

**ADVANCED PATIENT MONITORING AND DISEASE PREDICTION SYSTEM USING MACHINE LEARNING**

**A PROJECT REPORT**

**Submitted by**

**ASHIK S**

**BALA SUBRAMNAIYAN S**

**DHARANISHAN K**

**GANESHRAJA A**

**in partial fulfilment for the award of the degree**

**of**

**BACHELOR OF ENGINEERING**

**in**

**ELECTRONICS AND COMMUNICATION**

**ENGINEERING**

**SRI ESHWAR COLLEGE OF ENGINEERING**

**(Autonomous) COIMBATORE-641 202**

**ANNA UNIVERSITY: CHENNAI 600 025**

**MAY 2022**

**ANNA UNIVERSITY : CHENNAI 600 025**

**BONAFIDE CERTIFICATE**

Certified that this project report “**ADVANCED PATIENT MONITORING AND DISEASE PREDICTION SYSTEM USING MACHINE LEARNING**” is the bonafide work of

|  |  |
| --- | --- |
| **ASHIK S** | **[722818106012]** |
| **BALA SUBRAMANIYAN S** | **[722818106015]** |
| **DHARANISHAN K** | **[722818106027]** |
| **GANESHRAJA A** | **[722818106032]** |

who carried out the project work under my supervision

SIGNATURE SIGNATURE

…………………………… ……………………………

Dr. N. SHANMUGASUNDARAM, Mr.C.UDHAYAKUMAR , M.E., Ph.D., M.E.,(Ph.D).,

**HEAD OF THE DEPARTMENT,** **SUPERVISOR,** Professor & Head, Assistant Professor,

Department of ECE, Department of ECE,

Sri Eshwar College of Engineering, Sri Eshwar College of Engineering

Kinathukadavu, Kinathukadavu,

Coimbatore - 641 202. Coimbatore - 641 202.

Submitted for the **ANNA UNIVERSITY** practical examination-project work

viva-voce held on ………………………

………………………………. …………………………

**(Internal Examiner) (External Examiner)**

**ACKNOWLEDGEMENT**

The success of a work depends on a team and cooperation. We take this opportunity to express our gratitude and thanks to everyone who helped us in our project.

We would like to thank the management for excellent facilities and the constant support provided by them to complete this project. It is indeed our great honor bounded duty to thank our beloved Chairman **Mr. R. Mohanram**, for his academic interest shown towards the students. We are indebted to our Director **Mr**. **R. Rajaram**, for motivating and providing us with all facilities.

We wish to express our sincere regards and deep sense of gratitude to **Dr.** **SudhaMohanram, M.E., Ph.D.,** Principal, for the excellent facilities and encouragement provided during the course of the study and project.

We are indebted to **Dr. N.Shanmugasundaram, M.E., Ph.D.,** Professor & Head of Electronics and Communication Department, for having permitted us to carry out this project and for giving the complete freedom to utilize the resources of the department and for his motivation and valuable guidance given to us in completing this project.

We thank our project coordinator **Dr.T.Thamaraimanalan, M.E., Ph.D.,** Associate Professor/ECE, and project guide **Mr.C.Udhayakumar M.E., (Ph.D).,** Assistant Professor/ECE for his valuable support and encouragement given to us for this project.

We solemnly express our thanks to all the teaching and non-teaching staff of the Electronics and Communications Engineering Department, family and friends for their valuable support which inspired us to work on this project.

# ABSTRACT

IoT and machine learning (ML) are becoming increasingly efficient in the medical and telemedicine areas all around the world. This article describes a system that employs latest technology to give a more accurate method of forecasting disease.This technology uses sensors to collect data from the body of the patient. The obtained sensor information is collected with NodeMcU before being transferred to the Cloud Platform "ThinkSpeak" through an ESP8266 wifi module. ThinkSpeak is a cloud server that provides real-time data streams in the cloud. For the best results, data currently saved in the cloud is evaluated by one of the machine learning algorithms, the KNN algorithm. Based on the findings of the analysis and compared with the data sets, the disease is predicted and a prescription for the relevant disease is issued.

**Keywords:** IOT, Machine Learning, KNN Algorithm, Patient’s body parameters, Sensors.

**Table of Contents**

|  |  |  |
| --- | --- | --- |
| **CHAPTER NO.** | **TITLE** | **PAGE NO.** |
|  | ABSTRACT |  |
|  | LIST OF TABLES |  |
|  | LIST OF FIGURES |  |
|  | LIST OF ABBREVIATION |  |
| 1 | INTRODUCTION |  |
|  | 1.1 OVERVIEW | 1 |
| 2 | LITERATURE SURVEY |  |
|  | 2.1 LITERATURE REVIEW | 3 |
|  | 2.2 EXISTING SYSTEM | 10 |
|  | 2.3 DRAWBACKS OF EXISTING SYSTEM | 11 |
| 3 | PROPOSED SYSTEM |  |
|  | 3.1 BLOCK DIAGRAM | 12 |
|  | 3.2 WORKING | 12 |
|  | 3.3 OBJECTIVE | 13 |
| 4 | REQUIREMENTS OF THE PROJECT |  |
|  | 4.1 HARDWARE REQUIREMENTS | 14 |
|  | 4.2 SOFTWARE REQUIREMENTS | 14 |
| 5 | HARDWARE DESCRIPTION |  |
|  | 5.1 NODEMCU | 15 |
|  | 5.2 TEMPERATURE SENSOR (LM35) | 21 |
|  | 5.3 MAX30100 | 23 |
|  | 5.4 LIQUID CRYSTAL DISPLAY(LCD) | 24 |
|  | 5.5 POWER SUPPLY (STEP-DOWN TRANSFORMER) | 28 |
| 6 | SOFTWARE DESCRIPTION |  |
|  | 6.1 ARDUINO | 33 |
|  | 6.2 THINGSPEAK | 37 |
|  | 6.3 GOOGLE COLAB | 41 |
| 7 | COMPREHENSIVE RESULTS |  |
|  | 7.1 HARDWARE RESULTS | 54 |
|  | 7.2 SOFTWARE RESULTS | 54 |
| 8 | CONCLUSION AND THE FUTURE SCOPE | 56 |
|  | REFERENCES |  |
|  | APPENDIX |  |
|  | MICROCONTROLLER PROGRAMMING |  |
|  | DATA SET AND TRAINING |  |
|  | DATA CLASSIFICATION AND PREDICTION |  |

**List of Tables**

|  |  |  |
| --- | --- | --- |
| **TABLE NO.** | **TITLE** | **PAGE NO.** |
| 5.1 | PIN DESCRIPTION OF NODEMCU | 15 |
| 5.2 | PIN DESCRIPTION OF LCD | 26 |

**List of Figures**

|  |  |  |
| --- | --- | --- |
| **FIGURE NO** | **TITLE** | **PAGE NO.** |
| 3.1 | BLOCK DIAGRAM OF A SYSTEM | 12 |
| 5.1 | NODEMCU | 17 |
| 5.2 | CHOOSE A REQUIRED ESP8266 MODULE | 20 |
| 5.3 | LM35 | 22 |
| 5.4 | WORKING OF MAX30100 | 23 |
| 5.5 | LIQUID CRYSTAL DISPLAY | 25 |
| 5.6 | TRANSFORMER | 30 |
| 6.1 | CONVNETIONAL SENSOR NETWORKS | 37 |
| 6.2 | ENSOR NETWORK OVER CLOUD | 38 |
| 6.3 | STRUCTURE OF IOT | 39 |
| 6.4 | POSSIBLE OUTCOME OF A PROJECT | 40 |
| 6.5 | CREATING A NEW GOOGLE COLAB NOTEBOOK | 43 |
| 6.6 | CHANGING THE RUNTIME TYPE FOR GOOGLE COLAB | 44 |
| 7.1 | HARDWARE CONNECTION OF THE PROPOSED SYSTEM | 54 |
| 7.2 | MACHINE LEARNING OUTPUT OF THE PROPOSED SYSTEM | 54 |
| 7.3 | TEMPERATURE OF A PATIENT | 55 |
| 7.4 | GRAPHICAL REPRESENTATION OF THE OUTPUT | 55 |

**List of Abbreviation**

|  |  |
| --- | --- |
| IoT | Internet of Things |
| ML | Machine Learning |
| PCA | Patient Centric Agent |
| RPM | Remote Patient Monitoring |
| ECG | Electrocardiogram |
| ICU | Intensive Care Unit |
| SVM | Support Vector Machine |
| CVD | Cardio Vascular Disease |
| BSN Technology | Body Sensor Network Technology |
| KNN Algorithm | K-Nearest Neighbour Algorithm |
| MATLAB | Matrix Laboratory |
| GPU | Graphical Processing Unit |
| WSN | Wireless Sensor Network |
| LCD | Liquid Crystal Display |

# Chapter 1

# INTRODUCTION

## OVERVIEW

**Telemedicine** is one of the most promising areas in which the Internet of Things (IoT) has proven to be cost-effective and productive in healthcare applications, particularly in the field of patient monitoring. In the field of cardiology, telemedicine combined with sensors can be used to a greater extent. This project describes the experience, the technique used, and the different design considerations that need be made in order to make telemedicine in patient monitoring systems effective. The method uses Internet of Things (IoT) technology for uploading to Webserver and telecommunication technologies for providing medical information and services. The sensors are used to capture biological parameters and are linked to mobile phones, allowing various patients to access data in concurrently.

The **discipline of biomedicine** is no longer isolated in today's world of automation. Engineering and technology have shown to be important in the realm of biomedical research. It not only made doc-tors more efficient, but it also helped them improve the entire medicine process. In a number of healthcare situations, machine learning (ML) is already aiding. In health maintenance, machine learning aids in the analysis of numerous data parameters, the recommendation of findings, and the generation of appropriate reports.

The **Internet of Things (IoT)** describes the network of physical object “things” that are embedded with sensors, software, and other technologies for the purpose of connecting and exchanging data with other devices and systems over the internet. These devices range from ordinary household objects to sophisticated industrial tools. All these devices can communicate with each other and take important actions that would provide timely help to save someone’s life. After collecting the data, an IoT healthcare device would send this critical information to the cloud so that doctors can act upon it.

**Machine learning** is a subfield of artificial intelligence, which is broadly defined as the capability of a machine to imitate intelligent human behavior. Artificial intelligence systems are used to perform complex tasks in a way that is similar to how humans solve problems.

**Machine learning** techniques in **healthcare** use the increasing amount of health data provided by the Internet of Things to improve patient outcomes. These techniques provide promising applications as well as significant challenges. The three main areas machine learning is applied to include medical imaging, natural language processing of medical documents, and genetic information. Many of these areas focus on diagnosis, detection, and prediction. A large infrastructure of medical devices currently generates data but a supporting infrastructure is oftentimes not in place to effectively utilize such data

In multispecialty hospitals with a large number of wards and a large number of patients in each ward, doctors are unable to monitor the patient at all times. For this, the doctor creates time slots, and each ward is visited after a certain amount of time has passed. However, patients may experience is-sues in the intervals between these time frames. This causes patient inconvenience, and hospital management may feel helpless in the face of the problem.

# CHAPTER 2

# LITERATURE SURVEY

## 2.1 LITERATURE REVIEW

### 2.1.1 IMPROVING PATIENT SAFETY AND CLINICIAN WORKFLOW IN THE GENERAL CARE SETTING WITH ENHANCED SURVEILLANCE MONITORING

**AUTHORS:** Susan p.mcgrath,Irina M.Perreard, Melissa D.Garland ,Kelli A. Converse, Todd A.Mackenzie

**DESCRIPTION:**

Clinical monitoring systems have been implemented in the inpatient hospital setting for decades, with little attention given to systems analysis or assessment of impact on clinician workflow or patient care. This study provides an example of how system level design and analysis can be applied in this domain, with specific focus on early detection of patient deterioration to mitigate failure to rescue events. Wireless patient sensors and pulse oximetry-based surveillance system monitors with advanced display and information systems capabilities were introduced to 71 general care beds in two units. Nursing workflow was redesigned to integrate use of the new system and its features into patient assessment activities. Patient characteristics, vital sign documentation, monitor alarm, workflow, and system utilization data were collected and analyzed for the period five months before and five months after implementation. Comparison unit data were also collected and analyzed for the same periods. A survey pertaining to staff satisfaction and system performance was administered after implementation. Statistical analysis was performed to examine differences in the before and after data for the target and control units. The enhanced monitoring system received high staff satisfaction ratings and significantly improved key clinical elements related to early recognition of changes in patient state, including reducing average vital signs data collection time by 28%, increasing patient monitoring time (rate ratio 1.22), and availability and accuracy of patient information. Impact on clinical alarms was mixed, with no significant increase in clinical alarms per monitored hour.

### 2.1.2 CONTINUOUS PATIENT MONITORING WITH A PATIENT CENTRIC AGENT: A BLOCK ARCHITECTURE.

**AUTHORS:** Md. Ashraf Uddin, Andrew Stranieri, Iqbal Gondal, Venki Balasubramanian.

**DESCRIPTION :**

The Internet of Things (IoT) has facilitated services without human intervention for a wide range of applications including continuous Remote Patient Monitoring (RPM). However, the complexity of RPM architectures, the size of datasets generated and limited power capacity of devices make RPM challenging. In this paper, we propose a tier based End-to-End architecture for continuous patient monitoring that has a Patient Centric Agent (PCA) as its center piece. The PCA manages a Blockchain component to preserve privacy when data streaming from body area sensors needs to be stored securely. The PCA based architecture includes a lightweight communication protocol to enforce security of data through different segments of a continuous, real time patient monitoring architecture. The architecture includes the insertion of data into a personal Blockchain to facilitate data sharing amongst healthcare professionals and integration into electronic health records while ensuring privacy is maintained. The Blockchain is customized for RPM with modifications that include having the PCA select a Miner to reduce computational effort, enabling the PCA to manage multiple Blockchains for the same patient, and the modification of each block with a prefix tree to minimize energy consumption and incorporate secure transaction payments. Simulation results demonstrate that security and privacy can be enhanced in Remote Patient Monitoring with the PCA based End to End architecture.

### 2.1.3 DEVELOPMENT OF WIRELESS TRANSDUCER FOR REAL-TIME REMOTE PATIENT MONITORING.

**AUTHORS:** Yang Yang; I Zhu;Ke Ma;Roy B. V. B. Simorang-kir;Nemai Chandra Karmakar; Karu P. Esselle

**DESCRIPTION:**

A novel wireless transducer that uses analog-based technology at 2.4 GHz is presented in this letter. The transducer consists of an electrocardiography (ECG) detection circuit and a novel three-stage amplitude modulation transmitter that up-converts the ECG signal to a 2.4-GHz carrier frequency. To minimize the effects due to local oscillator leakage as well as the interference at the image frequency, the intermediate frequency is carefully selected, and a bandpass filter with a very sharp selectivity is designed. As demonstrated by the experimental results, the full-wave ECG signals can be successfully demodulated from the transmitted signal using the presented transducer. This enables the possibility of using analog-based technology for remote patient monitoring in real time.

### 2.1.4 LOW-POWER WEARABLE ECG MONITORING SYSTEM FOR MULTIPLE-PATIENT REMOTE MONITORING.

**AUTHORS:** Elisa Spanò; Stefano Di Pas-coli; Giuseppe Iannaccone

**DESCRIPTION:**

Many devices and solutions for remote ECG monitoring have been proposed in the literature. These solutions typically have a large marginal cost per added sensor and are not seamlessly integrated with other smart home solutions. Here we propose an ECG remote monitoring system that is dedicated to non-technical users in need of long-term health monitoring in residential environments and is integrated in a broader Internet-of-Things (IoT) infrastructure. Our prototype consists of a complete vertical solution with a series of advantages with respect to the state of the art, considering both prototypes with integrated front end and prototypes realized with off-the-shelf components:

1. ECG prototype sensors with record-low energy per effective number of quantized levels
2. an architecture providing low marginal cost per added sensor/user,
3. the possibility of seamless integration with other smart home systems through a single internet-of-things infrastructure.

### 2.1.5 CLINICIAN-DRIVEN DESIGN OF VITALPAD –AN INTELLIGENT MONITORING AND COMMUNICATION DEVICE TO IMPROVE PATIENT SAFETY IN THE INTENSIVE CARE UNIT.

**AUTHOR:** Luisa Flohr;Shaylene Beaudry; K Taneille Johnson; Nicholas West; Catherine M Burns; J Mark Ansermino; Guy A Dumont; David Wensley;Peter Skippen; Mat-thias Görges.

**DESCRIPTION:**

The paediatric intensive care unit (ICU) is a complex environment, in which a multidisciplinary team of clinicians (registered nurses, respiratory therapists, and physicians) continually observe and evaluate patient information. Data are provided by multiple, and often physically separated sources, cognitive workload is high, and team communication can be challenging. Objective: Our aim is to combine information from multiple monitoring and therapeutic devices in a mobile application, the Vital-PAD, to improve the efficiency of clinical decision-making, communication, and thereby patient safety. Methods: We observed individual ICU clinicians, multidisciplinary rounds, and handover procedures for 54 hours to identify data needs, workflow, and existing cognitive aid use and limitations. A prototype was developed using an iterative participatory design approach; usability testing, including general and task-specific feedback, was obtained from 15 clinicians. Results: Features included map overviews of the ICU showing clinician assignment, patient status, and respiratory support; patient vital signs; a photo-documentation option for arterial blood gas results; and team communication and reminder functions. Discussion: Clinicians reported the prototype to be an intuitive display of vital parameters and relevant alerts and reminders, as well as a user-friendly communication tool.

Future work includes implementation of a prototype, which will be evaluated under simulation and real-world conditions, with the aim of providing ICU staff with a monitoring device that will improve their daily work, communication, and decision-making capacity. Mobile monitoring of vital signs and therapy parameters might help improve patient safety in wards with single-patient rooms and likely has applications in many acute and critical care settings.

### 2.1.6 Smart Health Monitoring System using IOT and Machine Learning Techniques.

**AUTHOR:** Honey Pandey, S. Prabha

**DESCRIPTION:**

Coronary illness is that the principle purpose for death around the world. Human services field contains a tremendous measure of information, for handling those information certain methods are utilized. Handling or processing is one in all methods regularly utilized. This strategy predicts the emerging potential outcomes of cardiovascular ailment. The results of this strategy are to fore see the previous cardiovascular malady. The task manages IOT using sensor (pulse sensor to watch pulse) with Arduino and furthermore the outcome can be checked in sequential screen. With the use of IFTTT the readings of sensor are perused in google sheet which is then changed over into csv go looking like data. The datasets utilized are grouped as far as therapeutic parameters which are additionally utilized for preparing and testing the information. This strategy assesses those parameters utilizing information preparing order method. With the work of AI calculations and classification. Initially, the dataset is dissected, watched and screened, at that point the obtained information is handled in python programming utilizing Machine Learning Algorithm to be specific Decision Tree Algorithm and Random backwoods classifier Algorithm. SVM (Support vector machine) shows the higher outcome as far as exactness for identifying heart illness. Henceforth the proposed framework is demonstrated to be solid one for foreseeing prior heart disease. The proposed hardware as well as software system helps patient to predict heart disease in early stages. It will be helpful for mass screening system in villages where hospital facilities are not available, i.e., rural areas.

### 2.1.7 Machine Learning Algorithms for Disease, Prediction Using IoT Environment.

**AUTHOR:** Samir S. Yadav, Shivajirao M. Jadhav.

**DESCRIPTION:**

In the most advanced healthcare application environment, the use of IoT technologies brings convenience to medical professionals and patients, since they have applied to health areas. In IoT, Body sensor network (BSN) technology plays a vital role in the healthcare system where lightweight wireless and low-powered sensor nodes used for monitoring the patients. In this paper, we propose a healthcare system using IoT and BSN technology. This system includes various sensors like pulse rate sensor, temperature sensor, and blood pressure sensor. These sensors sense the parameters and send the data to the controller. According to the conditions, the buzzer will on as temperature exceeds the given range. It carries the sensed data to the LCD to display on it. At the same time, data send to doctors using the internet, so that they can give quick and proper solution in real-time. Many patients suffer because of not getting the timely and appropriate solution and help for their problem. Proposed system hence offers the real-time solution and help in case of emergency. This system is convenient; therefore, a person can carry it with them. Thus continuous health checking is possible. The system also predicts the disease for a particular patient base on current reading using various supervised learning algorithms.

### 2.1.8 Heart Disease Detection by Using Machine Lear

### ning Algorithms and a Real-Time Cardiovascular Health Monitoring System.

**AUTHOR:** Nashif, Mohammad Hasan Imam.

**DESCRIPTION:**

Cardiovascular diseases are the most common cause of death worldwide over the last few decades in the developed as well as underdeveloped and developing countries. Early detection of cardiac diseases and continuous supervision of clinicians can reduce the mortality rate.

However, accurate detection of heart diseases in all cases and consultation of a patient for 24 hours by a doctor is not available since it requires more sapience, time and expertise. In this study, a tentative design of a cloud-based heart disease prediction system had been proposed to detect impending heart disease using Machine learning techniques. For the accurate detection of the heart disease, an efficient machine learning technique should be used which had been derived from a distinctive analysis among several machine learning algorithms in a Java Based Open Access Data Mining Platform, WEKA. The proposed algorithm was validated using two widely used open-access database, where 10-fold cross-validation is applied in order to analyze the performance of heart disease detection. An accuracy level of 97.53% accuracy was found from the SVM algorithm along with sensitivity and specificity of 97.50% and 94.94%respectively. Moreover, to monitor the heart disease patient round-the-clock by his/her caretaker/doctor, a real-time patient monitoring system was developed and presented using Arduino, capable of sensing some real-time parameters such as body temperature, blood pressure, humidity, heartbeat. The developed system can transmit the recorded data to a central server which are updated every 10 seconds. As a result, the doctors can visualize the patient’s real-time sensor data by using the application and start live video streaming if instant medication is required. Another important feature of the proposed system was that as soon as any real-time parameter of the patient exceeds the threshold, the prescribed doctor is notified at once through GSM technology.

### 2.1.9 Data Mining for Wearable Sensors in Health Monitoring Systems: A Review of Recent Trends and Challenges.

**AUTHOR:** Hadi Banaee, mobyen Uddin Ahmed, Amy Loutfi.

**DESCRIPTION:**

The past few years have witnessed an increase in the development of wearable sensors for health monitoring systems. This increase has been due to several factors such as development in sensor technology as well as directed efforts on political and stakeholder levels to promote projects which address the need for providing new methods for care given increasing challenges with an aging population.

An important aspect of study in such system is how the data is treated and processed. This paper provides a recent review of the latest methods and algorithms used to analyse data from wearable sensors used for physiological monitoring of vital signs in healthcare services. In particular, the paper outlines the more common data mining tasks that have been applied such as anomaly detection, prediction and decision making when considering in particular continuous time series measurements. Moreover, the paper further details the suitability of particular data mining and machine learning methods used to process the physiological data and provides an overview of the properties of the data sets used in experimental validation. Finally, based on this literature review, a number of key challenges have been outlined for data mining methods in health monitoring systems.

### 2.1.10 Image-Based Cardiac Diagnosis with Machine Learning.

**AUTHOR:** Carlos Martin-Isla, Victor M. Campello, Cristian Izquierdo, Zahra Raisi-Es-tabragh, Bettina Baeßler, Steffen E. Petersen and Karim Lekadir.

**DESCRIPTION:**

Cardiac imaging plays an important role in the diagnosis of cardiovascular disease (CVD). Until now, its role has been limited to visual and quantitative assessment of cardiac structure and function. However, with the advent of big data and machine learning, new opportunities are emerging to build artificial intelligence tools that will directly assist the clinician in the diagnosis of CVDs. This paper presents a thorough review of recent works in this field and provide the reader with a detailed presentation of the machine learning methods that can be further exploited to enable more automated, precise and early diagnosis of most CVDs.

## 2.2 EXISTING SYSTEM

The main vision of the healthcare industry is to provide better health care to all the people anywhere and at any time in the world. This should be done in a more patient friendly and economic manner. Therefore for increasing the patient care efficiency, there is a need to improve the patient monitoring devices.

Because technology has made life easier so that impact is shown to reduce the tension of patient.The body sensor network (BSN) technology is one of the most imperative technologies used in IoT-based modern healthcare system. It is basically a collection of low-power and lightweight wireless sensor nodes that are used to monitor the human body functions and surrounding environment. Since BSN nodes are used to collect sensitive (life-critical) information and may operate in hostile environments, accordingly, they require strict security mechanisms to prevent malicious interaction with the system.

## 2.3 DRAWBACKS OF EXISTING SYSTEM.

1. Prescription was not sent to the patient by doctor.
2. Wired network-restriction between the body movement.
3. Takes time to analyse the manual reading.
4. Interference of the multi device that share the channel.

# CHAPTER 3

# PROPOSED SYSTEM

## 3.1 BLOCK DIAGRAM

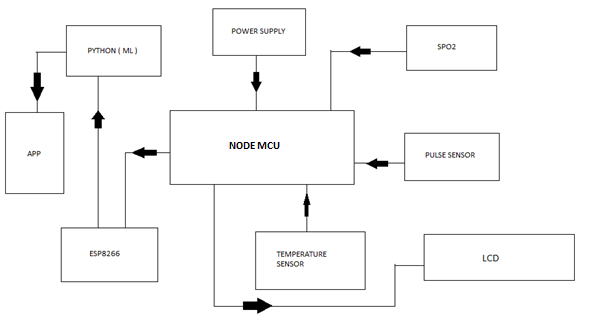


Fig. 3.1 Block Diagram of a System

## 3.2 WORKING

The proposed system is to help the needy and old age people to take their immediate medication properly. Power supply is given to the NodeMcU module, pulse oximetry sensor, temperature sensor, LCD.MAX30100 gives the solution by gathering both pulse rate and SpO2 of a person at the same time. It consumes low energy size and time.it is possible of generating a quick result. LM35 which is used as a temperature sensor, is a good analogue temperature sensor chip with linearity of 0.25`C and the accuracy of 0.5`C out of the box.

LM35 series with the output waveforms that are directly proportional to the temperature in degree Celsius. When the system gets started with the power the sensors started to sense the heart rate, oxygen saturation and temperature of the person’s body and transfer the data to the microcontroller. NodeMcu is then process the data and further transfer it to ThingSpeak cloud platform through its in-built Wi-Fi module.

Thingspeak is a great platform for integrating to a cloud server by offering real-time data and graphical analysis.it creates a channel to store data received from nodemcu and started to visualize the sensed data in a graphical way, which can be stored in the cloud like a database. Now Google CoLab is inter-connected with thingspeak and nodemcu. Google CoLab is a platform which gives unlimited access to GPUs and TPUs to everyone who requires in order to build a Machine Learning and Deep Learning models. The data sets which was already imported. Google CoLab compares the sensor values and data set values based on the backend instructions and display the output. These processes are done through the machine learning by the efficient KNN algorithm. K-nearest neighbour algorithm falls under the supervised learning category and is used for the classification and regression.it is a versatile algorithm also used for inputting missing values and resampling datasets. The instructions are coded using python language. Now the user and caretaker will get notified through the mobile app which is user-friendly, easily understandable.

## 3.3 OBJECTIVE

The main objective of our project is

* To build a smart and efficient patient monitoring system.
* To help the patient by forecasting the diseases based on their medical history.
* To provide a alert messages which notify the caretaker and the patient by providing a better solution.
* To effectively monitor the health of the patient.

# CHAPTER 4

# REQUIREMENTS OF PROJECT

## 4.1 HARDWARE REQUIREMENTS

* The hardware which used for this system are,
* NodeMCU
* Pulse Oximatory Sensor (Max30100)
* Temperature Sensor(LM35)
* Oxygen Saturation (SpO2 Sensor)(Max30100)
* Liquid Crystal Display (LCD)
* Power Supply

## 4.2 SOFTWARE REQUIREMENTS

* The softwares which used for this system are,
* Arduino IDE
* ThingSpeak
* Google CoLab

# CHAPTER 5

# HARDWARE DESCRIPTION

## 5.1 NODEMCU

Table 5.1 PIN Description of NodeMcu

|  |  |  |
| --- | --- | --- |
| **Pin Category** | **Name** | **Description** |
| Power | Micro-USB  3.3V  GND  Vin | **Micro-USB:** NodeMCU can be powered through the USB port  **3.3V:** Regulated 3.3V can be supplied to this pin to power the board  **GND:** Ground pins  **Vin:**External Power Supply |
| Control Pins | **EN, RST** | The pin and the button resets the microcontroller |
| Analog Pin | A0 | Used to measure analog voltage in the range of 0-3.3V |
| GPIO Pins | GPIO1 to GPIO16 | NodeMCU has 16 general purpose input-output pins on its board |
| SPI Pins | SD1, CMD, SD0, CLK | NodeMCU has four pins available for SPI communication. |
| UART Pins | TXD0, RXD0, TXD2, RXD2 | NodeMCU has two UART interfaces, UART0 (RXD0 & TXD0) and UART1 (RXD1 & TXD1). UART1 is used to upload the firmware/program. |
| I2C Pins |  | NodeMCU has I2C functionality support but due to the internal functionality of these pins, you have to find which pin is I2C. |

NodeMCU is an open-source Lua based firmware and **development board** specially targeted for IoT based Applications. It includes firmware that runs on the ESP8266 Wi-Fi SoC from Espressif Systems, and hardware which is based on the ESP-12 module.

### **5.1.1 NodeMCU Development Board Pinout Configuration**

#### **NodeMCU ESP8266 Specifications & Features**

* Microcontroller: Tensilica 32-bit RISC CPU Xtensa LX106
* Operating Voltage: 3.3V
* Input Voltage: 7-12V
* Digital I/O Pins (DIO): 16
* Analog Input Pins (ADC): 1
* UARTs: 1
* SPIs: 1
* I2Cs: 1
* Flash Memory: 4 MB
* SRAM: 64 KB
* Clock Speed: 80 MHz
* USB-TTL based on CP2102 is included onboard, Enabling Plug n Play
* PCB Antenna
* Small Sized module to fit smartly inside your IoT project

### **5.1.2 Brief About NodeMCU ESP8266**

The **NodeMCU ESP8266 development board** comes with the ESP-12E module containing the ESP8266 chip having Tensilica Xtensa 32-bit LX106 RISC microprocessor. This microprocessor supports RTOS and operates at 80MHz to 160 MHz adjustable clock frequency. NodeMCU has 128 KB RAM and 4MB of Flash memory to store data and programs. Its high processing power with in-built Wi-Fi / Bluetooth and Deep Sleep Operating features make it ideal for IoT projects.

NodeMCU can be powered using a Micro USB jack and VIN pin (External Supply Pin). It supports UART, SPI, and I2C interface.



Fig. 5.1 NodeMcu

### **Programming NodeMCU ESP8266 with Arduino IDE**

The NodeMCU Development Board can be easily programmed with Arduino IDE since it is easy to use.

### **Uploading your first program**

Once Arduino IDE is installed on the computer, connect the board with the computer using the USB cable. Now open the Arduino IDE and choose the correct board by selecting **Tools>Boards>NodeMCU1.0** (ESP-12E Module), and choose the correct Port by selecting **Tools>Port**. To get it started with the NodeMCU board and blink the built-in LED, load the example code by selecting **Files>Examples>Basics>Blink**. Once the example code is loaded into your IDE, click on the ‘upload’ button given on the top bar. Once the upload is finished, you should see the built-in LED of the board blinking.

### **Applications**

* Prototyping of IoT devices
* Low power battery operated applications
* Network projects
* Projects requiring multiple I/O interfaces with Wi-Fi and Bluetooth functionalities

### 5.1.3CIRCUIT DESIGN FOR PROGRAMMING ESP8266 USING ARDUINO

You have already seen the required components and the circuit diagram of the project. Now, let us try to understand the design of the circuit.

First and foremost, the ESP8266 Module works on 3.3V Power Supply and anything greater than that, like 5V for example, will kill the SoC. So, the VCC Pin and CH\_PD Pin of ESP8266 ESP-01 Module are connected to a 3.3V Supply.Next important thing to remember is that the ESP8266 WiFi Module has two modes of operation: Programming Mode and Normal Mode.In Programming Mode, you can upload the program or firmware to the ESP8266 Module and in Normal Mode, the uploaded program or firmware will run normally.In order to enable the Programming Mode, the GPIO0 pin must be connected to GND. In the circuit diagram, I’ve connected a SPDT switch to the GPIO0 pin.

Toggling the lever of SPDT will switch the ESP8266 between Programming mode (GPIO0 is connected to GND) and normal mode (GPIO0 acts as a GPIO Pin).Also, the RST (Reset) will play an important role in enabling Programming Mode. The RST pin is an active LOW pin and hence, it is connected to GND through a Push Button. So, whenever the button is pressed, the ESP8266 Module will reset.The RX and TX pins of the ESP8266 Module are connected to RX and TX Pins on the Arduino board. Since the ESP8266 SoC cannot tolerate 5V, the RX Pin of Arduino is connected through a level converter consisting of a 1KΩ and a 2.2KΩ Resistor.

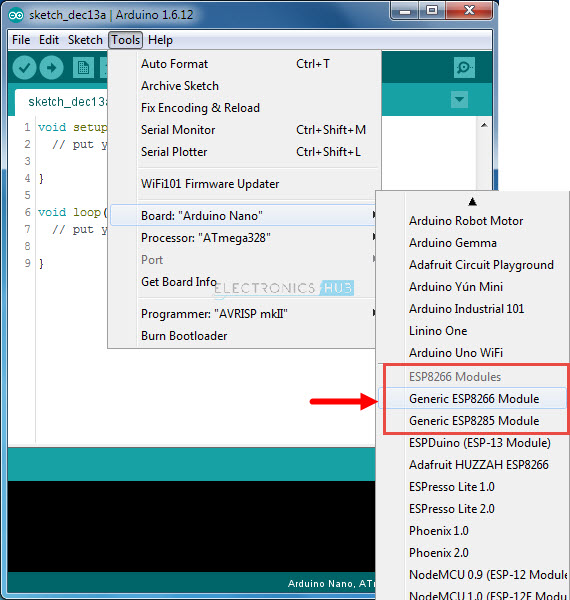
Finally the GPIO2 pin is connected to an LED to test the working of the program. All the necessary connections for enabling the Programming Mode in ESP8266 are mentioned below.

1. VCC – – > 3.3V
2. GND – – > GND
3. CH\_PD – – > 3.3V
4. RST – – > Normally Open; GND to Reset
5. GPIO0 – – > GND
6. TX – – > TX of Arduino
7. RX – – > RX of Arduino (through level converter)

### 5.1.4WORKING OF ESP8266 ARDUINO INTERFACE

Make sure that all the above mentioned connections are properly made. After connecting and configuring the ESP8266 in Programming Mode (GPIO0 is connected to GND), connect the Arduino to the system.

Once the ESP8266 Module is powered ON, Push the RST button and open the Arduino IDE. In the Board options (Tools –> Board), select the “Generic ESP8266” Board. Select the appropriate port number in the IDE.



Fig**.** 5.2 choosing a required ESP8266 Module

Now, open the Blink Sketch and change the LED Pin to 2. Here, 2 means GPIO2 pin of the ESP8266 Module. Before you hit the upload make sure that GPIO0 is connected to GND first and then press the RST button.

Hit the upload button and the code will take a while to compile and upload. You can see the progress at the bottom of the IDE. Once the program is successfully uploaded, you can remove the GPIO0 from GND. The LED connected to GPIO2 will blink.

## 5.2 TEMPERATURE SENSOR (LM35)

The LM35 series are precision integrated-circuit temperature sensors, whose output voltage is linearly proportional to the Celsius (Centigrade) temperature. The LM35 thus has an advantage over linear temperature sensors calibrated in ˚ Kelvin, as the user is not required to subtract a large constant voltage from its output to obtain convenient Centigrade scaling. The LM35 does not require any external calibration or trimming to provide typical accuracies of ±1⁄4˚C at room temperature and ±3⁄4˚C over a full −55 to +150˚C temperature range. Low cost is assured by trimming and calibration at the wafer level. The LM35’s low output impedance, linear output, and precise inherent calibration make interfacing to readout or control circuitry especially easy. It can be used with single power supplies, or with plus and minus supplies. As it draws only 60 µA from its supply, it has very low self-heating, less than 0.1˚C in still air. The LM35 is rated to operate over a −55˚ to +150˚C temperature range, while the LM35C is rated for a −40˚ to +110˚C range (−10˚ with improved accuracy). The LM35 series is available packaged in hermetic TO-46 transistor packages, while the LM35C, LM35CA, and LM35D are also available in the plastic TO-92 transistor package. The LM35D is also available in an 8-lead surface mount small outline package and a plastic TO-220 package.

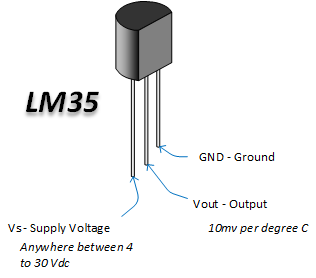


Fig. 5.3 LM35

### 5.2.1 FEATURES

* Calibrated directly in ˚ Celsius (Centigrade)
* Linear + 10.0 mV/˚C scale factor
* 0.5˚C accuracy guarantee able (at +25˚C)
* Rated for full −55˚ to +150˚C range
* Suitable for remote applications
* Low cost due to wafer-level trimming
* Operates from 4 to 30 volts
* Less than 60 µA current drain
* Low self-heating, 0.08˚C in still air
* Nonlinearity only ±1⁄4˚C typical
* Low impedance output, 0.1 Ω for 1 mA load

## 5.3 MAX30100:

The MAX30100 is an integrated pulse oximetry and heartrate monitor sensor solution. It combines two LEDs, a photodetector, optimized optics, and low-noise analog signal processing to detect pulse oximetry and heart-rate signals. The MAX30100 operates from 1.8V and 3.3V power supplies and can be powered down through software with negligible standby current, permitting the power supply to remain connected at all times.

### 5.3.1 Applications

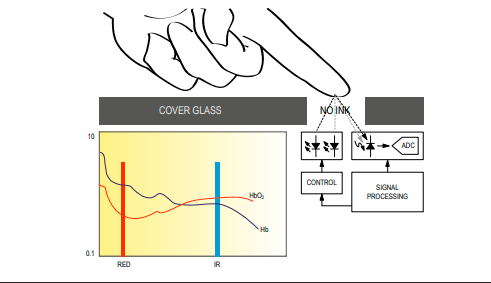
* Wearable Devices
* Fitness Assistant Devices
* Medical Monitoring Devices.

Fig. 5.4 Working of MAX30100

### 5.3.2 BENEFITS and Features

* Complete Pulse Oximeter and Heart-Rate Sensor Solution Simplifies Design
* Integrated LEDs, Photo Sensor, and High-Performance Analog Front -End
* Tiny 5.6mm x 2.8mm x 1.2mm 14-Pin Optically Enhanced System-in-Package
* Ultra-Low-Power Operation Increases Battery Life for Wearable Devices
* Programmable Sample Rate and LED Current for Power Savings
* Ultra-Low Shutdown Current (0.7µA, typ)
* Advanced Functionality Improves Measurement Performance
* High SNR Provides Robust Motion Artifact Resilience
* Integrated Ambient Light Cancellation
* High Sample Rate Capability
* Fast Data Output Capability

## 5.4 LIQUID CRYSTAL DISPLAY(LCD)

A Liquid Crystal Display (LCD) is an electronically-modulated optical device shaped into a thin, flat panel made up of any number of colour or monochrome pixels filled with liquid crystals and arrayed in front of a light source (backlight) or reflector. It is often utilized in battery-powered electronic devices because it uses very small amounts of electric power. LCD has material, which continues the properties of both liquids and crystals. Rather than having a melting point, they have a temperature range within which the molecules are almost as mobile as they would be in a liquid, but are grouped together in an ordered from similar to a crystal. They are used in similar applications where LEDs are used. These applications are display of display of numeric and alphanumeric characters in dot matrix and segmental displays.

LCD consists of two glass panels, with the liquid crystal materials sandwiched in between them. The inner surface of the glass plates is coated with transparent electrodes which define in between the electrodes and the crystal, which makes the liquid crystal molecules to maintain a defined orientation angle. When a potential is applied across the cell, charge carriers flowing through the liquid will disrupt the molecular alignment and produce turbulence.

When the liquid is not activated, it is transparent. When the liquid is activated, the molecular turbulence causes light to be scattered in all directions and the cell appears to be bright. Thus, the required message is displayed. When the LCD is in the off state, the two polarizers and the liquid crystal rotate the light rays, such that they come out of the LCD without any orientation, and hence the LCD appears transparent. The Fig. 5.5 shows the LCD display.

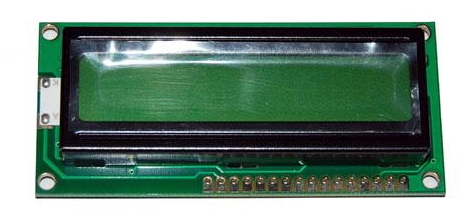


Fig. 5.5 Liquid Crystal Display

### 5.4.1 WORKING OF LCD DISPLAY

When sufficient voltage is applied to the electrodes the liquid crystal molecules would be aligned in a specific direction. The light rays passing through the LCD would be rotated by the polarizer, which would result in activating/highlighting the desired characters. The power supply should be of +5V, with maximum allowable transients of 10mV.

To achieve a better/suitable contrast for the display the voltage (V) at pin 3 should be adjusted properly. A module should not be removed from a live circuit.

The ground terminal of the power supply must be isolated properly so that voltage is induced in it. The module should be isolated properly so that stray voltages are not induced, which could cause a flicking display. LCD is lightweight with only a few, millimetres thickness since the LCD consumes less power, they are compatible with low power electronic circuits, and can be powered for long durations. LCD does not generate light and so light is needed to read the display. By using backlighting, reading is possible in the dark. LCDs have long life and a wide operating temperature range. Before LCD is used for displaying proper initialization should be done. LCD is used to display the blood group and blood glucose level.

### 5.4.2 LCD Pin description

The function of each pins of LCD is described below VCC, VSS and VEE while VDD and VSS provide +5V and ground, respectively, VEE is used for controlling LCD contrast.

The Table 5.2 illustrate the pin descriptions of LCD.

Table 5.2 Pin description of LCD

|  |  |  |
| --- | --- | --- |
| **Pin No.** | **Symbol** | **Function** |
| 1 | Vss | Ground terminal of Module |
| 2 | Vdd | Supply terminal of Module, +5v |
| 3 | Vo | Power supply for liquid crystal drive |
| 4 | RS | Register select  RS=0…Instruction register  RS=1…Data register |
| 5 | R/W | Read/Write  R/W=1…Read  R/W=0…Write |
| 6 | EN | Enable |
| 7-14 | DB0-DB7 | Bi-directional Data Bus. Data Transfer is performed once, through DB0-DB7,incase of interface data length is 8-bits;and twice, thru DB4-DB7 in the case of interface data length is 4-bits.Upper four bits first then lower four bits. |
| 15 | LAMP-(L-) | LED or EL lamp power supply terminals |
| 16 | LAMP+(L+) (E2) | Enable |

#### Register select

There are two important registers inside the LCD. The RS pin is used for selection as follows. If RS=0, the instruction code register is selected, allowing the user to send a command such as clear display, cursor at home, etc. If RS=1 the data register is selected, allowing the user to send data to be displayed on the LCD.

#### Read/Write

R/W input allows the user to write information to the LCD or read information from it. R/W=1 when reading; R/W=0 when writing.

#### Enable

The enable pin is used by the LCD to latch information presented on its data pins. When data is supplied to data pins, a high to low pulse must be applied to this pin in order for the LCD to latch in the data present at the data pins.

#### D0 - *D7*

The 8-bit data pins, D0 – D7, are used to send information to the LCD or read contents of the LCD’S internal registers. There are also instruction codes that can be sent to the LCD to clear the display or force the cursor to the home position or blink the cursor. RS=0 is used to check the busy flag bit to see if the LCD is ready to receive information. The busy flag is D7 and can be read when R/W=1 and RS=0, as follows: if R/W=1, RS=0.when D7=1, the LCD is busy taking care of internal operation and will not accept any new information, when D7=0, the LCD is ready to receive new information.

## 5.5 POWER SUPPLY (STEP-DOWN TRANSFORMER)

This document presents the solution for a 12V 1A flyback converter based on the Infineon OPTIREG™ TLE8386-2EL controller and IPD50N08S4-13 OptiMOS™-T2. The user is guided through the component selections, the circuit design and, finally, an overview of the experimental results are presented. The TLE8386-2EL is part of the Automotive OPTIREG™ family and it implements a low-side-sense current mode controller with built in protection features. The device is AECQ-100 qualified. The IPD50N08S4-13 is an AEC-Q101 qualified 80V N-channel enhanced mode MOSFET, it is part of the OptiMOS™-T2 family. Intended audience This document is intended for power supply design engineers, application engineers, students, etc., who need to design a Flyback converter for automotive power applications where a galvanic isolation between two voltage domains is required. In particular the focus is on a battery connected flyback that delivers up to 12W at 12V output voltage; the intention is to provide the user with all of the needed information to fully design and characterize the SMPS bringing it from an engineering concept to its production. Specific features and applications are: - 48V to 12V Automotive applications - Isolated current mode SMPS - Flyback regulators with auxiliary sensing.

### 5.5.1 **Transformer Specifications**

A **centre-tapped transformer** also known as **two phase three wire transformer** is normally used for rectifier circuits. When a digital project has to work with AC mains a Transformer is used to step-down the voltage (in our case, to 24V or 12V) and then convert it to DC by using a rectifier circuit. In a center-tapped transformer the peak inverse voltage is twice as in bridge rectifier hence this transformer is commonly used in full wave rectifier circuits.

The operation and theory behind a Center tapped transformer is very similar to a normal secondary transformer. A primary voltage will be induced in the primary coil (I1 and I3) and due to magnetic induction the voltage will be transferred to the secondary coil. Here in the secondary coil of a centre tapped transformer, there will be an additional wire (T2) which will be placed exactly at the center of the secondary coil, hence the voltage here will always be zero.

If we combine this zero potential wire (T2) with either T1 or T2, we will get a voltage of 12V AC. If this wire is ignored and voltage across T1 and T2 is considered then we will get a voltage of 24V AC. This feature is very useful for the function of a full wave rectifier.

Let us consider the voltage given by the first half of the secondary coil as Va and the voltage across the second half of the secondary coil as Vb as shown

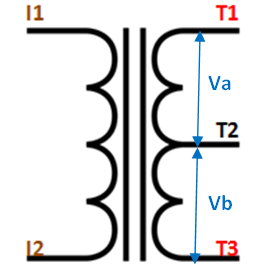


Fig. 5.6 Transformer

### 5.5.2 RECTIFER CIRCUIT:

We have learnt in rectifier circuits about converting a sinusoidal ac voltage into its corresponding pulsating dc. Apart from the dc component, this pulsating dc voltage will have unwanted ac components like the components of its supply frequency along with its harmonics (together called ripples). These ripples will be the highest for a single-phase half wave rectifier and will reduce further for a single-phase full wave rectifier. The ripples will be minimum for 3-phase rectifier circuits.

Such supply is not useful for driving complex electronic circuits. For most supply purposes constant dc voltage is required than the pulsating output of the rectifier. For most applications the supply from a rectifier will make the operation of the circuit poor. If the rectifier output is smoothened and steady and then passed on as the supply voltage, then the overall operation of the circuit becomes better. Thus, the output of the rectifier has to be passed though a filter circuit to filter the ac components. The filter is a device that allows passing the dc component of the load and blocks the ac component of the rectifier output. Thus the output of the filter circuit will be a steady dc voltage. The filter circuit can be constructed by the combination of components like capacitors, resistors, and inductors. Inductor is used for its property that it allows only dc components to pass and blocks ac signals. Capacitor is used so as to block the dc and allows ac to pass. All the combinations and their working are explained in detail below. Series Inductor Filter The circuit diagram of a full wave rectifier with a series inductor filter is given below. As the name of the filter circuit suggests, the Inductor L is connected in series between the rectifier circuit and the load. The inductor carries the property of opposing the change in current that flows through it. In other words, the inductor offers high impedance to the ripples and no impedance to the desired dc components. Thus the ripple components will be eliminated. When the rectifier output current increases above a certain value, energy is stored in it in the form of a magnetic field and this energy is given up when the output current falls below the average value. Thus all the sudden changes in current that occurs in the circuit will be smoothened by placing the inductor in series between the rectifier and the load. The waveform below shows the use of inductor in the circuit.

From the circuit, for zero frequency dc voltage, the choke resistance Ri in series with the load resistance RL forms a voltage divider circuit, and thus the dc voltage across the load is Vdc = RL/(Ri + RL) Vdc is the output from a full wave rectifier. In this case, the value of Ri is negligibly small when compared to RL. The effect of higher harmonic voltages can be easily neglected as better filtering for the higher harmonic components take place. This is because of the fact that with the increase in frequency, the reactance of the inductor also increases. It should be noted that a decrease in the value of load resistance or an increase in the value of load current will decrease the amount of ripples in the circuit. So, the series inductor filter is mostly used in cases of high load current or small load resistance. A simple series inductor filter may not be properly used. It is always better to use a shunt capacitor (C) with series inductor (L) to form an LC Filter. Shunt Capacitor Filter As the name suggests, a capacitor is used as the filter and this high value capacitor is shunted or placed across the load impedance. This capacitor, when placed across a rectifier gets charged and stores the charged energy during the conduction period. When the rectifier is not conducting, this energy charged by the capacitor is delivered back to the load. Through this energy storage and delivery process, the time duration during which the current flows through the load resistor gets increased and the ripples are decreased by a great amount. Thus for the ripple component with a frequency of ‘f’ megahertz, the capacitor ‘C’ will offer a very low impedance. The value of this impedance can be written as: Shunt Capacitor Impedance = 1/2 fC Thus the dc components of the input signal along with the few residual ripple components, is only allowed to go through the load resistance RLoad. The high amount of ripple components of current gets bypassed through the capacitor C. Now let us look at the working of Half-wave rectifier and Full-wave rectifier with Capacitor filters, their output filtered waveform, ripple factor, merits and demerits in detail.

# CHAPTER 6

# SOFTWARE DESCRIPTION

## 6.1 ARDUINO IDE

The Arduino development environment contains a text editor for writing code, a message area, a text console, a toolbar with buttons for common functions, and a series of menus. It connects to the Arduino hardware to upload programs and communicate with them.

### 6.1.1 Writing Sketches

Software written using Arduino are called sketches. These sketches are written in the text editor. Sketches are saved with the file extension .ino. It has features for cutting/pasting and for searching/replacing text. The message area gives feedback while saving and exporting and also displays errors. The console displays text output by the Arduino environment including complete error messages and other information. The bottom righthand corner of the window displays the current board and serial port. The toolbar buttons allow you to verify and upload programs, create, open, and save sketches, and open the serial monitor.

NB: Versions of the IDE prior to 1.0 saved sketches with the extension pde It is possible to open these files with version 1.0, you will be prompted to save the sketch with the .ino extension on save.

The Arduino environment uses the concept of a sketchbook: a standard place to store your programs (or sketches). The sketches in your sketchbook can be opened from the File Sketchbook menu or from the Open button on the toolbar. The first time you run the Arduino software, it will automatically create a directory for your sketchbook. You can view or change the location of the sketchbook location from with the Preferences dialog.

''Beginning with version 1.0, files are saved with a .ino file extension. Previous versions use the .pde extension. You may still open .pde named files in version 1.0 and later, the software will automatically rename the extension to .ino.

Tabs, Multiple Files, and Compilation

Allows you to manage sketches with more than one file (each of which appears in its own tab). These can be normal Arduino code files (no extension), C files (.c extension), C++ files (.cpp), or header files (.h).

### 6.1.2 Uploading

Before uploading your sketch, you need to select the correct items from the Tools Board and Tools Serial Portmenus. The boards are described below. On the Mac, the serial port is probably something like /dev/tty.usbmodem241(for an Uno or Mega2560 or Leonardo) or /dev/tty.usbserial-1B1 (for a Duemilanove or earlier USB board), or/dev/tty.USA19QW1b1P1.1 (for a serial board connected with a Keyspan USB-to-Serial adapter). On Windows, it's probably COM1 or COM2 (for a serial board) or COM4, COM5, COM7, or higher (for a USB board) - to find out, you look for USB serial device in the ports section of the Windows Device Manager. On Linux, it should be /dev/ttyUSB0,/dev/ttyUSB1 or similar.

Once you've selected the correct serial port and board, press the upload button in the toolbar or select the Upload item from the File menu. Current Arduino boards will reset automatically and begin the upload. With older boards (pre-Diecimila) that lack auto-reset, you'll need to press the reset button on the board just before starting the upload. On most boards, you'll see the RX and TX LEDs blink as the sketch is uploaded. The Arduino environment will display a message when the upload is complete, or show an error.

When you upload a sketch, you're using the Arduino bootloader, a small program that has been loaded on to the microcontroller on your board. It allows you to upload code without using any additional hardware. The bootloader is active for a few seconds when the board resets; then it starts whichever sketch was most recently uploaded to the microcontroller. The bootloader will blink the on-board (pin 13) LED when it starts (i.e. when the board resets).

### 6.1.3 LIbraries

Libraries provide extra functionality for use in sketches, e.g. working with hardware or manipulating data. To use a library in a sketch, select it from the Sketch Import Library menu. This will insert one or more  statements at the top of the sketch and compile the library with your sketch. Because libraries are uploaded to the board with your sketch, they increase the amount of space it takes up. If a sketch no longer needs a library, simply delete it from the top of your code.

There is a list of libraries in the reference. Some libraries are included with the Arduino software. Others can be downloaded from a variety of sources. Starting with version 1.0.5 of the IDE, you do can import a library from a zip file and use it in an open sketch. See these instructions for installing a third-party library.

### 6.1.4 PROGRAMMING

The Arduino Uno can be programmed with the Arduino software (download). Select "Arduino Uno from the Tools Board menu (according to the microcontroller on your board). For details, see the reference and tutorials.

The ATmega328 on the Arduino Uno comes preburned with a bootloader that allows you to upload new code to it without the use of an external hardware programmer. It communicates using the original STK500 protocol (reference, C header files).

You can also bypass the bootloader and program the microcontroller through the ICSP (In-Circuit Serial Programming) header; see these instructions for details.The ATmega16U2 (or 8U2 in the rev1 and rev2 boards) firmware source code is available . The ATmega16U2/8U2 is loaded with a DFU bootloader, which can be activated by:

• On Rev1 boards: connecting the solder jumper on the back of the board (near the map of Italy) and then resetting the 8U2.

• On Rev2 or later boards: there is a resistor that pulling the 8U2/16U2 HWB line to ground, making it easier to put into DFU mode.

You can then use Atmel's FLIP software (Windows) or the DFU programmer (Mac OS X and Linux) to load a new firmware. Or you can use the ISP header with an external programmer (overwriting the DFU bootloader). See this user-contributed tutorial for more information.

**Automatic (Software) Reset**

Rather than requiring a physical press of the reset button before an upload, the Arduino Uno is designed in a way that allows it to be reset by software running on a connected computer. One of the hardware flow control lines (DTR) of theATmega8U2/16U2 is connected to the reset line of the ATmega328 via a 100 nanofarad capacitor. When this line is asserted (taken low), the reset line drops long enough to reset the chip. The Arduino software uses this capability to allow you to upload code by simply pressing the upload button in the Arduino environment. This means that the bootloader can have a shorter timeout, as the lowering of DTR can be well-coordinated with the start of the upload.

This setup has other implications. When the Uno is connected to either a computer running Mac OS X or Linux, it resets each time a connection is made to it from software (via USB). For the following half-second or so, the bootloader is running on the Uno. While it is programmed to ignore malformed data (i.e. anything besides an upload of new code), it will intercept the first few bytes of data sent to the board after a connection is opened. If a sketch running on the board receives one-time configuration or other data when it first starts, make sure that the software with which it communicates waits a second after opening the connection and before sending this data.

The Uno contains a trace that can be cut to disable the auto-reset. The pads on either side of the trace can be soldered together to re-enable it. It's labeled "RESET-EN". You may also be able to disable the auto-reset by connecting a 110 ohm resistor from 5V to the reset line; see this forum thread for details.

## 6.2 ThingSpeak

### 6.2.1 Building of sensor network

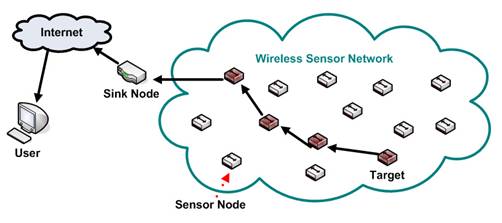


Fig. 6.1 Conventional Sensor Network

A conventional sensor network is a radio network of sensor nodes with ability to sense physical parameters, store sensed data, carry out simple processing on data and forward the data through radio interface. The objective of such network is to push the data to a sink node which can then forward the data to server ( or cloud) is shown in the figure 8.

### 6.2.2 Connection of sensor network to the cloud.

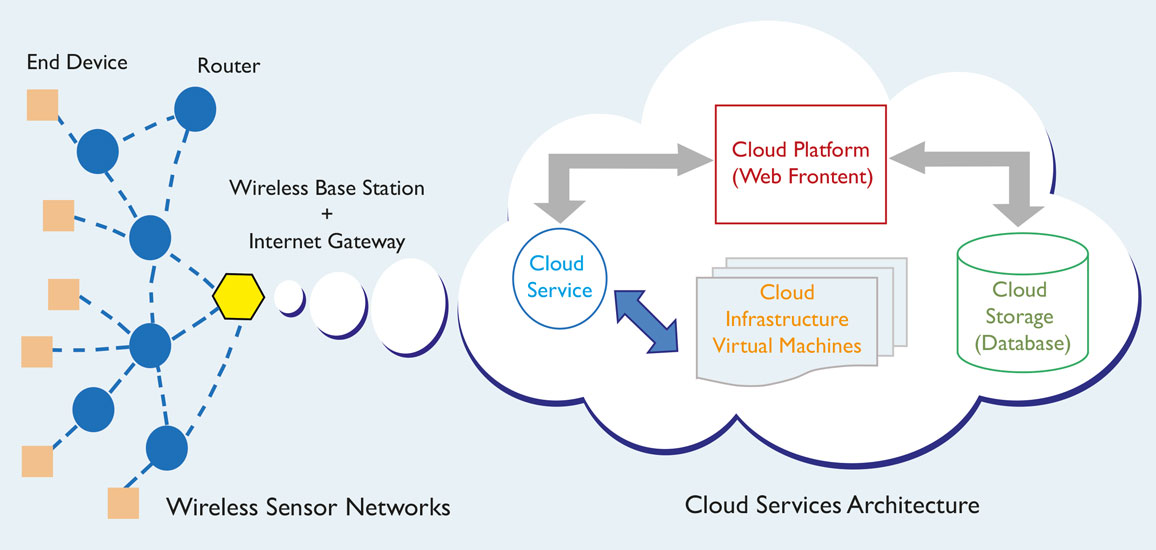


Fig. 6.2 Sensor Network over cloud

Figure 6.2 shows the connection between sensor network and the cloud. However many real time applications includes sensors spread over long areas. As such they are treated as independent networks. Internet of Things is a new paradigm of connecting devices like micocontrollers and smart objects to cloud. Using IoT services, we can now connect sensors to internet directly. One of common design of sensor network includes cluster based methods where clusters are at formed by group of nodes. These are also called coordinator nodes. These nodes gather data from all neihboring nodes. If these nodes can be linked to internet with their unique Ip addresses, then the sensor network can be infinitely scaled( theoretically).

### 6.2.3 Interfacing with the embedded system.

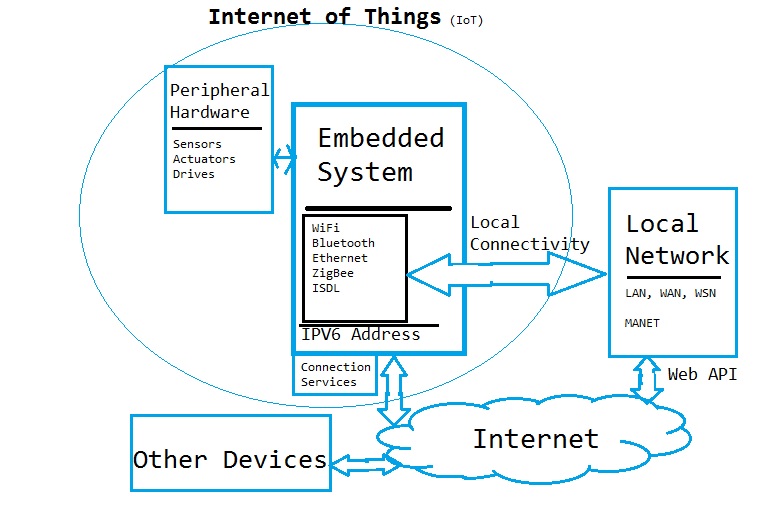


Fig. 6.3 Structure Of IOT

The basic structure of IoT is presented above. Our work would include following innovative research extensions to the exising WSN and IoT framework.

Firstly we would focus purely on sensors and that too on coordinator nodes in Peripheral hardware. Rather than working on integrating individual hardware over cloud, our method would assume entire standalone sensor network as a single peripheral and would connect that to cloud though IoT.

We would focus mainly on ZigBee as wireless technology as that is most accepted WSN standard.Our methods would provide not only communication services but also data gathering and analysis services. We would integrate both simulation as well as real time test beds to prove the designed concepts.

### 6.2.4 Possible outcome of the IoT:

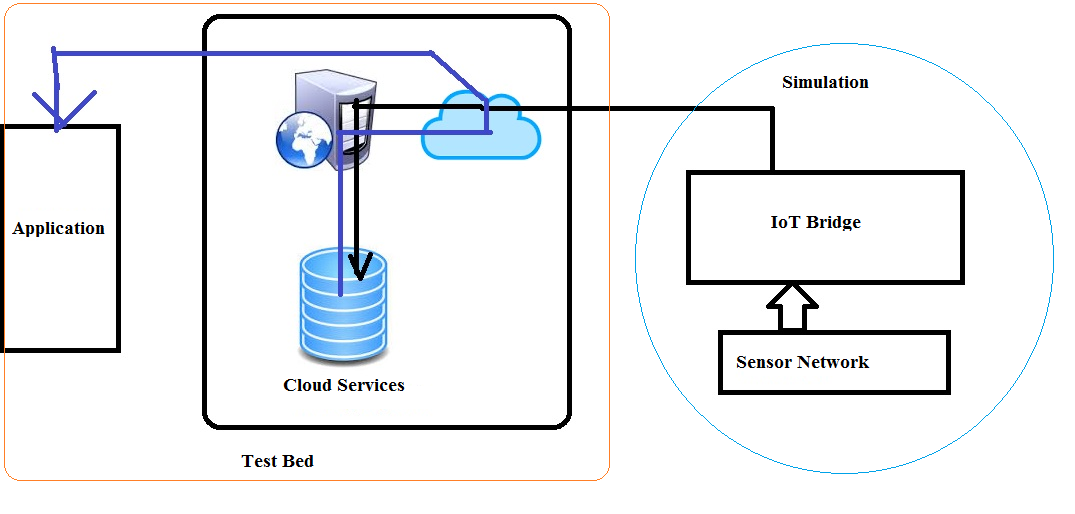


Fig. 6.4 possible outcome of project

In the project work we mainly focus on building more comprehensive state of art cloud extension of WSN through IoT. Firstly the research would focus towards bettering each of the current state of art building blocks including but not limited to sensor network, coordinator protocol, data analysis in sensor network, cloud services, IoT protocols and so on. One of the first expected result would be a unique framework to connect existing WSN to cloud. Then system should prove the advantage of such extension by demonstrating the scale of improvement in data analysis services .Results should prove that IoT can be used to create mesh sensor networks and enhanced bandwidth can be used to connect sensor networks with other control system.

All the information given by the sensor network should be passed to the cloud through the javascriptnode.js . Here the cloud is a private application which can be accessed by the public and this application can be easily installed by any user which will helps the user to collect the data from the sensor network and also helps to analyse and visualize the data from the sensor network .

This can be done by the matlab visualization which build in option provided in this apps. According to the data given by the apps the user can act based on the requirements .

### 6.2.5 Sending data to THING SPEAK APPS:

The Internet of Things provides access to a broad range of embedded devices and web services. Thing Speak is an open data platform and API for the Internet of Things that enables us to collect, store, analyse, visualize, and act on data from sensors or actuators, such as Arduino, and other hardware. For example, with Thing Speak one can create sensor-logging applications, location-tracking applications, and a social network of things with status updates, so that you could have your home thermostat control itself based on our current location.

The primary element of Thing Speak activity is the channel, which contains data fields, location fields, and a status field. After you create a Thing Speak channel, you can write data to the channel, process and view the data with MATLAB® code, and react to the data with tweets and other alerts.

The typical Thing Speak workflow lets you:

1. Create a Channel and collect data
2. Analyse and visualize the data
3. Act on the data using any of several Apps

## 6.3 google colab

Colaboratory, or “Colab” for short, is a product from Google Research. Colab allows anybody to write and execute arbitrary python code through the browser, and is especially well suited to machine learning, data analysis and education. More technically, Colab is a hosted Jupyter notebook service that requires no setup to use, while providing access free of charge to computing resources including GPUs.

Colab notebooks are stored in Google Drive, or can be loaded from GitHub. Colab notebooks can be shared just as you would with Google Docs or Sheets. Simply click the Share button at the top right of any Colab notebook, or follow these Google Drive file sharing instructions.

If you choose to share a notebook, the full contents of your notebook (text, code, output, and comments) will be shared. You can omit code cell output from being saved or shared by using **Edit > Notebook settings > Omit code cell output when saving this notebook**. The virtual machine you’re using, including any custom files and libraries that you’ve setup, will not be shared. So it’s a good idea to include cells which install and load any custom libraries or files that your notebook needs.

### 6.3.1 GOOGLE COLAB QUICK GUIDE

To create your Google Colab file and get started with Google Colab, We can go to [Google Drive](https://drive.google.com/) and create a Google Drive account if we do not have one. Now, click on the “New” button at the top left corner of your Google Drive page, then click on More -> Google Colaboratory.

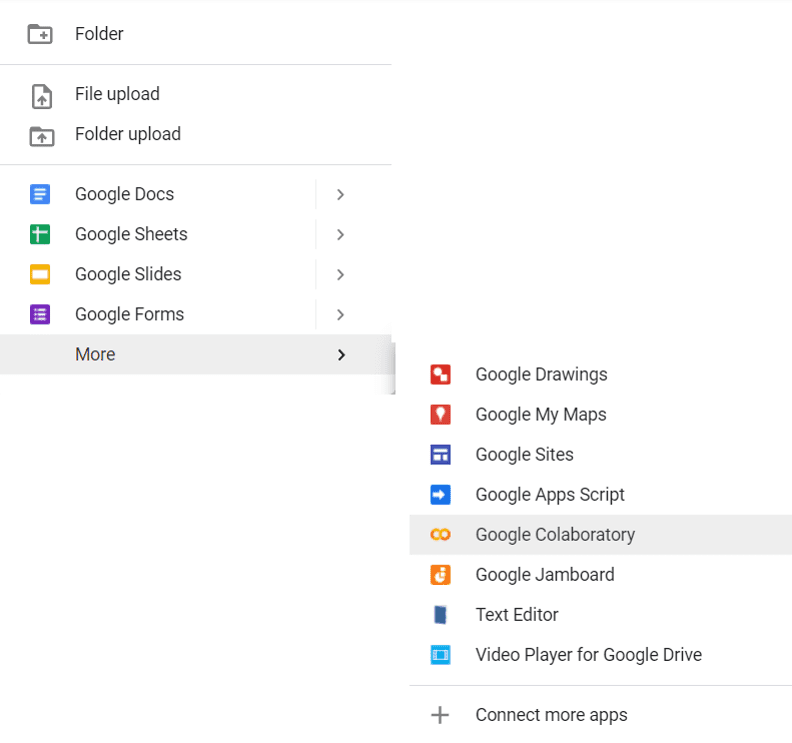


Fig. 6.5 Creating a New Google CoLab Notebook

From here, we can share our Google Colab file with others using the Share button on the top right-hand corner or start coding!

The hotkeys on Colab and that on Jupyter notebooks are similar. These are some of the useful ones:

* Run cell: Ctrl + Enter
* Run cell and add new cell below: Alt + Enter
* Run cell and goto cell below: Shift + Enter
* Indent line by two spaces: Ctrl + ]
* Unindent line by two spaces: Ctrl + [

But there’s also one extra that’s pretty useful that lets we only run a particular selected part of the code in a cell:

* Run selected part of a cell: Ctrl + Shift + Enter

Just like the Jupyter notebook, we can also write text with Markdown cells. But Colab has an additional feature that automatically generates a table of contents based on our markdown content, and we can also hide parts of the code based on their headings in the markdown cells.

If you run Jupyter on your own computer, you have no choice but to use the CPU from your computer. But in Colab, you can change the **runtime** to include GPUs and TPUs in addition to CPUs because it is executed on Google’s cloud. You can switch to a different runtime by going to Runtime ▷ Change runtime type:

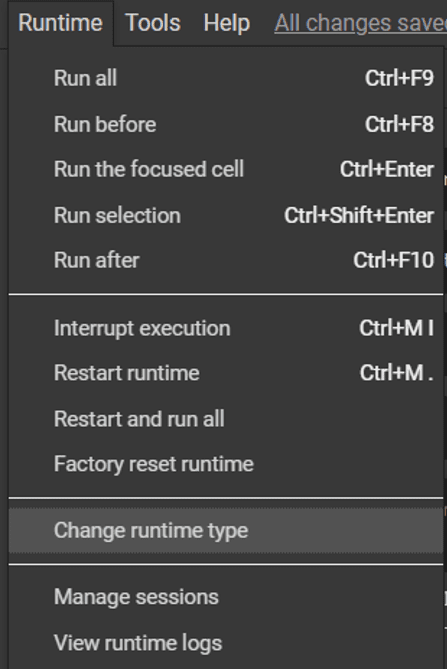
[](https://machinelearningmastery.com/wp-content/uploads/2022/04/Colab-Runtime-Menu.png)

Fig. 6.6 Changing the Runtime Type for Google Colab

Unlike your own computer, Google Colab does not provide you with a terminal to enter commands to manage your Python environment. To install Python libraries and other programs, we can use the ! character to run shell commands just like in Jupyter notebooks, e.g. !pip install numpy (but as we’ll see later on, Colab already comes pre-installed with a lot of the libraries we’ll need, such as NumPy)

### 6.3.2 Python

Python is an interpreter, high-level, general-purpose programming language. Created by Guido van Rossum and first released in 1991, Python's design philosophy emphasizes code readability with its notable use of significant whitespace.

Python is an easy to learn, powerful programming language. It has efficient high-level data structures and a simple but effective approach to object- oriented programming. Python’s elegant syntax and dynamic typing, together with its interpreted nature, make it an ideal language for scripting and rapid application development in manya reason most platforms and

May be freely distributed. The same site also contains distributions of and pointers to many free third party Python modules, programs and tools, and additional documentation. The Python interpreter is easily extended with new functions and data types implemented in C or C++ (or other languages callable from C). Python is also suitable as an extension language for customizable applications. This tutorial introduces the reader informally to the basic concepts and features of the Python language and system. It helps to have a Python interpreter handy for hands-on experience, but all examples are self-contained, so the tutorial can be read off- line as well. For a description of standard objects and modules, see library-index. Reference-index gives a more formal definition of the language. To write extensions in C or C++, read extending-index and c-api-index. There are also several books covering Python in depth. This tutorial does not attempt to be comprehensive and cover every single feature, or even every commonly used feature. Instead, it introduces many of Python’s most notes worthy features, and will give you a good idea of the language’s flavor and style. After reading it, you will be able to read and write Python modules and programs, and you will be ready to learn more about the various Python library modules described in library-index. If you do much work on computers, eventually you find that there’s some task you’d like to automate. For example, you may wish to perform a search-and-replace over a large number of text files, or rename and rearrange a bunch of photo files in a complicated way. Perhaps you’d like to write a small custom database, or a specialized

GUI application or a simple game. If you’re a professional software developer, you may have to work with several C/C++/Java libraries but find the usual write/compile/test/re-compile cycle is too slow. Perhaps you’re writing a test suite for such a library and find writing the testing code a tedious task. Or maybe you’ve written a program that could use an extension language, and you don’t want to design and implement a whole new language for your application.

Typing an end-of-file character (Control-D on Unix, Control-Z on Windows) at the primary prompt causes the interpreter to exit with a zero exit status. If that doesn’t work, you can exit the interpreter by typing the following command: quit(). The interpreter’s line-editing features include interactive editing, history substitution and code completion on systems that support read line. Perhaps the quickest check to see whether command line editing is supported is typing Control-P to the first Python prompt you get. If it beeps, you have command line editing; see Appendix Interactive Input Editing and History Substitution for an introduction to the keys. Ifnothing appears to happen, or if ^P is echoed, command line editing isn’t available; you’ll only be able to use backspace to remove characters from the current line. The interpreter operates somewhat like the Unix shell: when called with standard input connected to a tty device, it reads and executes commands interactively; when called with a file name argument or with a file as standard input, it reads and executes a script from that file.

A second way of starting the interpreter is python -c command [arg] ..., which executes the statement(s) in command, analogous to the shell’s -c option. Since Python statements often contain spaces or other characters that are special to the shell, it is usually advised to quote commands in its entirety with single quotes. Some Python modules areal so useful as scripts. These can be invoked using python-m module [arg]...,which executes the source file for the module as if you had spelled out its full name on the command line. When a script file is used, it is sometimes useful to be able to run the script and enter interactive mode afterwards. This can be done by passing -i before the script.

There are tools which use doc strings to automatically produce online or printed documentation or to let the user interactively browse through code; it’s good practice to include doc strings in code that you write, so make a habit of it. The execution of a function introduces a new symbol table used for the local variables of the function. More precisely, all variable assignments in a functions to read the value in the local symbol table; whereas variable references first look in the local symbol table, then in the local symbol tables of enclosing functions, then in the global symbol table, and finally in the table of built-in names. Thus, global variables cannot be directly assigned a value within a function (unless named in a global statement), although they may be referenced. The actual parameters (arguments) to a function call are introduced in the local symbol table of the called function when it is called; thus, arguments are passed using call by value (where the value is always an object reference, not the value of the object).1 When a function calls another function, a new local symbol table is created for that call. A function definition introduces the function name in the current symbol table.

The value of the function name has a type that is recognized by the interpreter as a user-defined function. This value can be assigned to another name which can then also be used as a function.

Annotations are stored in the annotations attribute of the function as a dictionary and haven o effect on any other part of the function. Parameter annotations are defined by a colon after the parameter name, followed by an expression evaluating to the value of the annotation. Return annotations are defined by a literal ->, followed by an expression, between the parameter list and the colon denoting the end of the def statement.

The comparison operators in and not in check whether a value occurs (does not occur) in a sequence. The operator is and does not compare whether two objects are really the same object; this only matters for mutable objects like lists. All comparison operators have the same priority, which is lower than that of all numerical operators. Comparisons can be chained. For example, a<b==c tests whether a is less than b and more over b equals c. Comparisons may be combined using the Boolean operators and the outcome of a comparison (or of any other Boolean expression) may be negated with not. These have lower priorities than comparison operators; between them, not has the highest priority and or the lowest, so that A and not B or C is equivalent to (A and (not B)) or C. As always, parentheses can be used to express the desired composition. The Boolean operators and are so-called short-circuit operators: their arguments are evaluated from left to right, and evaluation stops as soon as the outcome is determined. For example, if A and C are true but Bis false, A and B and C does not evaluate the expression C. When used as a general value and not as a Boolean, the return value of a short-circuit operator is the last evaluated argument.

Classes provide a means of bundling data and functionality together. Creating a new class creates a new type of object, allowing new instances of that type to be made. Each class instance can have attributes attached to it for maintaining its state. Class instances can also have methods (defined by its class) for modifying its state. Compared with other programming languages, Python’s class mechanism adds classes with a minimum of new syntax and semantics. It is a mixture of the class mechanisms found in C++ and Modula-3. Python classes provide all the standard features of Object Oriented Programming: the class inheritance mechanism allows multiple base classes, a derived class can override any methods of its base class or classes, and a method can call the method of a base class with the same name.

Objects can contain arbitrary amounts and kinds of data. As is true for modules, classes partake of the dynamic nature of Python: they are created at runtime, and can be modified further after creation. In C++ terminology, normally class members (including the data members) are public (except see below Private Variables), and all member functions are virtual. A sin Modula-3, there are no short hands for referencing the object’s members from its methods: the method function is declared with an explicit first argument representing the object, which is provided implicitly by the call. A sin small talk, classes themselves are objects. This provides Semantics for importing and renaming. Unlike C++ and Modula-3, built-in types can be used as base classes for extension by the user. Also, like in C++, most built-in operators with special syntax (arithmetic operators, sub scripting etc.) can be redefined for class instances.(Lacking universally accepted terminology to talk about classes, I will make occasional use of Smalltalk and C++ terms. I would use Modula-3 terms, since its object- oriented semantics are closer to those of Python than C++, but I expect that few readers have heard of it.)

Objects have individuality, and multiple names (in multiple scopes) can be bound to the same object. This is known as aliasing in other languages. This is usually not appreciated on a first glance at Python, and can be safely ignored when dealing with immutable basic types (numbers, strings, tuples).However, aliasing has a possibly surprising effect on these mantic of Python code involving mutable objects such as lists, dictionaries, and most other types. This is usually used to the benefit of the program, since aliases behave like pointers in some respects. For example, passing an object is cheap since only a pointer is passed by the implementation; and if a function modifies an object passed as an argument, the caller will see the change — this eliminates the need for two different argument passing mechanisms as in Pascal.

A namespace is a mapping from names to objects. Most name spaces are currently implemented as Python dictionaries, but that’s normally not noticeable in any way (except for performance), and it may change in the future. Examples of name spaces are: these to f built-in names (containing functions such as abs(), and built-in exception names); the global names in a module; and the local names in a function invocation. In a sense the set of attributes of an object also form a namespace. The important thing to know about namespaces is that there is absolutely no relation between names in different namespaces; for instance, two different modules may both define a function maximize without confusion — users of the modules must prefix it with the module name. By the way, I use the word attribute for any name following a dot — for example, in the expression z. real, real is an attribute of the object z. Strictly speaking, references to names in modules are attribute references: in the expression modname.funcname, modname is a module object and funcname is an attribute of it.

In this case there happens to be a straight forward mapping between the module’s attributes and the global names defined in the module: they share the same namespace!1 Attributes may be read-only or writable. In the latter case, assignment to attributes is possible. Module attributes are writable: you can write modname. The answer = 42. Writable attributes may also be deleted with the del statement. For example, del mod name. the answer will remove the attribute the answer from the object named by mod name. Namespaces are created at different moments and have different lifetimes. The namespace containing the built-in names is created when the Python interpreter starts up, and is never deleted. The global namespace for a module is created when the module definition is read in; normally, module namespaces also last until the interpreter quits. The statements executed by the top-level invocation of the interpreter, either read from a script file or interactively, are considered part of a module called main, so they have their own global namespace. (The built-in names actually also live in a module; this is called built ins.) The local namespace for a function is created when the function is called, and deleted when the function returns or raises an exception that is not handled within the function. (Actually, forgetting would be a better way to describe what actually happens.) Of course, recursive invocations each have their own local namespace.

To speed uploading modules, Python caches the compiled version of each module in the pycache directory under the name module.version.pyc, where the version encodes the format of the compiled file; it generally contains the Python version number. For example, in CPython release 3.3 the compiled version of spam.py would be cached as pycache/spam.cpython-33.pyc. This naming convention allows compiled modules from different releases and different versions of Python to coexist.

Python checks the modification date of the source against the compiled version to see if it’s out of date and needs to be recompiled. This is a completely automatic process. Also, the compiled modules are platform-independent, so the same library can be shared among systems with different architectures. Python does not check the cache in two circumstances. First, it always recompiles and does not store the result for the module that’s loaded directly from the command line. Second, it does not check the cache if there is no source module. To support anon-source (compiled only) distribution, the compiled module must be in the source directory, and there must not be a source module. Some tips for experts:

You can use the -O or -OO switches on the Python command to reduce the size of a compiled module. The -O switch removes assert statements, the -OO switch removes both assert statements and doc strings. Since some programs may rely on having these available, you should only use this option if you know what you’re doing. “Optimized” modules have an opt- tag and are usually smaller. Future releases may change the effects of optimization.

A program doesn’t run any faster when it is read from a .pyc file than when it is read from a .py file; the only thing that’s faster about .pyc files is the speed with which they are loaded.The module compile all can create .pyc files for all modules in a directory.There is more detail on this process, including a flow chart of the decisions.

### 6.3.3 NumPy library

NumPy is a Python library for creating and manipulating vectors and matrices. This Colab is not an exhaustive tutorial on NumPy. Rather, this Colab teaches you just enough to use NumPy in the Colab exercises of Machine Learning Crash Course. Numpy is the core library for scientific computing in Python.

It provides a high-performance multidimensional array object, and tools for working with these arrays. If you are already familiar with MATLAB, you might find this tutorial useful to get started with Numpy.

#### Basic operation of NumPy

**>>> import** **numpy** **as** **np**

**>>>** x = np.array([1, 2, 3])

**>>>** x

array([1, 2, 3])

**>>>** y = np.arange(10)

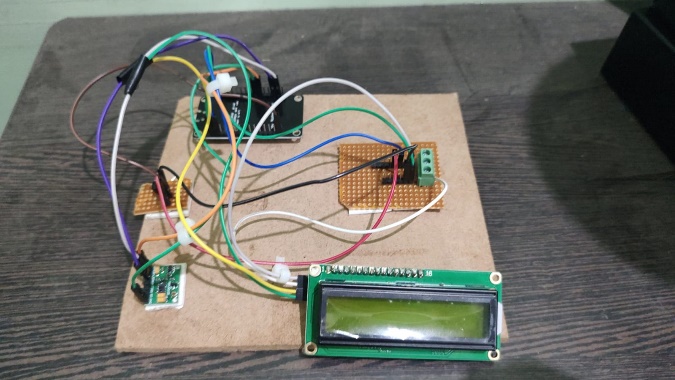
**>>>** y

array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])

# chapter 7

# comprehensive results

## 7.1 hardware results



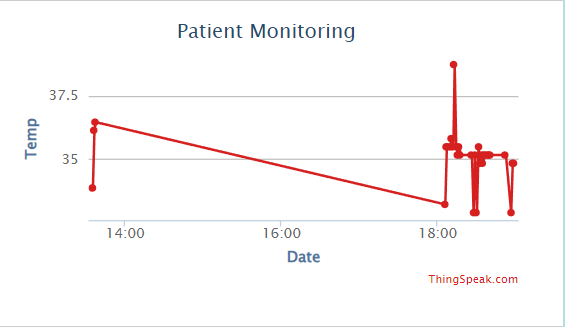
*Fig. 7.1 Hardware connection of the proposed System*

The Fig 7.1 Shows the hardware of the proposed system. In this project the max30100 and temperature sensor senses the patient’s temperature, oxygen saturation and pulse rate. these three parameters are sent to the NodeMcu and displayed temporarily in LCD panel.

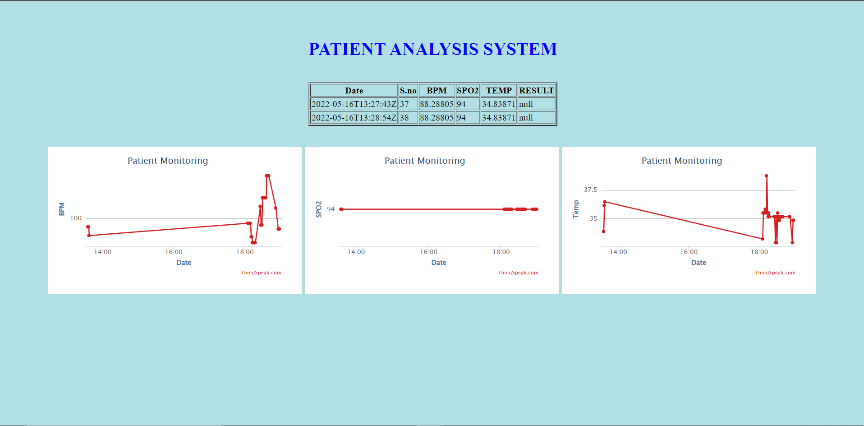
## 7.2 software results



*Fig. 7.2 Machine learning accuracy level test Output*



*Fig. 7.3 Temperature of a patient*



*Fig. 7.4 Graphical representation of the output*

These figure depicts the output of the proposed system by general method and graphical method. The patients are possibly gets alerted through the data displayed in the system and can be treated well.

# chapter 8

# CONCLUSION AND FUTURE SCOPE

Thus, All the parameters of the patient is sent and Machine learning is processed and the status and prescription is viewed in the Mobile Application.ML based Patient Monitoring systems are especially useful because they let the patients live their life while at the same time afford constant medical attention. The need for visiting the clinic/doctor is pushed to only deserving cases

Although many patient monitoring system is familiar with in-hospital medical care, patients are relatively less experienced and confident with using this system. Hence, promoting patients' active and voluntary participation is very important. Besides interaction between medical providers and patients, communication among chronic disease patients is also important.

# references

[1] H. Pandey and S. Prabha, "Smart Health Monitoring System using IOT and Machine Learning Techniques," 2020 Sixth Interna-tional Conference on Bio Signals, Images, and Instrumentation (ICBSII), 2020, pp. 1-4, doi: 10.1109/ICBSII49132.2020.9167660

[2] Machine Learning Algorithms for Disease, Prediction Using Iot Environment. Samir S. Yadav, Shivajirao M. Jadhav, Ma-chine Learning Algorithms for Disease, Predic-tion Using Iot Environment, International Jour-nal of Engineering and Advanced Technology (IJEAT)ISSN: 2249 – 8958, Volume-8 Issue-6, August 2019.

[3] Heart Disease Detection by Using Ma-chine Learning Algorithms and a Real-Time Cardiovascular Health Monitoring System ShadmanNashif, Mohammad Hasan Imam Department of Computer Science & Engineering, Rajshahi University of Engineering & Technology (RUET), Rajshahi, Bangla-desh.Department of Electrical & Electronic Engineering, American International Univer-sity-Bangladesh (AIUB), Dhaka, Bangladesh. DOI: 10.4236/wjet.2018.64057, World Journal of Engineering and Technology

[4] Data Mining for Wearable Sensors in Health Monitoring Systems: A Review of Re-cent Trends and Challenges.Sensors 2013,13, 17472-17500; doi:10.3390/s131217472

[5] Image-Based Cardiac Diagnosis with Machine Learning. Carlos Martin-Isla, Victor M. Campello, Cristian Izquierdo, Zahra Raisi-Es-tabragh, Bettina Baeßler, Steffen E. Petersen, and Karim Lekadir.

[6] heart disease monitoring and predicting by using machine learning based on IoT technology,20 5th International Conference on Innovative Technologies in Intelligent Systems and Industrial Applications (CITISIA) ©2020 IEEE | DOI: 10.1109/CITISIA50690.2020.9371772

[7] HeartCare: IoT based heart disease prediction system |2019 International Conference on Information Technology (ICIT)| 10.1109/ICIT48102.2019.00022 | IEEE

# appendix

# coding of a system

## *I*. Microcontroller programming (Arduino IDE)

#include <LiquidCrystal\_I2C.h>

#include < Wire .h>

#include "MAX30100\_PulseOximeter.h"

#include <SoftwareSerial.h>

SoftwareSerial esp(2,3);

int x=0;

float a,b,c;

#define DEBUG true

#define IP "184.106.153.149"// thingspeak.com ip

String Api\_key = "GET /update?key="OEMFMV390IANB4YF"; //change it with your api key like "GET /update?key=Your Api Key"

int error;

// Create the lcd object address 0x27 and 16 columns x 2 rows

LiquidCrystal\_I2C lcd (0x27, 16,2); //we can find the address through I2C address scanner

#define REPORTING\_PERIOD\_MS 3000

PulseOximeter pox;

uint32\_t tsLastReport = 0;

void onBeatDetected()

{

Serial.println("Beat!");

}

void setup()

{

esp.begin(115200);

send\_command("AT+RST\r\n", 2000, DEBUG); //reset module

send\_command("AT+CWMODE=1\r\n", 1000, DEBUG); //set station mode

send\_command("AT+CWJAP=\"Project\",\"12345678\"\r\n", 2000, DEBUG); //connect wifi network

while(!esp.find("OK")) { //wait for connection

Serial.println("Connected");}

Serial.begin(115200);

Serial.print("Initializing pulse oximeter..");

lcd.init ();

lcd. backlight ();

lcd.print("Initializing...");

delay(3000);

lcd.clear();

// Initialize the PulseOximeter instance

// Failures are generally due to an improper wiring, missing power supply

if (!pox.begin()) {

Serial.println("FAILED");

for(;;);

} else {

Serial.println("SUCCESS");

}

pox.setIRLedCurrent(MAX30100\_LED\_CURR\_4\_4MA);

// Register a callback for the beat detection

pox.setOnBeatDetectedCallback(onBeatDetected);

}

void loop()

{

// Make sure to call update as fast as possible

pox.update();

if (millis() - tsLastReport > REPORTING\_PERIOD\_MS) {

// Serial.print("Heart rate:");

//Serial.print(pox.getHeartRate());

//Serial.print("bpm / SpO2:");

//Serial.print(pox.getSpO2());

//Serial.println("%");

int reading = analogRead(A1);

// Convert the reading into voltage:

float voltage = reading \* (5000 / 1024.0);

// Convert the voltage into the temperature in degree Celsius:

float temperature = voltage / 10;

Serial.println(temperature);

// temperature =500-temperature;

// Print the temperature in the Serial Monitor:

//Serial.print(temperature);

//Serial.print(" \xC2\xB0"); // shows degree symbol

//Serial.println("C");

lcd.clear();

lcd.setCursor(0,0);

lcd.print("BPM:");

a=pox.getHeartRate();

lcd.print(a);

lcd.print(" T-");

c=temperature;

lcd.print(c);

lcd.setCursor(0,1);

lcd.print("SpO2:");

b=pox.getSpO2();

lcd.print(b);

lcd.print("%");

lcd.print(" ");

tsLastReport = millis();

}

if((a>60)&&(b>80))

{

start: //label

error=0;

updatedata();

if (error==1){

goto start; //go to label "start"

}

while(1);

}

}

void updatedata(){

String command = "AT+CIPSTART=\"TCP\",\"";

command += IP;

command += "\",80";

Serial.println(command);

esp.println(command);

delay(2000);

if(esp.find("Error")){

return;

}

command = Api\_key ;

command += "&field1=";

command += a;

command += "&field2=";

command += b;

command += "&field3=";

command += c;

command += "\r\n";

Serial.print("AT+CIPSEND=");

esp.print("AT+CIPSEND=");

Serial.println(command.length());

esp.println(command.length());

if(esp.find(">")){

Serial.print(command);

esp.print(command);

}

else{

Serial.println("AT+CIPCLOSE");

esp.println("AT+CIPCLOSE");

//Resend...

error=1;

}

}

String send\_command(String command, const int timeout, boolean debug)

{

String response = "";

esp.print(command);

long int time = millis();

while ( (time + timeout) > millis())

{

while (esp.available())

{

char c = esp.read();

response += c;

}

}

if (debug)

{

Serial.print(response);

}

return response;

}

## *II*.dataset and training

import numpy as np

import pandas as pd

from sklearn.model\_selection import train\_test\_split

import matplotlib.pyplot as plt

import seaborn as sns

import pickle

data = pd.read\_csv('/content/data\_kct.csv')

data.head()

data.shape

X = data.iloc[:,:-1]

X.head()

y = data.iloc[:,-1]

y.head()

data['target'].value\_counts()

X\_train,X\_test,y\_train,y\_test = train\_test\_split(X,y,test\_size=0.2,random\_state=1)

sns.countplot(x='target',data=data)

plt.show()

X\_train.shape

X\_train.head()

y\_test.shape

y\_test.head()

from sklearn.neighbors import KNeighborsClassifier

model = KNeighborsClassifier(n\_neighbors=3)

model.fit(X\_train,y\_train)

filename = 'Patient\_model.sav'

pickle.dump(model, open(filename, 'wb'))

y\_pred = model.predict(X\_test)

from sklearn import metrics

acc=(metrics.accuracy\_score(y\_pred,y\_test))

print("Accuracy is:",acc)

print("Confusion Matrix is: ",metrics.confusion\_matrix(y\_pred,y\_test))

## *III.*data classfication and prediction

import pickle

import urllib.request

import json

from time import sleep

conn = urllib.request.urlopen("https://api.thingspeak.com/channels/692121/feeds.json?results=1")

response = conn.read()

print ("http status code=%s" % (conn.getcode()))

data=json.loads(response)

x=int(data['feeds'][0]['entry\_id'])

y=x

conn.close()

while x==y:

conn = urllib.request.urlopen("https://api.thingspeak.com/channels/692121/feeds.json?results=1")

response = conn.read()

#print ("http status code=%s" % (conn.getcode()))

data=json.loads(response)

y=int(data['feeds'][0]['entry\_id'])

conn.close()

conn = urllib.request.urlopen("https://api.thingspeak.com/channels/692121/feeds.json?results=1")

response = conn.read()

print ("http status code=%s" % (conn.getcode()))

data=json.loads(response)

a=float(data['feeds'][0]['field3'])

b=float(data['feeds'][0]['field4'])

conn.close()

filename = 'Patient\_model.sav'

loaded\_model = pickle.load(open(filename, 'rb'))

person\_reports = [[a,b]]

disease\_predicted = loaded\_model.predict(person\_reports)

print("ANALYSING....")

if disease\_predicted[0]==0:

print("The person may have no disease")

#sleep(30)

conn = urllib.request.urlopen("https://api.thingspeak.com/update?api\_key=3J6LJDI7PCAU2IVL&field3=0")

elif disease\_predicted[0]==1:

print("The person may be in Fever take Paracetamol")

#sleep(30)

conn = urllib.request.urlopen("https://api.thingspeak.com/update?api\_key=3J6LJDI7PCAU2IVL&field3=1")

elif disease\_predicted[0]==2:

print("The person may be in Hypertension take nisoldipine")

#sleep(30)

conn = urllib.request.urlopen("https://api.thingspeak.com/update?api\_key=3J6LJDI7PCAU2IVL&field3=2")

elif disease\_predicted[0]==3:

print("The person may have Covid Visit Hospital" )

# 





