IOT BASED HEALTH MONITORING SYSTM

SONARGAON UNIVERSITY (SU)

Supervised By

Md. IQBAL KABIR

Lecturer

Department of EEE

Sonargaon University (SU)

Submitted By:

MD Arafat	:	EEE 1903018004 (18C)
MD Ruhul Amin	:	EEE 1903018056 (18C)
MD Fazle Rabbi	:	EEE 1903018130 (18C)
MD Khalid Saifullah	:	EEE 1903018124 (18C)

Department of Electrical & Electronic Engineering (EEE)
Sonargaon University (SU)
147/I, Panthapath,
Dhaka-1215, Bangladesh

Date of Submission: January 2023

January 2023

DECLARATION

It is declared hereby that this thesis paper or any part of it has not been submitted to anywhere else for the award of any degree.

Signature of the candidates

---------- **Name: MD Arafat** **ID: EEE 1903018004 (18C)**	---------- **Name: MD Ruhul Amin** **ID: EEE 1903018056 (18C)**
---------- **Name : MD Fazle Rabbi** **ID: EEE 1903018130(18C)**	---------- **Name: MD Khalid Saifullah** **ID: EEE 1903018124 (18C)**

Under Supervision of

.....
Md. IQBAL KABIR
Lecturer
Department of EEE
Sonargaon University (SU)

Certification

This is to certify that this project entitled "**IOT Based Health Monitoring system**" is done by the following students under my direct supervision. This project work has been carried out by them in the laboratories of the Department of Electrical and Electronic Engineering under the Faculty of Engineering, Sonargaon University (SU) in partial fulfillment of the requirements for the degree of Bachelor of Science in Electrical and Electronic Engineering.

Supervisor

……

Md. IQBAL KABIR
Lecturer

Department of Electrical and Electronic Engineer (EEE)

Sonargaon University (SU)

ACKNOWLEDGEMENT

The report titled as on **"IOT Based Health Monitoring System**" has been prepared to fulfill the requirement of our practicum program. In the process of doing and preparing our practicum report, we would like to pay our gratitude to some persons for their enormous help and vast co-operation.

At first, we would like to show our gratitude to the University authority to permit us to do our practicum. Specially, we would like to thank to our honorable teacher **Md. Iqbal Kabir,** Lecturer, Department of Electrical & Electronics Engineering, Sonargaon University, Dhaka, for his valuable and patient advice, sympathetic assistance, co-operation, contribution of new idea. Deep theoretical and hardware knowledge & keen interest of our supervisor in this field influenced us to carry out this project. His endless patience, scholarly guidance, continual encouragement, constant and energetic supervision, constructive criticism, valuable advice, reading many inferior draft and correcting them at all stage have made it possible to complete this project.

Finally, we would like to thanks again to the respected **Vice- Chancellor of SU, Professor**

Dr. Md. Abul Bashar also thanks to **Head of the Department of SU , Electrical & Electronics Engineering, Professor Dr. M. Bashir Uddin** because they are designated such an environment for learning through which we got the opportunity to acquire knowledge under B.Sc in EEE program, and that will be very helpful for our prospective career.

We are, indeed, grateful to all those from whom we got sincere cooperation and help for the preparation of this report

Abstract

The Heart Rate Monitoring system is developed using IOT technology with an objective of detecting the heartbeat of the patient in order to monitor the risk of heart attack and also the regular checkup. Body health monitoring is very important to us to make sure our health is in excellent condition. A sensor in this health monitoring system will collect information about the patient's health condition. It is smaller in size, faster, and more affordable. This system can be used to measure the oxygen saturation level, heart rate, and temperature of the human body and display the results on a web-based platform. IoT enables healthcare professionals to be more watchful and connect with the patients proactively. Data collected from IoT devices can help physicians identify the best treatment process for patients and reach the expected outcomes. An HMS is a vital tool that provides real-time data using wireless sensor networks. The sensors that attach to a patient's body to monitor the aforementioned patient health parameters and the software that an HMS includes helps transmit the information to their doctor or a hospital

Table of Contents

Name of the Contents		Page No.
ACKNOWLEDGEMENT		**I**
ABSTRACT		**II**
CHAPTER NAME		**III**
LIST OF FIGURES		**IV**
TABLES		**V**
LIST OF ABBREVIATIONS		**VI**
Chapter 1: Introduction		
1.1	Introduction	**01**
1.2	PROPOSED SYSTEM	**02**
1.3	Motivation	**03**
1.4	Objective	**04**
1.5	Sensors	**05**
1.6	IoT Server	**06**
Chapter 2: **THEORY OF THE PROJECT**		
2.1	Experimental Setup	**05**
2.2	EXPERIMENTAL RESULTS	**07**
2.3	Related Works	**07**
2.4	Thesis Contribution	**08**
Chapter 3: **DESIGN AND FABRICATION**		
3.1	Hardware	**09**
3.2	Block Diagram of This Project	**09**
3.3	Circuit Diagram of This Project	**10**
3.4	The list of devices used in the project is given below	**10**
3.4.1	Node MCU	**11**
3.4.2	SMPS	**14**
3.4.3	Max30100 Pulse Oximeter	**14**
3.4.4	LCD Display	**16**
3.4.5	Temperature Sensor	**17**
3.4.6	Smartphone	**18**
3.5	Project Implementation Result	**19**

4.1	Arduino IDE	**20**
4.2	Proteus 8 Professional	**27**
4.3	Bylink App	**29**
Chapter 5: **RESULT & DISCUSSION**		
5.1	Introduction	**35**
5.2	Result	**35**
5.3	Discussion & Cost Estimation	**37**
5.4	Application & Conclusion	**38**
Chapter 6: **CONCLUSION**		
5.1	Advantages	**18**
5.2	Disadvantages	**18**
5.3	Future Scopes	**19**
5.4	Conclusion	**19**
References		**20**
APPENDIX		**21**
A. Programming Codes		**22-26**

List of Figures

Figure No	Figure Contains	Page No
Figure 2.1	**AC to DC Converter**	**04**
Figure 2.2	L7805CV +5V linear voltage regulator	**05**
Figure 2.3	2-Channel Relay Module	**05**
Figure 2.4	Capacitors	**06**
Figure 2.5	BC 547N	**07**
Figure 2.6	PNP Transistor	**07**
Figure 2.7	Node MCU ESP8266	**08**
Figure 2.8	Peltier Module	**09**
Figure 2.9	Implemented DC power unit.	**09**
Figure 2.10	Heatsink	**10**
Figure 2.11	Temperature Sensor	**11**
Figure 2.12	Heatsink Paste	**11**
Figure 3.1	Block diagram of the proposed system	**13**
Figure 3.2	Complete circuit diagram of the project	**14**
Figure 4.1	Project Overview	**16**
Figure 4.2	Showing Temperature	**17**
Figure 4.3	**Environment Position.**	**17**

Chapter 1: Introduction

1.1 Introduction

Health is always a major concern in every growth the human race is advancing in terms of technology. Like the recent corona virus attack that has ruined the economy of China to an extent is an example how health care has become of major importance. In such areas where the epidemic is spread, it is always a better idea to monitor these patients using remote health monitoring technology. So Internet of Things (IoT) based health monitoring system is the current solution for it [1]. Remote Patient Monitoring arrangement empowers observation of patients outside of customary clinical settings (e.g. at home), which expands access to human services offices at bring down expenses [2]. The core objective of this project is the design and implementation of a smart patient health tracking system that uses Sensors to track patient health and uses internet to inform their loved ones in case of any issues. The objective of developing monitoring systems is to reduce health care costs by reducing SMS based patient flourishing viewing and IOT based patient checking framework. In IOT based framework, subtle parts of the patient flourishing can be seen by different clients [4]. The explanation behind this is the information should be checked by passing by a site or URL. While, in GSM based patient viewing, the flourishing parameters are sent utilizing GSM by strategies for SMS. In most of the rural areas, the medical facility would not be in a hand reach distance for the natives [5].So normally the people physician office visits, hospitalizations, and diagnostic testing procedure [3]. Each of our bodies utilizes temperature and also pulse acknowledging to peruse understanding wellbeing. The sensors are linked to a microcontroller to track the status which is thus interfaced to a LCD screen and additionally remote association with have the capacity to exchange alarms. If framework finds any sudden changes in understanding heart beat or body temperature, the framework consequently alarms the client about the patient's status over IOT and furthermore indicates subtle elements of pulse and temperature of patient live in the web. In this manner IOT set up tolerant wellbeing following framework viably utilizes web to screen quiet wellbeing measurements and spare persists time. There is a significant capability between neglect any kind of minor health issues which is shown in early stages by variation of vital elements like body temperature, heartbeat rate etc. Once the health issue has been increased to a critical stage and the life of the person is endangered, then they take medical assistance, which can cause an unnecessary waste of their earnings. This also comes into account especially when certain epidemic is spread in an area where the reach of doctors is impossible. So to avoid the spread of

disease, if a smart sensor is given to patients, who can be monitored from a distance would be a practical solution to save many lives [6]. In this paper section II describes about the proposed system, section III describes about the experimental setup including the circuit and section IV about the algorithm used in the implementation. The paper discusses the experimental results in section V.

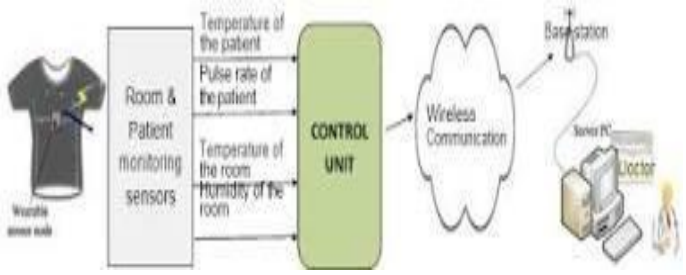


Figure 1: Proposed System

1.2 PROPOSED SYSTEM

The core objective of this project is the design and implementation of a smart patient health tracking system. Fig.1 shows the overview of the proposed system. The sensors are embedded on the patient body to sense the temperature and heartbeat of the patient. Two more sensors are place at home to sense the humidity and the temperature of the room where the patient is staying. These sensors are connected to a control unit, which calculates the values of all the four sensors. These calculated values are then transmitted through a IoT cloud to the base station. From the base station the values are then accessed by the doctor at any other location. Thus based on the temperature and heart beat values and the room sensor values, the doctor can decide the state of the patient and appropriate measures can be taken.

1.3 Motivation

The core to our motivation was to think ahead of our time and to make a contribution on the sector Internet of Things (IOT) which is undoubtedly the next big thing on the technological market. And IOT has proven itself to be a very handy possession. The idea of this project came to us by observing how easy it is to connect or integrate the normal home appliances with the internet. This led us to the secluded health sector which is still lagging behind from other industries in terms of technology. We observed many case even from our daily lives that in the hectic world we are living right now, it is difficult to keep a watch over everything. Specially, to keep an eye on the elderly people is difficult and costly as well. These real life situations gave us the ideas and motivation to

integrate the two sectors so that it helps in developing such a project which will create an economic and efficient health monitoring system and will pave the way for future work on the field of IOT.

1.4 Objective

- Making an automated system which will help to monitor host remotely is our primary objective.
- Making an alarm or reaction system which will react whenever there is an alarming situation.
- Providing a way to remotely monitor the temperature, pulse, counting the bowel discharge in a day and also the mount of sleep of the patient via Thinkspeak.
- Analyzing the collected data using the built in Matlab of the Thinkspeak sever to detect future hazards.
- Sending alarming messages via e-mail and twitter to the concerning authority or people if any abnormality is detected.
- Contributing in the field of IOT to pave a way for future project in the technological development

1.5 Sensors

The temperature sensor connected to the analog pin of the arduino controller is converted into digital value with the help of ADC [10]. Using this digital data, the controller converts it into the actual temperature value in degree Celsius using the equation:

temperature (°C) = [raw ADC value*5/4095- (400/1000)]*(19.5/1000)

The heartbeat sensor is based on the principle of photo phlethysmography. It measures the change in volume of blood through any organ of the body which causes a change in the light intensity through that organ (a vascular region) [7]. The digital pulses are given to a microcontroller for calculating the heat beat rate, given by the formula:

BPM (Beats per minute) = 60*f, where f is the pulse frequency A humidity sensor (or hygrometer) senses, measures and reports both moisture and air temperature. Humidity sensors work by detecting changes that alter electrical currents or temperature in the air. The relative humidity is calculated as given below: Voltage = (ADC Value/1023.0)*5.0; Percent relative humidity = (Voltage-0.958)/0.0307;or off.

1.6 IoT Server

At whatever point the patient goes to the healing center premises, sensors sense the physiological signs and these signs are changed over to electrical signs [8]. Then simple electrical flag is changed over to advanced flag (computerized information) which is put away in RFID. The put way computerized information is transmitted through Zigbee Protocol to the neighborhood server Zigbee is appropriate convention for this framework. It comprises of greatest number of cell hubs. It is more favored for gadgets which are littler in measure and expend less vitality. From nearby server the information is exchanged to the therapeutic server through WLAN. Medicinal server comprises of substantial database as given in Table I. At the point when the information is exchanged to the therapeutic server, it checks whether the patient has any past medicinal record then the server adds the new information to that record and exchanges to the specialist. In the event that patient does not have any past therapeutic record then the server makes new ID and stores the information in its database [9]. This information is exchanged to the specialist for diagnosis. The complete Data transmission using IoT is given in Fig. 2.

Table I: Database Structure

Sensor data	Datatype
Patient ID	Int
Body Temperature	Float
Pulse rate	Float
Room temperatur	Float
Room humidity	Float



Figure 2: Data Transmission using IoT

Chapter 2: Literature Review

2.1 Experimental Setup

The body temperature, humidity and pulse rate sensors are monitored and initially displayed on LCD as explained in the flowchart in Fig. 3 [10 – 29]. The values from the sensors especially the body temperature sensor and the pulse rate sensor is stored in the database. For body temperature, the range is defined as in Table II. The membership function of the temperature range as in Fig. 4 can be explained as:

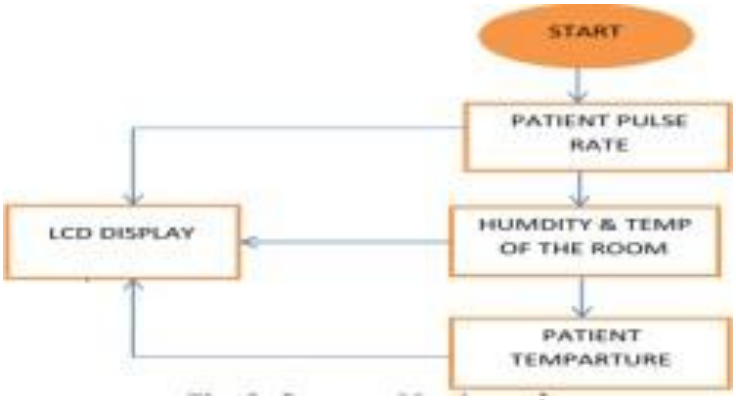


Fig. 3: Sensors Monitored

$$\text{Low} = \begin{cases} 1, x<36°c \\ 0, x>36°c \end{cases}$$

$$\text{Normal} = \begin{cases} 1, 36.0°c \leq x \leq 37.5℃ \\ 0, x>37.5℃ \; and \; x<36℃ \end{cases}$$

$$\text{High} = \begin{cases} 1, x>37.5℃ \\ 0, x<37.5℃ \end{cases}$$

Table II: Body Temperature

Body Temperature	State
36.0 –37.5 ℃	Normal
>37.5 °C	High
<36.6℃	Low

Similarly, to determine the health state of the patient, different range of pulse rate reading is also considered [10] as in Table III. The membership function of the pulse rate in Fig. 5 is as given below:

Low $= \begin{cases} 1, x<60\,BPM \\ 0, x>60\,BPM \end{cases}$

Normal $= \begin{cases} 1, 60\ BPM \leq X \leq 100\ BPM \\ 0, X>100\ BPM\ and\ X<60 BPM \end{cases}$

High $= \begin{cases} 1, x>100\ BPM \\ 0, x<10\ BPM \end{cases}$

Table III: Pulse Rate

Pulse rate	State
60 BPM - 100 BPM	Normal
>100 BPM	High
<60 BPM	Low

Based on these different range values, the rules for diagnosing the disease of the patient is performed. The output health state is diagnosed with the following membership function: Healthy, Unwell, Hypothermia, Fever and Needs a detailed health checkup as shown in Fig.6. The membership function of the output health state is defined as given below:

Checkup $= \begin{cases} 1, x<20 \\ 0, x>20 \end{cases}$

Unwell= $\begin{cases} 1, 20 \leq X \leq 40 \\ 0, X>40\ and\ X<20 \end{cases}$

Hypothermia= $\begin{cases} 1, 40 \leq x \leq 60 \\ 0, x<60\ and\ x<40 \end{cases}$

Healthy $= \begin{cases} 1, x>80 \\ 0, x<80 \end{cases}$

The rules for diagnosing the output health condition are as given in Table IV:

Table IV: Rules for Diagnosing Disease

Pulse rate	Body Temperature		
	Low	Normal	High
Low	Health Checkup	Unwell	Health Checkup
Normal	Hypothermia	Healthy	Fever
High	Health Checkup	Unwell	Health Checkup

These rules for diagnosis can be summarized by considering all the combinations of membership functions of the body temperature and the pulse rate as given below. • If the pulse rate and body temperature are (Low & Low) OR (Low & High) OR (High & Low) OR (High & High) •

Then the patient has to immediately go for a detailed Health Checkup. • If the pulse rate and body temperature are (Low & normal) OR (High & Normal), then the patient is considered to be unwell. • If the pulse rate and body temperature are (Normal & Low) then the patient is considered to be in a hypothermia state. • If the pulse rate and body temperature are (Normal & High) then the patient is considered to behaving fever. • If the pulse rate and body temperature are (Normal & Normal) then the patient is considered to be healthy.

2.2 EXPERIMENTAL RESULTS

The body temperature sensor, pulse rate sensor, room temperature and humidity sensor values are calibrated using the microcontroller. The complete prototype of the health monitoring system with the sensors are shown in Fig. 5 and Fig.6,where it shows the output values of the sensors calculated and displayed in a LCD display, so that these values are visible even to the patient.

Based on these values received, the disease of the patient is diagnosed by applying the rules set in Table IV. The diagnosis of the health state performed by the medical practitioner is shown in Fig 10. The medications can be prescribed and appropriate action can be suggested by the doctor even from a distance. When the pulse rate value was taken as 82.8 BPM (Normal) and the body temperature as 37°C (Normal), then the output value of health state is 91.4, which comes in the membership functionHealthy. So for all combinations of the input sensors, the output health state can be simulated and diagnosed using the Matlab simulation

2.3 Related Works:

An extensive research on the topic related to the system shows a very few of the related works could actually build their own preliminary framework and prototype of the system. Some of the works like the research conducted on the Ambient Assisted Living (AAL) [2] actually did more of a literature survey of the state of its present condition of the monitoring system via IoT. They also tried to identify and highlight the critical issues and the quality of service as well as the user driven experiences in their work. Some, other worked on showing or highlighting the importance of IoT in the health sector and some proposals for the health monitoring architectures. Some related findings used specific models for the health monitoring aspect. Like the abstraction of Model Driven Tree Reference Model (MDTRM) , where they explained the necessity of this model in the health field as well as identifying the complexities of the models. They also benched marked the models which came really handy for the initial phase of this research. Some other related model we found are General Domain Model Architecture (GDMA) , the health monitoring and sensing with cloud

processing was also a helpful source behind the research, as it was useful for generating ideas to get raw data's from wearable devices which are compatible and capable of measuring many physical value which we can use to obtain meaningful results. Masimo Radical-7 , a health monitor for clinical environment helps to collect data and wirelessly transmits it for ongoing display. This provides high resolution display of information with higher graphical capabilities. It also has a touch based user interface. But as it can be already assumed how cost effective it is, it can't send an alarm message to notify for any emergencies. Free Scale Home Health Hub reference platform store patient data to cloud via various sensors, where the people related to the patient can have an access. This platform too can't notify for any alarming situation to the people engaged with the patient. 13 Some surveys of ours also lead us to projects which even discussed to monitor the health whole area through wireless network sensors. They also tried to share their ideas by giving a model of their framework like cloud based processing and big data.

2.4 Thesis Contribution

Of all these ideas, models, frameworks and platforms that are surveyed for this research, we differ with all the above with a very basic and fundamental ways. The above researches conducted may use many monitoring variable on the contrary we are using more specific variable. Still, the basic functionality difference is that our motive to develop this project is to generate a functional response and to give a feedback to the relatives of the host, so that they can quickly take steps for the wellbeing of the host. The source of this response will be e-mail and social networking site twitter. But the basic idea of the research is totally of a different paradigm. Though, it is true that health monitoring and the prediction from anomalies as well as giving a useful feedback to the use is neither a very easy process nor the framework a very easy to set foot to. But, this research will take the health monitoring which now one of the most exciting topics (related to IoT) to a very different level. The useful information gathered from the research conducted has provided very handy to carry out and provide a workable model which will be discussed in the next segment of the paper. Workable data is something which we had to toil to obtain but the data's were managed properly to continue the research. Thus, most the related works done related to our project was highlighted in this segment and how our research differs from those works were also explained thoroughly. Some of those researches conducted really helped us understanding the importance of this sector and inspired to go the extra-mile on this field. The platform monitoring the daily activities of the host will be useful for the physician attending the host. The daily activities monitoring conducted by the system will help the physician to develop a conclusion and prediction to diagnose the host in a swift manner.

Chapter 3: Hardware Implementation

This chapter discusses the hardware required for the project implementation, discusses with figure the circuit connections and pseudo codes for the implementation.

3.1 Hardware

To conduct this project we need sensors so that we can monitor the health condition of the elderly people. In order to do so, we chose four different criteria's to monitor the elderly people. The four criteria's we chose are body temperature, pulse rate of the host, how long the person is sleeping and the number of time a person is using the toilet thus calculating how much time the host is spending there. So, for those four criteria's we chose the following sensors.

3.2 Block Diagram of This Project

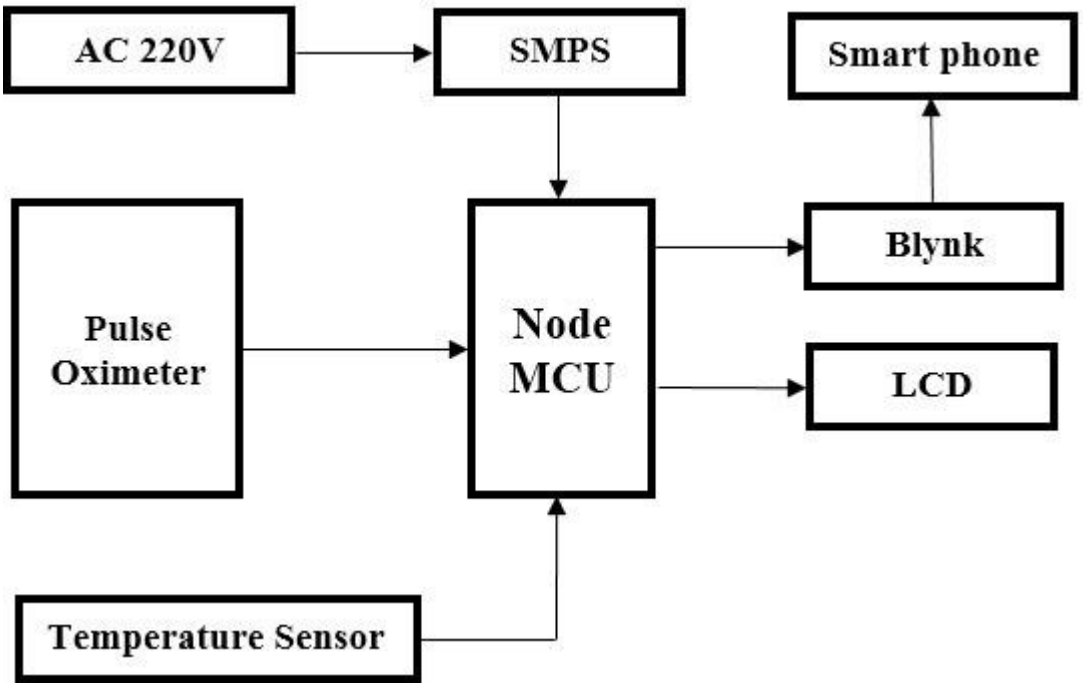


Fig: 3.1 Block Diagram

3.3 Circuit Diagram of This Project

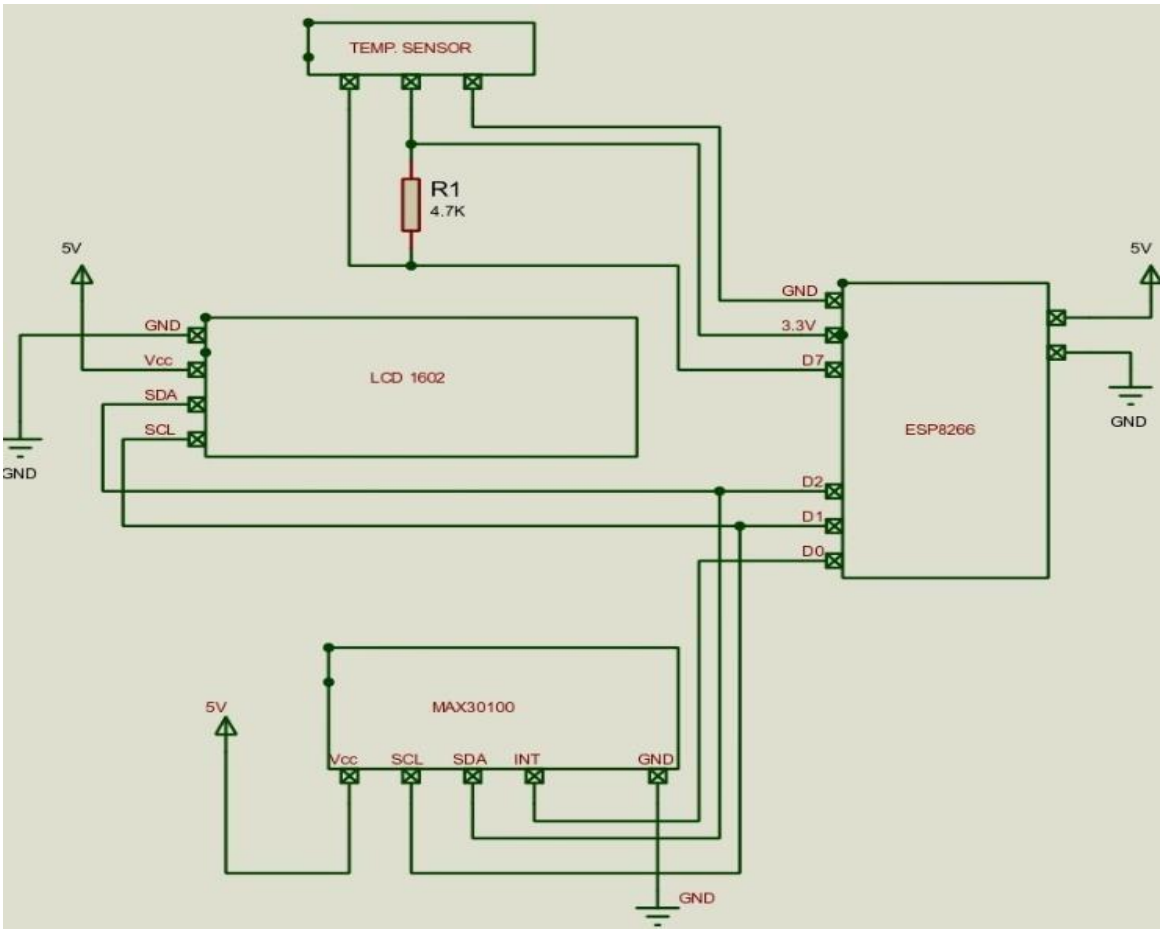


Fig: 3.2 Circuit Diagram

3.4 The list of devices used in the project is given below

- Node MCU.
- SMPS
- Max30100 Pulse Oximeter.
- LCD Display.
- Temperature Sensor
- Smart phone

3.4.1 Node MCU

The Here we use a Node MCU or ESP8266 development Board which is transformation of ESP12E wifi module and containing the ESP8266 chip which capacity is 32-bit LX106 RISE. It supports RTOS and it can operate 80MHz to 160MHz with adjustable clock frequency. It contain 1280 KB RAM and also 4 MB of flash memory to store data and programs. In this module we find wi-fi and Bluetooth and deep sleep operating features that make it ideal for IoT Projects. This ESP8266 is a family of Wi-Fi modules developed by the Chinese company Expressive. It is a System on a Chip, abbreviated simply as Sock with integrated TCP/IP protocol stack. It consists of a Ten silica L106 32-bit micro controller and a Wi-Fi transceiver. It has 11 input/output (I/O) pins. And it has a dedicated analog input for the programming purpose. It is programmable and it can be modified in several ways and that is the reason why this chip is the most prevalent IoT device in the market. In our system, we are using this Wi-Fi module because of its higher reliability and easy programming. (Peter, 2007) Wi-Fi module acts as a micro web server and hence eliminates the need for the wired connection between the Arduino board and computer which has many advantages. Main advantages are the reduction in size, cost, and increased convenience.

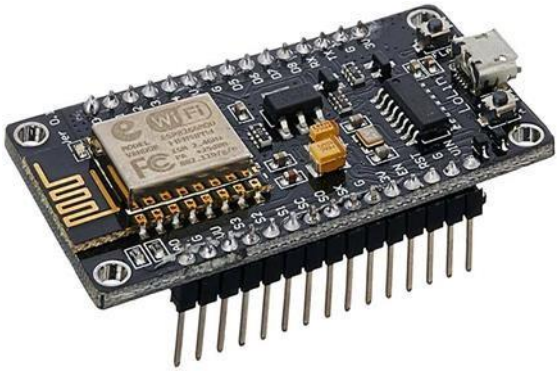


Figure 3.7: Node MCU ESP8266

The Wi-Fi module will require an active internet connection and it will act as the gateway for the Arduino board to communicate with the internet. This proposed system is flexible. I.e. apart from the above-listed components, our system can be interfaced with other sensors according to the user requirement. As Arduino.cc began developing new MCU boards based on non-AVR processors like the ARM/SAM MCU used in the Arduino Due, they needed to modify the Arduino IDE so it would be relatively easy to change the IDE to support alternate tool chains to allow Arduino C/C++ to be compiled for these new processors. They did this with the introduction of the Board Manager and the SAM Core. A "core" is the collection of software components required by the Board Manager and the Arduino IDE to compile an Arduino C/C++ source file for the target

MCU's machine language. Some ESP8266 enthusiasts developed an Arduino core for the ESP8266 WiFi SoC, popularly called the "ESP8266 Core for the Arduino IDE".[18] This has become a leading software development platform for the various ESP8266-based modules and development boards, including Node MCUs. The Node MCU (Node Micro Controller Unit) is an open-source software and hardware development environment built around an inexpensive System-on-a-Chip (SoC) called the ESP8266. The ESP8266, designed and manufactured by Espressif Systems, contains the crucial elements of a computer: CPU, RAM, networking (WiFi), and even a modern operating system and SDK. That makes it an excellent choice for Internet of Things (IoT) projects of all kinds.

The NodeMCU (Node MicroController Unit) is an open-source software and hardware development environment built around an inexpensive System-on-a-Chip (SoC) called the ESP8266. The ESP8266, designed and manufactured by Espressif Systems, contains the crucial elements of a computer: CPU, RAM, networking (WiFi), and even a modern operating system and SDK. That makes it an excellent choice for Internet of Things (IoT) projects of all kinds. However, as a chip, the ESP8266 is also hard to access and use. You must solder wires, with the appropriate analog voltage, to its pins for the simplest tasks such as powering it on or sending a keystroke to the “computer” on the chip. You also have to program it in low-level machine instructions that can be interpreted by the chip hardware. This level of integration is not a problem using the ESP8266 as an embedded controller chip in mass-produced electronics. It is a huge burden for hobbyists, hackers, or students who want to experiment with it in their own IoT projects.

But, what about Arduino? The Arduino project created an open-source hardware design and software SDK for their versatile IoT controller. Similar to NodeMCU, the Arduino hardware is a microcontroller board with a USB connector, LED lights, and standard data pins. It also defines standard interfaces to interact with sensors or other boards. But unlike NodeMCU, the Arduino board can have different types of CPU chips (typically an ARM or Intel x86 chip) with memory chips, and a variety of programming environments. There is an Arduino reference design for the ESP8266 chip as well. However, the flexibility of Arduino also means significant variations across different vendors. For example, most Arduino boards do not have WiFi capabilities, and some even have a serial data port instead of a USB port.

However, as a chip, the ESP8266 is also hard to access and use. You must solder wires, with the appropriate analog voltage, to its pins for the simplest tasks such as powering it on or sending a keystroke to the “computer” on the chip. You also have to program it in low-level machine instructions that can be interpreted by the chip hardware. This level of integration is not a problem

using the ESP8266 as an embedded controller chip in mass-produced electronics. It is a huge burden for hobbyists, hackers, or students who want to experiment with it in their own IoT projects. But, what about Arduino? The Arduino project created an open-source hardware design and software SDK for their versatile IoT controller. Similar to Node MCU, the Arduino hardware is a microcontroller board with a USB connector, LED lights, and standard data pins. It also defines standard interfaces to interact with sensors or other boards. But unlike Node MCU, the Arduino board can have different types of CPU chips (typically an ARM or Intel x86 chip) with memory chips, and a variety of programming environments. There is an Arduino reference design for the ESP8266 chip as well. However, the flexibility of Arduino also means significant variations across different vendors. For example, most Arduino boards do not have WiFi capabilities, and some even have a serial data port instead of a USB port. The Node MCU Development Board can be easily programmed with Arduino IDE since it is easy to use. Programming Node MCU with the Arduino IDE will hardly take 5-10 minutes. All you need is the Arduino IDE, a USB cable and the Node MCU board itself. You can check this Getting Started Tutorial for Node MCU to prepare your Arduino IDE for Node MCU. Besides adding WiFi capability, the main claim to fame for the ESP8266 processor over the AVR processor of the standard Arduino is that it has a larger 4 MB of Flash memory and runs at clock speeds of 80 MHz and can sometimes optionally be overclocked to 160 MHz and therefore has a very fast processing speed. The Digital I/O all support PWM and interrupts. In addition they can be configured to have pull-up or pull-down resistors. Though there are 11 digital I/O pins, 2 are typically reserved for use as the TX/RX lines if serial communications are used which leaves 9 digital I/O.

On the down-side, it has only 1 analog input which is probably the most significant limitation for some sensor type applications. That can always be overcome by using an external Analog Mux module like our 16-channel 74HC4067 or our ADS1115 4-channel 16-bit ADC module below if more analog I/O is desired. The module can be powered via the USB port or by using an external 7-12V power supply connected to the VIN pin. The module runs at 3.3V, so keep that in mind when working with I/O. The digital I/O is stated as being 5V tolerant, but the analog input needs to be limited to < 3.3V.The blue on-board LED is connected to D0 (GPIO16) and can be accessed using the LED_BUILTIN constant. The module has a typical ‘Reset’ button as well as a ‘Flash’ button. The Flash button is used when programming using the original Node MCU firmware. If the module is being used with the Arduino IDE, the Flash button does not need to be used to program the board and it will program just as any Arduino board would.

The module comes loaded with the Node MCU software that accepts the standard AT command set. The module was initially designed primarily to be programmed using Lua which is an interpreted language, but Lua has been mostly abandoned because it takes up a lot of memory and is slow because it is interpreted and more importantly it is generally buggy. It can also be programmed in C using the Arduino IDE and is how the modules are most often used. An example program is shown down below. If a program is download via the IDE, it will overwrite the Node MCU software or whatever else was loaded before. If that is a problem for what you want to do, the Node MCU software can always be reloaded. here are many instructions for installing and using ESP8266 based boards with the Arduino IDE, but here is a short-hand version. Note that once the ESP8266 board type is added to the IDE, there will be many more items added to the Tools drop down menu.

3.4.2 SMPS

The S.M.P.S operated at input regular voltage AC 100V – 240V.
The S.M.P.S should be capable of a total continuous DC power output of 40 Watts.
The S.M.P.S should be capable of a total peak 50 Watts.



Fig-3.7: SMPS

3.4.3 Max30100 Pulse Oximeter

MAX30100 is an integrated pulse oximetry and heart-rate monitor sensor solution. It integrates two LEDs (IR and Red), a photodetector (Red), optimized optics, and low-noise analog signal processing to detect pulse oximetry and heart-rate signals. It is fully configurable through software

registers and the digital output data is stored in a 16-deep FIFO within the device.It has an I2C digital interface to communicate with a host microcontroller.

The pulse oximetry subsystem in MAX30100 consists of ambient light cancellation (ALC), 16-bit sigma delta ADC, and proprietary discrete time filter. It has an ultra-low-power operation which makes it ideal for battery operated systems.MAX30100 operates on a supply in the range of 1.8 to 3.3V.It can be used in wearable devices, fitness assistant devices, medical monitoring devices, etc.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.



Figure: Max30100 Pulse Oximeter

Features:

- Integrated LEDs, Photo Sensor, and High-Performance Analog Front -End
- 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

Applications:

- Fitness Assistant Devices
- Medical Monitoring Devices
- Wearable Devices
- **Package includes:**
- 1 x Max30100 Pulse Oximeter Heart-Rate Sensor Module

3.4.4 LCD Display

LCD20*04, or 20*04 character-type liquid crystal display, is a kind of dot matrix module to show letters, numbers, and characters and so on. It's composed of 5x7 or 5x11 dot matrix positions; each position can display one character. There's a dot pitch between two charactersand a space between lines, thus separating characters and lines. The model 20*04 means it displays 4 lines of 20characters. Generally, LCD20*04 has parallel ports, that is, it would control several pins at the sametime. LCD20*04 can be categorized into eight-port and four-port connections. If the eight- port connection is used, then all the digital ports of the Sun Founder Uno board are almost completely occupied. If you want to connect more sensors, there will be no ports available. Therefore, the four- port connection is used here for better application.

Pin	Function
VSS	connected to ground
VDD	connected to a +5V power supply
VO	to adjust the contrast
RS	A register select pin that controls where in the LCD's memory you are writing data to. You can select either the data register, which holds what goes on the screen, or an instruction register, which is where the LCD's controller looks for instructions on what to do next.
R/W	A Read/Write pin to select between reading and writing mode
E	An enabling pin that reads the information when High level (1) is received. The instructions are run when the signal changes from High level to Low level.
D0-D7	to read and write data
A	Pins that control the LCD backlight. Connect A to 3.3v.
K	Pins that control the LCD backlight. Connect K to GND.

Tabel.01:Lcd display outline



Fig- 3.10: LCD Display

3.4.5 Temperature Sensor

The DS18B20 is a digital thermometer that accurately measures temperature in the range -10°C to +85°C and also includes alarm functions and trigger

Temperature sensors work by providing readings via electrical signals. Sensors are composed of two metals that generate an electrical voltage or resistance when a temperature change occurs by measuring the voltage across the diode terminals. When the voltage increases, the temperature also increases. A temperature sensor is a key component of any process heating application as it provides temperature feedback about the process, which can be used to monitor or control the process.



Fig 1: Temperature Sensor

A temperature sensor is an electronic device that measures the temperature of its environment and converts the input data into electronic data to record, monitor, or signal temperature changes. There are many different types of temperature sensors. There are Temperature Sensor applications in many industries including medical, motorsport, HVAC, agriculture, industrial, aerospace and

automotive. The four most common types of temperature sensors, ranging in responsiveness and accuracy from high to low are: Negative Temperature Coefficient (NTC) Thermistors. Resistance Temperature Detectors (RTDs) Thermocouples. Semiconductor-Based Sensors. Temperature sensors work by providing readings via electrical signals. Sensors are composed of two metals that generate an electrical voltage or resistance when a temperature change occurs by measuring the voltage across the diode terminals. When the voltage increases, the temperature also increases

3.4.6 Smartphone



Figure: Smartphone

A smartphone is a cellular telephone with an integrated computer and other features not originally associated with telephones such as an operating system, web browsing, and the ability to run software applications. IBM announced the first smartphone in 1992, which wouldn't be available for purchase for two more years. The Simon Personal Communicator (SPC) included many features familiar to modern smartphone users, such as a touch screen that required a stylus (a what?)

Uses include:

- Sending and receiving emails, text, photographs and multimedia messages.
- Registering contacts.
- Calculator, currency conversions, alarm, etc. functions.
- Browsing the Internet using a mobile browser.
- Playing games.
- Video chat.
- Mobile payment for goods or services.

- Barcode scanning.

3.5 Project Implementation Result

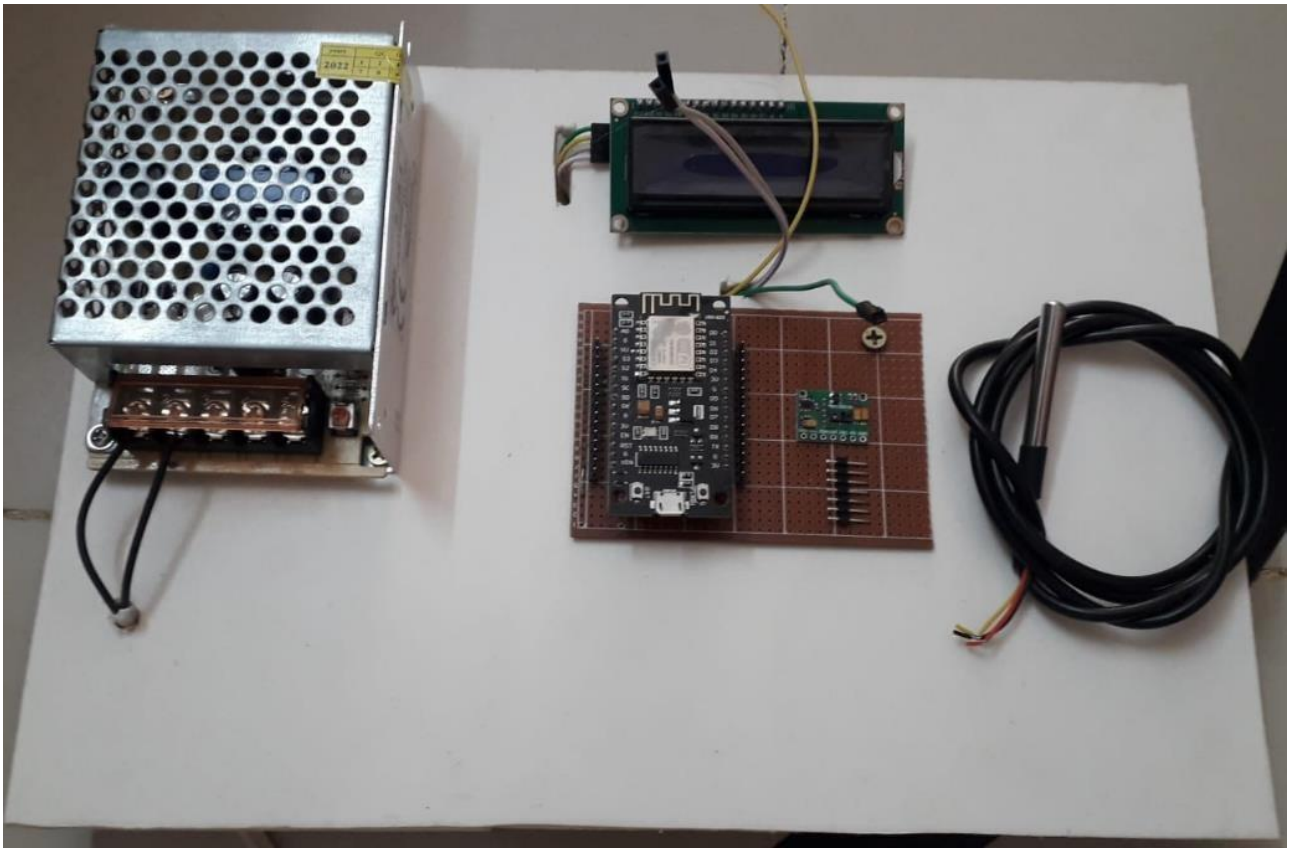


Fig-3.17: Project picture

Chapter 4: Software and Simulation

4.1 Arduino IDE

The Arduino IDE is an open-source software, which is used to write and upload code to the Arduino boards. The IDE application is suitable for different operating systems such as Windows, Mac OS X, and Linux. It supports the programming languages C and C++. Here, IDE stands for Integrated Development Environment.

The program or code written in the Arduino IDE is often called as sketching. We need to connect the Genuino and Arduino board with the IDE to upload the sketch written in the Arduino IDE software. The sketch is saved with the extension '.ino.'

The Arduino IDE will appear as:

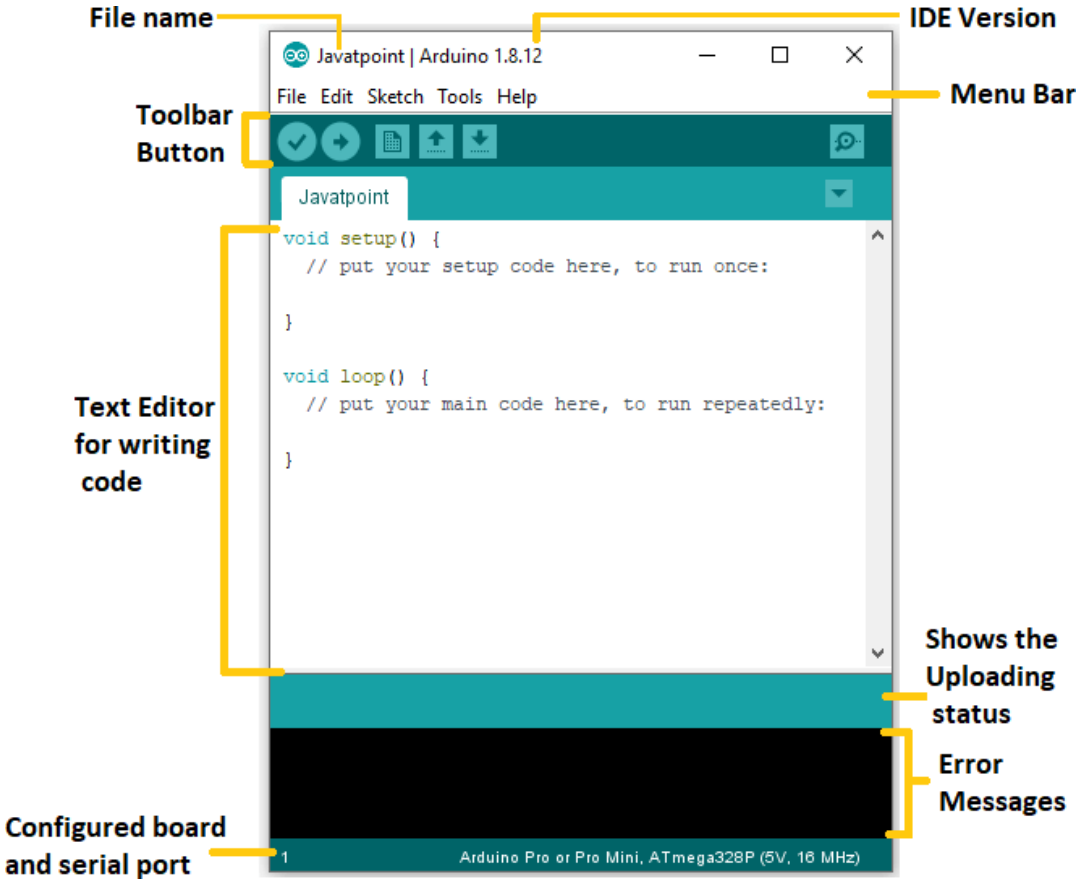


Figure 4.1: Java Point

Toolbar Button

The icons displayed on the toolbar are **New, Open, Save, Upload,** and **Verify**.

It is shown below:

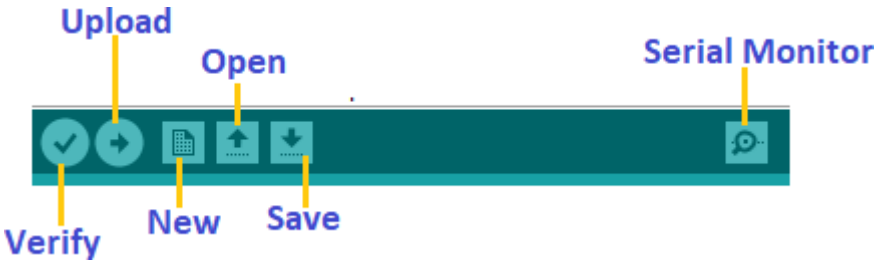


Figure 4.2: Toolbar Button

Upload

The Upload button compiles and runs our code written on the screen. It further uploads the code to the connected board. Before uploading the sketch, we need to make sure that the correct board and ports are selected.

We also need a USB connection to connect the board and the computer. Once all the above measures are done, click on the Upload button present on the toolbar.

The latest Arduino boards can be reset automatically before beginning with Upload. In the older boards, we need to press the Reset button present on it. As soon as the uploading is done successfully, we can notice the blink of the Tx and Rx LED.

If the uploading is failed, it will display the message in the error window.

We do not require any additional hardware to upload our sketch using the Arduino Boot loader. A Boot loader is defined as a small program, which is loaded in the microcontroller present on the board. The LED will blink on PIN 13.

Open

The Open button is used to open the already created file. The selected file will be opened in the current window.

Save

The save button is used to save the current sketch or code.

New

It is used to create a new sketch or opens a new window.

Verify

The Verify button is used to check the compilation error of the sketch or the written code.

Serial Monitor

The serial monitor button is present on the right corner of the toolbar. It opens the serial monitor.

When we connect the serial monitor, the board will reset on the operating system Windows, Linux, and Mac OS X. If we want to process the control characters in our sketch, we need to use an external terminal program. The terminal program should be connected to the COM port, which will be assigned when we connect the board to the computer.

Menu Bar

- File

When we click on the File button on the Menu bar, a drop-down list will appear. It is shown below:

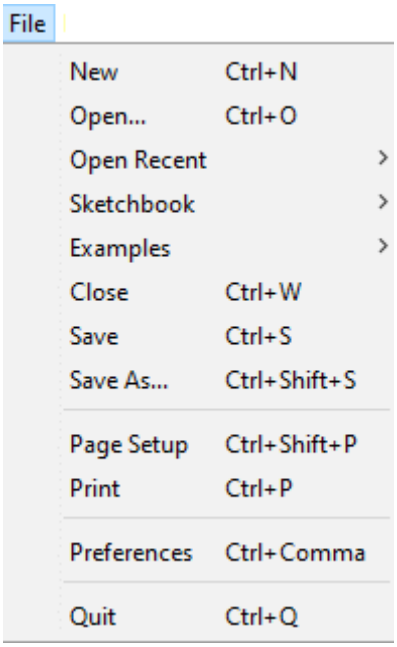


Figure 4.3: File

Let's discuss each option in detail.

New

The New button opens the new window. It does not remove the sketch which is already present.

Open

It allows opening the sketch, which can be browsed from the folders and computer drivers.

Open Recent

The Open Recent button contains the list of the recent sketches.

Sketchbook

It stores the current sketches created in the Arduino IDE software. It opens the selected sketch or code in a new editor at an instance.

Examples

It shows the different examples of small projects for a better understanding of the IDE and the board. The IDE provides examples of self-practice.

Close

The Close button closes the window from which the button is clicked.

Save

The save button is used to save the current sketch. It also saves the changes made to the current sketch. If we have not specified the name of the file, it will open the 'Save As...' window.

Save As...

We can save the sketch with a different name using the 'Save As...' button. We can also change the name accordingly.

Page Setup

It allows setting the page margins, orientation, and size for printing. The 'Page Setup' window will appear as:

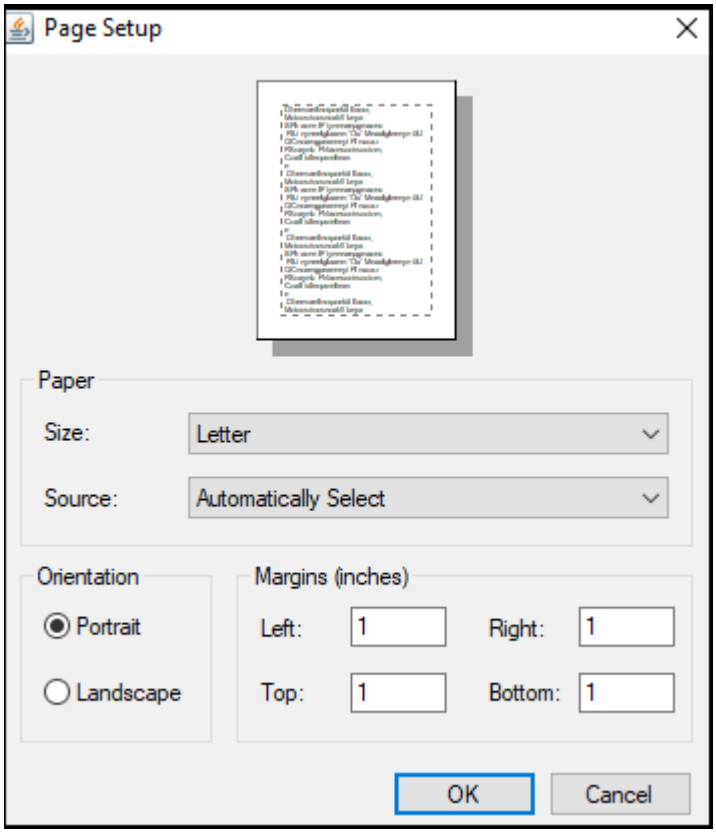


Figure 4.4: Page Setup

Print

According to the settings specified in the ‘Page Setup’, it prepares the current sketch for printing.

Preferences

It allows the customization settings of the Arduino IDE.

Quit

The Quit button is used to close all the IDE windows. The same closed sketch will be reopened when we will open the Arduino IDE.

- Edit

When we click on the Edit button on the Menu bar, a drop-down list appears. It is shown below:

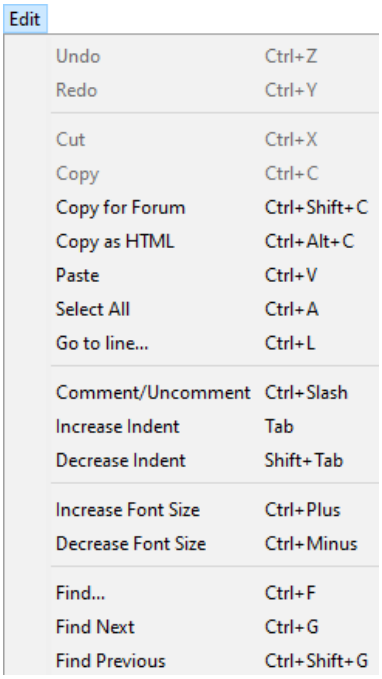


Figure 4.5: Edit

Let's discuss each option in detail.

Undo

The Undo button is used to reverse the last modification done to the sketch while editing.

Redo

The Redo button is used to repeat the last modification done to the sketch while editing.

Cut

It allows us to remove the selected text from the written code. The text is further placed to the clipboard. We can also paste that text anywhere in our sketch.

Copy

It creates a duplicate copy of the selected text. The text is further placed on the clipboard.

Upload

The Upload button is used to configure the code to the specified board through the port.

Upload Using Programmer

It is used to override the Boot loader that is present on the board. We can utilize the full capacity of the Flash memory using the 'Upload Using Programmer' option. To implement this, we need to

restore the Boot loader using the Tools-> Burn Boot loader option to upload it to the USB serial port.

Export compiled Binary

It allows saving a .hex file and can be kept archived. Using other tools, .hex file can also be sent to the board.

Show Sketch Folder

It opens the folder of the current code written or sketch.

Include Library

Include Library includes various Arduino libraries. The libraries are inserted into our code at the beginning of the code starting with the #. We can also import the libraries from .zip file.

Add File...

The Add File... button is used to add the created file in a new tab on the existing file.

For example, let's add 'Blink' file to the 'Java point' file. The tab will now appear as:

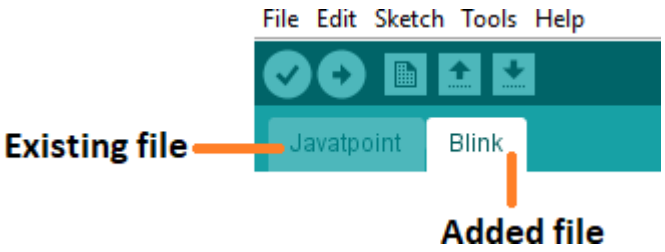


Figure 4.6: Added file

We can also delete the corresponding file from the tab by clicking on the small triangle -> Delete option.

Tools

When we click on the Tools button on the Menu bar, a drop-down list appears. It is shown below:

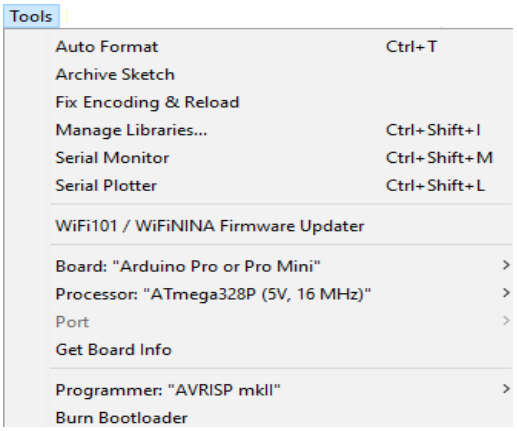


Figure 4.7: Tools

Let’s discuss each option in detail.

Auto Format

The Auto Format button is used to format the written code. For example, lining the open and closed curly brackets in the code.

Archive Sketch

The copy of the current sketch or code is archived in the .zip format. The directory of the archived is same as the sketch.

Fix Encoding and Reload

This button is used to fix the inconsistency between the operating system char maps and editor char map encoding.

Manage Libraries…

It shows the updated list of all the installed libraries. We can also use this option to install a new library into the Arduino IDE.

Serial Monitor

It allows the exchange of data with the connected board on the port.

Serial Plotter

The Serial Plotter button is used to display the serial data in a plot. It comes preinstalled in the Arduino IDE.

WiFi101/WiFi NINA Firmware Updater

It is used to check and update the Wi-Fi Firmware of the connected board.

Board

We are required to select the board from the list of boards. The selected board must be similar to the board connected to the computer.

Processor

It displays the processor according to the selected board. It refreshes every time during the selection of the board.

Port

It consists of the virtual and real serial devices present on our machine.

Get Board Info

It gives the information about the selected board. We need to select the appropriate port before getting information about the board.

Programmer

We need to select the hardware programmer while programming the board. It is required when we are not using the onboard USB serial connection. It is also required during the burning of the Boot loader.

Burn Boot loader

The Boot loader is present on the board onto the microcontroller. The option is useful when we have purchased the microcontroller without the boot loader. Before burning the boot loader, we need to make sure about the correct selected board and port.

Help

When we click on the Help button on the Menu bar, a drop-down list will appear. It is shown below:

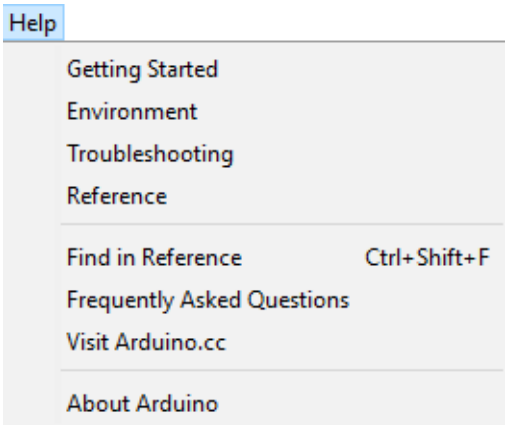


Figure 4.8: Help

The Help section includes several documents that are easy to access, which comes along with the Arduino IDE. It consists of the number of options such as Getting Started, Environment, Troubleshooting, Reference, etc. We can also consider the image shown above, which includes all the options under the Help section. Some documents like Getting started, Reference, etc., can be accessed without the internet connection as well. It will directly link us to the official website of Arduino.

4.2 Proteus 8 Professional

This is a brief introduction to Proteus application. The special features and advantages of this application will be immediately below. Hopefully this article will give you the most extensive and useful view on how to use Proteus 8 Professional.

Want to check the circuit diagram? Seeing circuit layout seems like a good choice but it is not easy to do it for large circuit boards. Or you can try creating a circuit diagram (PCB) printout and testing

but it takes a lot of time-consuming operations. It seems, using Proteus 8 Professional is the most ideal. Proteus 8 Professional is a software used to plot, sketch PCB and simulate diagrams.

This software was developed by **Lab center Electronic Ltd**.

Features of Proteus 8 Professional

Very easy to draw diagrams on Proteus 8 Professional. Click "select device" and select the desired detail, you can draw the wire by clicking on the output of electricity or Vcc, Ground, etc.
Is the circuit completed? The simulation feature of Proteus 8 Professional can display most details in the circuit. There are two options for this feature: run the simulation and run each table. "Running simulation" shows circuit board at normal speed (if the circuit is not too complicated). "Running each table" will run each circuit board at every click. This customization is useful for fixing digital circuits. You can simulate microcontrollers. Depending on each microcontroller will be outlined based on PIC24, dsPIC33, 8051, Arduino, or ARM7. You can download the compiler and dump hex file into the microcontroller contained in Proteus. Moreover, real-time integration with simulation using switches, resistors, photoelectric resistors, etc.

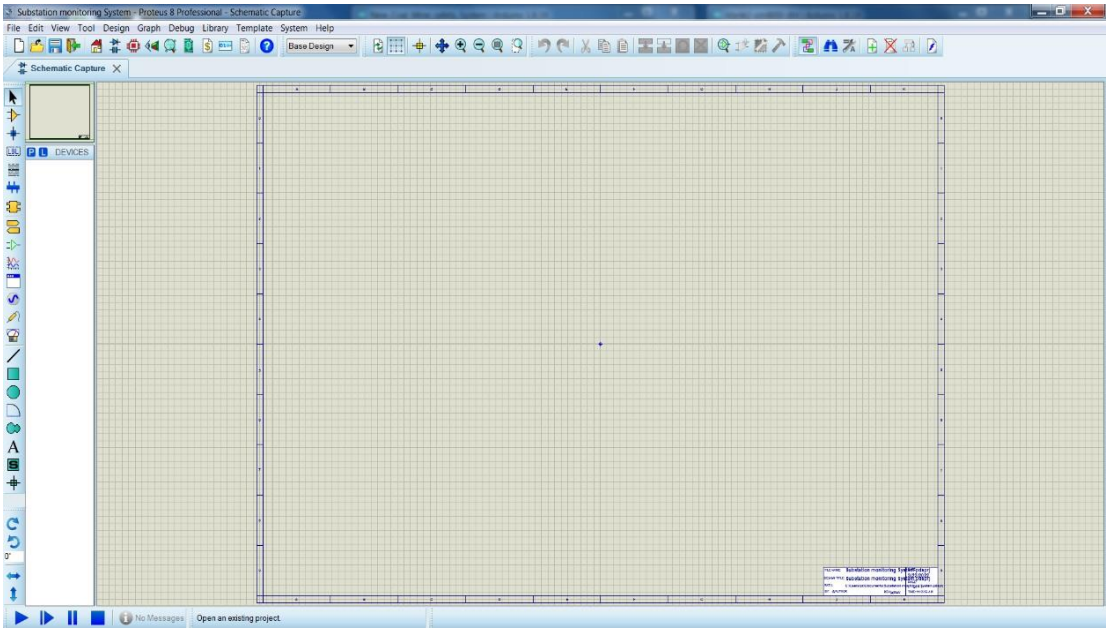


Figure 4.9: Schematic Capture

Is an easy-to-use feature in Proteus. You can create your own designs or make Proteus work for you. Creating your own design is easy, you just need to put the details into the diagram and draw the electrical circuit through. Don't worry about breaking any design rules because it will automatically detect errors. If you want Proteus to do it for you, just put the details in the corresponding position and run it automatically. It will draw the way to set the circuit and choose

the best version. And now there's another custom "Auto placer", which requires you to set the board size by drawing the shape and size of the board. After that, it automatically puts the details into the mold. After that, all you have to do is make a circuit diagram.

Pros and cons Proteus 8 Professional

Advantages:

- You can interact with test run simulations, use switches, etc.
- There are virtual instruments available for measuring electricity.

Defect:

- User interface should be better.

4.3 Blynk App

With **Blynk**, you can create smartphone applications that allow you to easily interact with microcontrollers or even full computers such as the **Raspberry Pi.** The main focus of the Blynk platform is to make it super-easy to develop the mobile phone application. As you will see in this course, developing a mobile app that can talk to your Arduino is as easy as dragging a widget and configuring a pin. With Blynk, you can control an LED or a motor from your mobile phone with literally zero programming. This is actually the first experiment that I will demonstrate in this course. But don't let this simplicity make you think that Blynk is only useful for trivial applications. Blynk is a robust and scalable tool that is used by hobbyists and the industry alike.

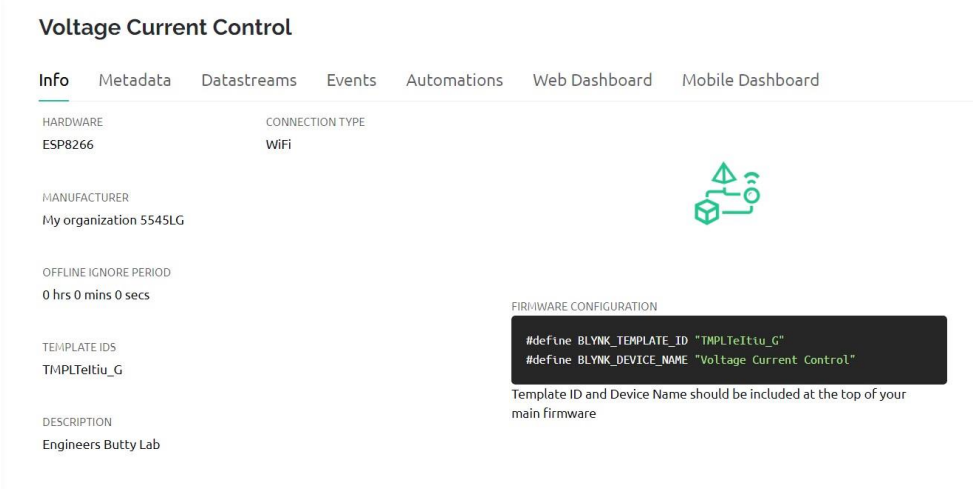


Figure 4.10: Info

You can use it to monitor the soil humidity of your vegetable garden and turn on the water, or open up your garage door, with your phone. You can also use it to control smart furniture that can learn

from your routines, or embed IoT and AI to traditional industrial products such as a boiler, or for improving the integrity and safety of oilfields. Blynk is free to use for personal use and prototyping. Their business model generates profits by selling subscriptions to businesses that want to publish Blynk-powered apps for their hardware products or services.

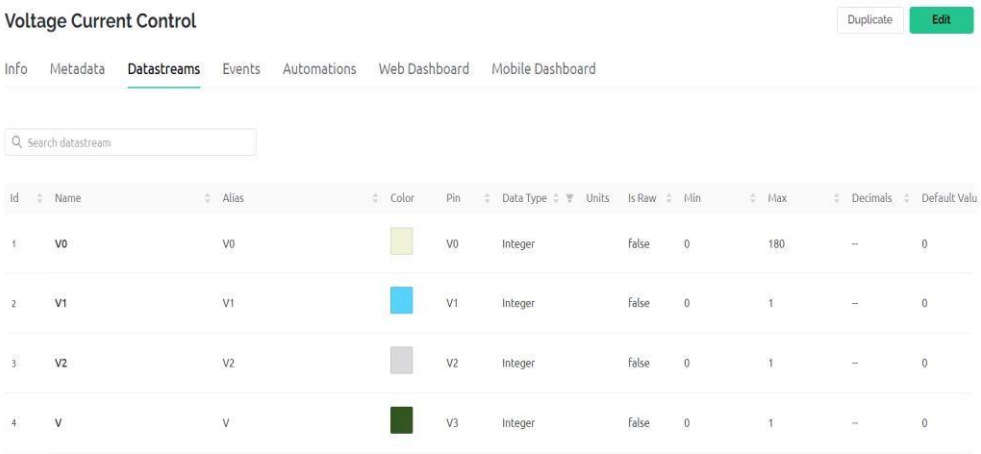


Figure 4.11: DataStream's

The Blynk innovation is the smartphone app. The Blynk app is is really an app editor. It allows you to create one or more projects. Each project can contain graphical widgets, like virtual LEDs, buttons, value displays and even a text terminal, and can interact with one or more devices. With the help of the Blynk library, it is possible to control Arduino or ESP32 pins directly from your phone, without having to write any code at all. It is also possible to share a project with friends and even customers so that they can access the connected devices but not be able to modify the project. Imagine a scenario where you build a smartphone application where you can control lights, window blinds and room temperature from your phone. You can share the project with other family members so that they can also access the functionality.

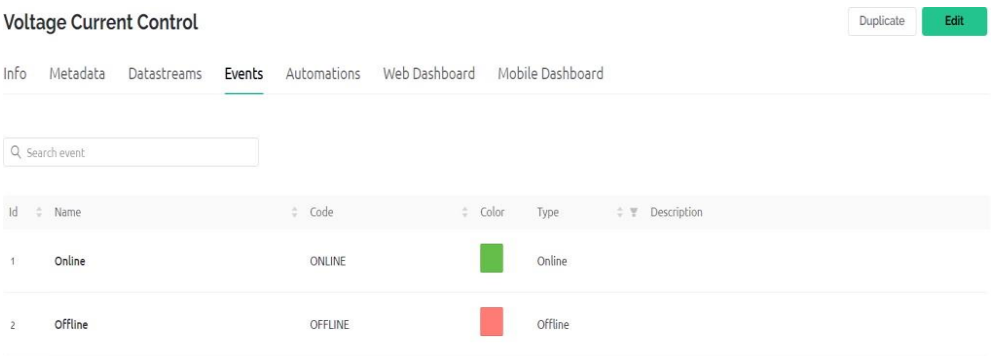


Figure 4.12: Events

I am amazed by the range of devices and connectivity types that the Blynk platform supports. The support is implemented by means of a Blynk library that targets a device and connectivity type combination. For example, if you want to use your Arduino Uno with an Ethernet shield, you would use the library "BlynkSimpleEthernet" which contains the Blynk firmware plus the required connectivity support.

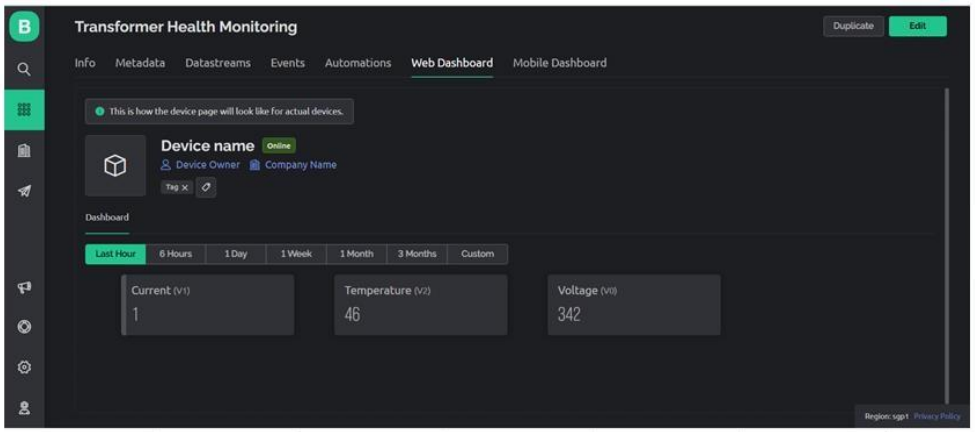


Figure 4.13: Web Dashboard

Both sketches would use the same Blynk infrastructure, such as the physical pins that are part of the Arduino's hardware, as well as the virtual pins which are implemented in software by the Blynk Platform. This means that with a bit of planning, you can write sketches that can be easily shared among different target devices. You can potentially write a sketch for an Arduino Uno and and with minimal modification run it on an Arduino MKR1010. Blynk also supports clients that are not a microcontroller. You can write client code is JavaScript, Python or Lua thanks to the available Blynk libraries for these languages. This means that you can have a Blynk project that interacts with an Arduino and with Python code running on a Linux virtual machine somewhere on the Cloud. The possibilities are really very exciting.

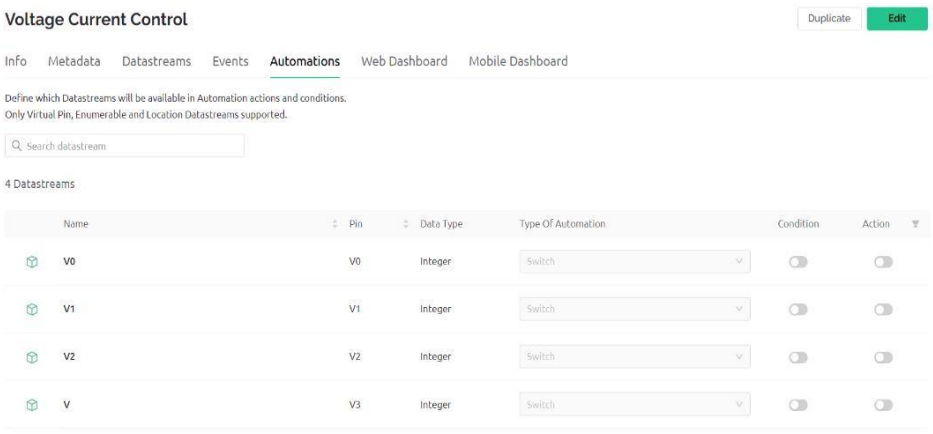


Figure 4.14: Automations

Unlike IoT platform such as IFTTT, Twillio, and even Adafruit IO, you can host a private instance of the full Blynk server and connect your smartphone Blynk app to it. The Blynk Cloud server is an excellent choice for most projects, as it is always there, ready to use. We will use the Cloud server in the first few experiments in this course to help you get started with minimal effort. However, as you will see, the Cloud Blynk server has imposed limitations. Some limitations are due to the topology of the server: depending on your geographical location, the server may be in a different continent, which makes communications between the app, the devices and the server slow due to the amount of time it takes packets to travel across the Internet.

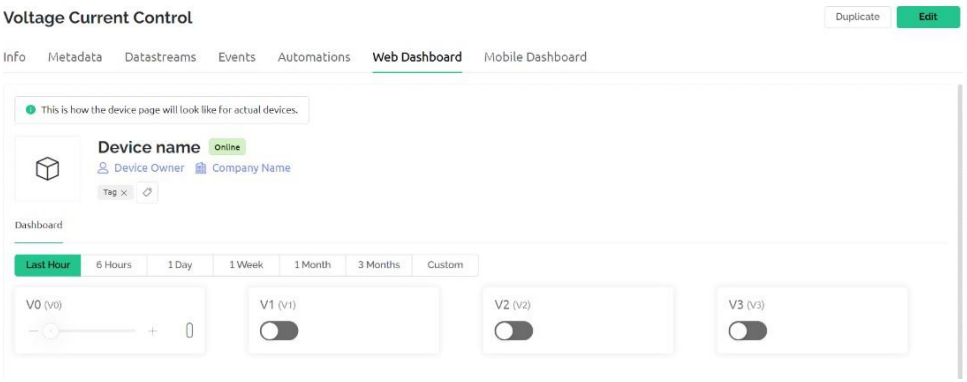


Figure 4.15: Web Dashboard

Another imposed limitation is that in the Cloud server, you can only use a small number of widgets. Blynk is using the concept of "energy" to implement a pricing system for its widgets. In the Cloud server you may start a new project with 1000 energy units. An LED widget may cost you 200 units, leaving 800 units for other widgets. On a private server, you can set your own energy limits. You can configure your server to allocate 100,000 energy units to new users. It's totally up to you. Of course, you can purchase additional energy units to spent on the Cloud Blynk server, and this is a legitimate consideration especially if the users of your Blynk project are distributed around the world (therefore latency is not a big issue).

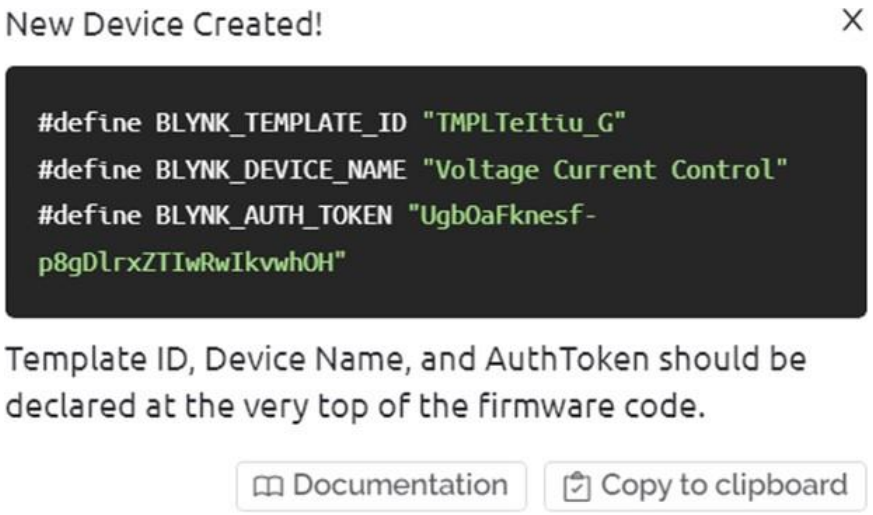


Figure 4.16: New Device Created

However, a private Blynk server gives you additional benefits:

- Essentially unlimited energy units, so that you can build any Blynk application you can think of.
- Minimal latency, which is useful when your application is used in a limited geographic area and responsiveness is important.
- Total control of your data. You can keep your own backups of your Private server, migrate your server to a new host, implement whichever security mechanisms you wish, and finely control your users.

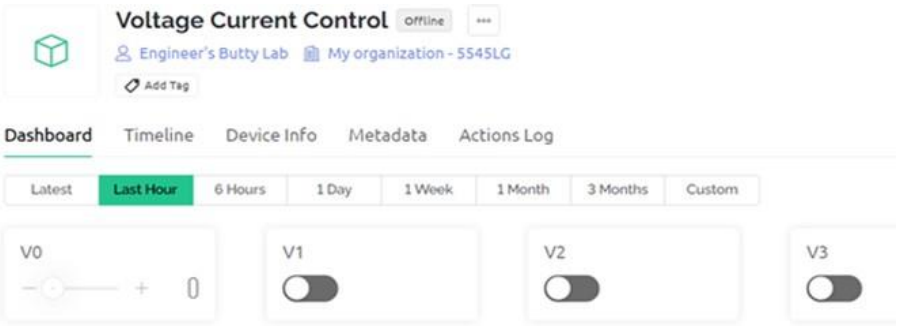


Figure 4.17: Dashboard

In this course, I will show you how to install and configure a private Blynk server so you can see the related benefits and costs. You can run an instance of the Blynk server on any host that has a

Java runtime environment. In this course, I use a $15 Raspberry Pi Zero W, running the Raspberry Pi operating system. This is a computer with just half a gigabyte of RAM, and still runs the Blynk server without any issues at all. You can use any Raspberry Pi with WiFi, it does not need to be a Zero W. In fact, you can use any Linux or Windows host you want, including virtual machines.

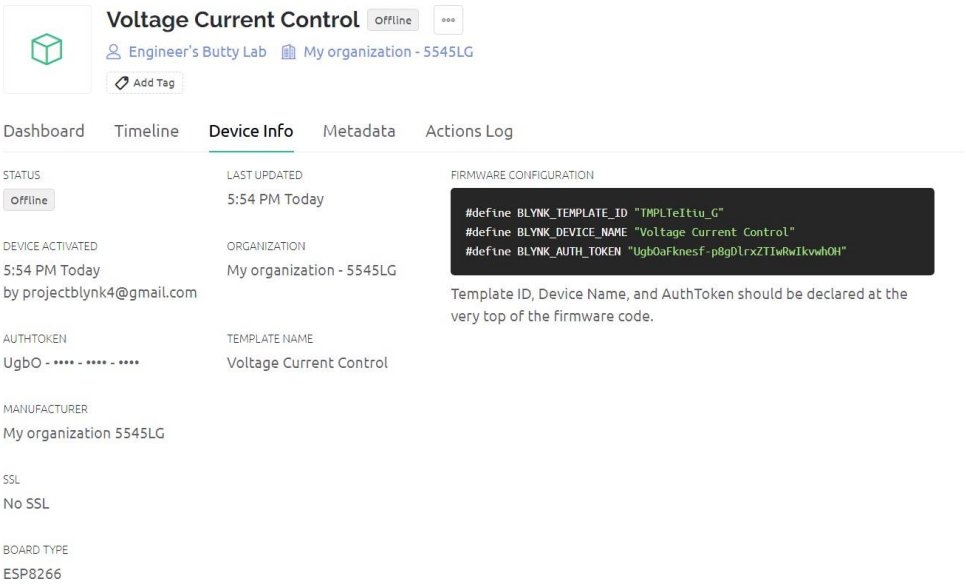
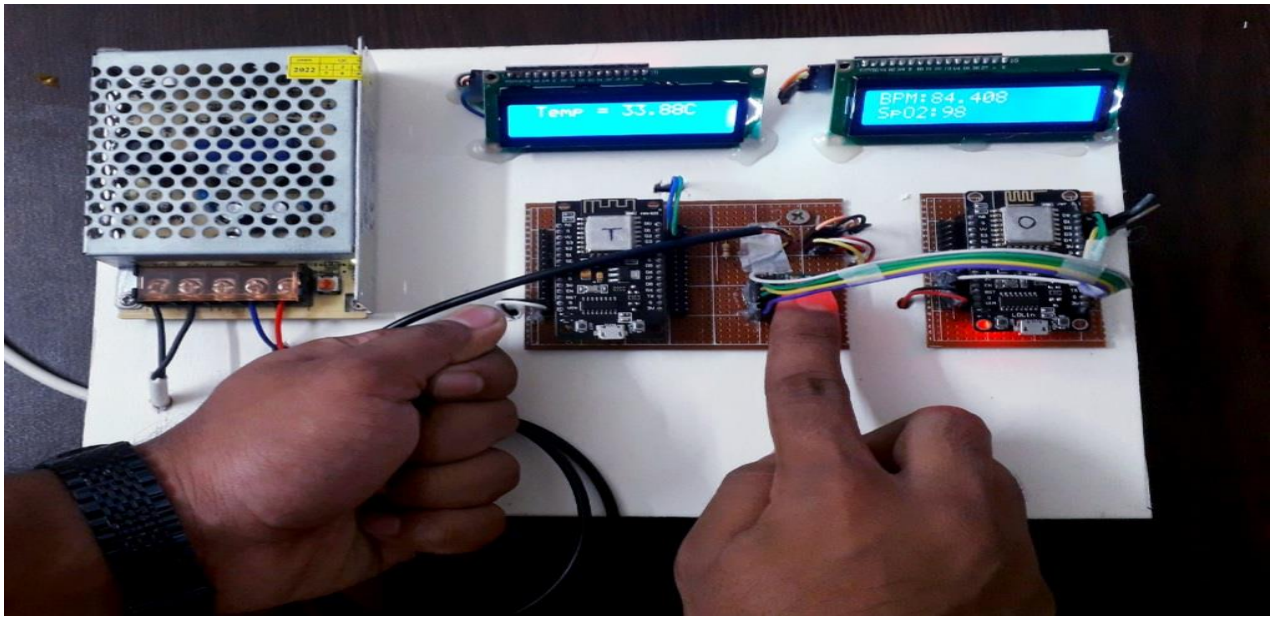


Figure 4.18: Device Info

Chapter 5: Result

The system consists of two parts: the hardware and the mobile application. Both parts are essential for the system, and users can obtain results from both. This system was built by the circuit diagram and flow chart as shown in Figures 2 and 3. Figure 18 shows the system prototype. Here, the Arduino Uno, LCD display, buzzer, pulse sensor, temperature sensor, node MCU, and Bluetooth module were added to implement the system. In the display of the system, the measured pulse rate, SpO2 level, and body temperature are shown; if the measured pulse rate and SpO2 level go out of range, the buzzer rings. This system prototype is simple and easy to use. It may be easily moved from one location to another because it is a lightweight prototype. Because all of the components are precisely positioned, the overall result is satisfactory.



Chapter 6: Discussions and Conclusions

6.1 Discussions

This paper presents the design and implementation of a health monitoring system using the Internet of Things (IoT). In present days, with the expansion of innovations, specialists are always looking for innovative electronic devices for easier identification of irregularities within the body. IoT-enabled technologies enable the possibility of developing novel and noninvasive clinical support systems. This paper presents a health care monitoring system. In particular, COVID-19 patients, high blood pressure patients, diabetic patients, etc., in a rural area in a developing country, such as Bangladesh, do not have instant access to health or emergency clinics for testing. Buying individual instruments or continuous visitation to hospitals is also expensive for the regular population. The system we developed will measure a patient's body temperature, heartbeat, and oxygen saturation (SpO2) levels in the blood and send the data to a mobile application using Bluetooth. The mobile application was created via the Massachusetts Institute of Technology (MIT) inventor app and will receive the data from the device over Bluetooth. The physical, logical, and application layers are the three layers that make up the system. The logical layer processes the data collected by the sensors in the physical layer. Media access management and intersensor communications are handled by the logical layer. Depending on the logical layer's processed data, the application layer makes decisions. The main objective is to increase affordability for regular people. Besides sustainability in the context of finance, patients will have easy access to personal healthcare. This paper presents an IoT-based system that will simplify the utilization of an otherwise complicated medical device at a minimum cost while sitting at home. A 95 percent confidence interval with a 5 percent maximum relative error is applied to all measurements related to determining the patient's health parameters. The use of these devices as support tools by the general public in a certain situation could have a big impact on their own lives.

6.2 Conclusions

The Internet of Things is considered now as one of the feasible solutions for any remote value tracking especially in the field of health monitoring. It facilitates that the individual prosperity parameter data is secured inside the cloud, stays in the hospital are reduced for conventional routine examinations and most important that the health can be monitored and

disease diagnosed by any doctor at any distance. In this paper, an IoT based health monitoring system was developed. The system monitored body temperature, pulse rate and room humidity and temperature using sensors, which are also displayed on a LCD. These sensor values are then sent to a medical server using wireless communication. These data are then received in an authorized personals smart phone with IoT platform. With the values received the doctor then diagnose the disease and the state of health of the patient.

6.3 Advantages

- Data By making use of more inserted data, we can treat patients more efficiently. ...
- Tracking and monitoring Through the use of IoT, we can keep track of data more efficiently. ...
- Connectivity of devices. ...
- Time Through IoT, we can save a lot of time for diagnosis and treatment. ...
- Research.

6.4 Challenges

There are three basic challenges we face during the project implementation

Pulse Sensor

The data that is retrieved from the pulse sensor can give some error reading sometimes. Due to unavailability of better pulse sensor in our country we had to order it from USA. And the data comparison of the local and foreign sensor was noticeable. This delayed our implementation of the project in some ways as pulse sensor is an integral part of this project. A pulse wave is the change in the volume of a blood vessel that occurs when the heart pumps blood, and a detector that monitors this volume change is called a pulse sensor. The pulse felt on the neck is called the carotid pulse. When felt on the groin, it is called the femoral pulse. The pulse at your wrist is called the radial pulse. The pulse rate is a measurement of the heart rate, or the number of times the heart beats per minute. As the heart pushes blood through the arteries, the arteries expand and contract with the flow of the blood. Taking a pulse not only measures the heart rate, but also can indicate the following: Heart rhythm.

6.5 Applications

Future works

- Integrating more sensors for more specific data acquisition and analysis.
- Will be applicable in army services in active situation.
- Will be used to provide health service to rural areas in affordable price.
- Huge database will be built for doctors to diagnose people from different areas and cultures.

Our project can be considered as platform to develop in the field of IoT on the health sector. In developing countries like ours, this kind of innovative and cost effective project can improve the future of technology. So, we are looking forward to implement the project in order to make an impact in the new era of technology.

6.6 Future works

Multiple parameters like Blood pressure, retinal size, age and weight can be included as controlling parameters in the future.

This system also developed by using advanced GSM and GPRS technology in future.

To combat such situations, medical IoT devices and applications can gather vital data and transfer it to doctors and health personnel for real-time tracking. Also, these mobile applications and IoT devices can also send notifications regarding a patient's critical conditions irrespective of place, and time. Additionally, this knowledge helps EMTs choose the most appropriate institution for a patient. Similar technology in the form of an IoT-enabled badge that tracks blood pressure, pulse, body temperature, and breathing rate may be used in emergency medical treatment in the future of IoT in healthcare

References:

[1] Madhuri G.Hiremath1, Veeresh Pujari2, Dr. Baswaraj Gadgay. “IOT Based Energy Monitoring & Control Devices”, International Journal for Research in Applied Science & Engineering Technology (IJRASET), Volume 5 Issue VI, June 2017

[2] Rajiv .K. Bhatia and Varsha Bodade,"Smart Grid Security and Privacy: Challenges, Literature Survey and Issues", International Journal of Advanced Research in Computer Science and Software Engineering, Volume 4, Issue 1, January 2014.

[3] Mitali Mahadev Raut, 2Ruchira Rajesh Sable, 3Shrutika Rajendra Toraskar. “Internet of Things(IOT) Based Smart Grid”, International Journal of Engineering Trends and Technology (IJETT) , Volume 34 Number 1- April 2016

[4] Stamatis Karnouskos,"The cooperative Internet ofThings enabled Smart Grid", SAP Research,Vincenz-PriessnitzStrasse 1, D-76131, Karlsruhe, Germany,2013.

[5] https://en.wikipedia.org/wiki/Liquid-crystal_display

[6] https://www.eeweb.com/companyblog/allegro_microsystems/hall-effect-linear-currentsensor-ic

[7] http://esp8266.net

Appendix:

Program:

```
#define BLYNK_PRINT Serial


#include <ESP8266_Lib.h>
#include <BlynkSimpleShieldEsp8266.h>
#include <SoftwareSerial.h>
#include "EmonLib.h"             // Include Emon Library
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

// for esp8266
SoftwareSerial EspSerial(7, 8); // RX, TX

char auth[] = "keN0TNT4BgDsSMyJYhqxqu65EskLTKyn";
char ssid[] = "student";
char pass[] = "iotstudent";

#define ESP8266_BAUD 9600
ESP8266 wifi(&EspSerial);

EnergyMonitor emon1;             // Create an instance

float involt;


LiquidCrystal_I2C lcd(0x27, 20, 4);




// define for Frequency
int input = 12;
int high_time;
int low_time;
float time_period;
float freq;
float frequency;


float power;
float supplyVoltage;

//variable for current sensor
#define acs712 A1
```

```
int acspower = 5;
float amperage = 0.00;
float mean = 0.0;


//current sample count
long lastsample = 0;
long samplesum = 0;
int sampleCount = 0;
float vpc = 4.8828125;
float value;

//variable for current sensor2
#define acs7121 A2
int acspower1 = 5;
float amperage1 = 0.00;
float mean1 = 0.0;


//current sample count
long lastsample1 = 0;
long samplesum1 = 0;
int sampleCount1 = 0;
float vpc1 = 4.8828125;
float value1;


//flag
int flag = 0;
int flag1 = 0;

void setup() {

  pinMode(input, INPUT);
  pinMode(2, OUTPUT);
  pinMode(3, OUTPUT);
  digitalWrite(2, HIGH);
  digitalWrite(3, HIGH);

  Serial.begin(9600);
  lcd.begin();
  lcd.backlight();

  emon1.voltage(0, 640.25, 1.7); // Voltage: input pin, calibration 234.26, phase_shift

  EspSerial.begin(ESP8266_BAUD);
  delay(10);

  Blynk.begin(auth, wifi, ssid, pass);
}
```

```
void loop() {

  emon1.calcVI(20, 1000);        // Calculate all 2000. No.of half wavelengths (crossings),
time-out
  //emon1.serialprint();           // Print out all variables (realpower, apparent power, Vrms,
Irms, power factor)



  float supplyVoltage   = emon1.Vrms;             //extract Vrms into Variable


  high_time = pulseIn(input, HIGH);
  low_time = pulseIn(input, LOW);

  time_period = high_time + low_time;
  frequency = 1000 / time_period;



  if (supplyVoltage < 100.00)
  {
   supplyVoltage = 0.0;
  }


  if (frequency >= 60.0)
  {
   frequency = 0.0;
  }


  // current measure1

  for (sampleCount = 0; sampleCount <= 1000; sampleCount ++)
  {
   samplesum += (analogRead(acs712) - 508);
   //    sampleCount++;
   delay(1);
  }

  mean = samplesum / 1000;
  // Serial.println(mean);



  amperage = (mv / 16);
```

```
samplesum = 0;

  Serial.println(" amp: " + String(amperage));


// current measure2

for (sampleCount1 = 0; sampleCount1 <= 1000; sampleCount1 ++)
{
  samplesum1 += (analogRead(acs7121) - 508);
  //    sampleCount1++;
  delay(1);
}

mean1 = samplesum1 / 1000;

}


amperage1 = (mv1 / 16);



samplesum1 = 0;

Serial.println(" amp1: " + String(amperage1));




Serial.println(supplyVoltage);
Serial.println(frequency);
Serial.println("");

Blynk.virtualWrite(V0, supplyVoltage);
Blynk.virtualWrite(V1, frequency);
Blynk.virtualWrite(V2, amperage);
Blynk.virtualWrite(V3, amperage1);


delay(1000);
lcd.clear();
lcd.setCursor(0, 0);
lcd.print("Voltage:");
lcd.print(supplyVoltage);
lcd.print("V");

lcd.setCursor(0, 1);
lcd.print("Frequency:");
```

```
  lcd.print(frequency, 1);
  lcd.print("Hz");

  lcd.setCursor(0, 2);
  lcd.print("System 1:");
  lcd.print(amperage);
  lcd.print("A");

  lcd.setCursor(0, 3);
  lcd.print("System 2:");
  lcd.print(amperage1);
  lcd.print("A");


}
```