

FOG COMPUTING BASED IOT FOR HEALTH CARE MONITORING

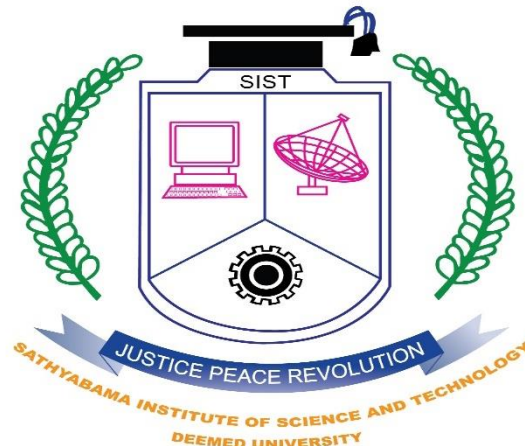
Submitted in partial fulfilment of the requirements
for the degree of
Bachelor of Technology
in
Electronics and Communication Engineering

By
Ravi Kumar. K (38130184)

Naresh. R (38130185)

Under the guidance of

Dr. V. J. K. KISHOR SONTI, M. Tech.. Ph.D.,



Department of Electronics and Communication Engineering

SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)

Accredited with Grade "A" by NAAC

JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI - 600 119

MAY-2022



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE

www.sathyabama.ac.in

DEPARTMENT OF Electronics and Communication Engineering

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the Bonafide work of **RAVI KUMAR K [38130184]** , **NARESH R [3810185]** who carried out the project entitled **"Fog Computing Based IOT For Health Monitoring"** under my / our supervision from Nov 2021 to May 2022

Dr. V. J. K. KISHOR SONTI, M.Tech.. Ph.D.,	
Internal Guide (Name in Capital letters with Signature)	External Guide (if Applicable) (Name in Capital letters with signature)

Head of the Department

DR. T. RAVI, M.E., Ph.D.

Submitted for Viva voce Examination held on _____

Internal Examiner

Dr. V. J. K. KISHOR SONTI, M.Tech. Ph.D.,

External Examiner

(Name in Capital letters with signature)

DECLARATION

I **RAVI KUMAR K [38130184]** and **NARESH RAYAPATTI [38130185]** hereby declare that the Project Report entitled “**FOG COMPUTING BASED IOT FOR HEALTHCARE MONITORING**” done by me under the guidance of **Dr. V. J. K. KISHOR SONTI, M.Tech.. Ph.D., Assoc Professor, department of ECE. (Internal)** and _____ (External) at **SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY, CHENNAI ,TAMILNADU ,600119.** is submitted in partial fulfilment of the requirements for the award of Bachelor of Engineering / Technology degree in **ELECTRONICS AND COMMUNICATION ENGINEERING.**

DATE: 30-04-2022

PLACE: CHENNAI

SIGNATURE OF THE CANDIDATE:

1. Ravi Kumar K [38130184].

2. R. Naresh [38130185].

ACKNOWLEDGEMENT

We are pleased to acknowledge our sincere thanks to Board of Management of **SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY** for their kind encouragement in doing this project and for completing it successfully. We are grateful to them.

We convey our thanks to **DR. N. M. NANDHITHA, M.E., Ph.D. Dean, School of Electrical and Electronics Engineering** and **DR. T. RAVI, M.E., Ph.D. Head of the Department, Department of Electronics and Communication Engineering** for providing us necessary support and details at the right time during the progressive reviews.

We would like to express our sincere and deep sense of gratitude to our Project Guide

Dr. V. J. K. KISHOR SONTI, M.Tech. Ph.D., Assoc Professor, department of ECE. for his valuable guidance, suggestions and constant encouragement paved way for the successful completion of the project work.

We wish to express our thanks to all teaching and Non-teaching staff members of the **Department of Electronics and Communication Engineering** who were helpful in many ways for the completion of the project.

We express our gratitude to our parents for their constant encouragement and support for the completion of the project.

ABSTRACT:

Advances in the information and communication technologies have led to the emergence of Internet of Thing(IoT).IoT allows many physical devices to capture transmit data, through the internet, providing more data interoperability methods. Nowadays IoT plays an important role not only in communication, but also in monitoring, recording, storage and display. Hence the latest trend in Healthcare communication method using IoT is adapted. Monitored on a continual basis, aggregated and effectively analyzed - such information can bring about a massive positive transformation in the field of healthcare. Our matter of concern in this project is to focus on the development and implementation of an effective healthcare monitoring system based on IoT. The proposed system monitors the vital health parameters and transmits the data through a wireless communication, which is further transferred to a network via a Wi-Fi module. The data can be accessed anytime promoting the reception of the current status of the patient. In case any abnormal behavior or any vital signs are recognized, the caretaker, as well as the doctors are notified immediately through a message service or an audio signaling device (buzzer). In order to design an efficient remote monitoring system, security plays an important part. Cloud computing and password protected Wi-Fi module handles authentication, privacy and security of patient details by allowing restricted access to the database. Hence the system provides quality healthcare to all. This paper is a review of Healthcare Monitoring system using IoT

Wireless sensor networks (WSNs) are widely used in the area of health informatics. Wireless and wearable sensors have become prevalent devices to monitor patients at risk for chronic diseases. This helps ascertain that patients comply by the treatment plans and also safeguard them during sudden attacks. The amount of data that are gathered from various sensors is numerous. In this paper, we propose to use fog computing to help monitor patients suffering from chronic diseases such that the data are collected and processed in an efficient manner. The main challenge would be to only sort out context-sensitive data that are relevant to the health of the patient. Just having a simple sensor-to-cloud architecture is not viable, and this is where having a fog computing layer makes a difference. This increases the efficiency of the entire system, as it not only reduces the amount of data that is transported back and forth between the cloud and the sensors but also eliminates the risk that a data center failure bears with it. We also analyze the security and deployment issues of this fog computing layer.

TABLE OF CONTENTS

TITLE	PAGE NO
ABSTRACT	v
LIST OF FIGURES	ix
LIST OF TABLES	xi

CHAPTER NO	TITLE	PAGE NO
1.	INTRODUCTION	
	1.1 IOT	1
	1.2 HEALTHCARE	1
	1.3 FOG COMPUTING	2
2.	LITERATURE REVIEW	3
3.	AIM AND SCOPE OF THE PRESENT INVESTIGATION	
	3.1 RESEARCH IN FOG COMPUTING	7
	3.2 MOTIVATION	8
	3.3 KEY OBJECTIVES OF HEALTHCARE SYSTEM	8
	3.4 CHALLENGES IN HEALTH MONITORING SYSTEM	8
	3.5 PROPOSED RESEARCH	9
	3.6 SENSORS TIER	10
	3.7 FOG COMPUTING	
	3.7.1 WHAT IS THE NEED FOR FOG COMPUTING	11
	3.7.2 LIMITATIONS OVERCOMES IN FOG	11

3.7.3 APPLICATIONS OF FOG	12
4. MATERIALS AND METHODS ALGORITHMS USED	
4.1 COMPONENTS REQUIRED	13
4.1.1 HARDWARE	13
4.1.2 SOFTWARE	13
4.2 BLOCK DIAGRAM	14
4.3 MODULE EXPLANATION	
4.3.1 NODE MCU	15
4.3.2 NODEMCU ESP2866 SPECIFICATION	15
4.3.3 GENERAL PURPOSE INPUT/OUTPUT	17
4.3.4 SECURE DIGITAL INPUT/OUTPUT	17
4.3.5 SERIAL PERIPHERAL INTERFACE	18
4.3.6 GENERAL SPI (MASTER/SLAVE)	18
4.3.7 I2C INTERFACE	18
4.3.8 UNIVERSAL ASYNCHRONOUS RECEIVER	19
4.3.9 PULSE WIDTH MODULATION	20
4.4 FUNCTIONAL OVERVIEW	
4.4.1 ESP8266 SOFTWARE COMPILING DOWNLOADING	21
4.4.2 ESP8266 FLASH SOFTWARE DOWNLOADING	22
4.5 AURDINO IDE + NODE MCU	
4.5.1 POWER PINS	23
4.5.2 NODE MCU COMPATIBILITY WITH ARDUINO IDE	25
4.5.3 INSIGHT INTO ESP8266 NODE MCU	26
4.6 ESP-12E MODULE	26
4.7 POWER REQUIREMENT	
4.7.1 INSTALLING THE ESP8266 CORE ON WINDOWS	28
4.8 ARDUINO EXAMPLE BLINK	30
4.9 DHT 11	33
4.9.1 WIRING- CONNECTING DHT11 SENSOR TO ESP8266	34

4.9.2	INSTALLING DHT SENSOR LIBRARY	35
4.10	ACCESSING WEB SERVER	38
4.11	I2C LCD	39
4.11.1	16*2 I2C DISPLAY	39
4.11.2	PARTS REQUIRED	40
4.11.3	WIRING LCD TO ESP32	41
4.11.4	WIRING THE LCD TO THE ESP8266	42
4.12	PREPARING THE ARDUINO IDE	44
4.12.1	ARDUINO IDE WITH ESP32	44
4.12.2	ARDUINO IDE WITH ESP8266	44
4.12.3	DISPLAY STATIC TEXT ON THE LCD	45
4.12.4	HOW THE CODE WORKS	45
4.12.5	DISPLAYING SCROLLING TEXT ON LCD	47
4.13	WORKING	48
4.13.1	PULSE SENSOR	48
4.13.2	MEMS SENSOR	50
4.13.3	BUZZERS	51
5.	RESULT AND DISCUSSION	
5.1	SIMULATION OF THE FOG ENVIRONMENT	52
5.2	THINGSPEAK	53
5.2.1	WHAT IS THINGSPEAK	53
5.2.2	ACCOUNT SETTINGS	53
6.	CONCLUSION AND FUTURE WORK	
6.1	CONCLUSION	58
6.2	FUTURE WORK IN HEALTH CARE	58
	REFERENCES	59

LIST OF FIGURES

FIGURE NO	TITLE	PAGE NO
3.1	FOG BLOCK DIAGRAM	9
3.2	FOG LAYERS	10
4.1	BLOCK DIAGRAM	14
4.2	NODE MCU	15
4.3	NODE MCU	16
4.4	ARDUINO+NODEMCU0-1	25
4.5	ARDUINO + NODEMCU1	26
4.6	ESP-12E1	27
4.7	POWER REQUIREMENT	28
4.8	SETUP 1	29
4.9	SETUP 1.2	30
4.10	ARDUINO EXAMPLE :BLINK1	31
4.11	ARDUINO EXAMPLE :BLINK2	32
4.12	DHT11 1	33
4.13	DHT11 2	34
4.14	DHT11 2.1	35
4.15	ARDUINO SETUP 1	36
4.16	ARDUINO SETUP 2	37
4.17	ACCESSING THE WEB SERVER1	38
4.18	I2C LCD 1	39
4.19	I2c LCD 2	40
4.20	I2C LCD 3	41
4.21	I2C TO NODE MCU 1	41

4.22	I2C TO NODE MCU 2	43
4.22.1	SERIAL MONITOR OUTPUT	45
4.23	LCD DISPLAY 1	45
4.24	SCROLLING TEXT	47
4.25	PULSE SENSOR 1	48
4.26	PULSE SENSOR 2	49
4.27	MEMES SENSOR 1	50
4.28	BUZZERS 1	51
5.1	API KEY SETTINGS 1	54
5.2	API KEY SETTINGS 2	55
5.3	SERIAL MONIOTR RESULTS	55
5.4	RESULTS STORED IN THE THINGSPEAK	56
5.5	CONNECTED PROJECT	57
5.6	MEMS SENSOR OUTPUT	57

LIST OF TABLES

TABLE NO	TITLE	PAGE NO
4.1	NODE MCU DEVELOPMENT BOARD	
	PINOUT CONFIGURATION	16
1.1	PIN DEFINITIONS OF SDIOS	17
1.2	PIN DEFINITIONS OF SPIS	18
1.3	PIN DEFINITIONS OF I2C	19
1.4	PIN DEFINITIONS OF UART	19
1.5	PIN DEFINITIONS OF PWM	20
1.6	I2C VS ESP32	42
1.7	LCD VS ESP8266	43

Keywords

- Fog computing
- Cloud computing
- Internet of things
- Healthcare
- Taxonomy
- Survey

CHAPTER 1

INTRODUCTION:

With each passing day, the way the world interacts changes. The past few years have seen an increase in the usage of cloud computing for a huge number of applications. But now, the cloud does not cater to just simple applications and technological needs; the Internet of things (IoT) is proving to be the next technological trend when it comes to cloud computing. In healthcare applications, wireless sensor networks (WSN) have started playing a huge role in the way patients are being monitored. Wireless sensors in the form of wireless wearable accessories or devices are attached to a patient such that this information can be used for the monitoring process. The sensors can be of various forms and sizes [1, 2] as long as they are relevant to the need.

1.1 IOT

The wireless sensor networks generate a huge amount of data. These data that have been collected from all the devices connected to the network may be useful as well as redundant. All these unprecedented amounts of data can overwhelm the data storage systems and the data analysis applications. The weeding out of irrelevant data has to be a context-sensitive process. Hence, the sensors would have to send the data collected to computing devices that are capable of performing tasks of analysis, aggregation, and storage. In many cases, each patient requires a high number of sensors, and hence, creating an infrastructure dedicated to an individual becomes inefficient. Hence, IoT provides an alternative approach in which sensor devices are used in a common infrastructure. These sensor devices can then forward the data to a cloud server.

1.2 HEALTHCARE

For many healthcare applications, having a simple sensor-to-cloud architecture is not viable, especially due to the fact that most hospitals would not prefer patient data to be stored outside [3]. Also, there is always the bleak case of there being a network failure or a data center failure, which puts patients' health at risk. This is where fog computing aids healthcare applications.

Using only the cloud may cause delay during the transfer of data from the sensors to the cloud and the cloud to the hospitals or personal physicians. In healthcare, we have emergency response systems that require real-time operations in which efficiency and time play an important part. Hence, transfer of such immense amounts of data back and forth is not an efficient option not only due to latency issues but also

due to security. The risks involved here are not only infringement of data but also risks to the health of patients. Hence, the classical centralized cloud computing architecture has to be extended to a distributed one. A distributed architecture refers to tasks being divided and then offloaded to more than one node.

In edge computing, the main computationally intensive operations are performed at the edge of the network instead of holding it in the cloud or on a centralized data warehouse. Edge computing uses computing resources near IoT devices for local storage and preliminary data processing. According to Cisco [5], by 2020, 50 billion devices will be connected to the Internet. Hence, edge computing will also require greater flexibility in order to manage this huge influx of devices. Edge devices cannot handle multiple IoT applications competing for their limited resources, which results in resource contention and increases processing latency. A distributed feature added to this will help in scalability and reduces the risk of exposure of data and hence increases security and eliminates most of the privacy concerns [6].

1.3 FOG COMPUTING

A fog computing layer integrates edge devices with cloud resources and hence extends the existing cloud infrastructure [7]. In this architecture, the application resides not only in the cloud but also in the devices closest to the patients and the infrastructure components between them. The term infrastructure refers to access points, gateways, and routers. The main objective of healthcare applications is to provide constant supervision on the health of a patient. Implementing this fog layer provides for the successful fulfillment of this requirement. Also, due to the fact that data are stored in data centers in the cloud, the problem of infrastructure, maintenance, upgrades, and costs is solved. In this paper, we strive to improve current healthcare systems by implementing a context-sensitive fog computing environment. We discuss the various computational tasks involved in healthcare that will be performed in the fog layer, the cloud, or at the user devices and sensors. The security of the information passing the system will be improved as the exposure of data is limited due to the fact that it does not have to travel to and fro in the network. The services that the cloud performs can be distributed to other nodes in such a way that the overhead time taken for data to go back and forth from the cloud to devices is compensated or reduced. Distributing the services among various nodes overcomes some of the challenges that previous healthcare monitoring applications have faced.

CHAPTER 2

LITERATURE SURVEY:

- [1] D. K. Rathore A. U. (2013). Wireless patient health monitoring system. International Conference on Signal Processing and Communication (ICSC), (pp. 415-418). Noida. doi:10.1109/ICSPCom.2013.6719824
- [2] D. Borthaku H. D. (2017). Smart fog: Fog computing framework for unsupervised clustering analytics in wearable Internet of Things. IEEE Global Conference on Signal and Information Processing (GlobalSIP), (pp. 472-476). Montreal, QC. doi:10.1109/GlobalSIP.2017.8308687
- [3] S. Durga, R. N. (2019). Survey on IOT and Deep Learning Algorithms used in Internet of Things (IoT) Healthcare., (pp. 1018-1022). Erode, India. doi:10.1109/ICCMC.2019.8819806
- [4] Nakashima N, N. M. (2019). Recommended configuration for personal health records by standardized data item sets for diabetes mellitus and associated chronic diseases. 10, pp. 868-875. J Diabetes Investig. doi:10.1111/jdi.13043
- 5] www.kaggle.com/uciml/pima-indians-diabetes-database.
- [6]Shakir, et.al produced the list of correlation between the attributes of both Fog and Cloud Computing which may differ in outline, devices, arrangements, and administrations for associations and clients. This comparison resulted that the performance of FC was considered well in both data processing service and low bandwidth consumption.
- [7] Mohammed, et.al explained that the FC as the complementary approach to Cloud Computing. As today's research world in the urge of designing the product-service-system instead of product, as well as in the context of Industry 4.0, the system has to deal with large amount of time-critical data and in the need of immediate analysis report. In this paper, they elaborated the benefits of Cloud and FC in same concert.
- [8]Sourav, et.al Cloud-FC from the perspective of architectures, applications, and their security challenges. They discussed the three layered architecture for FC and proposed the novel architectures like Energy Lattices, UXFog, MediFog, Connected Parking System, and FoAgro in order to utilize the features of FC and be benefited

[9]. P.P. Ray: In this author introduce, Firstly the background and definition of IoT. Secondly, thorough discussions on fundamentals behind IoT architectures are elaborated. Next, several key domains where IoT based research works are currently going on are visited. Afterward, detailed analyses of the research challenges are mentioned. Resulting graph attains the state of-the-art research based motives on the aforementioned domains. A novel concept-„lo” is also proposed that is based on various theoretical nomenclature and external inputs. Different from other IoT survey papers, a main contribution of this paper is that it focuses on area specific architectures of IoT applications and highlights the challenges and possible research opportunities for future IoT researchers who would work in architectural as well as in IoT as a whole.[1]

[10]. Sinan T. Shukur et al: In this paper the author summarizes, the emergence of IoT, new regulatory approaches to ensure its energy, scalability, security and privacy, human-in-the-loop, big data, etc. become necessary. The IoT revolution is expanding connectivity via the internet and a wide range of applications (e.g. actuators, sensors and other embedded systems). This will have an effect on the quality, different life styles and the way we behave and interact with humans, machines and devices in the future. Therefore, new research challenges and problems will emerge due to the large scale device proliferation and their inter-communication. This paper gives an overview of the key issues related to the IOT services and technologies. A number of researcher challenges have been described, which are expected to become a major research trends in the next decade. A number of previous works have been analyzed, and most relevant WSN and IOT applications were presented. [11]

[11]. J. Sathish Kumar et al: In this author presented Internet of Things with architecture and design goals. They surveyed security and privacy concerns at different layers in IoTs. In addition, they identified several open issues related to the security and privacy that need to be addressed by research community to make a secure and trusted platform for the delivery of future Internet of Things. It was also discussed the applications of IoTs in real life. In future, research on the IoTs will remain a hot issue. Lot of knotty problems is waiting for researchers to deal with [3].

[12]. Pallavi Sethi et al : In this survey paper Author"s presented a survey of the current technologies used in the IoT domain as of 2016. Currently, this field is in a very nascent stage. The technologies in the core infrastructure layers are showing signs of maturity. However, a lot more needs to happen in the areas of IoT applications and communication technologies. These fields will definitely mature and impact human life in inconceivable ways over the next decade [9].

[13]. Mayra Samaniego et al: In this paper basically discussion is on the study and development of Internet of Things (IoT) applications, web and mobile, is on the increase. Applications, working with data obtained from different areas such as transportation, smart homes, health care, public services, industry and many others. Previous studies have focused on managing the obtained data. However, managing the heterogeneous resources that get that data is an area that demands more attention. This work addresses the management of resources in the Internet of Things. This is achieved by proposing a virtual-resource edge layer, which enables access and configuration to constrained physical resources. The architecture presented focuses on the use of virtual resources as a management concept and identifies different approaches in the performance evaluation on edge computing devices. Using the IoT protocol CoAP, virtual resources are exposed in the edge network. An evaluation of a Go CoAP virtual resource is presented [11].

[14]. Tuhin Borgohain et al: In this paper they have surveyed all the security flaws existing in the Internet of Things that may prove to be very detrimental in the development and implementation of IoT in the different fields. So adoption of sound security measures countering the above detailed security flaw as well as implementation of various intrusion detection systems cryptographic and stenographic security measures in the information exchange process and using of efficient methods for communication will result in a more secure and robust IoT infrastructure. In conclusion, they would like to suggest that more effort on development of secured measures for the existing IoT infrastructure before going for further development of new implementation methods of IoT in daily life would prove to be a more fruitful and systematic method

[15]. Internet of Things (IoT) and cloud computing plays a vital role in today's Tele-monitoring health system. This system keeps track of patient's physiological parameters through collection of body sensors' data using Raspberry Pi board. The patient's health card are developed by the doctors and displayed on a webpage where doctors and patients can access and communicate each other without physical presence [1]. Using cloud computing, the data can be stored, updated and accessed from anywhere in the world. It is very suitable for rural areas where medical facilities are not available. In Remote health monitoring system using IoT, Body wireless sensor Network (BWSN) is used to transmit the patients' health parameters collected through Raspberry Pi microcontroller to the physicians and caretaker wirelessly [2]. Being long range wireless technology, emergency situation of the patient's health is quickly detected and timely intervention leads to save the life of the patient. Owing to costlier healthcare and long waiting time in hospitals, the concept of in-home patient monitoring system have

been emerging in the recent years. This system collects data of various body parameters through Biosensors, wearable devices and smart textiles and it transmits the data to central node server securely through Cipher text Policy Attribute Based Encryption (CP-ABE) method. In turn, the server shares the collected data to the hospitals for further treatment. The server rings alarm to the ambulance [3] during emergency situation. It is very beneficial for elders and chronic patients who require continuous monitoring. The specialized healthcare monitoring system for elderly people is a growing need in the aging population world. This system performs basic health checkups by measuring the body parameters regularly and report the data to the doctors. The result data are then displayed as statements in a web application where doctors and patients can interact with each other.

[16].Evaluation is of two parts: 1) Qualitative interviewing and 2) Quantitative Survey. The main challenge is to make elders equipped with for growing new International Journal of Pure and Applied Mathematics Special Issue 250 technologies and to become familiarity towards Smartphone, computer, etc.

Chapter – 3

Aim and Scope of the present investigation

Related Work

3.1.Research in Fog Computing

The main applications where fog computing will prove very useful are time-sensitive applications where a huge amount of data has to be processed [8]. The following are the application areas that can benefit from fog computing.(1)Healthcare applications: this is an area in which real-time processing plays an important role. Hence, data will have to be processed very fast, and the response time will have to be as less as possible [4, 9, 10].(2)Augmented reality: for augmented reality (AR), overlaying useful information onto the physical world in real time is very important. Using fog computing will help achieve this aspect of augmented reality [11].(3)Smart utility services: utility services like electricity, water, telephone, and so on can be managed by fog computing.(4)Traffic management system: fog computing can increase the efficiency of the traffic signal system by reducing the latency. The interactions between vehicles, traffic signals, and access points can be enhanced by the fog [12].(5)Caching and processing: fog computing can also be used for improving the performance of websites [13]. Certain websites have a lot of databases and data to be processed, for example, social networking sites and library or online shopping malls. These websites can use the fog layer for caching and preprocessing its data and hence reducing time and space complexity.(6)Gaming: in the past few years, there has been a huge evolution in the gaming industry. Apart from games being computationally complex, they are mostly multiplayer these days and depend greatly on real-time processing.(7)Decentralized smart building control: similar to smart utility services, even in the case of smart building, control fog computing will play a huge role in making it more efficient and secure.

Fog computing offers enormous advantages for delay-sensitive fog-based application. Chen et al. [14] implemented a prototype of a smart gateway for the use of WSN in healthcare systems at home. The system is able to transmit reports at real time in a low power embedded system. Hong et al. [15] presented Mobile Fog, which is a programming model for Internet applications that are geographically distributed and latency-sensitive.

3.2.Motivation

The use of smart devices has fortuitously been exercised in healthcare. These days, it is commonplace to find a range of healthcare gadgets that can be used by patients at home or even worn by them. The gadgets mostly encompass sensors. These sensors generally incorporate transducers and are capable of detecting electrical, thermal optical, chemical, and other signals [10].

The main motivation of this paper is to enhance health monitoring systems that are based on IoT devices such that the information collected from WSNs are processed efficiently, and the context-sensitive data that are relevant to the patient are considered. For this, we implement a fog layer that improves the latency of health monitoring systems and enables real-time health monitoring. By this, we strive to also ensure security for the information of the patients such that patient privacy is maintained, and also, tampering of data by third party is avoided.

3.3.Key Objectives of Healthcare Systems

Some of the key objectives of healthcare systems are [16](1)improved clinical decision making,(2)reduced duplication of diagnostic testing, imaging, and history taking,(3)better medication management,(4)increased adoption of screening programs and preventive health measures.

3.4.Challenges in Implementing a Health Monitoring System

In order to successfully implement a health monitoring system, a number of challenges have to be addressed [16]:(1)Safeguarding privacy and security(2)Technical problems(3)Organizational barriers(4)Financial costs(5)Different policies(6)Training programs for practitioners and healthcare providers.

3.5. Proposed Research

In this section, we present a tri-tier architecture for context- and latency-sensitive health monitoring using cloud and fog computing. The tri-tiers consist of cloud computing, fog computing, and sensors which work in conjunction with one another. Sensors consist of wearable or not wearable devices that are attached to the patients in the form of smart watches, fitness bands, smart phones, wearable glasses, and so on. The applications used for health monitoring will have components running in the edge devices situated in the fog layer, the wearable sensors, or the cloud. The edge devices may be controlled by the cloud and the fog layers. Information will flow across this tri-tiered infrastructure.

In context-sensitive health monitoring, personalized care can be given to each patient. Context can be classified into extrinsic and intrinsic context. In case of healthcare, extrinsic context is influenced by external factors like the environment that is surrounding the patient. Environmental sensors can be used to extract the extrinsic parameters of a user, and the intrinsic context can be extracted by biosensors. Both intrinsic and extrinsic sensors provide relevant information that may be used for monitoring patients' health. But depending on the disease of the patient, the type of data that is relevant differs. Hence, maintaining context-sensitive data processing has to also be done by the fog layer.

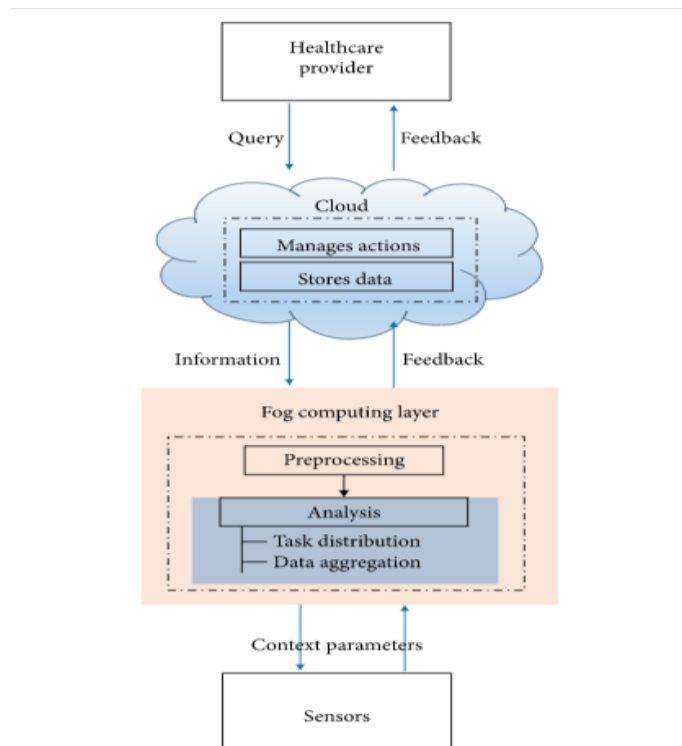


FIGURE :3.1. FOG BLOCK DIAGRAM

The schematic diagram of the proposed architecture is given in Figure 3. In this architecture of context- and latency-sensitive health monitoring systems, we ensure that all the key objectives of health monitoring and more are all accomplished.

3.6. Sensors Tier

These are the devices that gather information from the patients. These sensors gather both extrinsic and intrinsic values. Extrinsic characteristics are the temperature, location, and so on. Intrinsic characteristics are the blood pressure, blood glucose level, heartbeat, and so on that are collected by the patient's wearable sensors. The patient can also enter data into his or her smart phone, and these data will then be made available for processing. The job of the sensors is to collect all this data and send them to the fog computing layer.

3.7.FOG COMPUTING

The Distributed platforms, management Unlike traditional centers, are over geographically heterogeneous spanning domains. multiple interested proposals service mobility across platforms, and technologies preserve end-user and at content security and privacy across domains.

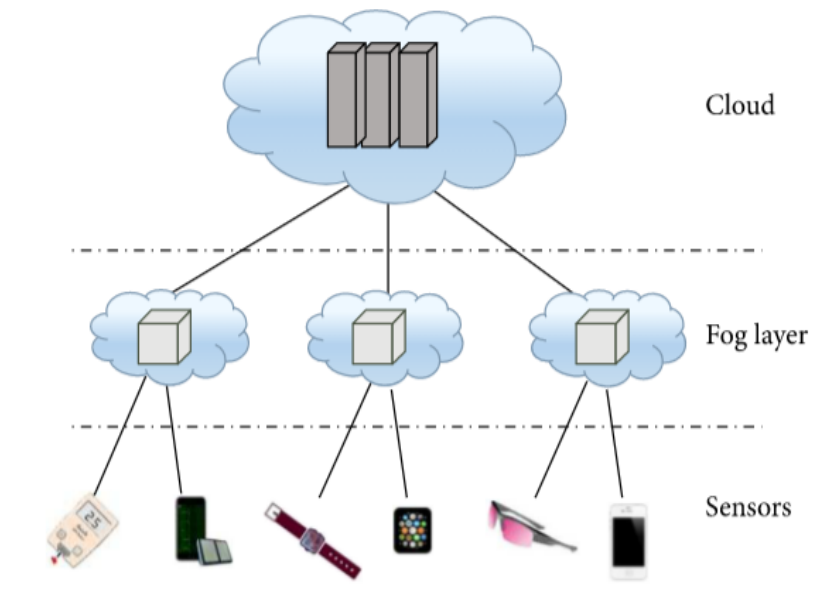


FIGURE 3.2: FOG LAYERS

IoT application which can regulate online patient health monitoring and periodical check-up for various chronic diseases. Essentially patient's criticality and abnormality condition can be easily identified in the EHMS application. IoT health wearable devices are used for collecting data from a patient in various living areas such as hospitals, homes, and work- stations, etc. Data collected are instantly stored in the local EHMS server use lot learning classification algorithms to build the training models. Collected patients information is used for prediction, review analysis, decision making, and data visualization. It can be shared by both doctor and Inpatient and Outpatient along with a patient caretaker. EHMS can provide qualitative and security services like regular monitoring, valuable collection of data, proper diagnosis analysis, and in-time patient services.

3.7.1.What is the need for fog computing?

FOG COMPUTING EXTENDS THE CLOUD COMPUTING PARADIGM TO THE EDGE OF THE NETWORK. WHILE FOG AND CLOUD USE THE SAME RESOURCES (NETWORKING, COMPUTE, AND STORAGE) AND SHARE MANY OF THE SAME MECHANISMS AND ATTRIBUTES

(virtualization, multi-tenancy) the extension is a non-trivial one in that there exist some fundamental differences stemming from the reason fog computing was developed: to address applications and services that do not fit the paradigm of the cloud

3.7.2.Limitations overcomes in fog:

- ✓ reduction in data movement across the network resulting in reduced congestion
- ✓ elimination of bottlenecks resulting from centralized computing systems
- ✓ improved security of encrypted data as it stays closer to the end user

3.7.3.Applications of fog:

- ☐ Tech giants Cisco and IBM are the driving forces behind fog computing, and link their concept to the emerging Internet of Things (IoT).
- ☐ Most of the buzz around fog has a direct correlation with the emergence of the Internet of Things (IoT)
- ☐ Used in Healthcare Monitoring in Hospitals
- ☐ Connected cars: Self-driven or self-autonomous cars are now available in the market and they produce a large amount of data.
- ☐ Smart grids and smart cities: For effectively running of systems, utility systems are using real-time data. ...
- ☐ Real-time analytics

Chapter – 4

Experimental or materials and methods; algorithms used

4.1 COMPONENTS REQUIRED

1. HARDWARE

- I. PULSE SENSOR
- II. MEMS SENSOR
- III. I2C LCD
- IV. NODE MCU[ESP8266]
- V. BUZZER
- VI. DHT11

2. SOFTWARE :

- I. Aurdino IDE WIN 7

4.2 BLOCK DIAGRAM

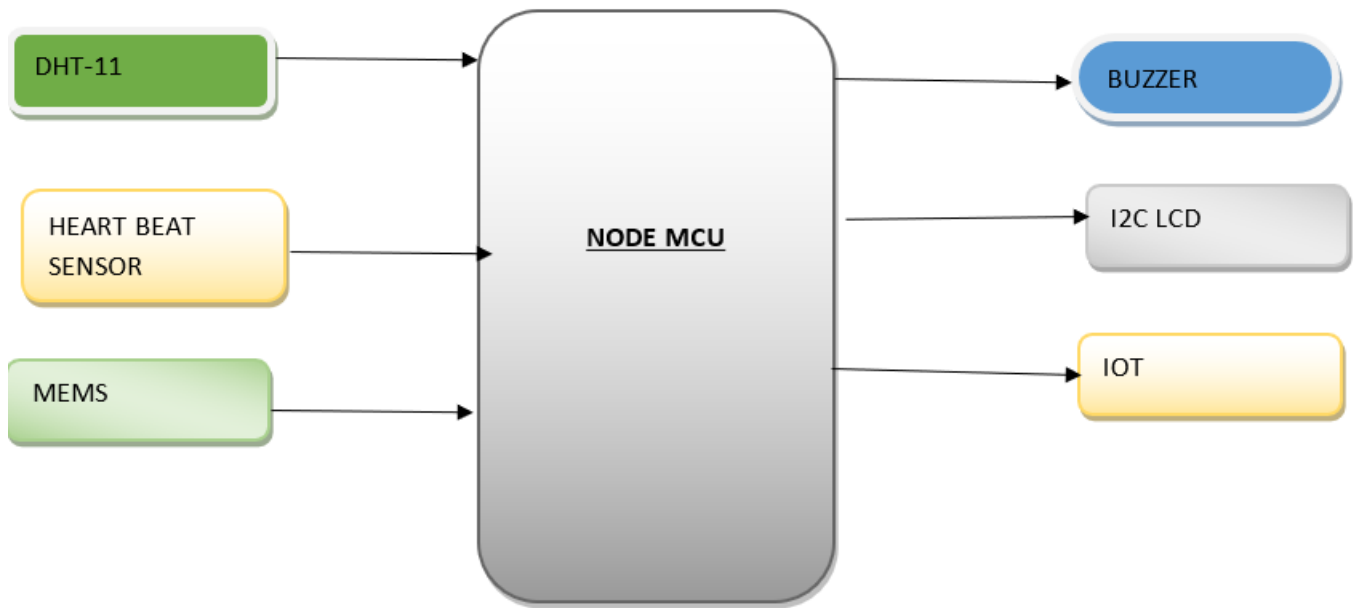


Fig:4.1.Block diagram

4.3 MODULE EXPLANATION

4.3.1. NODE MCU

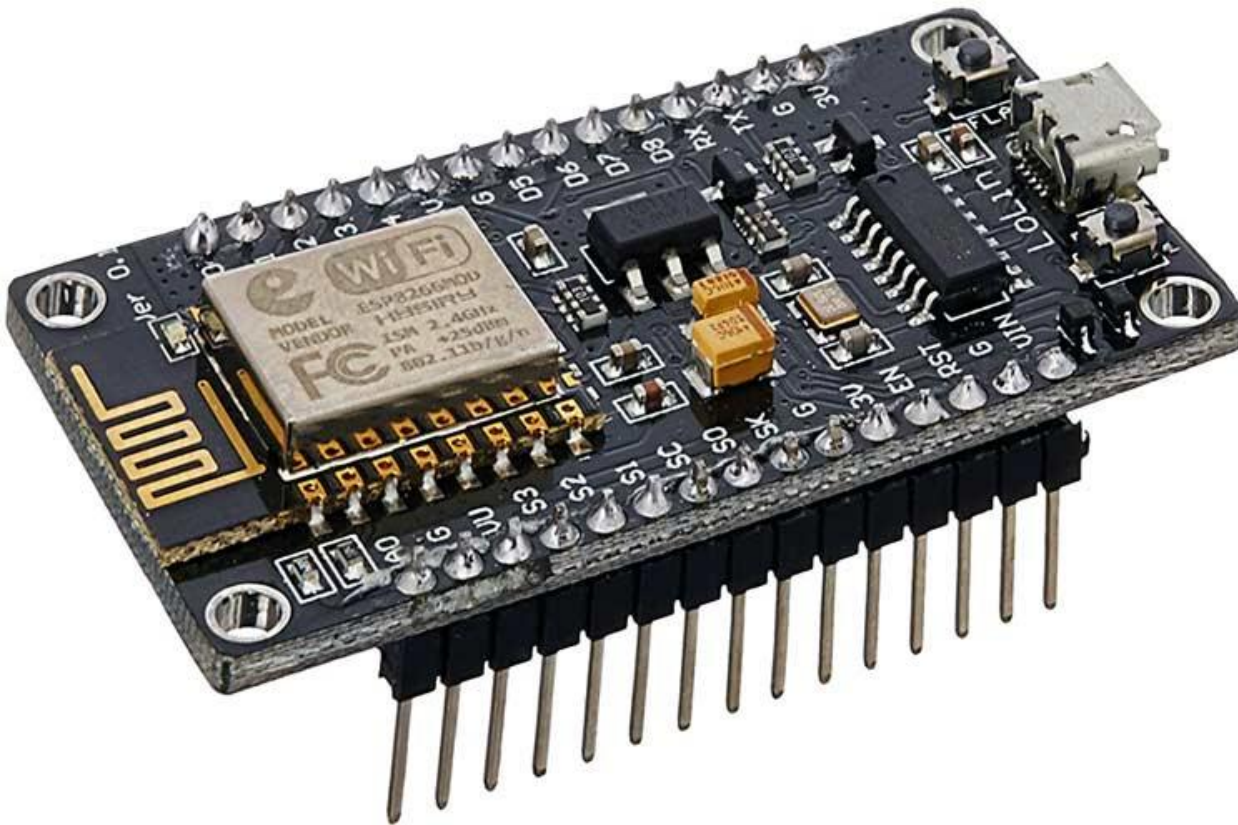


FIG 4.2: NODE MCU

Node MCU 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 Es press if Systems, and hardware which is based on the ESP-12 module.

TABLE 4.1 NODE MCU DEVELOPMENT BOARD PINOUT CONFIGURATION

Pin Category	Name	Description
Power	Micro-USB, 3.3V, GND, Vin	Micro-USB: Node MCU 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	Node MCU has 16 general purpose input-output pins on its board
SPI Pins	SD1, CMD, SD0, CLK	Node MCU has four pins available for SPI communication.

UART Pins	TXD0, RXD0, TXD2, RXD2	Node MCU has two UART interfaces, UART0 (RXD0 & TXD0) and UART1 (RXD1 & TXD1). UART1 is used to upload the firmware/program.
I2C Pins		Node MCU has I2C functionality support but due to the internal functionality of these pins, you have to find which pin is I2C.

TABLE:1 NODE MCU DEVELOPMENT BOARD PINOUT CONFIGURATION

4.3.2. Node MCU ESP8266 Specifications & Features:

- Microcontroller: Tensilica32-bit RISC CPU XtensaLX106
- 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 projects

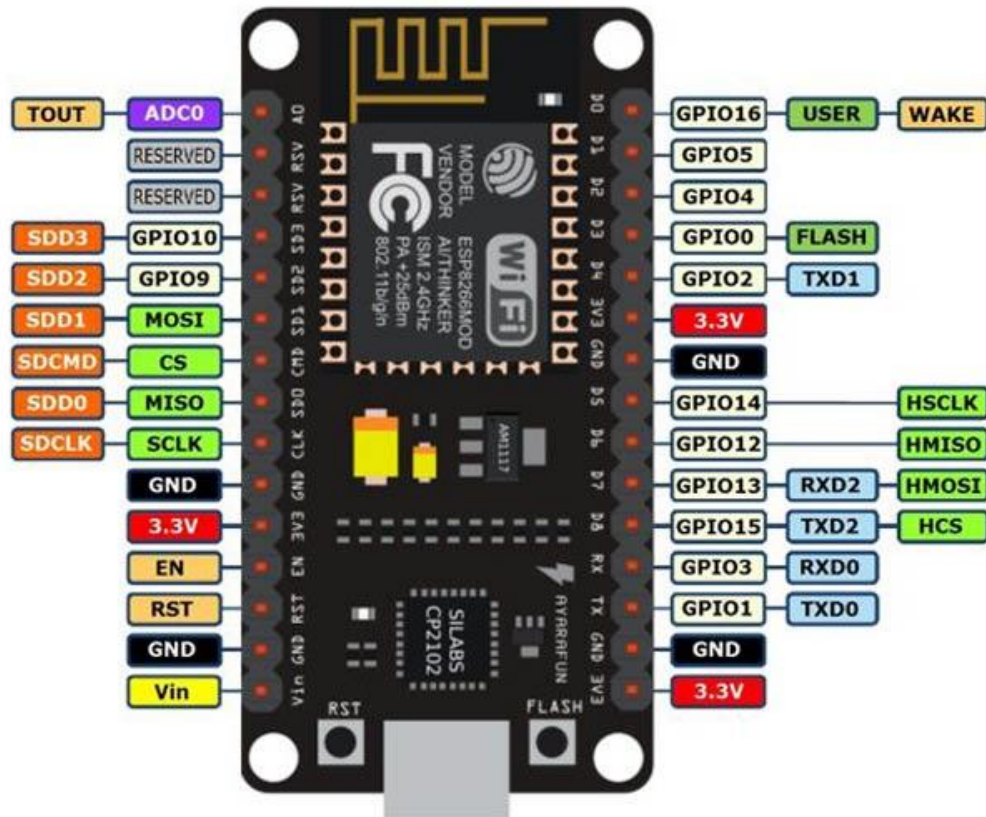


FIG.4.3. NODE MCU

4.3.3. General Purpose Input/Output Interface (GPIO):

ESP8266EX has 17 GPIO pins which can be assigned to various functions by programming the appropriate registers. Each GPIO can be configured with internal pull-up or pull-down, or set to high impedance, and when configured as an input, the data are stored in software registers; the input can also be set to edge-trigger or level trigger CPU interrupts. In short, the IO pads are bidirectional, non-inverting and tristate, which includes input and output buffer with tristate control inputs. These pins can be multiplexed with other functions such as I2C, I2S, UART, PWM, IR Remote Control, etc.

4.3.4. Secure Digital Input/Output Interface (SDIO):

ESP8266EX has one Slave SDIO, the definitions of which are described below. 4-bit 25 MHz SDIO v1.1 and 4-bit 50 MHz SDIO v2.0 are supported.

TABLE 1.1. PIN DEFINITIONS OF SDIOS

Table 1-1: Pin Definitions of SDIOs

Pin Name	Pin Num	IO	Function Name
SDIO_CLK	21	IO6	SDIO_CLK
SDIO_DATA0	22	IO7	SDIO_DATA0
SDIO_DATA1	23	IO8	SDIO_DATA1
SDIO_DATA_2	18	IO9	SDIO_DATA_2
SDIO_DATA_3	19	IO10	SDIO_DATA_3
SDIO_CMD	20	IO11	SDIO_CMD

4.3.5. Serial Peripheral Interface (SPI/HSPI):

ESP8266EX has 3 SPIs.

One general Slave/Master SPI

One Slave SDIO/SPI

One general Slave/Master HSPI

Functions of all these pins can be implemented via hardware. The pin definitions are described as below.

4.3.6. General SPI (Master/Slave):

TABLE 1.2: PIN DEFINITIONS OF SPIS

Table 1-2. Pin Definitions of SPIS

Pin Name	Pin Num	IO	Function Name
SDIO_CLK	21	IO6	SPICLK
SDIO_DATA0	22	IO7	SPIQ/MISO
SDIO_DATA1	23	IO8	SPID/MOSI
SDIO_DATA_2	18	IO9	SPIHD
SDIO_DATA_3	19	IO10	SPIWP
U0TXD	26	IO1	SPICS1
GPIO0	15	IO0	SPICS2

 Note:

SPI mode can be implemented via software programming. The clock frequency is 80 MHz at maximum.

4.3.7. I2C Interface:

ESP8266EX has one I2C used to connect with micro-controller and other peripheral equipments such as sensors. The pin definition of I2C is as below.

TABLE 1.3 PIN DEFINITIONS OF I2C

Table 1-4. Pin Definitions of I2C

Pin Name	Pin Num	IO	Function Name
MTMS	9	IO14	I2C_SCL
GPIO2	14	IO2	I2C_SDA

Both I2C Master and I2C Slave are supported. I2C interface functionality can be realized via software programming, the clock frequency reaches 100 kHz at a maximum. It should be noted that I2C clock frequency should be higher than the slowest clock frequency of the slave device.

4.3.8. Universal Asynchronous Receiver Transmitter (UART):

ESP8266EX has two UART interfaces UART0 and UART, the definitions are as below

TABLE 1.4: PIN DEFINITIONS OF UART

Table 1-6. Pin Definitions of UART

Pin Type	Pin Name	Pin Num	IO	Function Name
UART0	U0RXD	25	IO3	U0RXD
	U0TXD	26	IO1	U0TXD
	MTDO	13	IO15	U0RTS
	MTCK	12	IO13	U0CTS
UART1	GPIO2	14	IO2	U1TXD
	SD_D1	23	IO8	U1RXD

Data transfers to/from UART interfaces can be implemented via hardware. The data transmission speed via UART interfaces reaches 115200 x 40 (4.5 Mbps). UART0 can be used for communication. It supports fluid control. Since UART1 features only data transmit signal (Tx), it is usually used for printing log.

4.3.9. Pulse-Width Modulation (PWM):

ESP8266EX has four PWM output interfaces. They can be extended by users themselves. The pin definitions of the PWM interfaces are defined as below.

TABLE 1.5: PIN DEFINITIONS OF PWM

Table 1-7. Pin Definitions of PWM

Pin Name	Pin Num	IO	Function Name
MTDI	10	IO12	PWM0
MTDO	13	IO15	PWM1
MTMS	9	IO14	PWM2
GPIO4	16	IO4	PWM3

The functionality of PWM interfaces can be implemented via software programming. For example, in the LED smart light demo, the function of PWM is realized by interruption of the timer, the minimum resolution reaches as much as 44 ns. PWM frequency range is adjustable from 1000 μ s to 10000 μ s, i.e., between 100Hz and 1 kHz. When the PWM frequency is 1 kHz, the duty ratio will be 1/22727, and over 14 bit resolution will be achieved at 1 kHz refresh rate.

4.4 Functional Overview:

This protocol uses the SDIO mode of the ESP8266 to communicate with other processor's SPI hosts. The electrical interface is connected through signal line No.4, including the SCLK, MOSI, MISO and interrupt signal No.1 in the SPI protocol (note: no CS signal). Downloading the ESP8266 SDIO can be different from downloading other programs. When the ESP8266 starts, the system reads the pin shared by the SPI interface and the SDIO interface by default. Therefore, the SDIO module communication protocol should be used. The ESP8266 should start in the SDIO mode, and then, the host will start the chip in the ESP8266 RAM through the SDIO downloaded programs. The majority of the programs that directly use CPU CACHE to call FLASH can be burnt to the FLASH chip connected to the HSPI interface beforehand. Data received or sent by the ESP8266 SDIO is processed directly by the DMA module that supports linked list index. The ESP8266 can receive and send the SDIO packets efficiently without using the CPU. It does so through the address of the memory map linked list.

4.4.1. ESP8266 Software Compiling and Downloading:

- In the SDIO communication demo \esp_iot_sdk_v0.9.3_sdio_demo\ap, use the compiler to compile and generate the bin documents for downloading in order to complete the ESP8266 DEMO work.
- libmain.a in SDIO communication demo \esp_iot_sdk_v0.9.3_sdio_demo\lib is different from the version released in v0.9.3. When you use the released version of the SDK, use libmain.a in the DEMO to replace the original one. The new libmain.a will start the chip, and exchange the SPI module that reads FLASH and the HSPI mapping pin. Then, you can use DEMO to compile and generate.
- Copy eagle.app.v6.irom0text.bin in SDIO communication demo \esp_iot_sdk_v0.9.3_sdio_demo\bin to SDIO communication demoboard\XTCOM_UTIL. eagle.app.v6.irom0text.bin is all the functions of FLASH chip read directly through the SPI by CPU CACHE in the ESP8266 program.

- Run BinToArray.exe in SDIO communication demo\ . Transfer eagle.app.v6.flash.bin in SDIO communication demo \esp_iot_sdk_v0.9.3_sdio_demo\ bin to ANSI C format array. The new array will be saved in D:\. The target route of BinToArray.exe must be D:\. If there is not a D:\, you can (1) use a virtual machine with a D:\; (2) connect the device to a U disk named D:\; or (3) search online for a tool that can transfer bin to array.
- If there is a D:\, name hexarray.c in D:\ as eagle_fw.h, and define the array name as const unsigned char eagle_fw[] =..... Replace eagle_fw.h in SDIO communication demo\STM32\ Eagle_Wifi_Driver\ egl_drv_simulation\ (you can copy the array name and document name in the old eagle_fw.h, rename the hexarray.c and use it to replace the old eagle_fw.h.). Before starting the chip, write eagle.app.v6.flash.bin into the ESP8266 memory. eagle.app.v6.flash.bin should be transferred to array, and be written into the ESP8266 through STM32.
- Use the IAR platform to open EglWB.ewp.eww in SDIO communication demo\STM32\IAR\ to compile the programs.

4.4.2. ESP8266 FLASH Software Downloading :

1. Use the serial line to connect the ESP_IOT reference board and the computer, and connect them with a 5V power supply. Connect J67 to the 2 pins on the right (enable the FLASH chip in the HSPI interface), and J66 to the 2 pins on the left (disable the FLASH chip in the SPI interface). Set MTD0, GPIO0 and GPIO2 to the UART mode 0, 0,1 (up, up, down).
2. Double-click XTCOM_UTIL.exe in SDIO communication demo\XTCOM_UTIL. Click Tools ->Config Device, and choose Com interface. Baud Rate: 115200. Click Open, and you will see open Success. Click Connect, and push the H Flash board power, you will see the connection is completed.
3. Click API TEST(A)->(5) HSpiFlash Image Download, and choose eagle.app.v6.irom0text.bin in SDIO communication demo\XTCOM_UTIL. Offset: 0x40000. Click Download, and the downloading will be completed.

Use the pin header to connect the ESP_IOT reference board and the Red Dragon demo board. The details are shown below: In the Red Dragon demo board JP1: J62 pin headers in the ESP_IOT reference board (bottom-up)

GND ->1 VSS/GND

SPI_CLK ->4 SDIO_CLK

SPI_MOSI -> 5 SDIO_CMD

SPI_MISO -> 3 SDIO_DAT0

IRQ -> 2 SDIO_DAT1

The ESP_IOT reference board: change the jumper MTD0 to 1 (short the 2 pins below), GPIO0, GPIO2 random (1, x, x is the SDIO starting mode), CHIP_PD:ON (flip the switch downward). Keep jumper J66 connected to the 2 pins on the left, and jumper J67 connected to the 2 pins on the right. Connect the 5V power adapter to the ESP_IOT reference board and the Red Dragon demo board. Turn on the demo board power, download the compiled programs mentioned in Section 2.2 to STM32 in the IAR environment. Start the STM32 program, and turn on the ESP_IOT reference board power. the STM32 will write the starting program into the ESP8266, and after several seconds, it will automatically run the SDIO to return to the testing program.

4.5 Aurdino IDE + NODE MCU

4.5.1. Power pins :

There are four power pins. **VIN** pin and three **3.3V** pins.

- **VIN** can be used to directly supply the Node MCU/ESP8266 and its peripherals. Power delivered on **VIN** is regulated through the onboard regulator on the Node MCU module – you can also supply 5V regulated to the **VIN** pin
- **3.3V** pins are the output of the onboard voltage regulator and can be used to supply power to external components.

GND are the ground pins of Node MCU/ESP8266

I2C Pins are used to connect I2C sensors and peripherals. Both I2C Master and I2C Slave are supported. I2C interface functionality can be realized programmatically, and the clock frequency is 100 kHz at a maximum. It should be noted that I2C clock frequency should be higher than the slowest clock frequency of the slave device.

GPIO Pins Node MCU/ESP8266 has 17 GPIO pins which can be assigned to functions such as I2C, I2S, UART, PWM, IR Remote Control, LED Light and Button programmatically. Each digital enabled GPIO can be configured to internal pull-up or pull-down, or set to high impedance. When configured as an input, it can also be set to edge-trigger or level-trigger to generate CPU interrupts.

ADC Channel TheNode MCU is embedded with a 10-bit precision SAR ADC. The two functions can be implemented using ADC. Testing power supply voltage of VDD3P3 pin and testing input voltage of TOUT pin. However, they cannot be implemented at the same time.

UART Pins Node MCU/ESP8266 has 2 UART interfaces (UART0 and UART1) which provide asynchronous communication (RS232 and RS485), and can communicate at up to 4.5 Mbps. UART0 (TXD0, RXD0, RST0 & CTS0 pins) can be used for communication. However, UART1 (TXD1 pin) features only data transmit signal so, it is usually used for printing log.

SPI Pins Node MCU/ESP8266 features two SPIs (SPI and HSPI) in slave and master modes. These SPIs also support the following general-purpose SPI features:

- 4 timing modes of the SPI format transfer
- Up to 80 MHz and the divided clocks of 80 MHz
- Up to 64-Byte FIFO

SDIO Pins Node MCU/ESP8266 features Secure Digital Input/Output Interface (SDIO) which is used to directly interface SD cards. 4-bit 25 MHz SDIO v1.1 and 4-bit 50 MHz SDIO v2.0 are supported.

PWM Pins The board has 4 channels of Pulse Width Modulation (PWM). The PWM output can be implemented programmatically and used for driving digital motors and LEDs. PWM frequency range is adjustable from 1000 μ s to 10000 μ s (100 Hz and 1 kHz).

Control Pins are used to control the Node MCU/ESP8266. These pins include Chip Enable pin (EN), Reset pin (RST) and WAKE pin.

EN: The ESP8266 chip is enabled when EN pin is pulled HIGH. When pulled LOW the chip works at minimum power.

RST: RST pin is used to reset the ESP8266 chip.

WAKE: Wake pin is used to wake the chip from deep-sleep.

 Control Pins are used to control the Node MCU/ESP8266. These pins include Chip Enable pin (EN), Reset pin (RST) and WAKE pin.

EN: The ESP8266 chip is enabled when EN pin is pulled HIGH. When pulled LOW the chip works at minimum power.

RST: RST pin is used to reset the ESP8266 chip.

WAKE: Wake pin is used to wake the chip from deep-sleep.

4.5.2. Node MCU Compatibility with Arduino IDE:

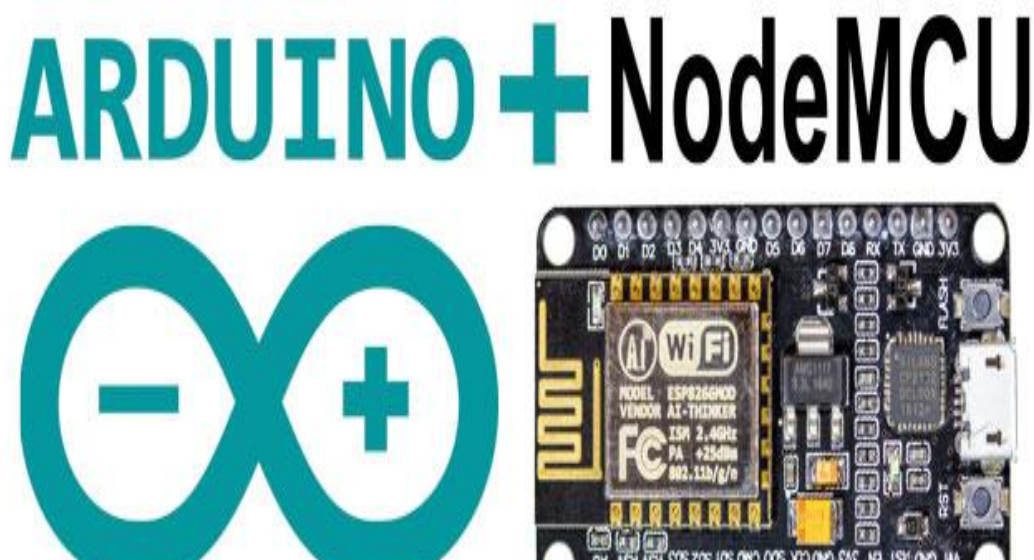


FIG 4.4.ARDUINO+NODEMCU 0-1

The Node MCU offers a variety of development environments, including compatibility with the Arduino IDE (Integrated Development Environment). The Node MCU/ESP8266 community took the IDE selection a step further by creating an Arduino add-on. If you're just getting started programming the ESP8266 or even an established developer, this is the highly recommended environment. Visit our dedicated page on setting up and configuring the Arduino IDE for a Node MCU ESP8266.

4.5.3. Insight Into ESP8266 Node MCU Features & Using It With Arduino IDE:

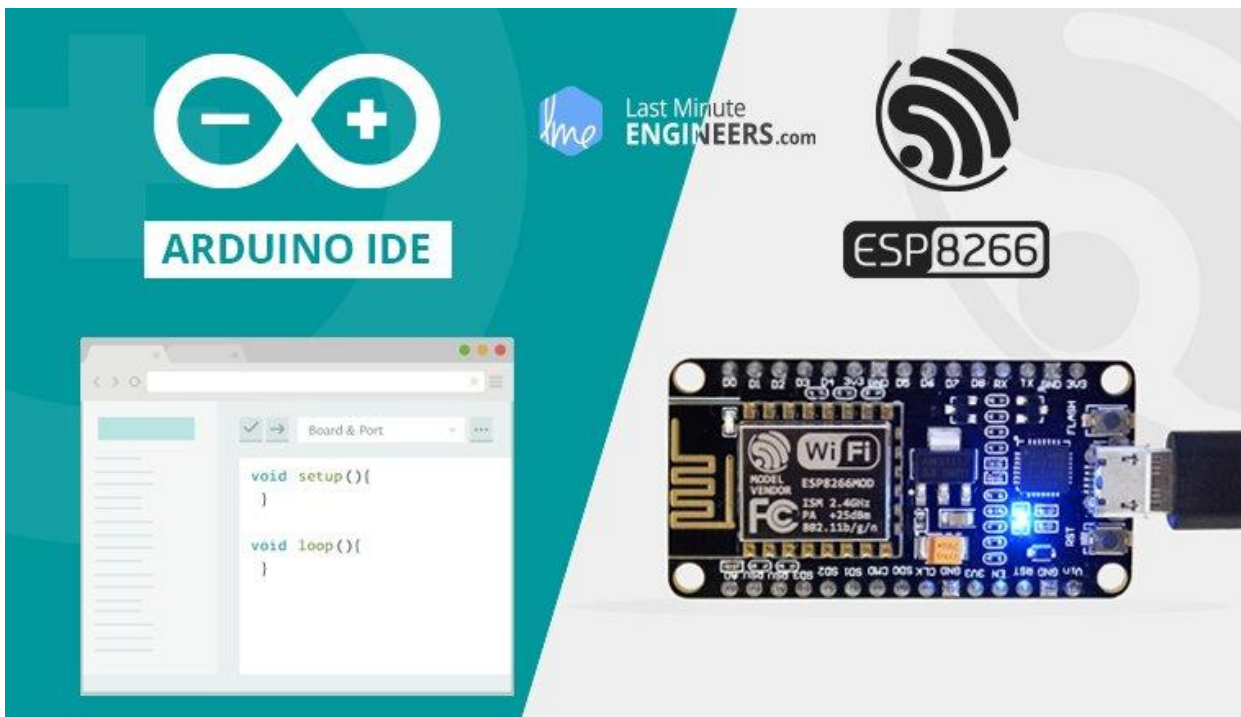


FIG4.5.ARDUINO+NODEMCU 1

The Internet of Things (IoT) has been a trending field in the world of technology. It has changed the way we work. Physical objects and the digital world are connected now more than ever. Keeping this in mind, Espressif Systems (A Shanghai-based Semiconductor Company) has released an adorable, bite-sized WiFi enabled microcontroller – **ESP8266**, at an unbelievable price! For less than \$3, it can monitor and control things from anywhere in the world – **perfect for just about any IoT project.**

4.6. ESP-12E Module

The development board equips the ESP-12E module containing ESP8266 chip having **TensilicaXtensa® 32-bit LX106 RISC microprocessor** which operates at **80 to 160 MHz** adjustable clock frequency and supports **RTOS**.



FIG4.6.ESP-12E 1

There's also **128 KB RAM** and **4MB of Flash memory** (for program and data storage) just enough to cope with the large strings that make up web pages, JSON/XML data, and everything we throw at IoT devices nowadays.

The ESP8266 Integrates **802.11b/g/n HT40 Wi-Fi transceiver**, so it can not only connect to a WiFi network and interact with the Internet, but it can also set up a network of its own, allowing other devices to connect directly to it. This makes the ESP8266 Node MCU even more versatile.

4.7. Power Requirement:

As the operating voltage range of ESP8266 is **3V to 3.6V**, the board comes with a LDO voltage regulator to keep the voltage steady at 3.3V. It can reliably supply up to 600mA, which should be more than enough when ESP8266 pulls as much as **80mA during RF transmissions**. The output of the regulator

is also broken out to one of the sides of the board and labeled a

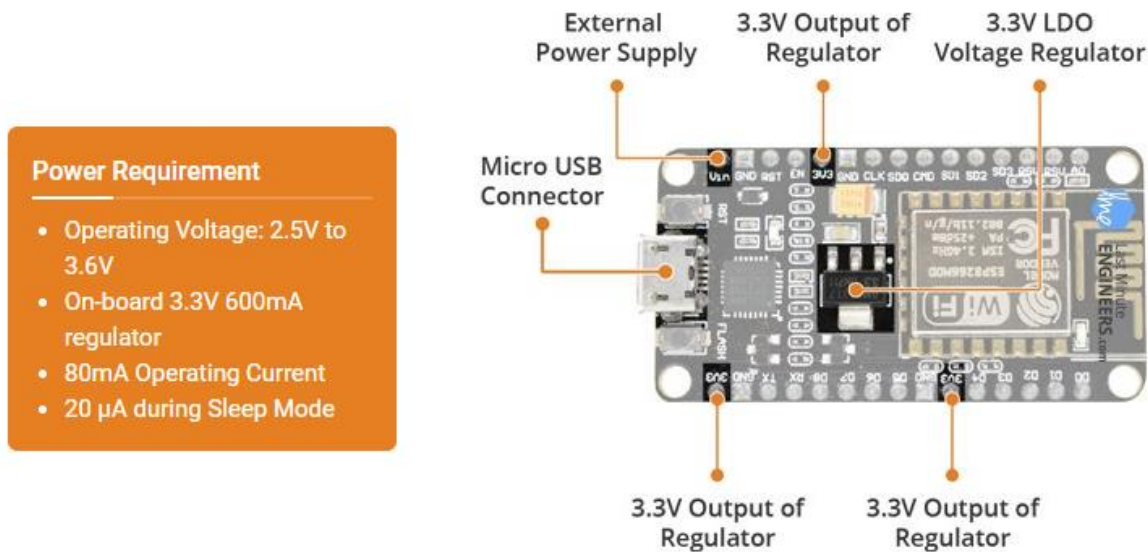


FIG.4.7 POWER REQUIREMENT 1

s 3V3. This pin can be used to supply power to external components.

Power to the ESP8266 Node MCU is supplied via the **on-board MicroB USB connector**. Alternatively, if you have a regulated 5V voltage source, the **VIN pin** can be used to directly supply the ESP8266 and its peripherals.

Warning:

The ESP8266 requires a 3.3V power supply and 3.3V logic levels for communication. The GPIO pins are not 5V-tolerant! If you want to interface the board with 5V (or higher) components, you'll need to do some level shifting.

4.7.1. Installing the ESP8266 Core on Windows OS :

Let's proceed with installing ESP8266 Arduino core.

The first thing is having latest Arduino IDE (Arduino 1.6.4 or higher) installed on your PC. If don't have it, we recommend upgrading now.

To begin, we'll need to update the board manager with a custom URL. Open up Arduino IDE and go to **File > Preferences**. Then, copy below URL into the **Additional Board Manager URLs** text box situated on the bottom of the window:

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

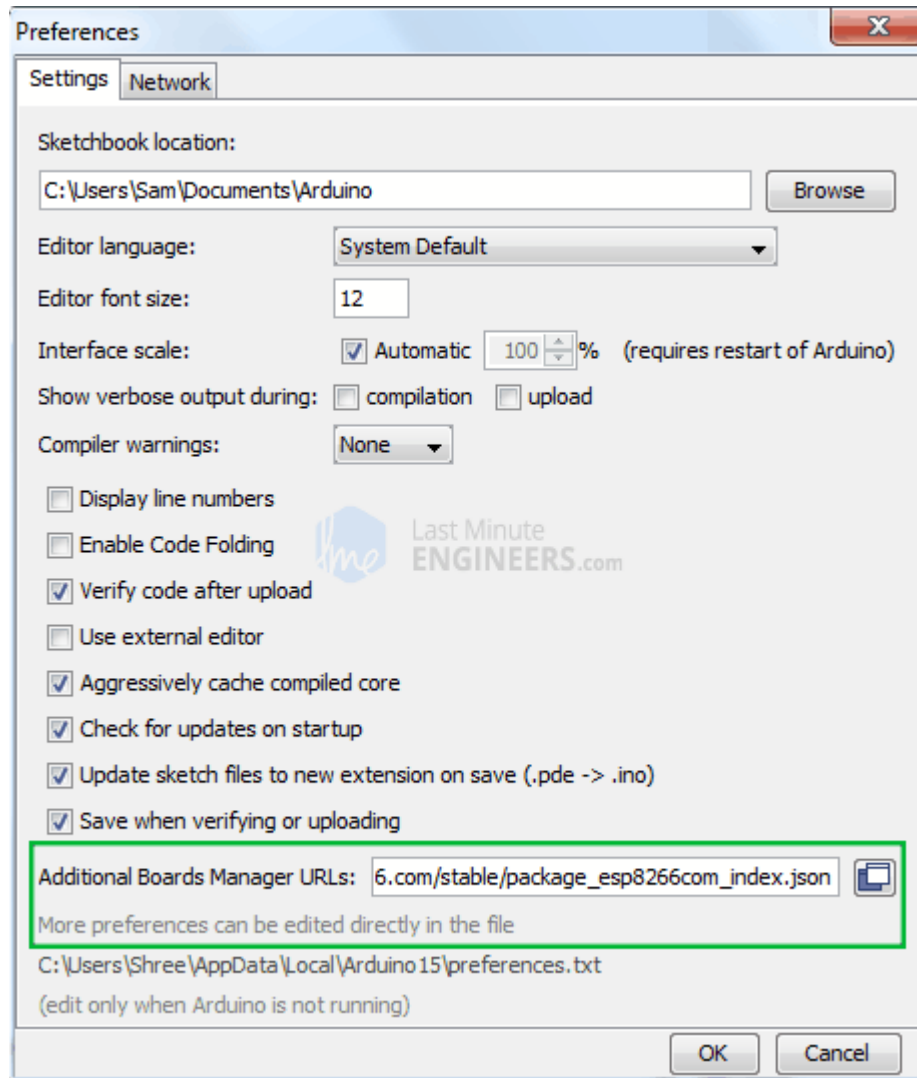


FIG.4.8.SETUP 1

Hit OK. Then navigate to the Board Manager by going to **Tools > Boards > Boards Manager**. There should be a couple new entries in addition to the standard Arduino boards. Filter your search by typing **esp8266**. Click on that entry and select Install.

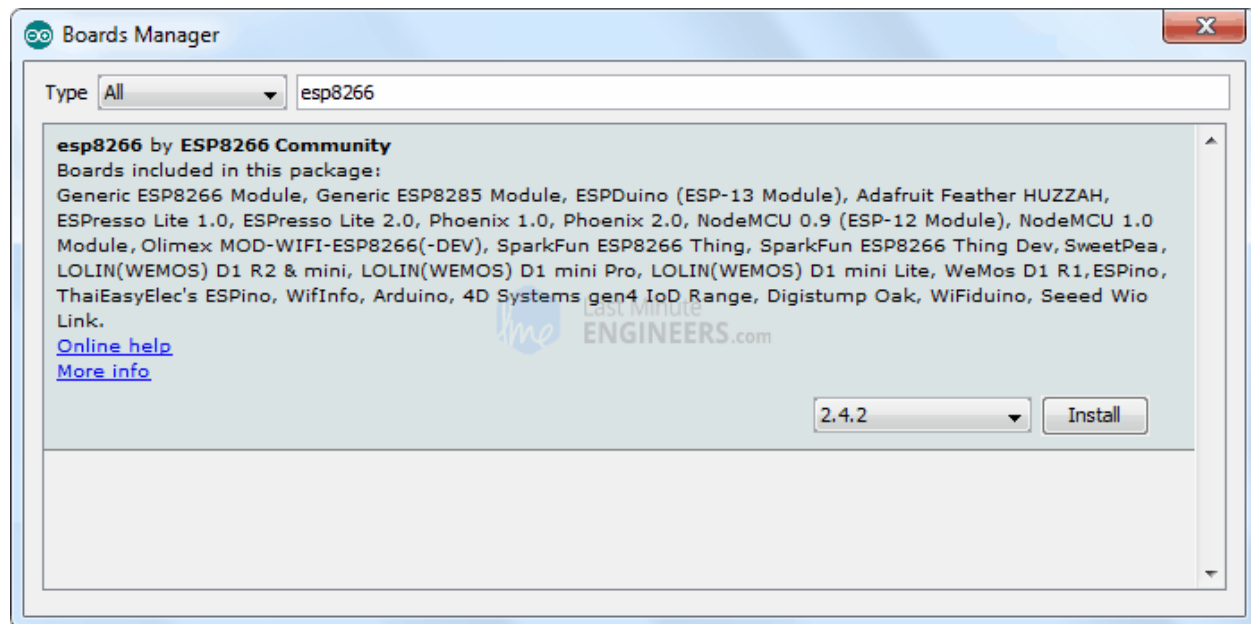


FIG.4.9.SETUP 2 1

The board definitions and tools for the ESP8266 include a whole new set of gcc, g++, and other reasonably large, compiled binaries, so it may take a few minutes to download and install (the archived file is ~110MB). Once the installation has completed, a small INSTALLED text will appear next to the entry. You can now close the Board Manager.

4.8. Arduino Example: Blink

To make sure ESP8266 Arduino core and the Node MCU are properly set up, we'll upload the simplest sketch of all – **The Blink!**

We will use the on-board LED for this test. As mentioned earlier in this tutorial, D0 pin of the board is connected to on-board Blue LED & is user programmable. Perfect!

Before we get to uploading sketch & playing with LED, we need to make sure that the board is selected properly in Arduino IDE. Open Arduino IDE and select **Node MCU 0.9 (ESP-12 Module)** option under your **Arduino IDE > Tools > Board** menu.

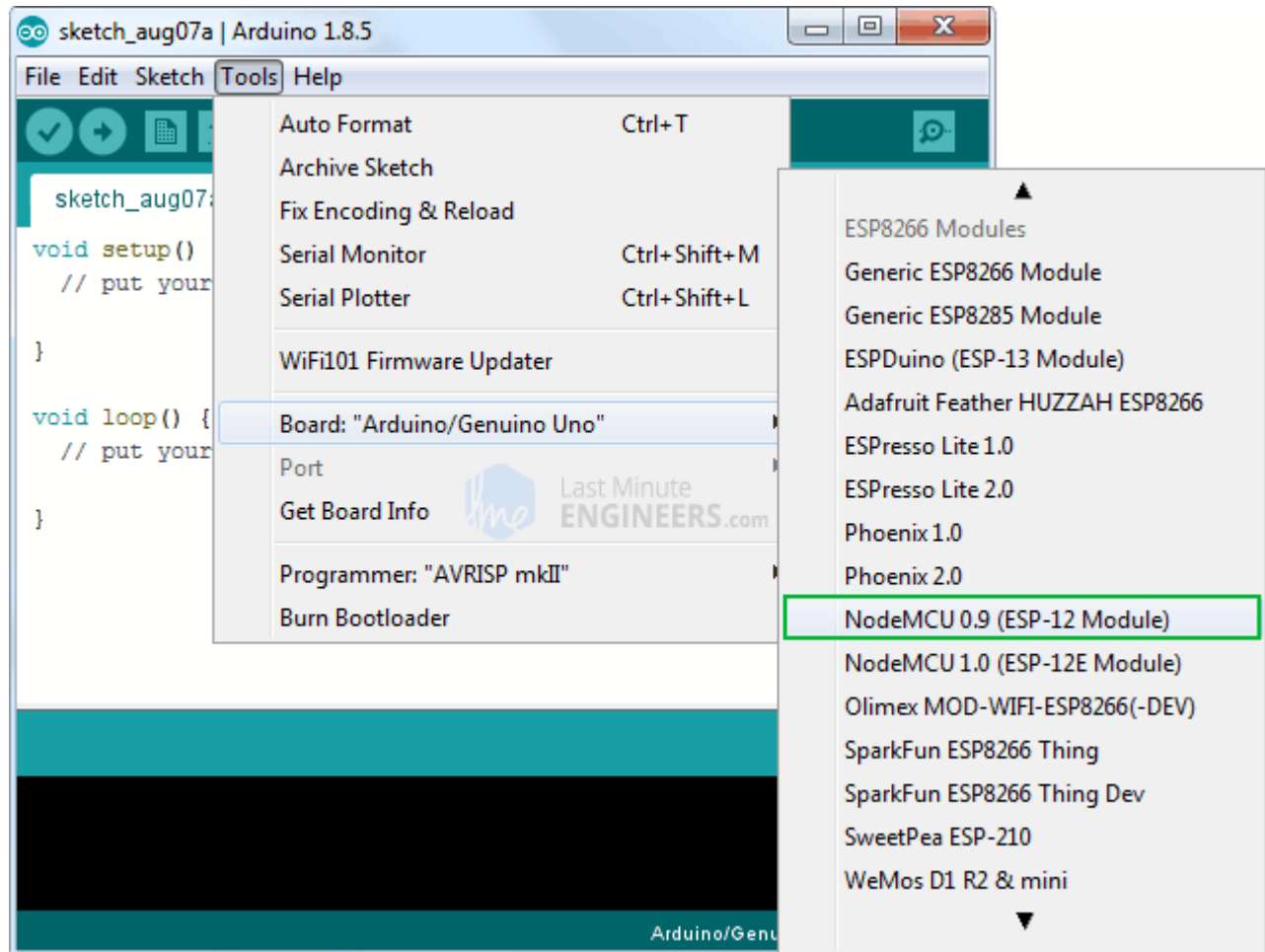


FIG.4.10.ARDUINO EXAMPLE: BLINK 1

Now, plug your ESP8266 Node MCU into your computer via micro-B USB cable. Once the board is plugged in, it should be assigned a unique COM port. On Windows machines, this will be something like **COM#**, and on Mac/Linux computers it will come in the form of **/dev/tty.usbserial-XXXXXX**. Select this serial port under the **Arduino IDE > Tools > Port** menu. Also select the **Upload Speed : 115200**

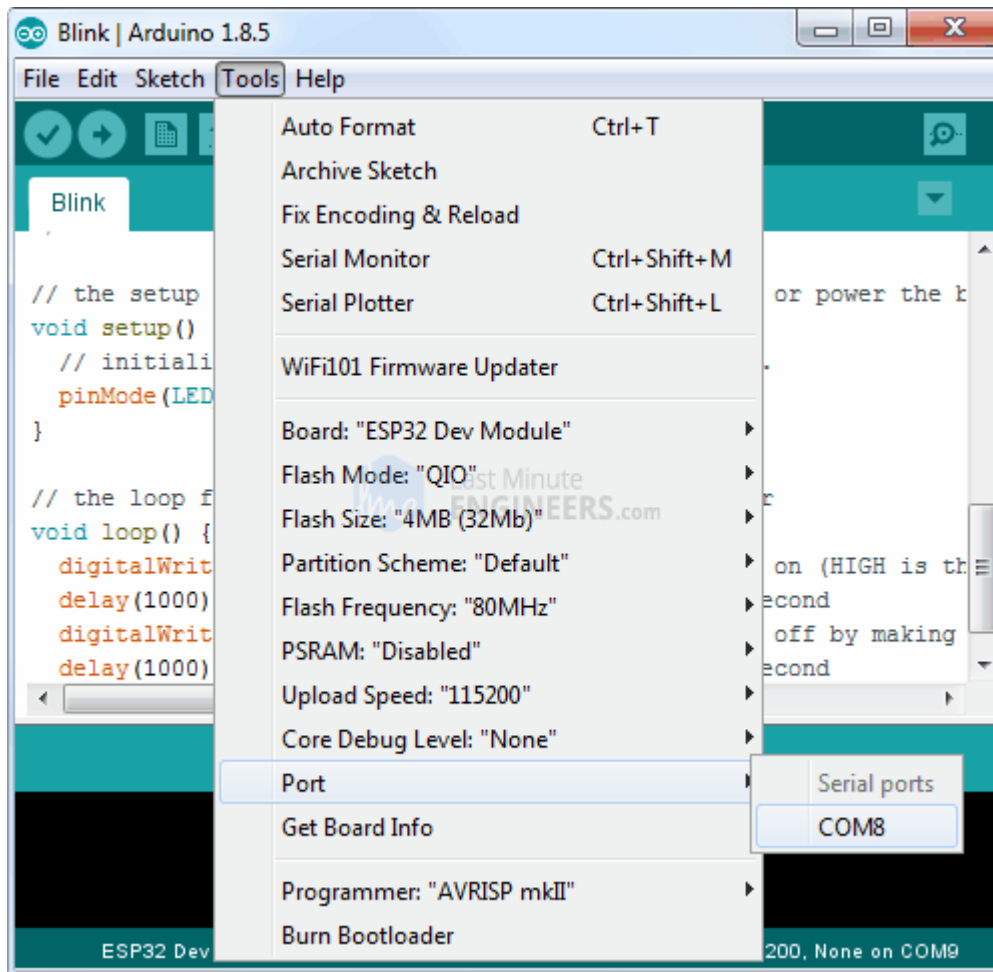


FIG.4.11.ARDUINO EXAMPLE: BLINK 2

Warning:

More attention needs to be given to selecting board, choosing COM port and selecting Upload speed. You may get **espcomm_upload_mem** error while uploading new sketches, if failed to do so.

Once the code is uploaded, LED will start blinking. You may need to **tap the RST button** to get your ESP8266 to begin running the sketch.

4.9. DHT 11

Interface DHT11 with ESP8266 Node MCU Using Web Server



FIG.4.12.DHT11 1

Have you ever wanted to have sensors scattered all around your house and garden reporting their temperature regularly to a central server? Then, this IoT project might be the solid launching point for you! This project uses ESP8266 Node MCU as the control device that easily connects to existing WiFi network & creates a Web Server. When any connected device accesses this web server, ESP8266 Node MCU reads in temperature & relative humidity from the DHT11 sensor & sends it to the web browser of that device with a nice interface. Excited? Let's get started!

It may look intimidating, but there are a few concepts you should be familiar with, before venturing further into this tutorial. If any of the concepts below sound foreign to you, consider reading (at least skimming) through that tutorial first:

4.9.1. Wiring – Connecting DHT11 sensor to ESP8266 Node MCU

Connecting DHT11/DHT22/AM2302 sensor to ESP8266 Node MCU is fairly simple. Start by placing the Node MCU on to your breadboard, ensuring each side of the board is on a separate side of the breadboard.

Now place the sensor on to your breadboard besides Node MCU. Connect VCC pin on the sensor to the 3.3V pin on the Node MCU and ground to ground. Also connect Data pin on the sensor to D8 pin of the ESP8266 Node MCU. Finally, we need to place a pull-up resistor of 10K Ω between VCC and data line to keep it HIGH for proper communication between sensor and Node MCU. If you happen to have a breakout board of the sensor, you need not add any external pull-up. It comes with a built-in pull-up resistor.

When you're done you should have something that looks similar to the illustration shown below.

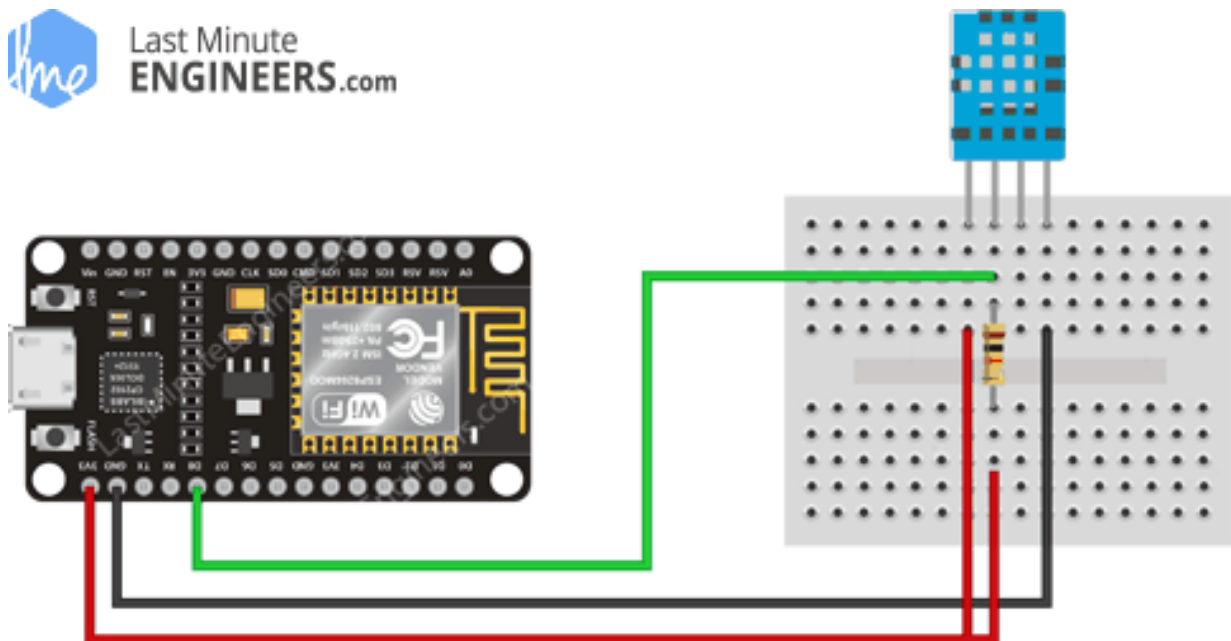


FIG.4.13.DHT11-2 1

Wiring DHT11 Temperature Humidity Sensor with ESP8266 Node MCU

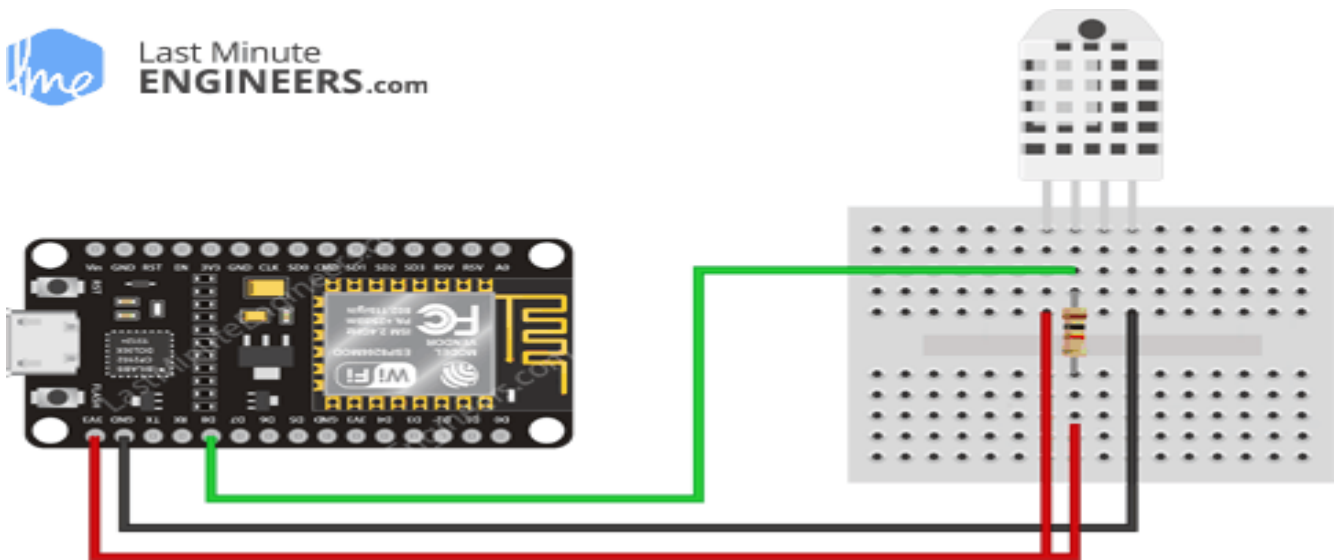


FIG.4.14.DHT11-2 2

Wiring DHT22 Temperature Humidity Sensor with ESP8266 Node MCU

4.9.2. Installing DHT Sensor Library

Communicating with DHT11, DHT22/AM2302 sensors is a bunch of work, as they have their own single wire protocol for data transfer. And this protocol requires precise timing. Fortunately, we don't have to worry much about this because we are going to use the DHT library from Adafruit which takes care of almost everything. The library is so powerful that it runs on both Arduino and ESP architecture.

To install the library navigate to the **Sketch > Include Library > Manage Libraries...** Wait for Library Manager to download libraries index and update list of installed libraries.

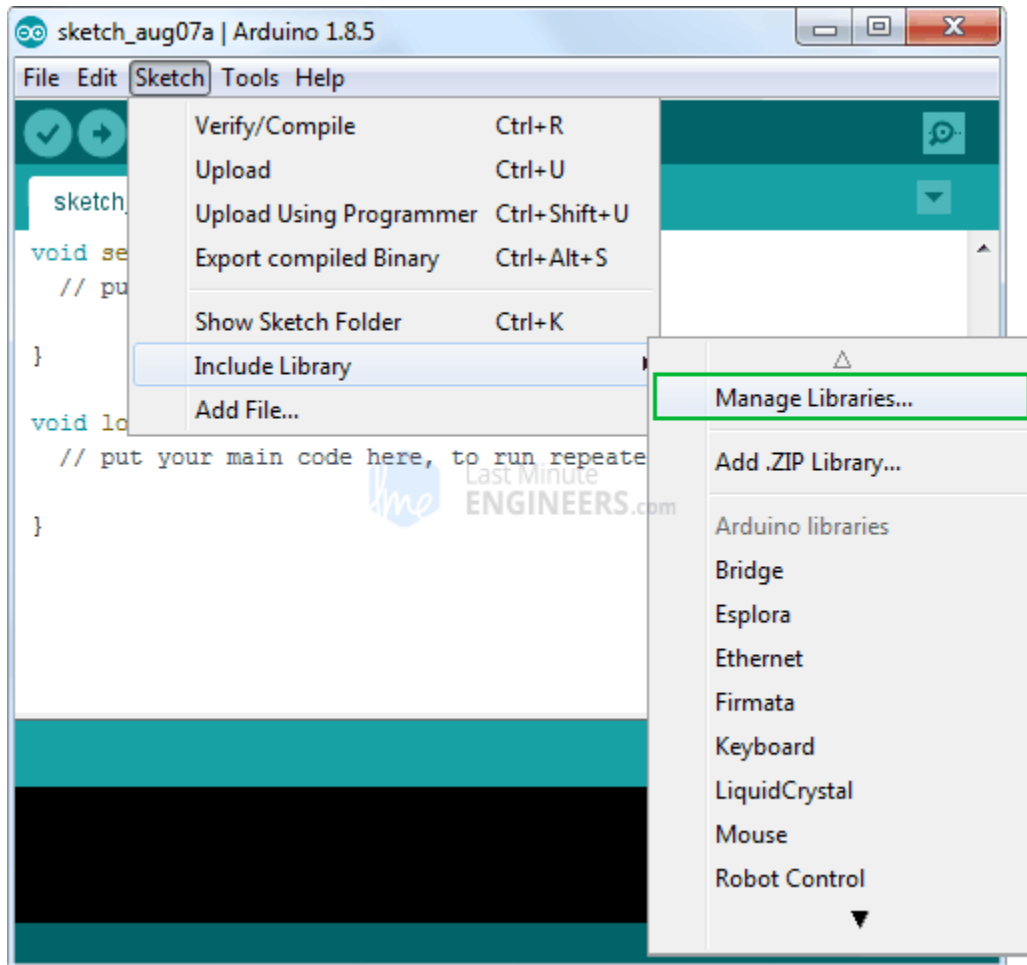


FIG.4.15.ARDUINO SETUP 1

Filter your search by typing '**DHT sensor**'. There should be a couple entries. Look for **DHT sensor library** by **Adafruit**. Click on that entry, and then select Install.

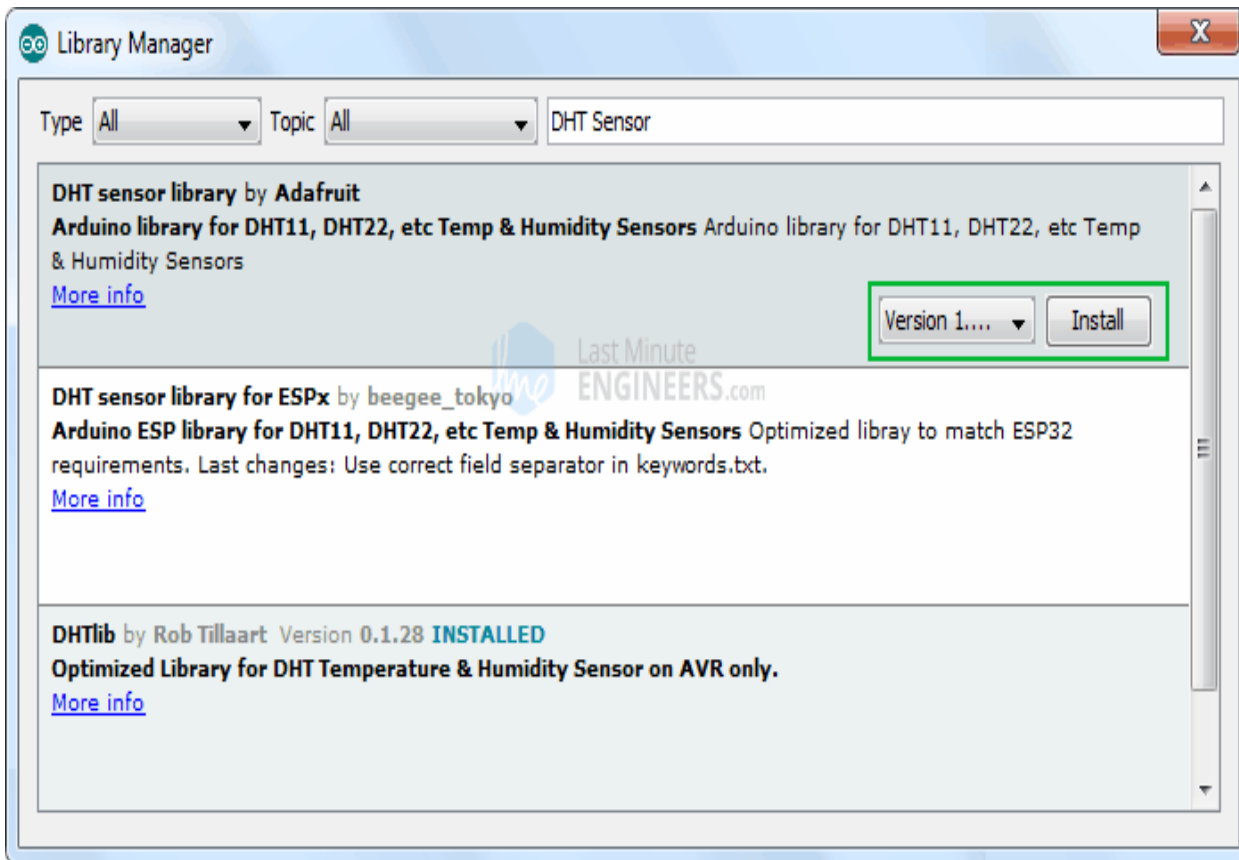


FIG.4.16.ARDUINO SETUP 2

The DHT sensor library uses the Adafruit Sensor support backend. So, search the library manager for **Adafruit Unified Sensor** and install that too (you may have to scroll a bit)

4.10. Accessing the Web Server

After uploading the sketch, open the Serial Monitor at a baud rate of 115200. And press the RESET button on the Node MCU. If everything is OK, it will output the dynamic IP address obtained from your router and show **HTTP server started** message.

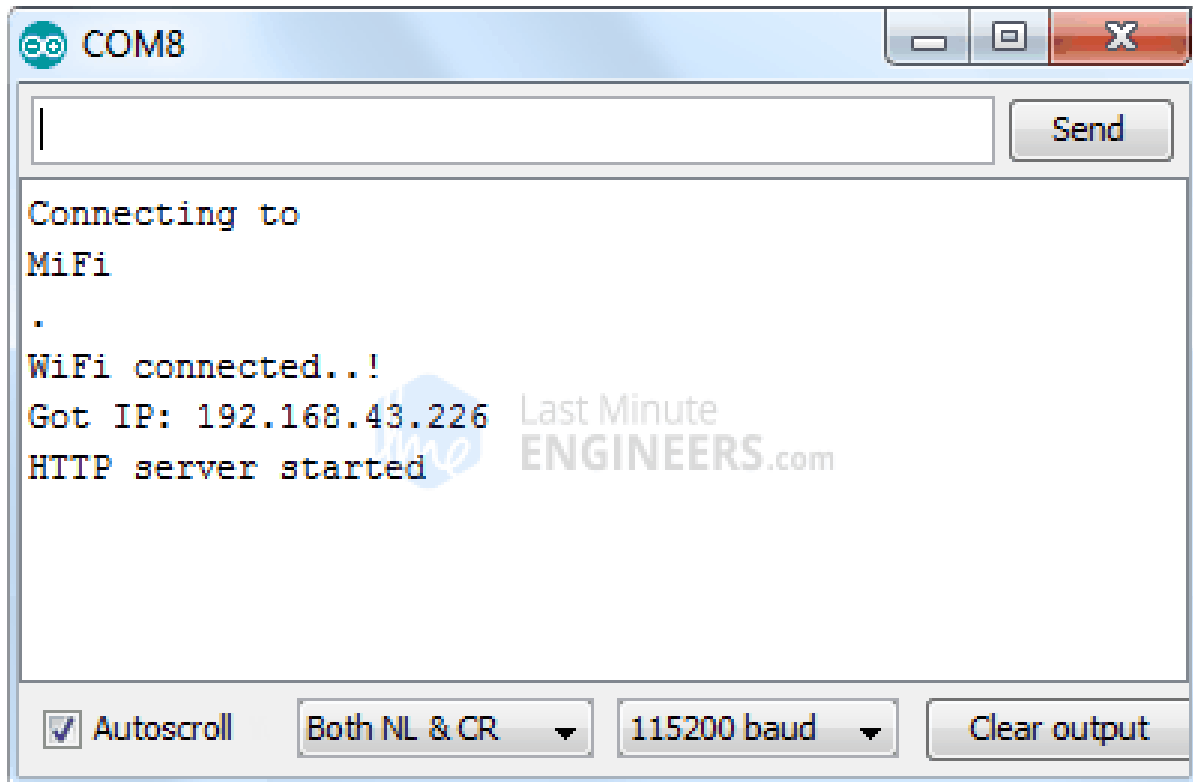


FIG.4.17-ACCESSING THE WEB SERVER 1

Next, load up a browser and point it to the IP address shown on the serial monitor. The ESP8266 Node MCU should serve up a web page showing temperature and relative humidity.

4.11. I2C LCD

4.11.1. 16×2 I2C Liquid Crystal Display

For this tutorial we'll be using a 16×2 I2C LCD display, but LCDs with other sizes should also work.

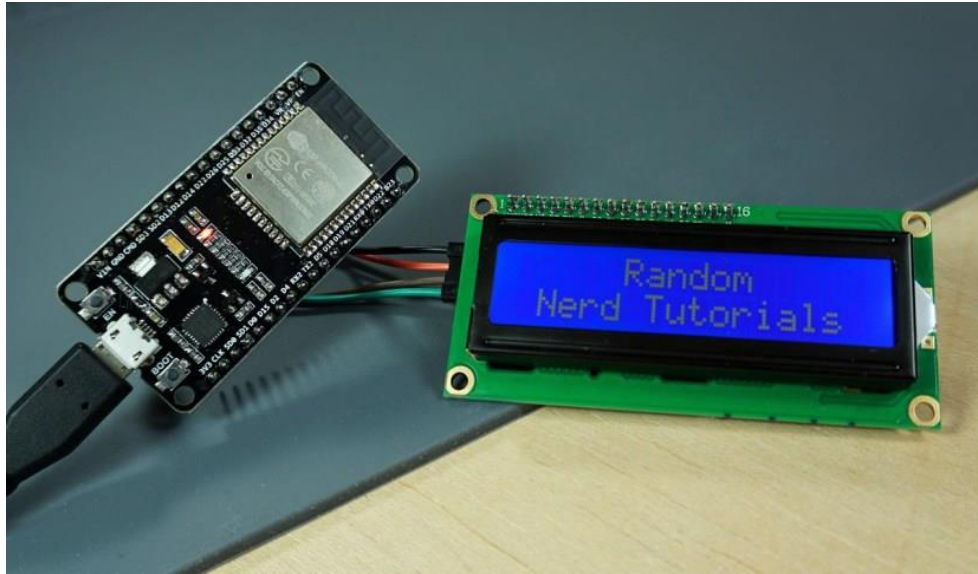


FIG.4.18.I2C LCD 1

The advantage of using an I2C LCD is that the wiring is really simple. You just need to wire the SDA and SCL pins.

Additionally, it comes with a built-in potentiometer you can use to adjust the contrast between the background and the characters on the LCD. On a “regular” LCD you need to add a potentiometer to the circuit to adjust the contrast.

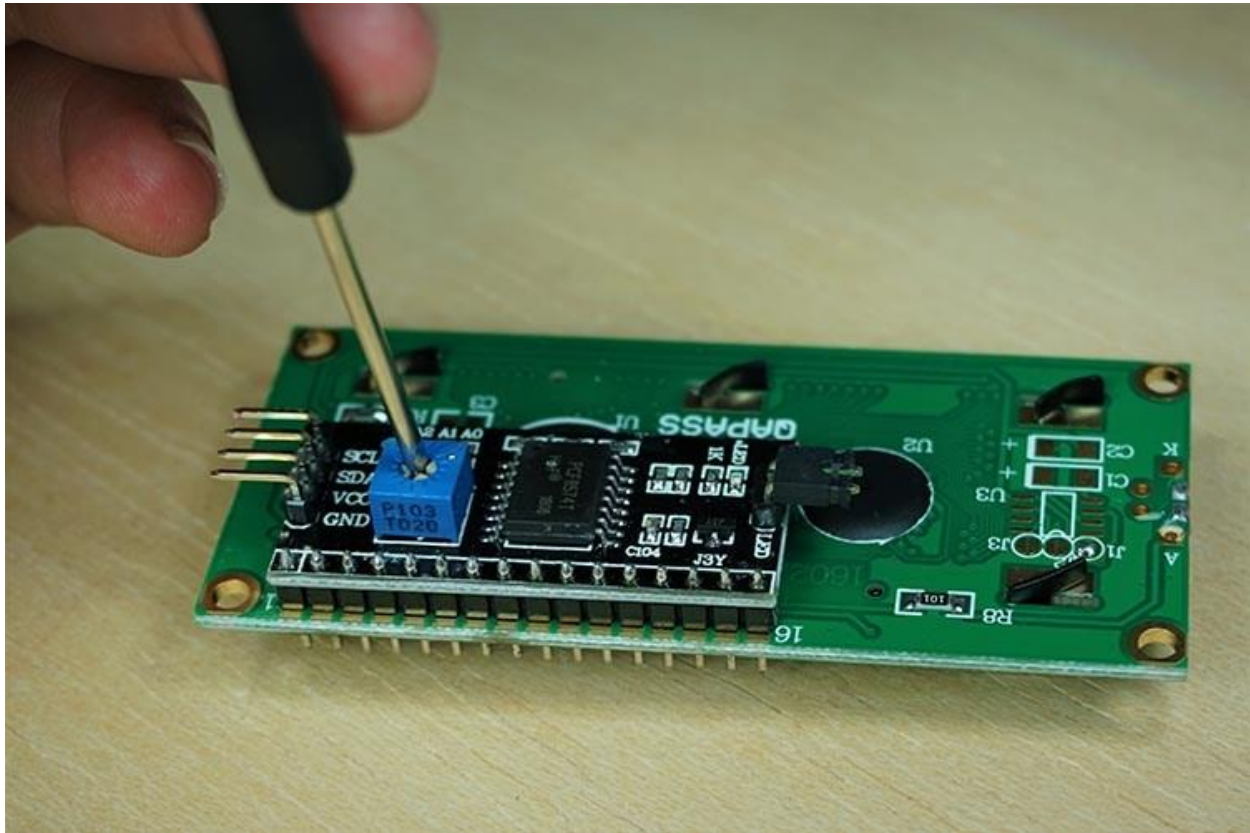


FIG.4.19.I2C LCD 2

4.11.2. Parts Required

To follow this tutorial you need these parts:

- ESP32 DOIT DEVKIT V1 Board – read [ESP32 Development Boards Review and Comparison](#)
- Optional – ESP8266 12-E – read [Best ESP8266 Wi-Fi Development Boards](#)
- 16×2 I2C Liquid Crystal Display (LCD)
- Female to female jumper wires

You can use the preceding links or go directly to [MakerAdvisor.com/tools](https://makeradvisor.com/tools) to find all the parts for your projects at the best price!

4.11.3. Wiring the LCD to the ESP32

This display uses I2C communication, which makes wiring really simple.

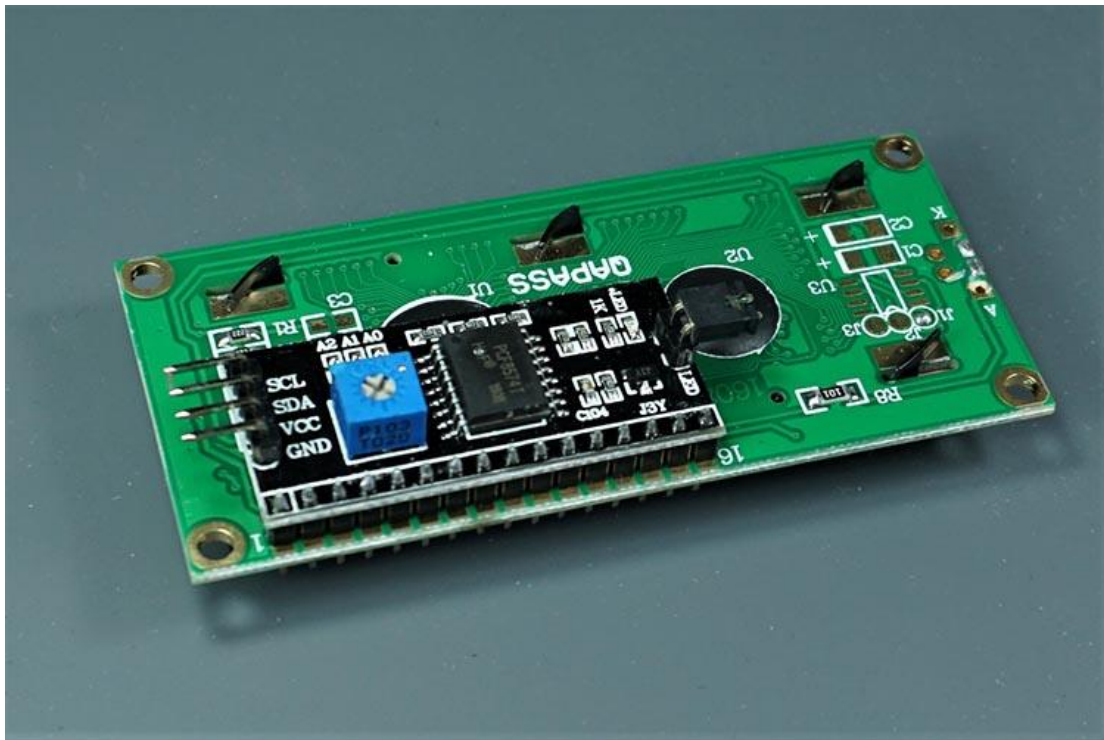
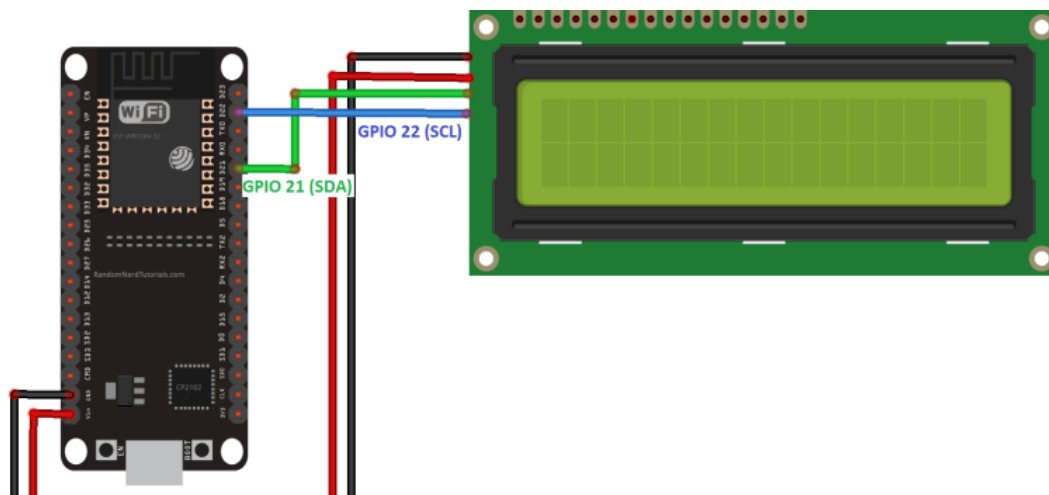


FIG.4.20.I2C LCD 3

Wire your LCD to the ESP32 by following the next schematic diagram. We're using the



FI.4.21.I2C TO NODE MCU 1

ESP32 default I2C pins (GPIO 21 and GPIO 22).

You can also use the following table as a reference.

TABLE 1.6: I2C VS ESP32

I2C LCD	ESP32
GND	GND
VCC	VIN
SDA	GPIO 21
SCL	GPIO 22

4.11.4. WIRING THE LCD TO THE ESP8266

You can also wire your LCD to the ESP8266 by following the next schematic diagram. We're using the ESP8266 default I2C pins (GPIO 4 and GPIO 5).

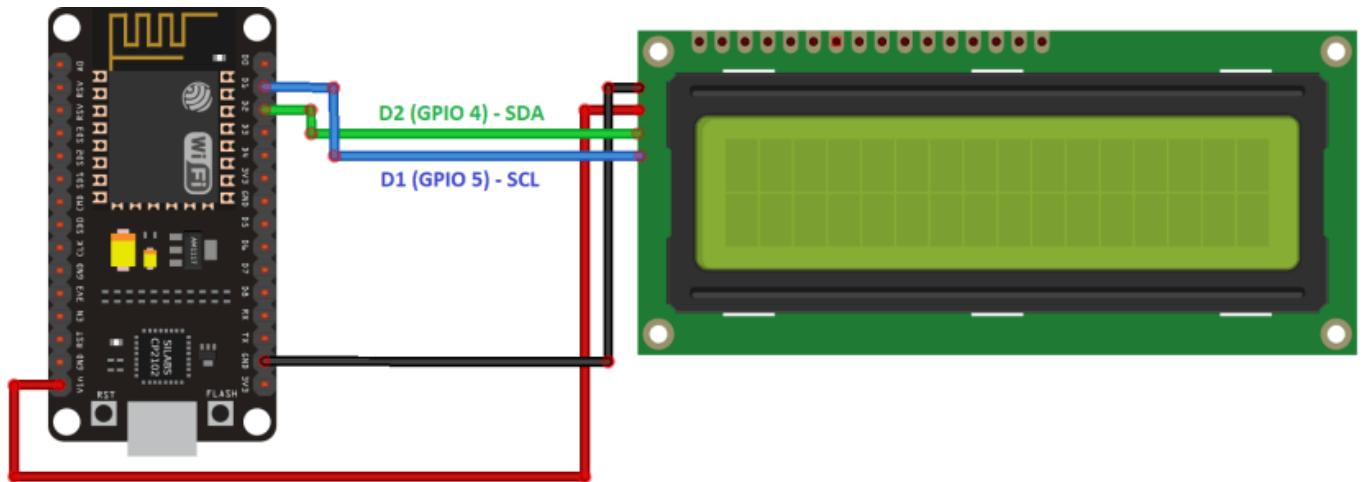


FIG.4.22.I2C TO NODE MCU 2

You can also use the following table as a reference.

TABLE 1.7: LCD VS ESP8266

I2C LCD	ESP8266
GND	GND
VCC	VIN
SDA	GPIO 4 (D2)
SCL	GPIO 5 (D1)

4.12. Preparing the Arduino IDE

Before proceeding with the project, you need to install the ESP32 or ESP8266 add-on in the Arduino IDE.

4.12.1. Arduino IDE with ESP32

Follow one of the next guides to prepare your Arduino IDE to work with the ESP32:

- **Windows** instructions – ESP32 Board in Arduino IDE
- **Mac and Linux** instructions – ESP32 Board in Arduino IDE

4.12.2. Arduino IDE with ESP8266 :

To install the ESP8266 add-on in your Arduino IDE, read the following tutorial: How to Install the ESP8266 Board in Arduino IDE.

INSTALLING THE LIQUIDCRYSTAL_I2C LIBRARY :

There are several libraries that work with the I2C LCD. We're using this library by Marco Schwartz.

Follow the next steps to install the library:

1. Click [here](#) to download the LiquidCrystal_I2C library. You should have a .zip folder in your Downloads
2. Unzip the .zip folder and you should get **LiquidCrystal_I2C-master** folder
3. Rename your folder from to **LiquidCrystal_I2C**
4. Move the **LiquidCrystal_I2C** folder to your Arduino IDE installation **libraries** folder
5. Finally, re-open your Arduino IDE
6. After uploading the code, open the Serial Monitor at a baud rate of 115200. Press the ESP32 EN button. The I2C address should be displayed in the Serial Monitor.

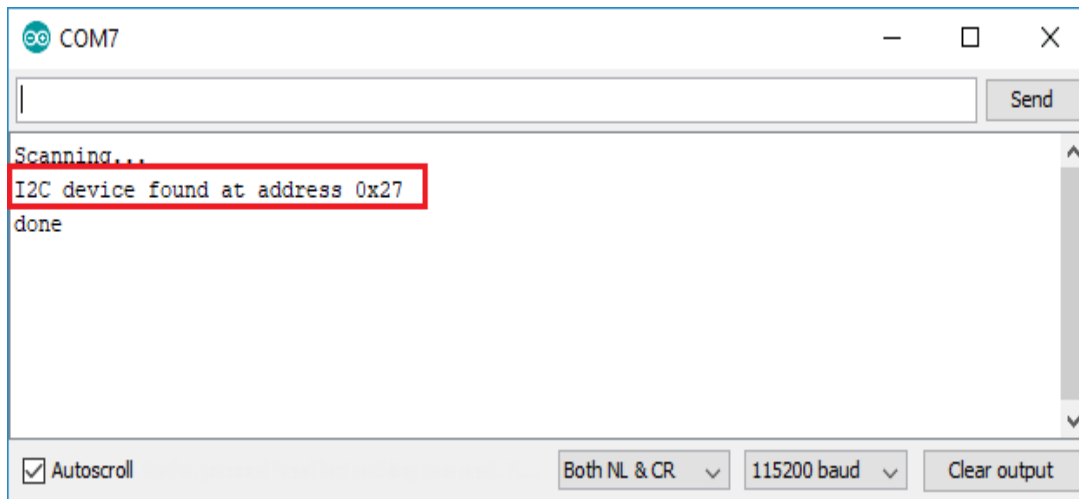


FIGURE 4.22.1: SERIAL MONITOR OUTPUT

7. In this case the address is **0x27**. If you're using a similar 16×2 display, you'll probably get the same address.

4.12.3. Display Static Text on the LCD

8. Displaying static text on the LCD is very simple. All you have to do is select where you want the characters to be displayed on the screen, and then send the message to the display.

9. Here's a very simple sketch example that displays "**HEARTBEAT**"

It displays the message in the first row, and then in the second row. In this simple sketch we show you



FIG.4.23.LCD DISPLAY 1

the most useful and important functions from the `LiquidCrystal_I2C` library. So, let's take a quick look at how the code works.

4.12.4. How the code works

First, you need to include the `LiquidCrystal_I2C` library.

```
#include <LiquidCrystal_I2C.h>
```

The next two lines set the number of columns and rows of your LCD display. If you're using a display with another size, you should modify those variables.

```
int lcdColumns=16;
```

```
int lcdRows=2;
```

Then, you need to set the display address, the number of columns and number of rows. You should use the display address you've found in the previous step.

```
LiquidCrystal_I2C lcd(0x27,lcdColumns,lcdRows);
```

In the `setup()`, first initialize the display with the `init()` method.

```
lcd.init();
```

Then, turn on the LCD backlight, so that you're able to read the characters on the display.

```
lcd.backlight();
```

To display a message on the screen, first you need to set the cursor to where you want your message to be written. The following line sets the cursor to the first column, first row.

```
lcd.setCursor(0,0);
```

Note: 0 corresponds to the first column, 1 to the second column, and so on...

Then, you can finally print your message on the display using the `print()` method.

```
lcd.print("Hello, World!");
```

Wait one second, and then clean the display with the `clear()` method.

```
lcd.clear();
```

After that, set the cursor to a new position: first column, second row.

```
lcd.setCursor(0,1);
```

Then, the process is repeated. So, here's a summary of the functions to manipulate and write on the display:

- `lcd.init()`: initializes the display
- `lcd.backlight()`: turns the LCD backlight on

- `lcd.setCursor(int column, int row)`: sets the cursor to the specified column and row
- `lcd.print(String message)`: displays the message on the display
- `lcd.clear()`: clears the display

This example works well to display static text no longer than 16 characters.

4.12.5. Display Scrolling Text on the LCD

Scrolling text on the LCD is specially useful when you want to display messages longer than 16 characters. The library comes with built-in functions that allows you to scroll text. However, many people experience problems with those functions because:

- The function scrolls text on both rows. So, you can't have a fixed row and a scrolling row;
- It doesn't work properly if you try to display messages longer than 16 characters.

So, we've created a sample sketch with a function you can use in your projects to scroll longer messages.



FIGURE 4.24: SCROLLING TEXT

4.13. WORKING:

There are two transistors in the center of the drawing. One has ten times the emitter area of the other. This means it has one tenth of the current density, since the same current is going through both transistors. This causes a voltage across the resistor R1 that is proportional to the absolute temperature, and is almost linear across the range. The "almost" part is taken care of by a special circuit that straightens out the slightly curved graph of voltage versus temperature.

The amplifier at the top ensures that the voltage at the base of the left transistor (Q1) is proportional to absolute temperature (PTAT) by comparing the output of the two transistors.

The amplifier at the right converts absolute temperature (measured in Kelvin) into either Fahrenheit or Celsius, depending on the part (LM34 or LM35). The little circle with the "i" in it is a constant current source circuit.

The two resistors are calibrated in the factory to produce a highly accurate temperature sensor.

The integrated circuit has many transistors in it -- two in the middle, some in each amplifier, some in the constant current source, and some in the curvature compensation circuit. All of that is fit into the tiny package with three leads.

4.13.1. PULSE SENSOR :

The **pulse sensor working principle** is very simple. This sensor has two surfaces, on the first surface, the light-emitting diode & ambient light sensor is connected. Similarly, on the second surface, the circuit is connected which is accountable for the noise cancellation & amplification.

The LED is located above a vein in a human body like ear tip or fingertip, however, it must be located on top of a layer directly. Once the LED is located on the vein, then the LED starts emitting light. Once the heart is pumping, then there will be a flow of blood within the veins. So if we check the blood flow, then we can check the heart rates also.

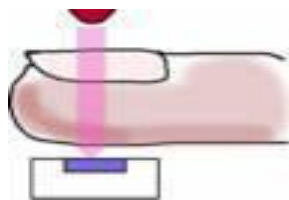


FIG.4.25.PULSE SENSOR 1

If the blood flow is sensed then the ambient light sensor will receive more light as they will be reproduced by the flow of blood. This small change within obtained light can be examined over time to decide our pulse rates.



FIG.4.26.PULSE SENSOR 2

Applications of Pulse Sensor :

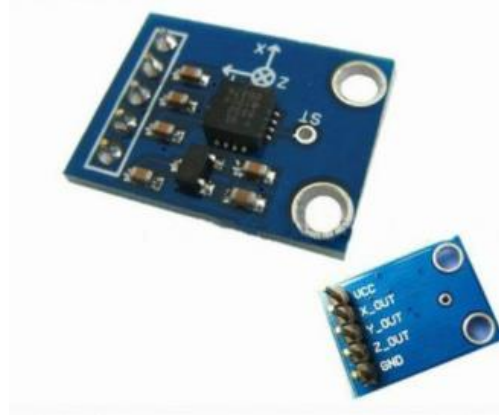
The applications of pulse rate sensor include the following.

- This sensor is used for Sleep Tracking
- This sensor is used for Anxiety monitoring
- This sensor is used in remote patient monitoring or alarm system
- This sensor is used in Health bands
- This sensor is used in complex gaming consoles

4.13.2. MEMS SENSOR

So what goes on inside that little sensor?

MEMS is a chip-based technology, known as a **Micro Electro-Mechanical System**. Sensors are composed of a suspended mass between a pair of capacitive plates. When tilt is applied to the sensor, the suspended mass creates a difference in electric potential. The difference is measured as a change



4.27.MEMS SENSOR 1

in capacitance.

Micro Electro Mechanical Systems (MEMS) is an enabling technology for the Internet of Things (IoT) because MEMS manufacturing makes possible small, low-cost, high-performance sensors and actuators. This chapter discusses some of useful and valuable aspects of MEMS technology for the IoT. While human senses have evolved to be quite good at specific functional capabilities, they are very limited compared to MEMS sensor technologies. MEMS sensors can have sensitivities far higher than any human, or even some animals, thereby enabling the ability to detect far smaller levels of an olfactory parameter, or any other parameter, than otherwise would be possible. The chapter reviews some of the more widely known and commonly used MEMS fabrication processes. It highlights a small number of the most popular methods of micromachining. The diversity, economic importance, and extent of potential applications of MEMS for the implementation of the IoT make it the hallmark technology of the future.

4.13.3. BUZZERS

A digital temperature sensor connected to NODE MCU measures body temperature of the patient.

A buzzer produces auditory beeps when the patient's heartbeat occurs / detected. This gives a brief insight to a healthcare professional how a patient's heart is performing in a particular health condition. Abnormal heartbeats can be detected by just listening to the beeps.

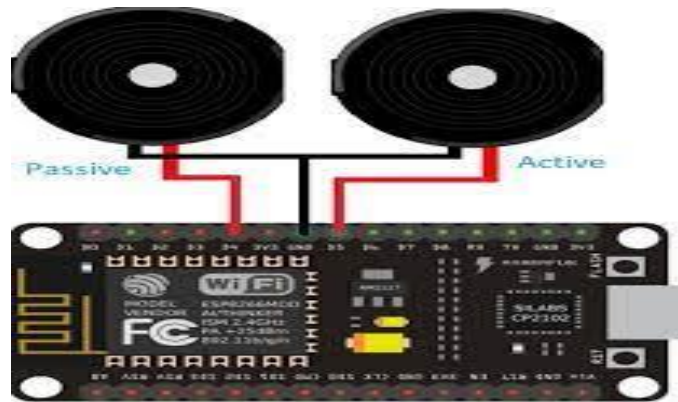


FIG.4.28.BUZZERS 1

This circuit is not only capable of sending patient's health data to a server but also can show real time data on a 16×2 LCD display. This is useful for a healthcare professional who is actively monitoring a patient on site.

RELATED WORK

This project is a combination of IOT and SOFTWARE.

Totally 2 people [Ravi and Naresh] worked in this project under the guidance of **Dr. V. J. K. KISHOR SONTI, M. Tech.. Ph.D., Associate Professor, Department of Electronics and Communication Engineering** Who is our internal Guide.

Ravi Took care of the software part and Naresh took care of the IOT part,

We both bought the Required components from the Hydrebad in Ameerpet from a shop.

Chapter – 5

Results and Discussion

In this section, we discuss the experiments performed so far and the analysis of the proposed system.

5.1. Simulation of the Fog Environment

For the purpose of comparing the use of fog computing versus just using the traditional cloud computing in healthcare, we used I Fog Sim toolkit to simulate the fog network [19].

The I Fog Sim simulates the specified configuration and gives the simulated outputs. This makes it convenient to observe end results when all required technology is not available. The simulator itself adds a bit overhead time. For this simulation, we conducted several test runs for 5 configurations of monitoring devices.

The average latency and network usage for the 5 configurations are given in Table 1. The table shows that the configurations of connected devices do not really affect the latency of our architecture that uses fog computing. The network usage of the architecture with fog computing is much lower than the architecture that uses only fog computing. The iFogSim toolkit is used to simulate the 5 different configurations. For each of the 5 configurations, the monitoring devices are varied. In Config 1, Config 2, Config 3, Config 4, and Config 5, they each have 4, 8, 16, 32, and 64 monitoring devices, respectively. So each configuration will give different results when simulated. The monitoring devices used in the configurations have a CPU length of 1000 million instructions, a network length of 20,000 bytes, and an average interarrival time of 5 ms. Figures 5 and 6 give the comparison of the latency and network usage for the various configurations. The simulation of configurations that use only the cloud is done using the CloudSim toolkit, and those that use the cloud in conjunction with the fog layer is done using the iFogSim toolkit. From both the charts, we can observe that the complexity of the fog layer does not affect the latency or the network usage. In fact, fog computing increases the efficiency of the entire system. From Table 1, we can observe that using fog layer indeed decreases the latency and network usage when compared to using only cloud computing. Figure 7 shows the energy consumption of fog computing versus only using the cloud. From the figure, it can be analyzed that the energy consumption for fog computing mainly takes places at the edge where most of the processing is done. On the other hand, in the case of cloud computing, the energy is mainly spent at the data centers or at the cloud.

5.2 THINGSPEAK:

5.2.1 What is ThingSpeak?

ThingSpeak is IoT Cloud platform where you can send sensor data to the cloud. You can also analyze and visualize your data with MATLAB or other software, including making your own applications.

The ThingSpeak service is operated by MathWorks. In order to sign up for ThingSpeak, you must create a new MathWorks Account or log in to your existing MathWorks Account.

ThingSpeak is free for small non-commercial projects.

ThingSpeak includes a Web Service (REST API) that lets you collect and store sensor data in the cloud and develop Internet of Things applications. It works with Arduino, Raspberry Pi and MATLAB (premade libraries and APIs exists) But it should work with all kind of Programming Languages, since it uses a REST API and HTTP.

5.2.2. Account Settings:

- MathWorks Account email and password are needed to sign in to ThingSpeak.
- If you edit your MathWorks Account settings, you will need to sign out of ThingSpeak and log back in.
- Your username is displayed as the author of your public channels.
- Your username must be 6-27 characters, containing only letters and numbers
- Time Zone is used when displaying data in your charts, and when scheduling your ThingSpeak apps.
- User API key is required to create and manage channels using the REST API.
- MQTT API key is required to subscribe to channel topics using the MQTT API. API keys enable you to write data to a channel or read data from a private channel. API keys are auto-generated fig.

Write API Key

Key

6G4B20WD814ITJD7

Generate New Write API Key

Read API Keys

Key

JP00NH16RAF8XQ1W

Note

Save Note

Delete API Key

Add New Read API Key

FIGURE:5.1. API KEYSETTINGS 1

- **Write API Key:** Use this key to write data to a channel. If you feel your key has been compromised, click **Generate New Write API Key**.
-
- **Read API Keys:** Use this key to allow other people to view your private channel feeds and charts. Click **Generate New Read API Key** to generate an additional read key for the channel.

My Profile

Email	rabtravi1069@gmail.com
	Edit MathWorks Account Settings
Time Zone	(GMT-05:00) Eastern Time (US & Canada) ▼
Username	mwa0000025739959
	<button>Edit</button>

API Keys

User API Key	5X25K9LRY2VIY3D2	<button>↻</button>
Alerts API Key	<no API key>	<button>↻</button>

FIGURE 5.2. API KEY SETTINGS 2

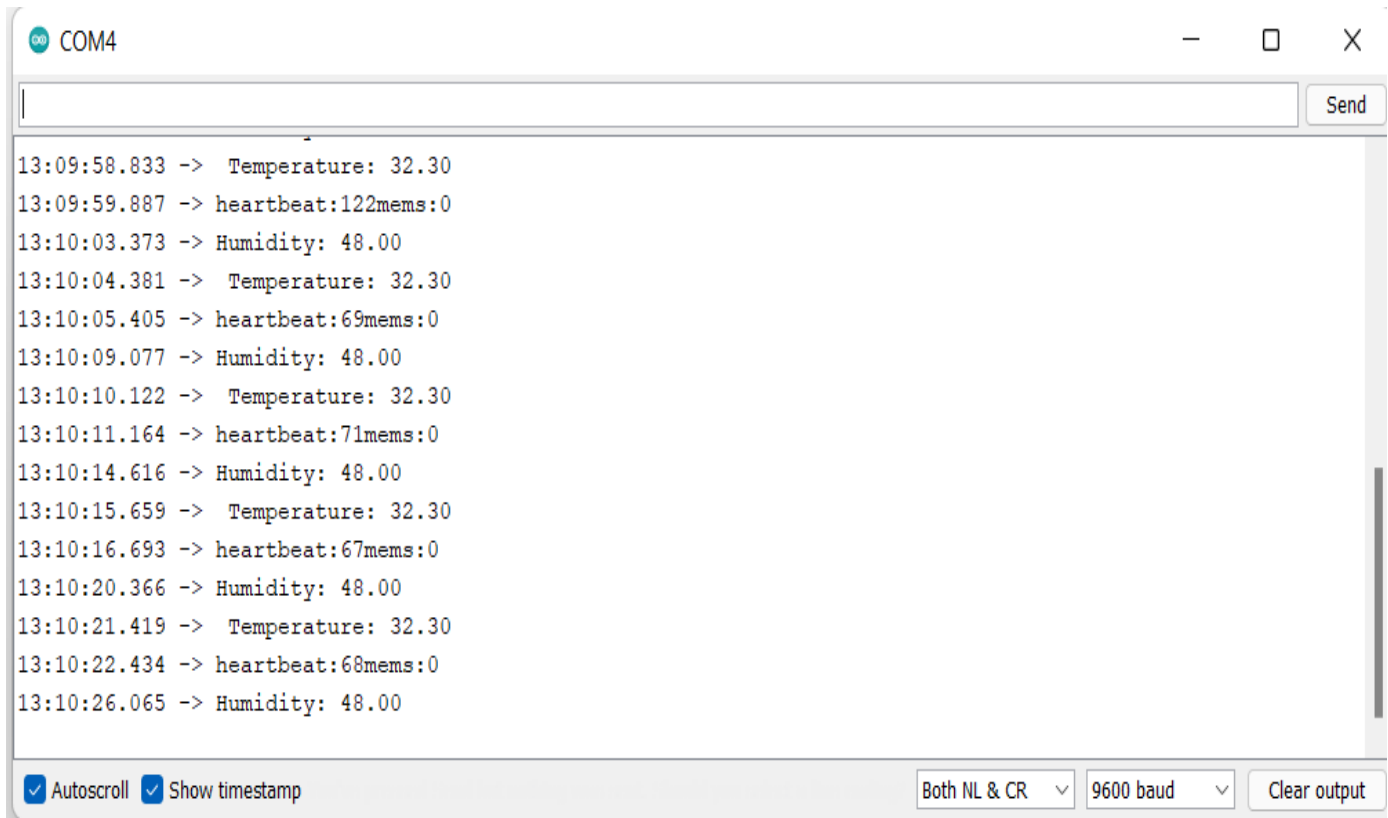


FIG:5.3 SERIAL MONIOTR RESULTS 1

Note: Use this field to enter information about channel read keys. For example, add notes to keep



FIG:5.4 RESULTS STORED IN THE THINGSPEAK

track of users with access to your channel data will be updated daily second by second in the thing speak platform.

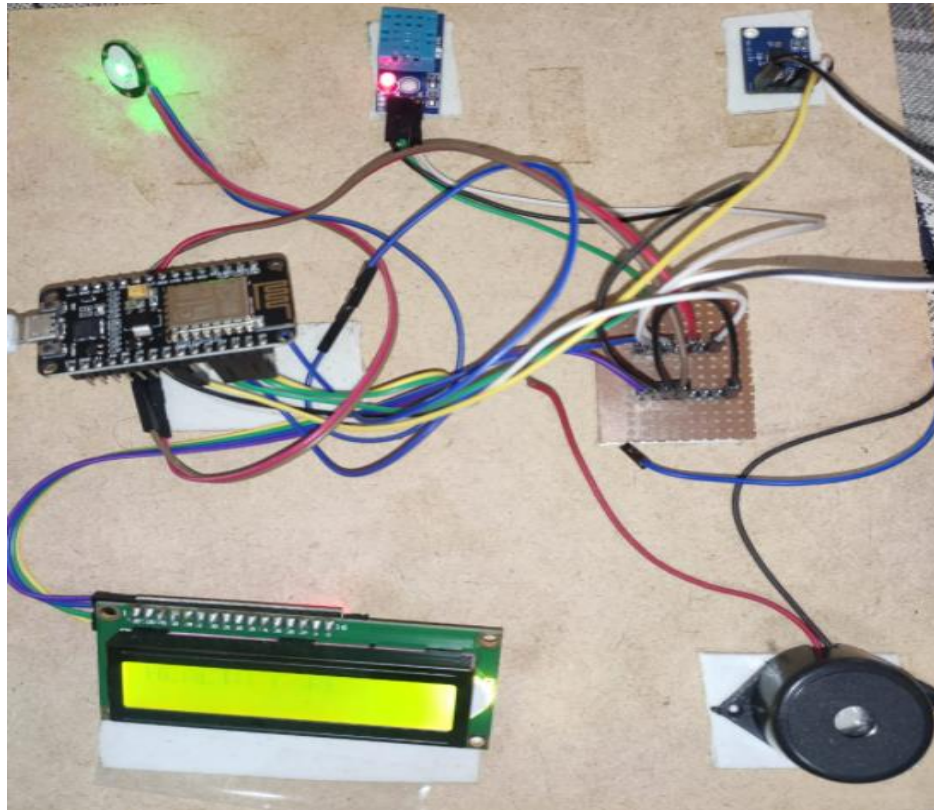


FIGURE 5.5: CONNECTED PROJECT

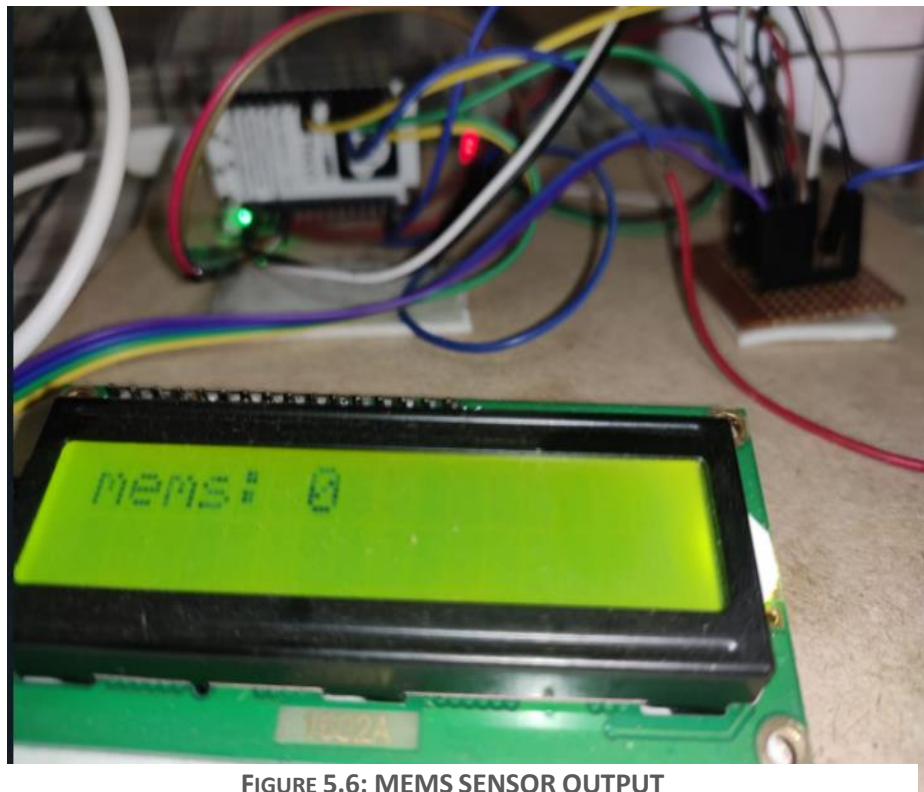


FIGURE 5.6: MEMS SENSOR OUTPUT

CHAPTER 6

CONCLUSION AND FUTURE WORKS

6.1. CONCLUSION:

In this paper, we reviewed the IOT based technologies being used as smart hospitals in present time, their mechanisms, advantages and disadvantage. Smart hospital or Telehealth, as per designed and implemented, has been performing successfully. Having numerous advantages, Telehealth telegraph has been proven to be effective, efficient and user-friendly system. Being as dynamic as internet, the framework of IOT based health services call for designing systems still more advanced to put through the existing drawbacks.

6.2. The Future of work in health care

Plenty of technology. Rapid change. Workforce uncertainty. And no shortage of well-entrenched “ways we’ve always done things.” Few industries match this description better than health care.

Although 100 percent of health care providers surveyed in the plan to make significant progress in adopting cognitive and artificial intelligence (AI) technologies in the next three to five years, and 33 percent say they consider it a priority to train employees so they can work side by side with robots and AI, none have made significant progress in adopting these technologies.²

One reason for this may be that many leaders of health care provider organizations anticipate that the scale and pace of change will overwhelm their workforce and compound current challenges, such as a short supply of nurses and a burned out physician population.

Today’s hospital systems are asking, “How can we hire and retain more nurses to combat our labor shortage?” The question they should ask instead is, “How can we augment nursing roles so that high-performing nurses wouldn’t want to work anywhere else?”

REFERENCES:

1. S. Potluri, A. Sarkar, E. T Yasin and S.N Mohanty, "IoT enabled cloud based healthcare system using Fog Computing: a Case Study", *Journal of Critical Reviews*, vol. 7, no. 6, 2020, ISSN 2394–5125.

Show in Context [Google Scholar](#)

2. H.J de Moura Costa, C.A daCosta, R. da Rosa Righi and R. Stoffel, "Fog computing in health: A systematic literature review", *Health Technol.*, vol. 10, pp. 1025-1044, 2020, [online] Available: <https://doi.org/10.1007/s12553-020-00431-8>.

Show in Context [CrossRef](#) [Google Scholar](#)

3. S. R Hassan, I. Ahmad, S. Ahmad, A. Alfaify and M. Shafiq, "Remote Pain Monitoring Using Fog Computing for e-Healthcare: An Efficient Architecture", *Sensors*, no. 22, pp. 6574, 2020, [online] Available: <https://doi.org/10.3390/s20226574>.

Show in Context [CrossRef](#) [Google Scholar](#)

4. G. Aceto, P. Valerio and P. Antonio, "Industry 4.0 and health: Internet of things big data and cloud computing for healthcare 4.0", *Journal of Industrial Information Integration*, vol. 18, pp. 100129, 2018.

Show in Context [CrossRef](#) [Google Scholar](#)

5. G. Harish, S. Nagaraju, B. Harish and M. Shaik, "A Review on Fog Computing and its Applications", *International Journal of Innovative Technology and Exploring Engineering (IJITEE)*, vol. 8, no. 6C2, 2019, ISSN 2278–3075.

[Google Scholar](#)

6. S. S Gill, R. Arya, G. Wander and R. Buyy, "Fog-Based Smart Healthcare as a Big Data and Cloud Service for Heart Patients Using IoT", *International Conference on Intelligent Data Communication Technologies and Internet of Things (ICICI)*, 2018.

[CrossRef](#) [Google Scholar](#)

7. Y. Kumar and M. Mahajan, "Intelligent Behavior Of Fog Computing With IOT For Healthcare System", *International Journal Of Scientific & Technology Research*, vol. 8, no. 07, July 2019, ISSN 2277–8616.

[Google Scholar](#)

8. S. Alhat, N. Bangal, A. Gaikwad and S. Khairnar, "Enhancing Data Security in IoT Healthcare Services using Fog Computing", *International Research Journal of Engineering and Technology (IRJET)*, vol. 06, no. 12, Dec 2019, ISSN 2395-0056.

[Google Scholar](#)

9. M. Al-khafajiy, L. Webster, T. Baker and A. Waraich, "Towards fog driven IoT healthcare: challenges and framework of fog computing in healthcare", pp. 1-7, 2018.

[CrossRef Google Scholar](#)

10. G. Rahman and C. Wen, "Fog Computing Applications Security and Challenges Review", *International Journal of Engineering & Technology*, vol. 7, no. 3, pp. 1615-1621, 2018.

[CrossRef Google Scholar](#)

11. A.A. Mutlag, M.K.A. Ghani, N. Arunkumar, M.A. Mohamed and O. Mohd, "Enabling technologies for fog computing in healthcare IoT systems", *Future Generation Computer Systems*, 2018, [online] Available: <https://doi.org/10.1016/j.future.2018.07.049>.

[Google Scholar](#)

12. B. H Malik, F. Mahmood, S. Shahzad, M. B Arif, W. U Khan, Sadaf Ilyas, et al., "From cloud computing to fog computing in Healthcare big data", *MATEC Web Conf.*, pp. 03011, 18, 2018, [online] Available: <https://doi.org/10.1051/matecconf/201818903011>.

[CrossRef Google Scholar](#)

13. T. Francis, "A Comparison of Cloud Execution Mechanisms Fog Edge and Clone Cloud Computing", *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 8, no. 6, pp. 4646-4653, December 2018, ISSN 2088–8708.

[CrossRef Google Scholar](#)

14. F. H. Atlam, R. J. Walters and G. B. Wills, "Fog Computing and the Internet of Things: A Review", *Big Data Cogn. Comput.*, vol. 2, no. 2, pp. 10, 2018, [online] Available: <https://doi.org/10.3390/bdcc2020010>.

[CrossRef Google Scholar](#)

15. Amy K. Paul and Merrick Schaefer, "Safeguards for the use of artificial intelligence and machine learning in global health", *Bulletin of the World Health Organization*, vol. 98, no. 4, pp. 282-284, 2020.

[CrossRef](#) [Google Scholar](#)

16. R. Mahmud, K. Ramamohanarao and R. Buyya, "Latency-Aware Application Module Management for Fog Computing Environments", *ACM Transactions on Internet Technology*, pp. 19, 2018.

Show in Context [CrossRef](#) [Google Scholar](#)

17. F. A. Kraemer, A. E. Braten, N. Tamkittikhun and D. Palma, "Fog Computing in Healthcare—A Review and Discussion", *IEEE Access*, vol. 5, pp. 9206-9222, 2017.

[View Article Full Text: PDF](#) (5382KB) [Google Scholar](#)

18. O. Osanaiye, S. Chen, Z. Yan, R. Lu, K. R. Choo and M. Dlodlo, "From Cloud to Fog Computing: A Review and a Conceptual Live VM Migration Framework", *IEEE Access*, vol. 5, pp. 8284-8300, 2017.

[View Article Full Text: PDF](#) (6884KB) [Google Scholar](#)

19. F. Al-Doghman, Z. Chaczko, A.R. Ajayan and R. Klempous, "A review on fog computing technology", *IEEE International Conference in Systems Man and Cybernetics (SMC)*, pp. 001 525-001 530, 2016.

[View Article Full Text: PDF](#) (20031KB) [Google Scholar](#)

20. T. N. Gia, M. Jiang, A. Rahmani, T. Westerlund, P. Liljeberg and H. Tenhunen, "Fog Computing in Healthcare Internet of Things: A Case Study on ECG Feature Extraction", *2015 IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable Autonomic and Secure Computing; Pervasive Intelligence and Computing*, pp. 356-363, 2015.

[View Article Full Text: PDF](#) (1093KB) [Google Scholar](#)