# ABSTRACT

Heart rate that is given by the number of times heart beats per minute is a crucial health parameter that indicates the soundness of the human health. Actually, heart patients are required to be monitored with utmost care, but every time it is not possible for a doctor to be present physically near the patient. So, the plan was to make a “Heart Rate Monitoring System” where the doctor and the assisting family members can access patient’s data over internet from any part of the world.

Heart Rate Monitoring System is an Internet of Things (IoT) based tool where the physical instrument will be with the patients and the software that monitors the heart rate can be accessed by the doctor and family members of the patients remotely through internet.

Heart rate is calculated using a Pulse sensor, it is a device which is used to calculate the pulses generated from the heart through any vein present inside the body. Heart rate can be monitored using two ways, the first is by using Liquid Crystal Display (LCD) and the second one is by using ThingSpeak IOT platform. I2C LCD module is used in order to lower the pin requirement for microcontroller. Connection with ThingSpeak IOT platform is provided by NodeMCU which is Wi- Fi module. It includes firmware which runs on ESP8266 Wi-Fi SoC

## INDEX

**TOPIC PAGENO**

**Title i**

[Certificate ii](#_TOC_250025)

[Acknowledgement iii](#_TOC_250024)

[Abstract iv](#_TOC_250023)

[Index v](#_TOC_250022)

[List of Figures vii](#_TOC_250021)

CHAPTER 1: INTRODUCTION TO EMBEDDED SYSTEMS

* 1. Introduction 7
  2. [Applications of Embedded System 10](#_TOC_250020)
  3. [Micro Processor for Embedded System 10](#_TOC_250019)

CHAPTER 2: HEART RATE MONITORING SYSTEM

* 1. [Introduction 15](#_TOC_250018)
  2. [Description 15](#_TOC_250017)
  3. [Hardware Requirement 16](#_TOC_250016)
     1. [NodeMCU ESP8266 Module 17](#_TOC_250015)
     2. [Pulse Sensor 24](#_TOC_250014)
     3. [LCD Module 27](#_TOC_250013)
     4. [I2C LCD Module 29](#_TOC_250012)
  4. [Software Requirement 30](#_TOC_250011)
     1. [Arduino IDE Software 30](#_TOC_250010)
     2. [ThingSpeak 32](#_TOC_250009)

[CHAPTER 3: PROCEDURE](#_TOC_250008)

* + 1. [Steps Involved 36](#_TOC_250007)
    2. [Hardware Setup 36](#_TOC_250006)
    3. [Software Setup 36](#_TOC_250005)
    4. [Arduino Code for Heart-Rate Monitoring 37](#_TOC_250004)
  1. [Flow Chart of the Project 40](#_TOC_250003)

CHAPTER 4: RESULT & ANALYSIS

* 1. Result 41
  2. [Applications 42](#_TOC_250002)
  3. [Conclusion 42](#_TOC_250001)
  4. [Future Scope 42](#_TOC_250000)

REEFERENCES 43

# LIST OF FIGURES

|  |  |  |
| --- | --- | --- |
| Fig 1.1 | Block diagram of Embedded System | 08 |
| Fig 2.3.1 | NodeMCU ESP8266 Module | 17 |
| Fig 2.3.1.2 | Pin Definition of NodeMCU ESP8266 | 18 |
| Fig 2.3.1.3 | NodeMCU Connection with pc | 21 |
| Fig 2.3.1.4 | Arduino IDE | 21 |
| Fig 2.3.1.5 | Serial monitor | 24 |
| Fig 2.3.2.1 | Pulse sensor | 25 |
| Fig 2.3.3.1 | LCD Specifications | 27 |
| Fig 2.3.3.2 | Liquid Crystal Display | 28 |
| Fig 2.3.4.1 | I2C Module | 29 |
| Fig 2.4.1 | Menu bar of Arduino IDE | 31 |
| Fig 2.4.2.1 | ThingSpeak Platform | 33 |
| Fig 2.4.2.2 | Channel Description | 34 |
| Fig 4.4.1 | Flow Chart | 40 |

**vii**



**HEART RATE MONITRING SYSTEM OVER INTERNET**

**CHAPTER – 1 INTRODUCTION**

* 1. **Introduction to Embedded System**

An embedded system is a special-purpose computer system designed to perform one or few dedicated functions, sometimes with real time computing constraints. It is usually embedded as a part of complete device including hardware and mechanical parts. In contrast general purpose computer, such as personal computer, can do many different tasks depending on programming. Embedded systems have become very important today as they control many of the common devices we use.

Since the embedded system is dedicated to specific tasks, design engineers can optimize it, reducing the cost of the product, or increasing the reliability and performance. Some embedded systems are mass produced, benefiting from economies of scale.

In general ―embedded system‖ is not an exactly defined term, as many systems have some elements of programmability. For example, Handheld computers share some elements with embedded systems – such as the operating systems and microprocessors which power them – but are not truly embedded systems, because they allow different applications to be loaded and peripherals to be connected.

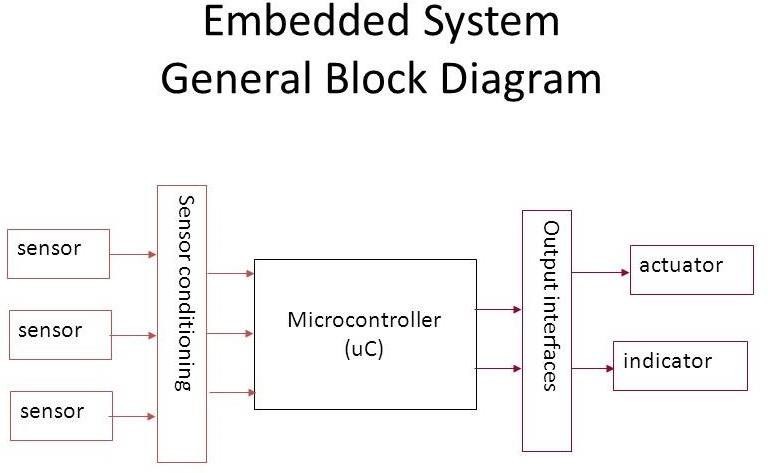
An embedded system is some combination of computer hardware and software, either fixed in capability or programmable, that is specifically designed for a particular kind of application device. Industrial machines, automobiles, medical equipment, cameras, household appliances, airplanes, vending machines, and toys (as well as the more obvious cellular phone and PDA) are among the myriad possible hosts of an embedded system. Embedded systems that are programmable are provided with a programming interface; an embedded systems programming is a specialized occupation.

The microprocessor-based systems built for controlling a function or range of function’s and is not designed to be programmed by the end user in the same way a PC is defined as an embedded system. An embedded system is designed to perform one particular task albeit with different choices and options.

Embedded systems contain processing or that are either microcontrollers or digital signal processors. Microcontrollers generally known as "chip", which may itself be packaged with the

microcontroller hybrid system of Application- Specific Integrated Circuit (ASIC). In general, input always comes from a detector or sensors in more specific word and mean while the output goes to the activator which may start or stop the operation of the machine or the operating system.

An embedded system is a combination of both hardware and software. Each embedded system is unique and the hardware is highly specialized in the application domain. Hardware consists of processors, microcontroller, IR sensors. On the other hand, Software are just like a brain of the whole embedded system as this consists of the programming languages used which make hardware work. As a result, embedded systems programming can be a widely varying experience.



#### Fig 1.1: Block diagram for Embedded System

An embedded system is a computer system with a dedicated function within a larger mechanical or electrical system, often with real-time computing constraints. It is embedded as part of a complete device often including hardware and mechanical parts. Embedded systems control many devices in common use today. 98 percent of all microprocessors are manufactured as components of embedded systems.

Examples of properties of typically embedded computers when compared with general- purpose counterparts are low power consumption, small size, rugged operating ranges, and low per-unit cost. This comes at the price of limited processing resources, which make them significantly more difficult to program and to interact with. However, by building intelligence mechanisms on top of the hardware, taking advantage of possible existing sensors and the

existence of a network of embedded units, one can both optimally manage available resources at those available. For example, intelligent techniques can be designed to manage power consumption of embedded systems.

Modern embedded systems are often based on microcontrollers (i.e. CPUs with integrated memory or peripheral interfaces), but ordinary microprocessors (using external chips for memory and peripheral interface circuits) are also common, especially in more- complex systems. In either case, the processor(s) used may be types ranging from general purpose to those specialized in certain class of computations, or even custom designed for the application at hand. A common standard class of dedicated processors is the digital signal processor (DSP).

Since the embedded system is dedicated to specific tasks, design engineers can optimize it to reduce the size and cost of the product and increase the reliability and performance. Some embedded systems are mass-produced, benefiting from economies of scale.

Embedded systems range from portable devices such as digital watches and MP3 player, to large stationary installations like traffic lights, factory controllers, and largely complex systems like hybrid vehicles, MRI, and avionics. Complexity varies from low, with a single microcontroller chip, to very high with multiple units, peripherals and networks mounted inside a large chassis or enclosure.

## Characteristics:

Embedded systems are designed to do some specific task, rather than be a general- purpose computer for multiple tasks. Some also have real-time performance constraints that must be met, for reasons such as safety and usability; others may have low or no performance requirements, allowing the system hardware to be simplified to reduce costs.

Embedded systems are not always standalone devices. Many embedded systems consist of small parts within a larger device that serves a more general purpose. For example, the Gibson Robot Guitar features an embedded system for tuning the strings, but the overall purpose of the Robot Guitar is, of course, to play music. Similarly, an embedded system in an automobile provides a specific function as a subsystem of the car itself.

The program instructions written for embedded systems are referred to as firmware, and are stored in read-only memory or Flash memory chips. They run with limited computer hardware resources: little memory, small or non-existent keyboard or screen.

Embedded systems range from no user interface at all, in systems dedicated only to one task, to complex graphical user interfaces that resemble modern computer desktop

operating systems. Simple embedded devices use buttons, LEDs, graphic or character LCDs (HD44780 LCD for example) with a simple menu system.

More sophisticated devices which use a graphical screen with touch sensing or screen- edge buttons provide flexibility while minimizing space used: the meaning of the buttons can change with the screen, and selection involves the natural behavior of pointing at what's desired. Handheld systems often have a screen with a "joystick button" for a pointing device.

Some systems provide user interface remotely with the help of a serial (e.g. RS- 232, USB, I²C, etc.) or network (e.g. Ethernet) connection. This approach gives several advantages: extends the capabilities of embedded system, avoids the cost of a display, simplifies BSP and allows one to build a rich user interface on the PC. A good example of this is the combination of an embedded web server running on an embedded device (such as an IP camera) or a network router. The user interface is displayed in a web browser on a PC connected to the device, therefore needing no software to be installed.

### Applications of Embedded system

* + - Military and aerospace software applications
    - Communication applications
    - Electronic applications and consumer devices
    - Industrial automation and process control software

## Micro Processor for Embedded System

Microprocessor is a computer processor that incorporates the functions of a computer's central processing unit (CPU) on a single integrated circuit (IC), or at most a few integrated circuits. The microprocessor is a multipurpose, clock driven, register based, programmable electronic device that accepts digital data or binary data as input, processes it according to instructions stored in its memory, and provides results as output. Microprocessors contain both combinational logic and sequential digital logic. Microprocessors operate on numbers and symbols represented in the binary numeral system.

The integration of a whole CPU onto a single chip or on a few chips greatly reduced the cost of processing power. Integrated circuit processors are produced in large numbers by highly automated processes resulting in a low per unit cost. Single-chip processors increase reliability as there are many fewer electrical connections to fail. As microprocessor designs get faster, the cost of manufacturing a chip (with smaller components built on a semiconductor chip the same size) generally stays the same.



**HEART RATE MONITRING SYSTEM OVER INTERNET**

Before microprocessors, small computers had been implemented using racks of circuit boards with many medium- and small-scale integrated circuits. Microprocessors integrated this into one or a few large-scale ICs. Continued increases in microprocessor capacity have since rendered other forms of computers almost completely obsolete (see history of computing hardware), with one or more microprocessors used in everything from the smallest embedded systems and handheld devices to the largest mainframes and supercomputers.

A microprocessor is a general-purpose system. Several specialized processing devices have followed from the technology. Microcontrollers integrate a microprocessor with peripheral devices in embedded systems. A digital signal processor (DSP) is specialized for signal processing. Graphics processing units may have no limited or general programming facilities. For example, GPUs through the 1990s were mostly non-programmable and have only recently gained limited facilities like programmable vertex shader.

32-bit processors have more digital logic than narrower processors, so 32-bit (and wider) processors produce more digital noise and have higher static consumption than narrower processors. Reducing digital noise improves ADC conversion results. So, 8-bit or 16-bit processors are better than 32-bit processors for system on a chip and microcontrollers that require extremely low-power electronics, or are part of a mixed-signal integrated circuit with noise-sensitive on-chip analog electronics such as high-resolution Analog to digital converters, or both.

Nevertheless, trade-offs apply: running 32-bit arithmetic on an 8-bit chip could end up using more power, as the chip must execute software with multiple instructions. Modern microprocessors go into low power states when possible, and a 8-bit chip running 32-bit software is active most of the time. This creates a delicate balance between software, hardware and use patterns, plus costs.

When manufactured on a similar process, 8-bit microprocessors use less power when operating and less power when sleeping than 32-bit microprocessors.

However, some people say a 32-bit microprocessor may use less average power than an 8-bit microprocessor when the application requires certain operations such as floating-point math that take many more clock cycles on an 8-bit microprocessor than a 32-bit microprocessor so the 8-bit microprocessor spends more time in high-power operating mode.

Microprocessors are silicon chips that contain a computer's central processing unit (CPU)—the device that executes commands entered into the computer. Along with clocks and



**HEART RATE MONITRING SYSTEM OVER INTERNET**

main memory, CPUs are among a computer's main components. The terms CPU and microprocessor often are used interchangeably. Essentially, microprocessors are responsible for manipulating data and performing numeric calculations and logical comparisons. At the heart of microprocessors are tiny electronic switches called transistors, which allow digital computers to process information in the form of electrical signals. These signals are in one of two states (on or off), and are represented by ones and zeroes, respectively. High-level programming languages like Java or C++, used to write popular software programs, eventually are translated to the machine language of ones and zeroes that computers understand.

Intel was the first company to produce a microprocessor for commercial use. Called the 4004, it was released in the early 1970s and contained slightly more than 2,000 transistors. By the early 2000s, microprocessors contained more than 5 million transistors on a single silicon chip. The more transistors a chip has, the more quickly it can process information. A microprocessor's clock speed defines the number of instructions it can carry out per second. This figure is expressed in Megahertz (MHz) or Gigahertz (GHz). In 2001 the processing speeds of some microprocessors exceeded 1.7 GHz.

In 1965, Intel Co-Founder Gordon E. Moore predicted the number of transistors manufacturers could fit onto a silicon chip would double every 18 months. Because his prediction proved to be accurate over time, it came to be known as Moore's Law. The law eventually will expire when it becomes physically impossible for manufacturers to fit any more transistors onto a single chip. This is expected to happen somewhere around 2017 or 2020 when transistors are atom-sized. At that time, a new computing architecture will be necessary. One possibility is quantum computing, which relies on atomic properties instead of transistors to determine the one’s and zero’s a computer understands. According to Infoworld "quantum computers rely on a particle's traits, such as the direction of its spin, for creating a state. For example, when the spin is up, a particle could be read as 'one,' and when its spin is down, the particle would be read as 'zero.'

In mid-2001 Intel announced experimental technology that it called "Wireless- Internet-On- A-Chip." Essentially, the technology consisted of a silicon chip that held a microprocessor, as well as Analog communication circuits and flash memory. According to Intel, the technology potentially would lead to the development of more powerful wireless Internet devices. Around the same time, Intel and Hewlett-Packard announced the launch of



**HEART RATE MONITRING SYSTEM OVER INTERNET**

the Itanium Processor, a new generation of microprocessor the companies co-developed for use in servers and workstation computers.

Microprocessors are the devices in a computer which make things happen. Microprocessors are capable of performing basic arithmetic operations, moving data from place to place, and making basic decisions based on the quantity of certain values.

The vast majority of microprocessors can be found in embedded microcontrollers. The second most common type of processors are common desktop processors, such as Intel's Pentium or AMD's Athlon. Less common are the extremely powerful processors used in high- end servers, such as Sun's SPARC, IBM's Power, or Intel's Itanium.

Historically, microprocessors and microcontrollers have come in "standard sizes" of 8 bits, 16 bits, 32 bits, and 64 bits. These sizes are common, but that does not mean that other sizes are not available. Some microcontrollers (usually specially designed embedded chips) can come in other "non-standard" sizes such as 4 bits, 12 bits, 18 bits, or 24 bits. The number of bits represent how much physical memory can be directly addressed by the CPU. It also represents the amount of bits that can be read by one read/write operation. In some circumstances, these are different; for instance, many 8-bit microprocessors have an 8-bit data bus and a 16-bit address bus.

8-bit processors can read/write 1 byte at a time and can directly address 256 bytes

1. bit processors can read/write 2 bytes at a time, and can address 65,536 bytes (64 Kilobytes) 32- bit processors can read/write 4 bytes at a time, and can address 4,294,967,295 bytes (4 Gigabytes). 64-bit processors can read/write 8 bytes at a time, and can address 18,446,744,073,709,551,616 bytes (16 Extra bytes).

## Micro Processor:

A microprocessor incorporates the functions of a CPU on a single integrated circuit or a few integrated circuits. It is a computer processor on a microchip and is a multipurpose, programmable device that uses digital data as input and provides results as an output once it processes the input according to instructions stored in its memory. Microprocessors use sequential digital logic as they have internal memory and operate on numbers and symbols represented in the binary numeral system. They are designed to perform arithmetic and logic operations that make use of data on the chip. General purpose microprocessors in PCs are used for multimedia display, computation, text editing and communication. Several microprocessors are part of embedded systems. These embedded microprocessors provide

digital control to several objects including appliances, automobiles, mobile phones and industrial process control.



**HEART RATE MONITRING SYSTEM OVER INTERNET**

The microprocessor contains all, or most of, the central processing unit (CPU) functions and is the "engine" that goes into motion when you turn your computer on. A microprocessor is designed to perform arithmetic and logic operations that make use of small number-holding areas called registers. Typical microprocessor operations include adding, subtracting, comparing two numbers, and fetching numbers from one area to another. These operations are the result of a set of instructions that are part of the microprocessor design.

When your computer is turned on, the microprocessor gets the first instruction from the basic input/output system (BIOS) that comes with the computer as part of its memory. After that, either the BIOS, or the operating system that BIOS loads into computer memory, or an application program is "driving" the microprocessor, giving it instructions to perform.

## Applications of Microprocessor:

* + DVD players
  + Cellular Telephones
  + Household Appliances
  + Car Equipment
  + Toys
  + Light Switches and Dimmers
  + Electrical Circuit Breakers
  + Smoke Alarms
  + Battery Packs
  + Car Keys
  + Power tool and Test Instruments.

# Introduction:

**CHAPTER – 2**

**Heart Rate Monitoring System**



**HEART RATE MONITRING SYSTEM OVER INTERNET**

Heart rate monitoring over internet is an Internet of Things based device which is used to monitor the heart rate or pulses of a patient without being remotely present with him. A Heart Rate Monitor is a device which calculates the pulses from a human body with the help of a pulse sensor. A Pulse Sensor uses the concept of light emission and absorption to calculate the pulse and converts it into an analog signal. This device is advantageous for patients who require continuous monitoring of their heart rate and is also advantageous for the doctors and the family members of the patient as they can check the real time pulse of their patient without being physically present with him. By the introduction of IoT in this project, we were successful to convert a basic heart monitor into a smart device with better data displaying. We use ThingSpeak as an IoT platform which shows all the data in forms of graphs. It shows real time data and past data as well.

# Description:

This Project consists of a NodeMCU which is the heart of this device. It is a microcontroller device integrated with Wi-Fi module. It has 30 pins with various uses. It supports analog and also digital data and gives us desired outputs. The input source is a pulse sensor which works on the concepts of Light emission and absorption. It emits light from the LED present on the sensor which falls on the blood vessels present under the skin. The movement of blood absorbs a part of light and reflects a part of light. The reflected light is absorbed by the LDR present on the sensor which converts that light into pulses. This data generated is shown onto the LCD screen physically present near to the patient. The pulses are also shown on Thingspeak IoT firmware which shows the pulses in the form of graphs. By using a unique ID and password for every individual patient , the doctors and also the family members of the patient can remotely access the patients real time pulse data without being physically present near them.

# HARDWARE REQUIREMENT:



**HEART RATE MONITRING SYSTEM OVER INTERNET**

This chapter briefly explains about the Hardware Implementation of the project. It discusses the design and working of the design with the help of block diagram and circuit diagram and explanation of circuit diagram in detail.

The implementation of the project design can be divided in two parts.

* + 1. Hardware implementation
    2. Firmware implementation

Hardware implementation deals in drawing the schematic on the plane paper according to the application, testing the schematic design over the breadboard using the various IC’s to find if the design meets the objective, carrying out the PCB layout of the schematic tested on breadboard, finally preparing the board and testing the designed hardware.

The project design and principle are explained in this chapter using the block diagram and circuit diagram. The block diagram discusses about the required components of the design and working condition is explained using circuit diagram and system wiring diagram.

Based on the Processor side Embedded Systems is mainly divided into 3 types:

1. **Micro Processor: -** are for general purpose ex: our personal computer
2. **Micro Controller: -** are for specific applications, because of cheaper cost
3. **DSP (Digital Signal Processor): -** are for high and sensitive application purpose

# NodeMCU ESP8266 module:



**HEART RATE MONITRING SYSTEM OVER INTERNET**

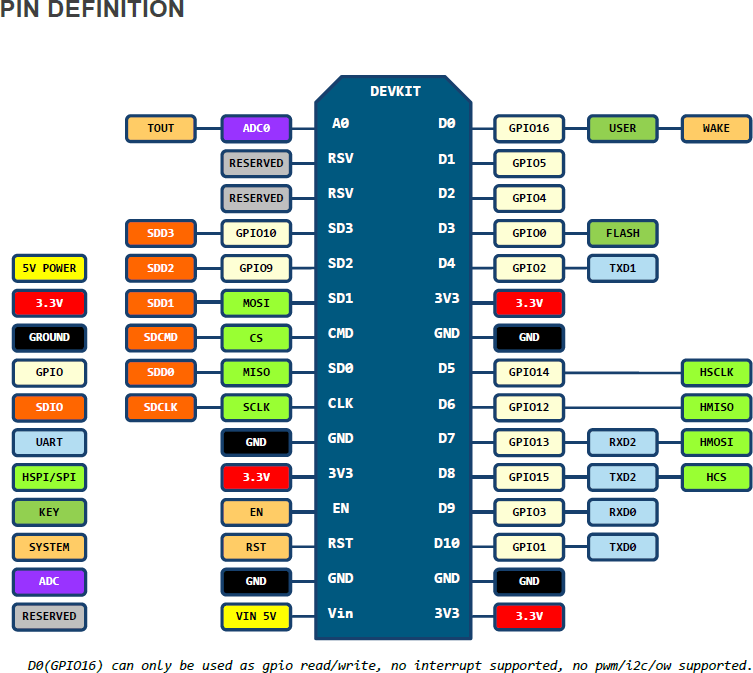


#### Fig 2.3.1 NodeMCU ESP8266 module

**NodeMCU** is an open source IoT platform. It includes firmware which runs on the ESP8266 Wi-Fi SoC from Espressif Systems, and hardware which is based on the ESP-12 module. NodeMCU was created shortly after the ESP8266came out. On December 30, 2013, Espressif Systems began production of the ESP8266



**HEART RATE MONITRING SYSTEM OVER INTERNET**



#### Fig 2.3.1.2 Pin Definition of NodeMCU ESP8266

NodeMCU is an open-source firmware and development kit that helps you to prototype or build IoT product. It includes firmware which runs on the ESP8266 Wi-Fi SoC from Espressif Systems, and hardware which is based on the ESP-12 module. The firmware uses the Lua scripting language. It is based on the eLua project, and built on the Espressif Non-OS SDK for ESP8266.

MCU stands for Microcontroller Unit - which really means it is a computer on a single chip. A microcontroller contains one or more CPUs (processor cores) along with memory and programmable input/output peripherals. They are used to automate automobile engine control, implantable medical devices, remote controls, office machines, appliances, power tools, toys etc.

**NodeMCU:**

NodeMCU Development board is featured with Wi-Fi capability, analog pin, digital pins and serial communication protocols.

To get start with using NodeMCU for IoT applications first we need to know about how to write/download NodeMCU firmware in NodeMCU Development Boards. And before that where this NodeMCU firmware will get as per our requirement.

There is online NodeMCU custom builds available using which we can easily get our custom NodeMCU firmware as per our requirement



**HEART RATE MONITRING SYSTEM OVER INTERNET**

## Writing codes for NodeMCU:

After setting up ESP8266 with Node-MCU firmware, let’s see the IDE (Integrated Development Environment) required for development of NodeMCU.

## NodeMCU with ESPlorer IDE:

Lua scripts are generally used to code the NodeMCU. Lua is an open source, lightweight, embeddable scripting language built on top of C programming language.

For more information about how to write Lua script for NodeMCU refer Getting started with NodeMCU using ESPlorerIDE

## NodeMCU with Arduino IDE:

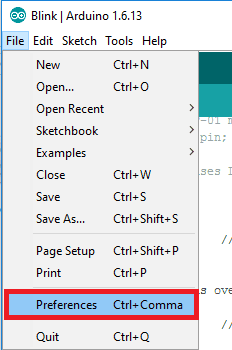
Here is another way of developing NodeMCU with a well-known IDE i.e. Arduino IDE. We can also develop applications on NodeMCU using Arduino development environment. This makes easy for Arduino developers than learning new language and IDE for NodeMCU. NodeMCU is Lua based firmware of ESP8266. Generally, ESPlorer IDE is referred for writing Lua scripts for NodeMCU. It requires to get familiar with ESPlorer IDE and Lua scripting language.

There is another way of developing NodeMCU with a well-known IDE i.e. Arduino IDE. We can also develop NodeMCU applications using Arduino development environment. This makes things easy for Arduino developers than learning new language and IDE for NodeMCU.

Let’s see about setting up Arduino IDE with NodeMCU.

First **Download Arduino IDE (version 1.6+)** https:/[/www](http://www.arduino.cc/en/Main/Software).[arduino.cc/en/Main/Software](http://www.arduino.cc/en/Main/Software)

#### Open Arduino IDE and Go to File -> Preference.



* + - * Now on Preference window, **Enter below link in Additional Boards Manager URLs**

<http://arduino.esp8266.com/stable/package_esp8266com_index.json>

* + - * Now close Preference window and **go to Tools -> Board -> Boards Manager**

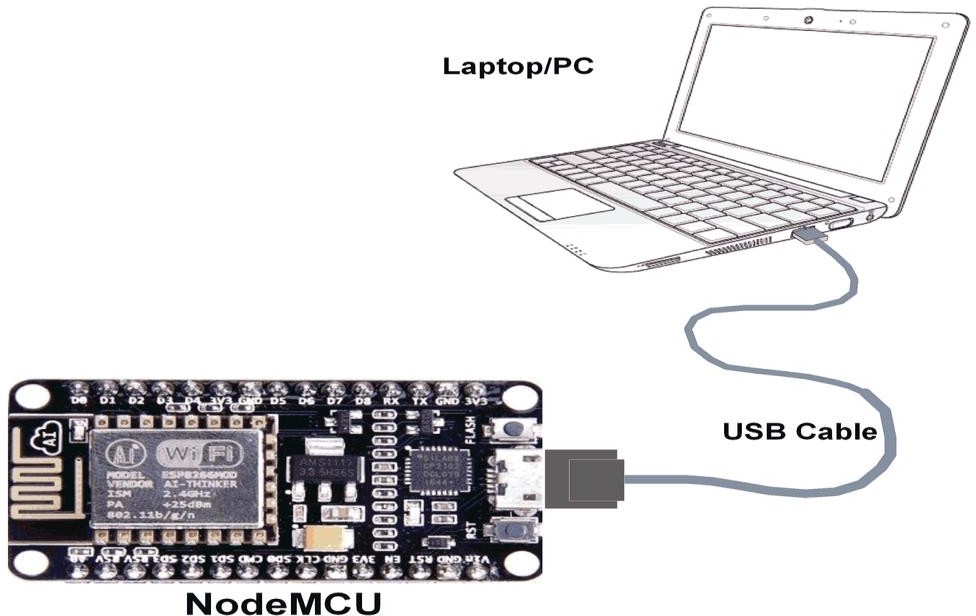
#### In Boards Manager window, Type ESP in the search box, esp8266 will be listed there below. Now select latest version of board and click on install.

* + - * After installation of the board is complete, **open Tools->Board->and select NodeMCU 1.0(ESP-12E Module)**

#### Now Your Arduino IDE is ready for NodeMCU

**Example**

Let’s see how to write simple serial print sketch using Arduino IDE for NodeMCU. First connect NodeMCU Development Kit with PC as shown in below figure.



#### Fig 2.3.1.3 NodeMCU connection with PC

* + - * After setting up Arduino IDE for NodeMCU, **open Arduino IDE and write simple sketch of serial print** as shown in below figure.



#### Fig 2.3.1.4 Arduino IDE 1

**Arduino Sketch**

void setup()

{

Serial.begin(9600); /\* initialise serial communication \*/

}

void loop()

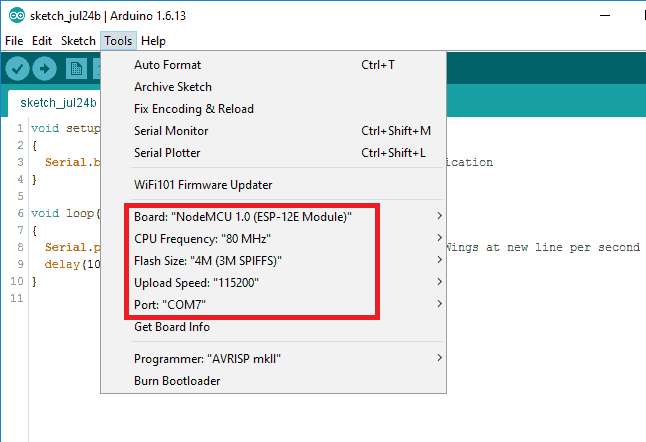
{

Serial.println("ElectronicWings"); /\* print Electronic Wings at new line per second \*/

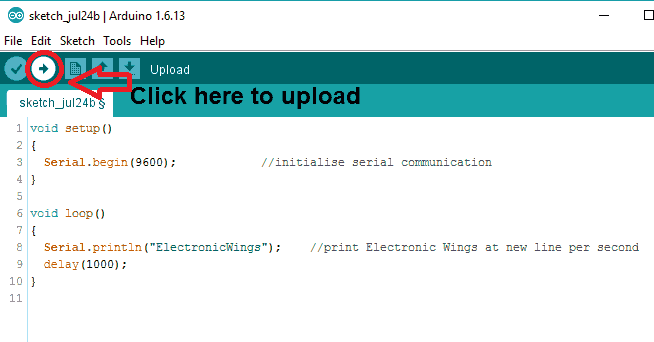
delay(1000);

}

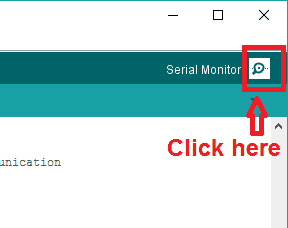
* + - * Ensure that you have selected the correct board as shown in below figure**. Also** make sure that you have selected the appropriate COM port.



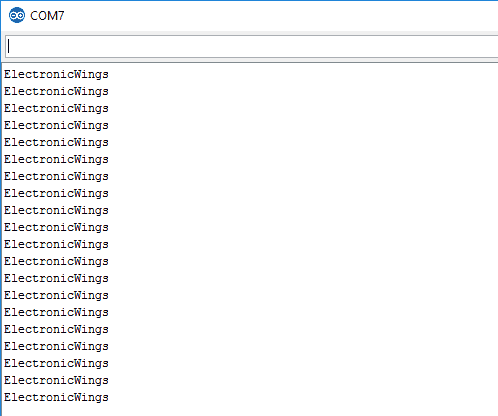
* + - * Now **compile & upload the written sketch** directly to the NodeMCU Dev Kit by clicking on upload button.



* + - * Now **Click on Serial Monitor (upper right corner) option** to check output on serial monitor window of Arduino IDE.



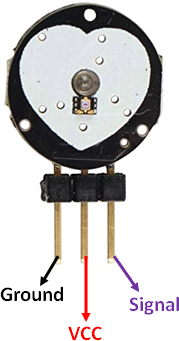
* + - * Serial monitor output window will pop up with output as shown in below figure.



#### Fig 2.3.1.5 Serial monitor

# Pulse Sensor:

Pulse Sensor is a well-designed plug-and-play heart-rate sensor for Arduino. It can be used by students, artists, athletes, makers, and game & mobile developers who want to easily incorporate live heart rate data into their projects. The sensor clips onto a fingertip or earlobe and plugs right into Arduino. It also includes an open-source monitoring app that graphs your pulse in real time.



## Fig. 2.3.2.1 Pulse Sensor

The Pulse Sensor can be connected to Arduino, or plugged into a breadboard. The front of the sensor is the pretty side with the Heart logo. This is the side that makes contact with the skin. On the front you see a small round hole, which is where the LED shines through from the back, and there is also a little square just under the LED. The square is an ambient light sensor, exactly like the one used in phones, tablets, and laptops, to adjust the screen brightness in different light conditions. The LED shines light into the fingertip or earlobe, or other capillary tissue, and sensor reads the light that bounces back. The back of the sensor is where the rest of the parts are mounted.

**Pulse sensor Specifications:**

* Operating voltage: 3.3V – 5V
* Current: 4mA
* Indicator LED

## Pin Configuration:

|  |  |  |  |
| --- | --- | --- | --- |
| **Pin Number** | **Pin Name** | **Wire Colour** | **Description** |
| 1 | Ground | Black | Connected to the ground of the system |
| 2 | Vcc | Red | Connect to +5V or +3.3V supply voltage |
| 3 | Signal | Purple | Pulsating output signal. |

**Measurement of Pulses:**

The working of the Pulse/Heart beat sensor is very simple. The sensor has two sides, on one side the LED is placed along with an ambient light sensor and on the other side we have some circuitry. This circuitry is responsible for the amplification and noise cancellation work. The LED on the front side of the sensor is placed over a vein in our human body. This can either be your Finger tip or you ear tips, but it should be placed directly on top of a vein.

Now the LED emits light which will fall on the vein directly. The veins will have blood flow inside them only when the heart is pumping, so if we monitor the flow of blood, we can monitor the heart beats as well. If the flow of blood is detected then the ambient light sensor will pick up more light since they will be reflected by the blood, this minor change in received light is analysed over time to determine our heart beats.

## Applications:

* Sleep Tracking
* Anxiety monitoring
* Remote patient monitoring/alarm system
* Health bands
* Advanced gaming consoles

# LCD Module:

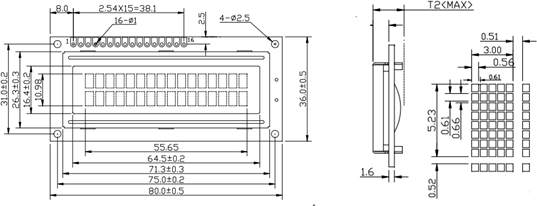


**HEART RATE MONITRING SYSTEM OVER INTERNET**

LCD modules are very commonly used in most embedded projects, the reason being its cheap price, availability and programmer friendly. Most of us would have come across these displays in our day to day life, either at PCO’s or calculators. The appearance and the pinouts have already been visualized above now let us get a bit technical.

**16×2 LCD** is named so because; it has 16 Columns and 2 Rows. There are a lot of combinations available like, 8×1, 8×2, 10×2, 16×1, etc. but the most used one is the 16×2 LCD. So, it will have (16×2=32) 32 characters in total and each character will be made of 5×8 Pixel Dots. A Single character with all its Pixels is shown in the below picture.

Now, we know that each character has (5×8=40) 40 Pixels and for 32 Characters we will have (32×40) 1280 Pixels. Further, the LCD should also be instructed about the Position of the Pixels. Hence it will be a hectic task to handle everything with the help of MCU, hence an **Interface IC like HD44780** is used, which is mounted on the backside of the LCD Module itself. The function of this IC is to get the **Commands and Data** from the MCU and process them to display meaningful information onto our LCD Screen. You can learn how to interface an LCD using the above-mentioned links. If you are an advanced programmer and would like to create your own library for interfacing your Microcontroller with this LCD module then you have to understand the HD44780 IC is working and commands which can be found its datasheet.



## Fig. 2.3.3.1 LCD SPECIFICATIONS



**Fig. 2.3.3.2 Liquid Crystal Display**

|  |  |  |
| --- | --- | --- |
| **Pin No:** | **Pin Name:** | **Description** |
| 1 | Vss (Ground) | Ground pin connected to system ground |
| 2 | Vdd (+5 Volt) | Powers the LCD with +5V (4.7V – 5.3V) |
| 3 | VE (Contrast V) | Decides the contrast level of display. Grounded to get maximum contrast. |
| 4 | Register Select | Connected to Microcontroller to shit between command/data register |
| 5 | Read/Write | Used to read or write data. Normally grounded to write data to LCD |
| 6 | Enable | Connected to Microcontroller Pin and toggled between 1 and 0 for data acknowledgement |
| 7 | Data Pin 0 | Data pins 0 to 7 forms a 8-bit data line. They can be connected to |
| 8 | Data Pin 1 |

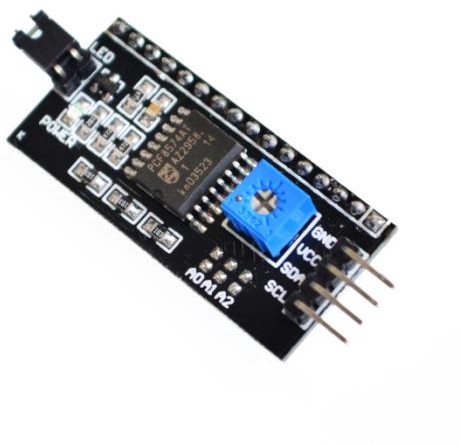
|  |  |  |
| --- | --- | --- |
| 9 | Data Pin 2 | Microcontroller to send 8-bit data.  These LCD’s can also operate on 4-bit mode in such case Data pin 4,5,6 and 7 will be left free. |
| 10 | Data Pin 3 |
| 11 | Data Pin 4 |
| 12 | Data Pin 5 |
| 13 | Data Pin 6 |
| 14 | Data Pin 7 |
| 15 | LED Positive | Backlight LED pin positive terminal |
| 16 | LED Negative | Backlight LED pin negative terminal |

# I2C LCD Module:

IC/I2C Interface Adapter Module is used for 16×2 LCD Display. It uses the PCF8574T IC chip which converts I2C serial data to parallel data for the LCD display. Also this interface module simplifies connecting an Arduino to a 16×2 Liquid Crystal display using only 4 wires.

## Specifications:

* + Supply Voltage: 5V
  + Interface: I2C
  + Compatible for 16×2 LCD
  + Brightness and Contrast can be adjusted by the Potentiometer



## Fig. 2.3.4.1 I2C Module

# SOFTWARE REQUIREMENT:



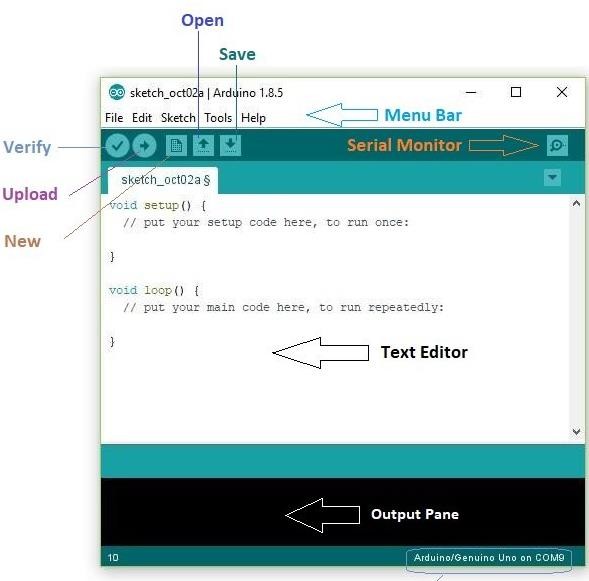
**HEART RATE MONITRING SYSTEM OVER INTERNET**

The software’s which we are using in this project are Arduino IDE and ThingSpeak IoT platform.

* + 1. **ARDUINO IDE SOFTWARE:**
* Arduino IDE is an open source software that is mainly used for writing and compiling the code into the Arduino Module.
* It is an official Arduino software, making code compilation too easy that even a common person with no prior technical knowledge can get their feet wet with the learning process.
* It is easily available for operating systems like MAC, Windows, Linux and runs on the Java Platform that comes with inbuilt functions and commands that play a vital role for debugging, editing and compiling the code in the environment.
* A range of Arduino modules available including Arduino Uno, Arduino Mega, Arduino Leonardo, Arduino Micro and many more.
* Each of them contains a microcontroller on the board that is actually programmed and accepts the information in the form of code.
* The main code, also known as a sketch, created on the IDE platform will ultimately generate a Hex File which is then transferred and uploaded in the controller on the board.
* The IDE environment mainly contains two basic parts: Editor and Compiler where former is used for writing the required code and later is used for compiling and uploading the code into the given Arduino Module.
* This environment supports both C and C++ languages.



**HEART RATE MONITRING SYSTEM OVER INTERNET**



#### fig 2.4.1 Menu Bar of Arduino IDE

The bar appearing on the top is called **Menu Bar** that comes with five different options as follow

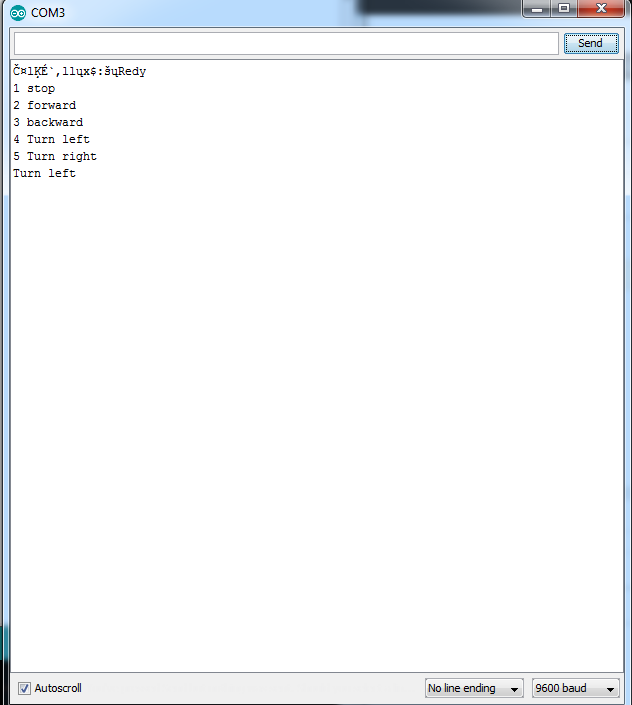
* **File** – You can open a new window for writing the code or open an existing one.
* **Edit** – Used for copying and pasting the code with further modification for font
* **Sketch** – For compiling and programming.
* **Tools** – Mainly used for testing projects. The Programmer section in this panel is used for burning a bootloader to the new microcontroller.
* **Help** – In case you are feeling sceptical about software, complete help is available from getting started to troubleshooting.
* The button appearing on the top right corner is a **Serial Monitor** – A separate pop-up window that acts as an independent terminal and plays a vital role for sending and receiving the Serial Data. You can also go to the Tools panel and select Serial

Monitor, or pressing Ctrl+Shift+M all at once will open it instantly. The Serial Monitor will actually help to debug the written Sketches where you can get a hold of how your program is operating. Your Arduino Module should be connected to your computer by USB cable in order to activate the Serial Monitor.



**HEART RATE MONITRING SYSTEM OVER INTERNET**

* You need to select the baud rate of the Arduino Board you are using right now. For my Arduino Uno Baud Rate is 9600, as you write the following code and click the Serial Monitor, the output will show as the image below.



# THINGSPEAK:

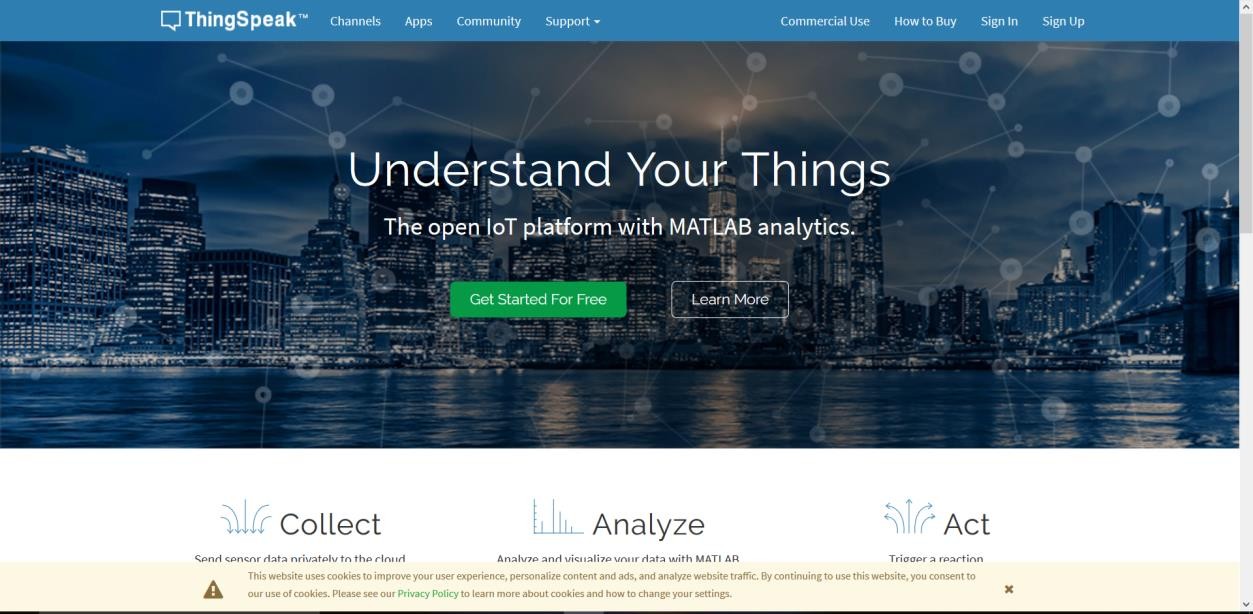
ThingSpeak is an IoT analytics platform service that allows you to aggregate, visualize, and analyse live data streams in the cloud. You can send data to ThingSpeak from your devices, create instant visualizations of live data, and send alerts using web services like Twitter. With MATLAB analytics inside ThingSpeak, you can write and execute MATLAB code to perform pre-processing, visualizations, and analyses. ThingSpeak enables

engineers and scientists to prototype and build IoT systems without setting up servers or developing web software. Key capabilities of ThingSpeak include:



**HEART RATE MONITRING SYSTEM OVER INTERNET**

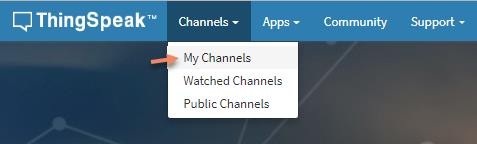
* Configure devices to send data to ThingSpeak using a REST API or MQTT.
* Aggregate data on-demand from devices and third-party sources.
* Get instant visualizations of live or historical sensor data.
* Pre-process and analyse your collected data using integrated MATLAB.
* Run your IoT analytics automatically based on schedules or events.
* Act on your data and communicate using third-party services like Twilio or Twitter.



## Fig. 2.4.2.1 ThingSpeak Platform

**Create a Channel:**

* + - 1. Sign In to ThingSpeak using your MathWorks Account, or create a new MathWorks account.
      2. Click **Channels** > **MyChannels**.

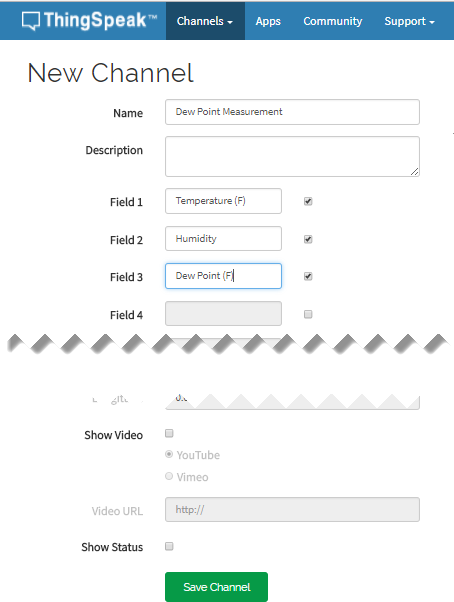


* + - 1. On the Channels page, click **New Channel**.



**HEART RATE MONITRING SYSTEM OVER INTERNET**

* + - 1. Check the boxes next to Fields 1–3. Enter these channel setting values:
         * **Name**: Dew Point Measurement
         * **Field 1**: Temperature (F)
         * **Field 2**: Humidity
         * **Field 3**: Dew Point



# Fig. 2.4.2.2 Channel Description

* + - 1. Click **Save Channel** at the bottom of the settings.



**HEART RATE MONITRING SYSTEM OVER INTERNET**

You now see these tabs:

* + - * + **Private View**: This tab displays information about your channel that only you can see.
        + **Public View**: If you choose to make your channel publicly available, use this tab to display selected fields and channel visualizations.
        + **Channel Settings**: This tab shows all the channel options you set at creation. You can edit, clear, or delete the channel from this tab.
        + **Sharing**: This tab shows channel sharing options. You can set a channel as private, shared with everyone (public), or shared with specific users.
        + **API Keys**: This tab displays your channel API keys. Use the keys to read from and write to your channel.
        + **Data Import/Export**: This tab enables you to import and export channel data.

# CHAPTER – 3 PROCEDURE



**HEART RATE MONITRING SYSTEM OVER INTERNET**

## Steps Involved:

* + - Firstly, we have written a code interfacing NodeMCU and pulse sensor using Arduino IDE
    - Later we compiled it and checked the output on serial monitor
    - Created one channel on thingspeak and linked the code to the channel
    - Connecting hardware interfacing NodeMCU, pulse sensor and LCD Module.
    - Attaching the pulse sensor to the index finger of a person and checking the live streaming of data on thingspeak
    - Recording the data for further analysis
    - Comparing the stored data and coming to a conclusion

## HARDWARE SETUP:

* + - Connect Pulse sensor ground pin to ground of NodeMCU.
    - Connect Pulse sensor data pin to Analog pin(A0) of NodeMCU.
    - Connect Pulse sensor power supply pin to 3.3v pin of NodeMCU.
    - Connect all the 16 pins of LCD module to the I2C LCD module.
    - Connect I2C LCD module ground pin to ground of NodeMCU.
    - Connect I2C LCD module VCC pin to Vin pin of NodeMCU.
    - Connect I2C LCD module SDA pin to D2 pin of NodeMCU.
    - Connect I2C LCD module SCL pin to D1 pin of NodeMCU.

## SOFTWARE SETUP:

* + - ThingSpeak
    - Log into thingspeak account
    - Create a ThingSpeak channel
    - Create 1 field for pulse sensor graphical representation.
    - Thingspeak is used for live streaming of data and when it comes to analysis part , the data from thingspeak is stored in excel sheet and do the analysis and comparison of the patients pulses.



**HEART RATE MONITRING SYSTEM OVER INTERNET**

# 3.3.1 ARDUINO CODE FOR HEART-RATE MONITORING:

#include <ESP8266WiFi.h> #include<Wire.h> #include<LiquidCrystal\_I2C.h>

String apiWritekey = "UXSFX7XXLPYM0RQN"; //Api write key from thingspeak.com const char\* ssid = "AndroidAP358E";

const char\* password = "gngj2397";

const char\* server = "api.thingspeak.com"; float resolution=3.3/1023;

WiFiClient client; LiquidCrystal\_I2C lcd(0x27, 16, 2); void setup() { Serial.begin(115200); WiFi.disconnect();

delay(10);

WiFi.begin(ssid, password); Serial.println(); Serial.println(); Serial.print("Connecting to "); Serial.println(ssid); WiFi.begin(ssid, password);

while (WiFi.status() != WL\_CONNECTED) { delay(500);

Serial.print(".");



**HEART RATE MONITRING SYSTEM OVER INTERNET**

}

Serial.println("");

Serial.print("NodeMcu connected to wifi..."); Serial.println(ssid);

Serial.println(); Serial.begin(115200); Wire.begin(D2,D1); lcd.init();

lcd.home();

lcd.backlight(); // Enable or Turn On the backlight

}

void loop() {

int heartValue = (analogRead(A0)/10); if (heartValue>40 && heartValue<90)

{

//int heartValue = (analogRead(A0)/10); if (client.connect(server,80)) {

String tsData = apiWritekey; tsData +="&field2=";

tsData += String(heartValue); tsData += "\r\n\r\n";

client.print("POST /update HTTP/1.1\n"); client.print("Host: api.thingspeak.com\n"); client.print("Connection: close\n");

client.print("X-THINGSPEAKAPIKEY: "+apiWritekey+"\n");

client.print("Content-Type: application/x-www-form-urlencoded\n"); client.print("Content-Length: ");



**HEART RATE MONITRING SYSTEM OVER INTERNET**

client.print(tsData.length()); client.print("\n\n"); client.print(tsData);

Serial.println("uploading to thingspeak"); Serial.println(heartValue); //printing the value tot the serial plotter lcd.setCursor(0,0);

// delay(500); lcd.print("BPM="); lcd.print(heartValue); delay(500);

//lcd.setCursor(0,5); lcd.print(" ");

}

/\*Serial.print("Temperature: "); Serial.print(temp);

Serial.println("uploaded to Thingspeak server ");\*/

}

client.stop();

Serial.println("Waiting to upload next reading..."); Serial.println();

}

# FLOW CHART OF THE PROJECT:



**HEART RATE MONITRING SYSTEM OVER INTERNET**

**Start**

**Pulse Sensor**

Node

MCU

**Stop**

**ThingSpeak Server**

**LCD Screen (Output)**

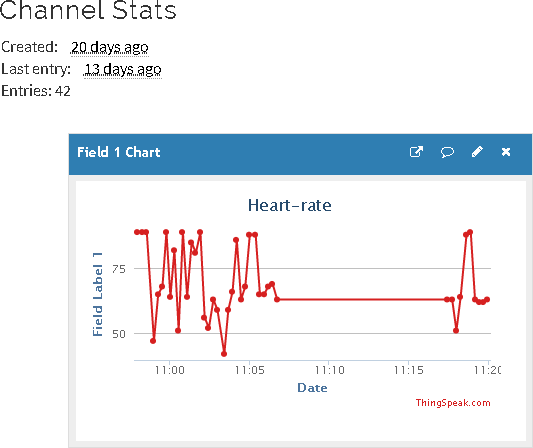
## Fig. 3.4.1 Flow Chart

**CHAPTER-4 RESULT AND ANALYSIS**



**HEART RATE MONITRING SYSTEM OVER INTERNET**

* 1. **RESULTS:**



## APPLICATIONS:

* + - Applicable in Medical science technology.
    - Applicable for Bedridden patients in house.

## CONCLUSION:

* + - Live streaming of patient’s pulse from any location.
    - Surveying the Storage of data collected about patents Heart rate for further analysis.
    - Comparing the stored data.
    - To create a platform to easily connect and access information and data.

## FUTURE SCOPE:

The Project can be extended for addition of

1. Temperature sensor for total health monitoring.
2. Addition of GSM module for emergency alerts.

# REFERENCNCES

1. A. Bourouis, M. Feham, A. Bouchachia, "Ubiquitous Mobile Health Monitoring System for Elderly (UMHMSE)", *International Journal of Computer Science and Information Technology*, vol. 2, no. 3, pp. 74-82, June 2011.
2. J. Holler et al., "From Machine-to-Machine to the Internet of Things" in Introduction to a New Age of Intelligence, Academic Press, 2014.
3. Anirvin Sharma, Tanupriya Choudhury, Praveen Kumar, "Health Monitoring & Management using IoT devices in a Cloud Based Framework", *Advances in Computing and Communication Engineering (ICACCE) 2018 International Conference on*, pp. 219-224, 201
4. *Thingspeak launched by ioBridge*, 2016, [online] Available: <https://en.wikipedia.org/wiki/ThingSpeak/>.
5. ESP8266: Get Started With ESP8266 Programming NodeMCU Using Arduino IDE , a book by Upskill Learning.
6. [www.circuitdigest.com](http://www.circuitdigest.com/)
7. [www.electronicshub.com](http://www.electronicshub.com/)
8. [www.instructables.com/circuits/arduino](http://www.instructables.com/circuits/arduino)