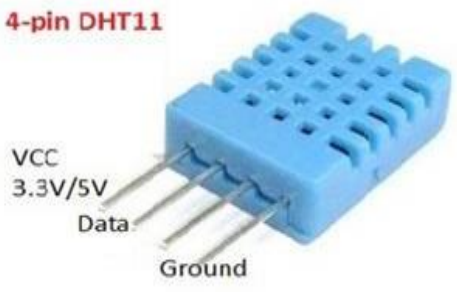
PRINCIPLE OF OPERATION-The DHT11 detects water vapor by measuring the electrical resistance between two electrodes. The humidity-sensing component is a moisture-holding substrate with electrodes applied to the surface. When water vapor is absorbed by the substrate, ions are released by the substrate which increases the conductivity between the electrodes. The change in resistance between the two electrodes is proportional to the relative humidity. Higher relative humidity decreases the resistance between the electrodes, while lower relative humidity increases the resistance between the electrodes.

The DHT11 measures temperature with a surface-mounted NTC temperature sensor (thermistor) built into the unit. An IC mounted on the back of the unit converts the resistance measurement to relative humidity. It also stores the calibration coefficients, and controls the data signal transmission between the DHT11 and the Arduino: The DHT11 uses just one signal wire to transmit data to the Arduino. Power comes from separate 5V and ground wires. A 10K Ohm pull-up resistor is needed between the signal line and 5V line to make sure the signal level stays high by default (see the datasheet for more info).

There are two different versions of the DHT11 you might come across. One type has four pins, and the other type has three pins and is mounted to a small PCB. The PCB-mounted version is nice because it includes a surface-mounted 10KOhm pull-up resistor for the signal line.

PIN DESCRIPTION OF DHT11

1. VCC pin is connected to 3- 5.5 volt DC.
2. Data pin is connected to serial data output.
3. NC is not connected.



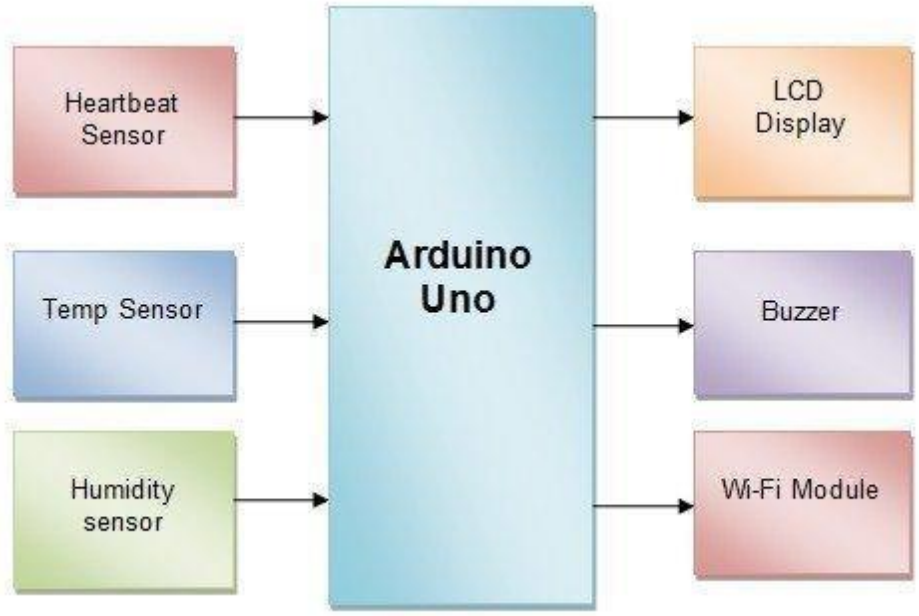
4. Ground pin is connected to the ground.

Detailed Specifications:

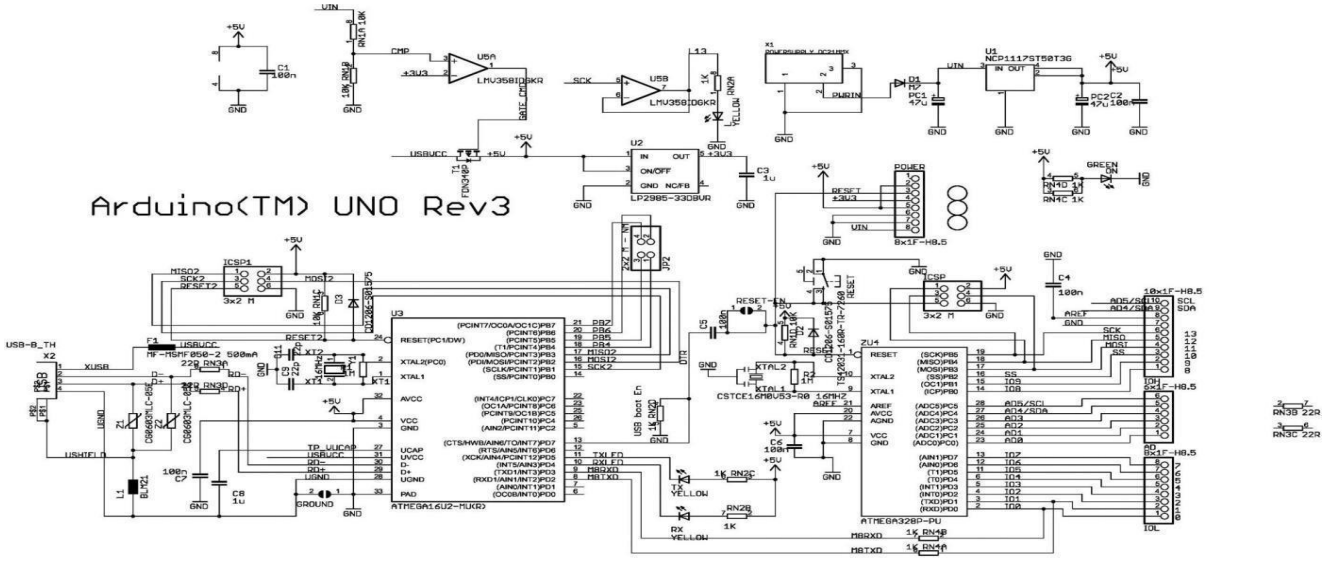
Parameters	Conditions	Minimum	Typical	Maximum
Humidity				
Resolution		1%RH	1%RH	1%RH
			8 Bit	
Repeatability			±1%RH	
Accuracy	25℃		±4%RH	
	0-50℃			±5%RH
Interchangeability	Fully Interchangeable			
Measurement Range	0℃	30%RH		90%RH
	25℃	20%RH		90%RH
	50℃	20%RH		80%RH
Response Time (Seconds)	1/e(63%)25℃, 1m/s Air	6 S	10 S	15 S
Hysteresis			±1%RH	
Long-Term Stability	Typical		±1%RH/year	
Temperature				
Resolution		1℃	1℃	1℃
		8 Bit	8 Bit	8 Bit
Repeatability			±1℃	
Accuracy		±1℃		±2℃
Measurement Range		0℃		50℃
Response Time (Seconds)	1/e(63%)	6 S		30 S

CHAPTER 4: HEALTHCARE MONITORING SYSTEM USING ARDUINO

BLOCK DIAGRAM OF THE PROJECT:



SCHEMATIC DIAGRAM OF ARDUINO

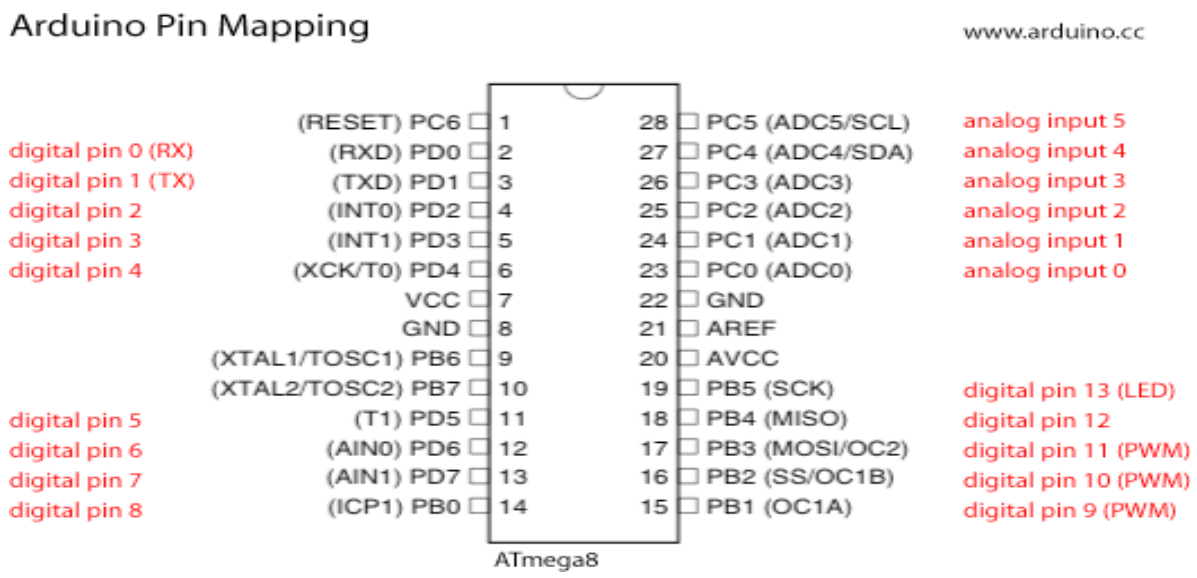


4.2: INTRODUCTION(ARDUINO UNO R3)

Arduino is an open-source computer hardware and software company, project, and user community that designs and manufactures single-board microcontrollers and microcontroller kits for building digital devices and interactive objects that can sense and control objects in the physical world. The project's products are distributed as open-source hardware and software, which are licensed under the GNU Lesser General Public License (LGPL) or the GNU General Public License (GPL), permitting the manufacture of Arduino boards and software distribution by anyone. Arduino boards are available commercially in preassembled form or as do-it-yourself(DIY) kits.

Arduino board designs use a variety of microprocessors and controllers. The boards are equipped with sets of digital and analog input/output (I/O) pins that may be interfaced with various expansion boards (*shields*) and other circuits. The boards feature serial communications interfaces, including Universal Serial Bus (USB) on some models, which are also used for loading programs from personal computers. The microcontrollers are typically programmed using a dialect of features from the programming languages C and C++. In addition to using traditional compiler toolchains, the Arduino project provides an integrated development environment(IDE) based on the Processing language project.

PIN DIAGRAM OF ARDUINO



Input and Output

Each of the 14 digital pins on the Arduino Uno can be used as an input or output, using pinMode(), digitalWrite(), and digitalRead() functions. They operate at 5 volts. Each pin can provide or receive a maximum of 40 mA and has an internal pull-up resistor (disconnected by default) of 20-50 KOhms. In addition, some pins have specialized functions:

Serial: pins 0 (RX) and 1 (TX). Used to receive (RX) and transmit (TX) TTL serial data. These pins are connected to the corresponding pins of the ATmega8U2 USB-to-TTL Serial chip.

External Interrupts: pins 2 and 3. These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value. See the attachInterrupt() function for details.

PWM: 3, 5, 6, 9, 10, and 11. Provide 8-bit PWM output with the analogWrite() function.

SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). These pins support SPI communication using the SPI library.

LED: 13. There is a built-in LED connected to digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.

The Uno has 6 analog inputs, labeled A0 through A5, each of which provide 10 bits of resolution (i.e. 1024 different values). By default they measure from ground to 5 volts, though is it possible to change the upper end of their range using the AREF pin and the analogReference() function. Additionally, some pins have specialized functionality:

TWI: A4 or SDA pin and A5 or SCL pin. Support TWI communication using the Wire library.

There are a couple of other pins on the board:

AREF. Reference voltage for the analog inputs. Used with analog Reference().

Reset. Bring this line LOW to reset the microcontroller. Typically used to add a reset button to shields which block the one on the board

When Arduino uno was used in the initial phase of the project Wi-Fi module ESP8266 was used along with it to send data from sensors to Thingspeak account.

4.3 :SOFTWARE DESIGN OF THE CIRCUIT:

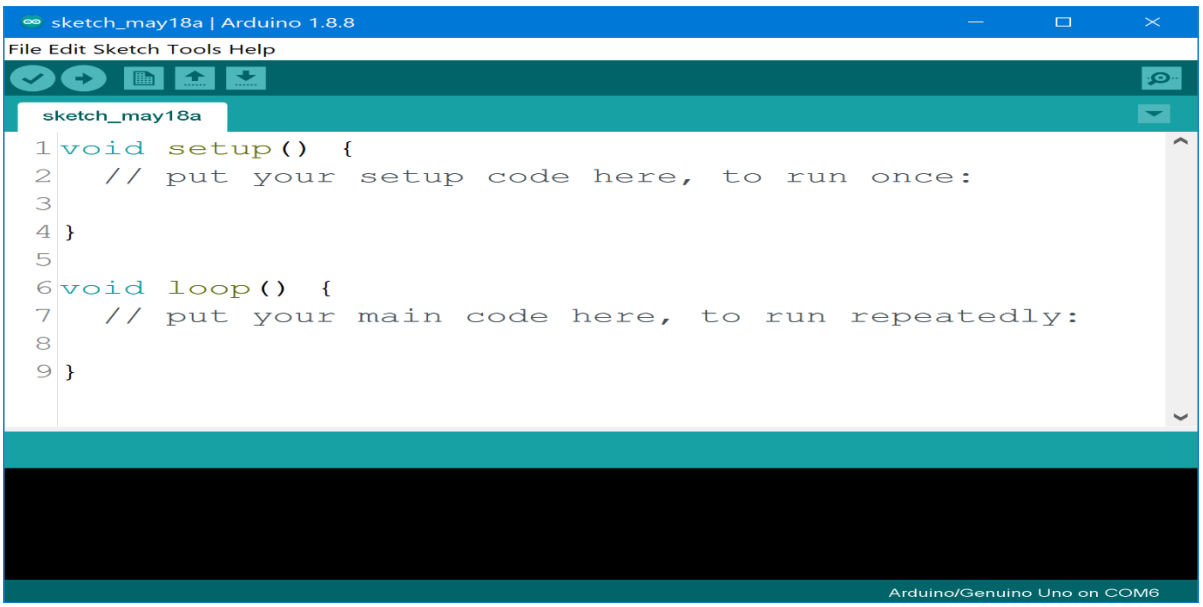
An **Integrated Development Environment** (**IDE**) is a software application that provides comprehensive facilities to computer programmers for software development. An IDE normally consists of a source code editor, build automation tools, and a debugger. Most modern IDEs have intelligent code completion. Some IDEs, such as NetBeans and Eclipse, contain a compiler, interpreter, or both; others, such as SharpDevelop and Lazarus, do not. The boundary between an integrated development environment and other parts of the broader *software development environment* is not well-defined. Sometimes an aversion control system, or various tools to simplify the construction of a Graphical User Interface (GUI), are integrated. Many modern IDEs also have a class browser, an object browser, and a class hierarchy diagram, for use in object-oriented software development.

. Integrated development environments are designed to maximize programmer productivity by providing tight-knit components with similar user interfaces. IDEs present a single program in which all development is done. This program typically provides many features for authoring, modifying, compiling, deploying, and debugging software. This contrasts with software development using unrelated tools, such as vi, GCC , or make.

One aim of the IDE is to reduce the configuration necessary to piece together multiple development utilities, instead providing the same set of capabilities as a cohesive unit. Reducing that setup time can increase developer productivity, in cases where learning to use the IDE is faster than manually integrating all of the individual tools. Tighter integration of all development tasks has the potential to improve overall productivity beyond just helping with setup tasks. For example, code can be continuously parsed while it is being edited, providing instant feedback when syntax errors are introduced. That can speed learning a new programming language and its associated libraries.

Some IDEs are dedicated to a specific programming language, allowing a feature set that most closely matches the programming paradigms of the language. However, there are many multiple-language IDEs.

While most modern IDEs are graphical, text-based IDEs such as Turbo Pascal were in popular use before the widespread availability of windowing systems like Microsoft Windows and the X Window System (X11). They commonly use function keys or hotkeys to execute frequently used commands or macros.



4.4: HARDWARE DESIGN OF THE CIRCUIT

To design any electrical circuit, either analog or digital, electrical engineers need to be able to predict the voltages and currents at all places within the circuit. Linear circuits, that is, circuits wherein the outputs are linearly dependent on the inputs, can be analyzed by hand using complex analysis. Simple nonlinear circuits can also be analyzed in this way. Specialized software has been created to analyze circuits that are either too complicated or too nonlinear to analyze by hand.

Circuit simulation software allows engineers to design circuits more efficiently, reducing the time cost and risk of error involved in building circuit prototypes. Some of these make use of hardware description languages such as VHDL or Verilog

CHAPTER 5 : HEALTHCARE MONITORING SYSTEM USING RASPBERRY PI 3.

5.1 :INTRODUCTION(RASPBERRY PI 3 MODEL B)

The Raspberry Pi is a credit card-sized single-board computer with an open-source platform that has a thriving community of its own, similar to that of the ARDUINO. It can be used in various types of projects from beginners learning how to code to hobbyists designing home automation systems.

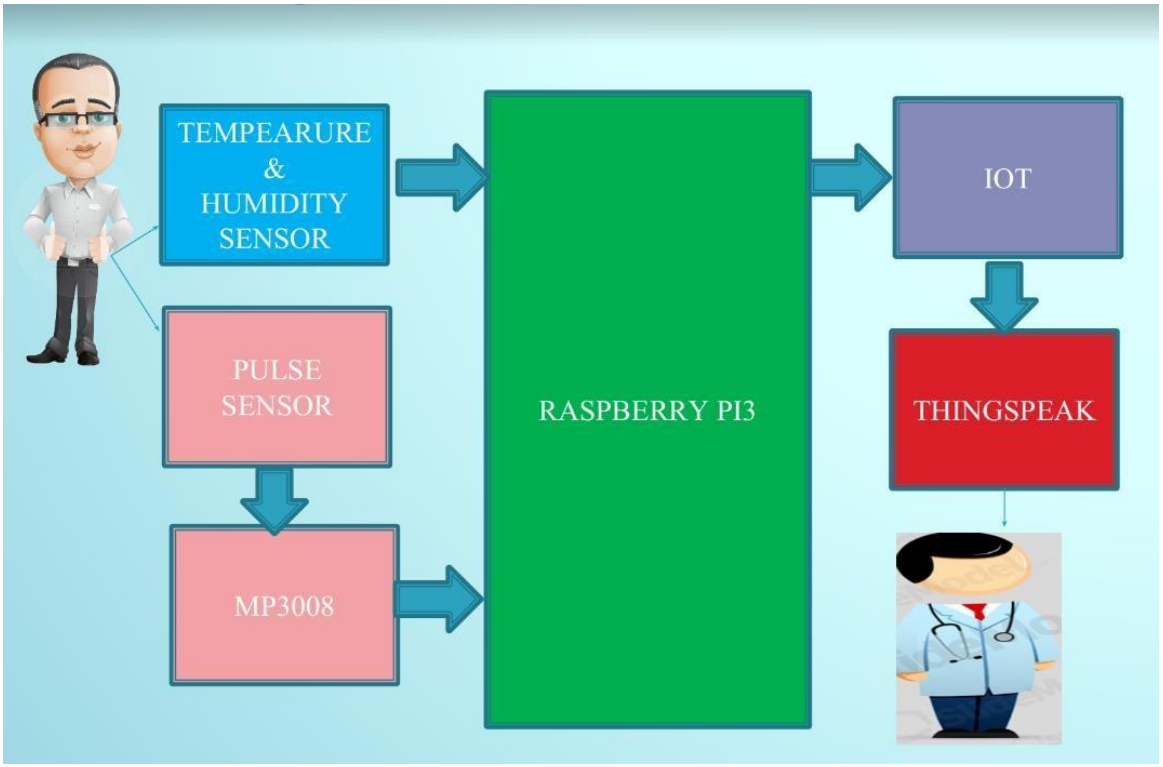
There are a few versions of the Raspberry Pi, but the latest version has improved upon its predecessor in terms of both form and functionality. The Raspberry Pi Model B features:

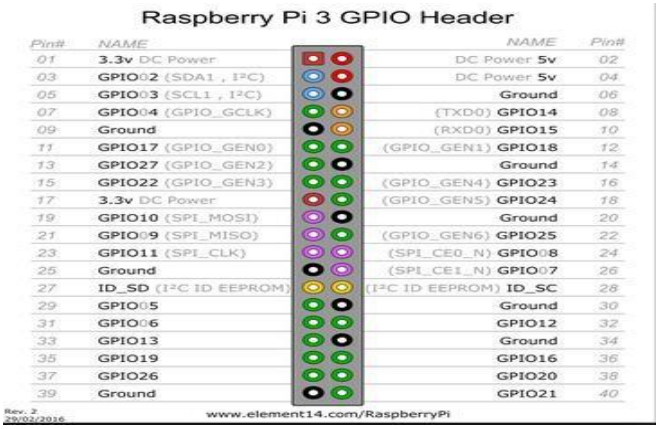
•More GPIO
•More USB
•Micro SD
•Lower power consumption
•Better audio
•Neater form factor

This higher-spec variant increases the Raspberry Pi GPIO pin count from 26 to 40 pins. There are now four USB 2.0 ports compared to two on the Model B. The SD card slot has been replaced with a more modern push-push type micro SD slot. It consumes slightly less power, provides better audio quality, and has a cleaner form factor.

To get started you need a **Raspberry Pi 3 Model B**, a 5 volts power supply of at least 2 amps with a micro USB cable and standard USB keyboard and mouse, a HDMI CABLE and monitor/TV for display, and a micro SD card with the operating system pre-installed. The NOOBS (New Out Of the Box Software) OS is recommended for beginners, and you may choose one of several from the download page.

5.2 :BLOCK DIAGRAM :





5.3:GPIOPIN DIAGRAM OF RASPBERRY PI 3 MODEL

5.3: HARDWARE REQUIREMENTS:

Before we plug into our Raspberry Pi, we require the following equipment.

1- Monitor with HDMI port

2- VGA connector

3- Micro USB power supply

4- A wired keyboard and mouse.

5- A micro SD card(8 GB or above)

6- A raspberry pi 3 model B

To get started we also need an operating system. The one I used in our project is Raspbian.

1. Begin by placing your SD card into the SD card slot on the Raspberry Pi. It will only fit one way.
2. Next, plug your keyboard and mouse into the USB ports on the Raspberry Pi.

3. Make sure that your monitor or TV is turned on, and that you have selected the right input (e.g. HDMI 1, DVI, etc.).

4. Connect your HDMI cable from your Raspberry Pi to your monitor or TV.

5. If you intend to connect your Raspberry Pi to the internet, plug an Ethernet cable into the Ethernet port, or connect a Wi-Fi dongle to one of the USB ports (unless you have a Raspberry Pi 3).

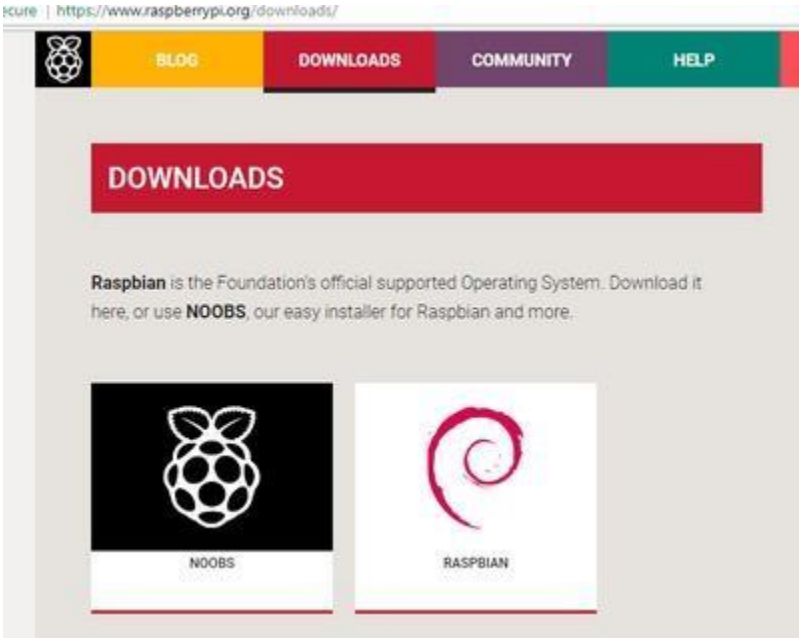
6. When you're happy that you have plugged all the cables and SD card in correctly, connect the micro USB power supply. This action will turn on and boot your Raspberry Pi.

5.4 : SOFTWARE REQUIREMENTS:

The recommended operating system for use with the Raspberry Pi is called Raspbian. Raspbian is a version of GNU/Linux, designed specifically to work well with the Raspberry Pi. You have several options when it comes to getting hold of a copy of Raspbian.

The easiest way to get NOOBS or Raspbian is to buy an SD card with the software already installed. You can get a pre-installed Raspbian card from RS or The Pi Hut.

Visit the official Raspberry Pi downloads page.



1. Click on NOOBS and download it. Extract the files from zip. Once the SD card is formatted, drag all the files in the extracted NOOBS folder and drop them onto the SD card drive.

2. The necessary files will then be transferred to your SD card.

3. When this process has finished, safely remove the SD card and insert it into your Raspberry Pi.

   If you want to connect your Raspberry Pi to the internet or local network then you will need to follow these steps:

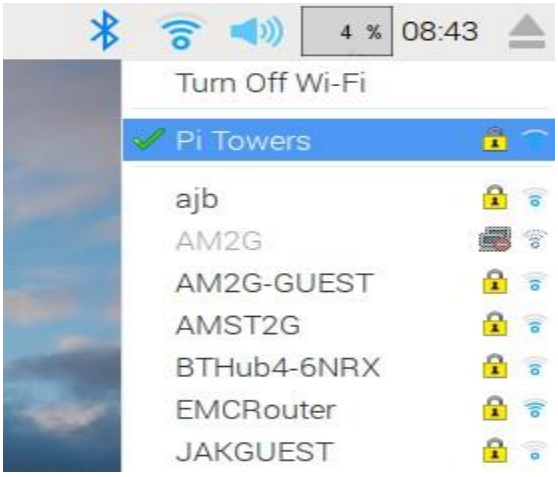
   CONNECTING TO WIRELESS LAN:

1. Wireless LAN connections can be made via the network icon at the right-hand end of the menu bar. If you are using a Raspberry Pi 3, or an earlier model with a Wi-Fi dongle plugged in, left-clicking this icon will bring up a list of available Wi-Fi networks.

2. If no networks are found, it will show the message "No APs found - scanning...": just wait a few seconds without closing the menu, and it should find your network.

3. The icons on the right show whether a network is secured or not, and its signal strength. Click the network that you want to connect to. If it's secured, a dialogue box will appear prompting you to enter the network key.

4. Enter the key and press **OK**, then wait a couple of seconds. The network icon will flash briefly to show that a connection is being made. Once it's ready, the icon stops flashing and shows the signal strength.



5.5: HEART RATE SENSING:

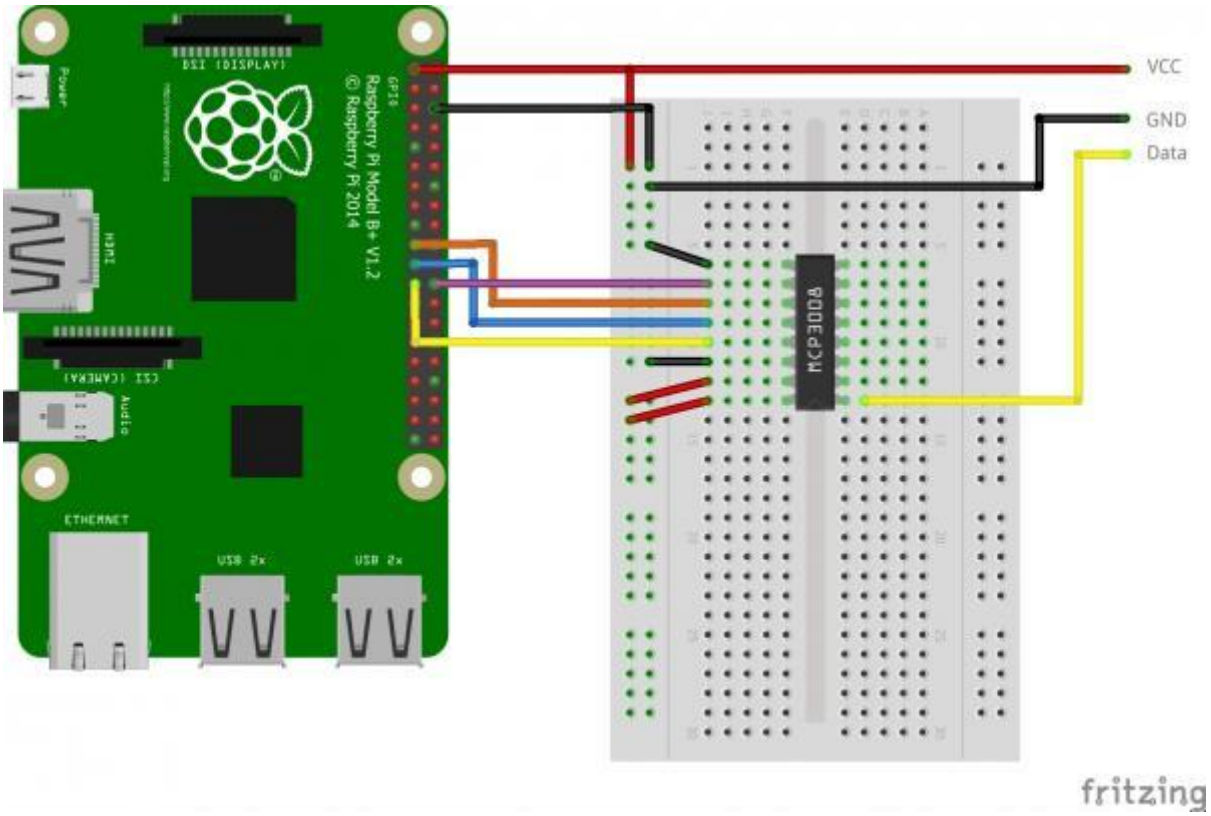
I used the following components :

- Heart rate Pulse sensor
- MCP3008 ADC
- Small Breadboard
- Jumper Wire

Connection

Since this sensor has also been designed for the Arduino, it does not provide a digitally readable signal. In order to be able to read the analog signal, an ADC such as the MCP3008 is required. It doesn't matter which channel of the ADC you are using, as long as you adjust it later in the code.

The positive pole of the pulse sensor is connected to 3.3V from the Raspberry Pi, as well as Minus to Ground. I have connected the signal/data pin (marked with an "S") to channel 0 of the MCP3008.



The ADC uses the SpiDev library, which must be installed.

Code for reading the Heart / Pulse Rate

If you don't use a current Raspbian / NOOBS operating system, you may need to update and install git.

```
sudo apt-get update

sudo apt-get upgrade sudo apt-get install git
```

On GitHub I have provided a library for recording the heart rate. You can download it as follows:

```
git clone https://github.com/tutRPi/Raspberry-Pi-Heartbeat-Pulse-Sensor
```

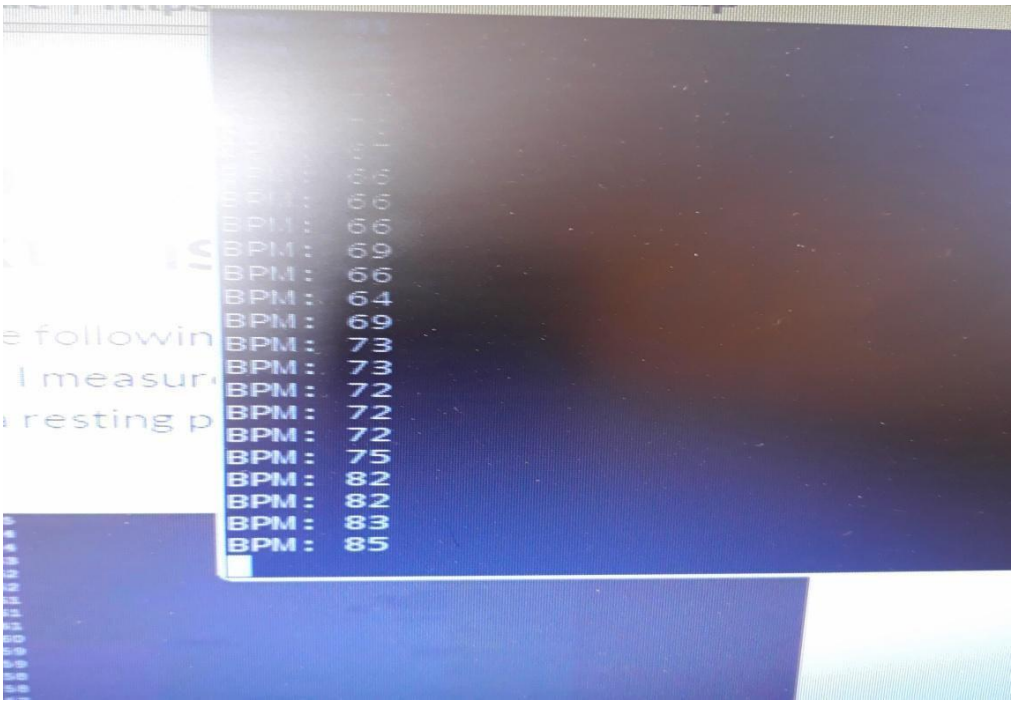
Also attached is a sample file, which reads the heart rate every second. In the video below you can see this visualized.

To start the example we switch to the folder and call the example.py.

```
cd Raspberry-Pi-Heartbeat-Pulse-Sensor/

sudo python example.py
```

If you want to stop, press CTRL+C.



5.6 :TEMPERATURE AND HUMIDITY SENSING:

We used the following components:

1- Raspberry pi 3 model B

2- DHT11 (temperature and humidity sensor)

3- 4.7 K resistance.

CONNECTIONS:

Connect ground of the dht11 to ground of raspberry pi 3 pin number 39(ground).

Connect data pin to pin number 16 of raspberry pi 3.

Connect Vcc of dht11 with pin 2(5 volts) of raspberry pi 3.

CODE :

```
import sys
import RPi.GPIO as GPIO
from time import sleep
import Adafruit_DHT
import urllib2
from pulsesensor import Pulsesensor
import time

p = Pulsesensor()

def getSensorData():
    RH, T = Adafruit_DHT.read_retry(Adafruit_DHT.DHT11, 23)
    bpm = p.BPM
    if bpm <= 0:
        print('No heartbeat found. Setting heartbeat to zero . . .')
        bpm = 0

    # return dict
    return (str(RH), str(T), str(bpm))

# main() function
def main():
    # use sys.argv if needed
    #if len(sys.argv) < 2:
       # print('Usage: python tstest.py PRIVATE_KEY')
       # exit(0)
    p.startAsyncBPM()
    print ('starting...')

    baseURL = 'https://api.thingspeak.com/update?api_key=%s' % 'J7EE2E1S6Z92FRYM'

    while True:
        try:
            RH, T, bpm = getSensorData()
            f = urllib2.urlopen(baseURL + "&field1=%s&field2=%s&field3=%s" % (RH, T, bpm))
            print f.read()
            f.close()
            sleep(1)
        except:
            print 'exiting.'
            p.stopAsyncBPM()
            break

# call main
if __name__ == '__main__':
    main()
NORMAL  combined.py                    python  utf-8[unix]  32/0x20 365/0x16D  39%  18/46 :  2  [3]trailing
[iot] 0:vim* 1:py-                                                    "~/Downloads/attachmen" 13:41 25-May-18
```

```
import sys
import RPi.GPIO as GPIO
from time import sleep
import Adafruit_DHT
import urllib2

def getSensorData():
    RH, T = Adafruit_DHT.read_retry(Adafruit_DHT.DHT11, 23)
    # return dict
    return (str(RH), str(T))

# main() function
def main():
    # use sys.argv if needed
    if len(sys.argv) < 2:
        print('Usage: python tstest.py PRIVATE_KEY')
        exit(0)
    print 'starting...'

    baseURL = 'https://api.thingspeak.com/update?api_key=%s' % sys.argv[1]

    while True:
```

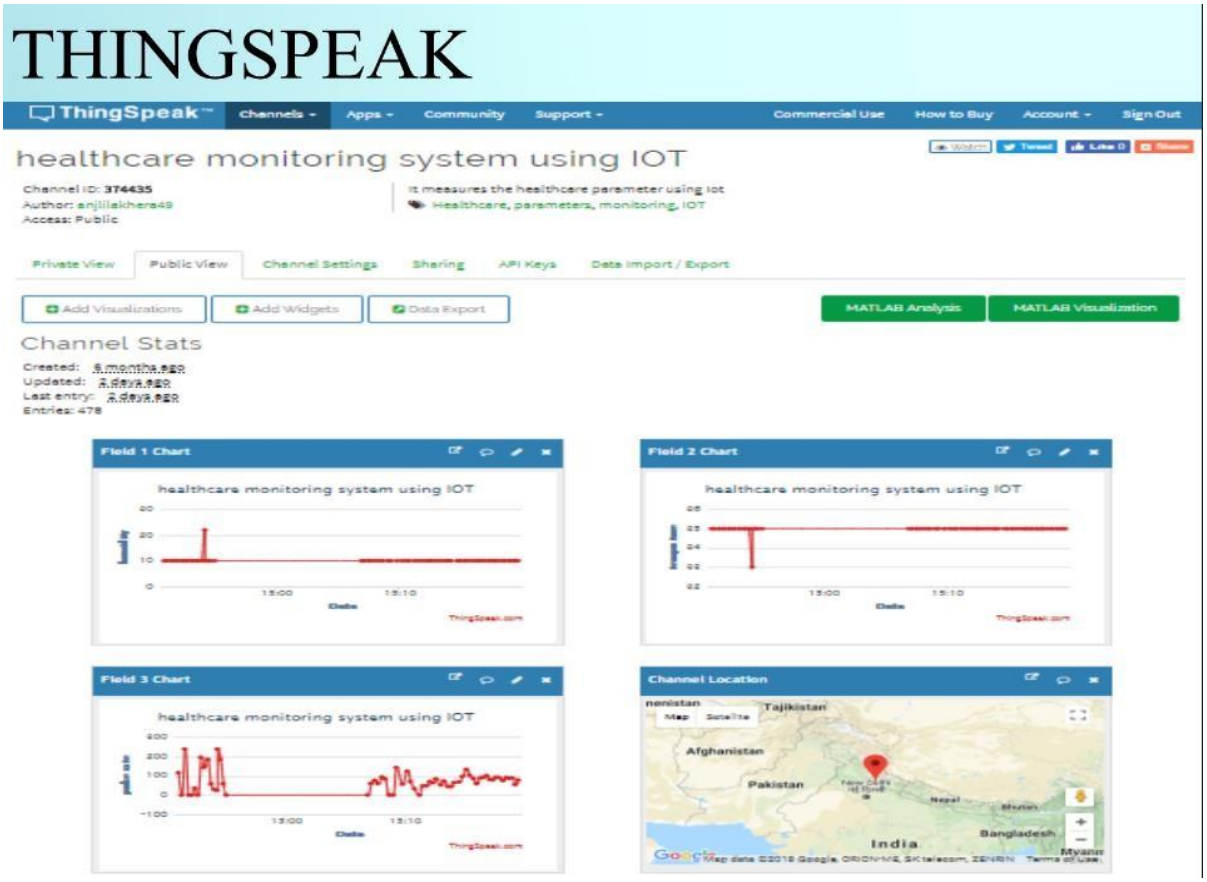
```
        try:
            RH, T = getSensorData()
            f = urllib2.urlopen(baseURL +
                                "&field1=%s&field2=%s" % (RH, T))
            print f.read()
            f.close()
            sleep(15)
        except:
            print 'exiting.'
            break

# call main
if__name__== '_main_':
    main()
```

You run it by uploading it to your Pi and doing:

sudo python dht11_thingspeak.py YOURWRITEAPIKEY

OUTPUT-



CHAPTER 6:

CONCLUSION

All the connections are made per the pin diagram of sensors with the Arduino. Important files are downloaded and then unzipped. Sketches are verified and uploaded to the Arduino board using Arduino IDE software. By this, all the data is sent to Thingspeak account which can be viewed by the browser using another Thingspeak account. By this, a device can be created which can be used in remote places and data can be sent to the doctor and an emergency healthcare facility can be provided in case of severe fluctuation of basic parameters of the body.

CHAPTER 7: BIBLIOGRAPHY

1. https://en.wikipedia.org/wiki/Electronic_circuit_design
2. https://en.wikipedia.org/wiki/Integrated_development_environment
3. https://www.arduino.cc/en/Guide/Environment
4. http://www.hobbyist.co.nz/?q=documentations/wiring-up-dht11-temp-humidity-sensor-to-your-arduino
5. https://steve.zazeski.com/arduino-dht11-temperature-and-humidity-data-logger/
6. http://www.circuitstoday.com/wp-content/uploads/2017/03/Pulse-sensor-Pin-out.png
7. raspberrypi.org
8. http://electronut.in/dht11-rpi-cloud-plot/
9. https://tutorials-raspberrypi.com/raspberry-pi-heartbeat-pulse-measuring/

THANK YOU