**AN APPROACH FOR WORKERS SAFETY MONITORING IN INDUSTRIAL CHEMICAL STORAGE TANKS MAINTENANCE USING IOT**

**A PROJECT REPORT**

***Submitted by***

**NANDA KUMAR . C - 513417106026**

**VIKRAM. S - 513417106048**

**AVINASH.A - 513417106003**

**ELAN SURIYA . R - 513417106013**

***In partial fulfillment for the award of the degree***

***of***

**BACHELOR OF ENGINEERING**

***in***

**ELECTRONICS AND COMMMUNICATION ENGINEERING**

****

**UNIVERSITY COLLEGE OF ENGINEERING KANCHIPURAM**

( A Constituent College of Anna University,Chennai)

**ANNA UNIVERSITY :: CHENNAI 600 025**

**APRIL 2021**

**ANNA UNIVERSITY : CHENNAI 600 025**

**BONAFIED CERTIFICATE**

Certified that this project report  **“ AN APPROACH FOR WORKERS SAFETY MONITORING IN INDUSTRIAL CHEMICAL STORAGE TANKS MAINTENANCE USING IOT”** is the bonafide work of **NANDA KUMAR.C [513417106026]** , **VIKRAM.S [513417106048] , ELAN SURIYA.R [513417106013], AVINASH.A[513417106003]** who carried out the project work under my supervision.

**SIGNATURE SIGNATURE**

**Dr.M.MalleswaranM.Tech.,Ph.D**., **Dr.M.Malleswaran M.Tech.,Ph.D., PROJECT GUIDE HEAD OF DEPARTMENT**

Associate professor,Associate professor ,

Department of ECE, Department of ECE,

University College of Engineering , University College of Engineering,

Kanchipuram-631 552 Kanchipuram-631 552

Submitted for the Project Viva Voice held on: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**INTERNAL EXAMINER EXTERNAL EXAMINER**

**ACKNOWLEDGEMENT**

We thank almighty of our heart who has been with us source, strength, compact and inspirations in the completion of the project.We are always thankful to our college dean **Dr. P. SAKTHIVEL** , who Endorsed us throughout this project.

We extend our profound thanks to the Head of the Department as well as our Project guide **Dr.M.MALLESWARAN., M.tech., Ph.D.,** for his valuable guidance , timely support during this project.We wish to acknowledge the valuable guidance provided by the faculty Members of the department of Electronics and Communication Engineering.

**ஆய்வு சுருக்கம்**

வேதியியல் தொழில்துறை துறையில், பல்வேறு செயல்முறைகளை மேற்பார்வை செய்வதற்கும் பராமரிப்பதற்கும் மனித சக்தி தேவைப்படுகிறது. இவற்றில் வேதியியல் வைத்திருக்கும் தொட்டிகளின் சுத்தம் மற்றும் பராமரிப்பு உயிருக்கு ஆபத்தானது.

தொழிலாளர்கள் ஆரோக்கியத்துடன் கையாளும் தரநிலைகள், நெறிமுறைகள் எல்லா நேரத்திலும் பாதுகாப்பை உறுதிப்படுத்தாது. வேதியியல் சேமிப்பு தொட்டிகளின் சூழலில் மற்றும் அதைச் சுற்றியுள்ள துப்புரவு மற்றும் பராமரிப்பின் போது தொழிலாளர்களின் பாதுகாப்பை தொடர்ந்து கண்காணிக்க இந்த திட்டம் உதவுகிறது. பார்வையாளர்கள் (பாதுகாப்பு அதிகாரிகள் அல்லது மேற்பார்வையாளர்கள்) தொழிலாளர்களின் சுகாதார நிலைமைகளை வெவ்வேறு எரிவாயு பிபிஎம் மதிப்புகள் மூலம் பகுப்பாய்வு செய்யலாம், அந்த சூழலில் தொழிலாளர்கள் சுத்தம் செய்வது பாதுகாப்பானதா இல்லையா என்பதைக் குறிக்கிறது. இதயத் துடிப்பு, வாயு, வெப்பநிலை போன்ற தொழிலாளர்களிடமிருந்து அளவுருக்களைக் கண்காணிக்க பல்வேறு வகையான சென்சார்கள் பயன்படுத்தப்பட்டன. உணர்திறன் மதிப்பு வாசல் மதிப்பு முறையை விட அதிகமாக இருக்கும்போது, ​​வாயுக்களின் செறிவு, உடல் வெப்பநிலை மற்றும் இதயத் துடிப்பு ஆகியவற்றை பகுப்பாய்வு செய்வதன் மூலம் பார்வையாளரை எச்சரிக்கையுடன் எச்சரிக்கிறது. நிகழ்நேர கண்காணிப்புக்கு அவற்றின் முடிவை வரைபடமாக்குதல். முன்மொழியப்பட்ட அமைப்பில், தொலைதூர இடத்திலிருந்து Adafruit.io மேகக்கணி மேடையில் சென்சார் மதிப்புகள் பதிவு செய்யப்பட்டு, திட்டமிடப்பட்டு பகுப்பாய்வு செய்யப்பட்டுள்ளன.

**ABSTRACT**

In Chemical industrial field ,manpower is required for supervising and maintenance of various process . Among these , the cleaning and maintenance

of chemical holding tanks is a life risking one. Though industries have their

standards dealing with workers health, protocols does not ensure the safety all the time . This project helps to provide continuous monitoring of the workers safety during cleaning and maintenance in and around the chemical storage tanks environment . Observer (safety officers or supervisors) can analyze the workers health conditions with different gas ppm values indicating whether it is safe for the workers to clean in that environment or not. Various types of sensors were utilized to monitor parameters from workers like heart rate , gas , temperature, etc. When the sensed value is higher than the threshold value system alerts the observer with an alarm by analyzing concentration of gases, body temperature and heart rate and graphing out their result for real time monitoring. In the proposed system , sensor values have been recorded , plotted and analyzed on Adafruit.io cloud platform from remote location.

**TABLE OF CONTENT**

**CHAPTER TITLE PAGE**

**ABSTRACT 5**

**LIST OF FIGURES 9**

**LIST OF ABBREVATIONS 10**

**1 INTRODUCTION 12**

1.1 OVERVIEW OF THE PROJECT 12

1.2 THE PROBLEM 13

1.3 AIM OF THE PROJECT 14

1.4 FLOW CHART 14

1.5 BLOCK DIAGRAM 15

2 **LITERATURE SURVEY 16**

2.1IOT DEVICE FOR SEWAGE GAS

MONNITORING AND ALERT SYSTEM 16

2.2 SMART SAFETY MONITORING SYSTEM

WITH TWO WAY COMMUNICATION 17

2.3 TOXIC ENVIRONMENT MONITORING

USING SENSORS BASED ON ARDUINO 18

**3 HARDWARE REQUIREMENTS 19**

3.1 NODEMCU – 12E 19

3.1.1 NODEMCU SPECIFICATION 20

3.1.2 PERIPHERALS 21

3.1.3 NODEMCU INBUILT

COMPONENTS 23

3.1.4 NODEMCU INTERNAL

BLOCK DIAGRAM 26

3.2 ANALOG MULTIPLEXER 32

3.3 MQ135 GAS SENSOR 35

3.4 LM35 TEMPERATURE SENSOR 37

3.5 PULSE SENSOR AMPED 40

**4 SOFTWARE REQUIREMENTS 43**

4.1 ARDUINO IDE 43

3.1.1 SETUP 43

3.1.2 TESTING 44

4.2 ADAFRUIT IO CLOUD SERVER 44

3.2.1 MQQT DATA FRAME 49

3.2.2 ADAFRUIT IO SETUP 52

**5 IMPLEMENTATION 53**

5.1 HARDWARE IMPLEMENTATION 53

5.2 SOFTWARE IMPLEMENTATION 54

**6 SYSTEM TESTING 60**

6.1 TEST APPROACH 60

6.2 FEATURES TO BE TESTED 60

6.3 TESTING TOOLS AND

ENVIRONMENT 61

6.4 TEST CASES 61

**7 RESULT AND DISCUSSION 63**

**8 CONCLUSION 66**

**REFERENCES 67**

**LIST OF FIGURES**

**FIG .NO TITLE PAGE. NO**

Fig 1.1 Flow Chart 14

Fig 1.2 Block diagram 15

Fig 3.1 Nodemcu 19

Fig 3.2 Nodemcu pin diagram 20

Fig 3.3 Nodemcu inbuilt components 23

Fig 3.4 CP2102 driver 24

Fig 3.5 UART diagram 24

Fig 3.6 UART dataframe 25

Fig 3.7 Nodemcu internal block diagram 26

Fig 3.8 CD4067 analog multiplexer 32

Fig 3.9 Mutiplexer truth table 34

Fig 3.10 MQ135 gas sensor 35

Fig 3.11 MQ135 module circuit diagram 36

Fig 3.12 LM35 temperature sensor 37

Fig 3.13 LM35 internal circuit diagram 38

Fig 3.14 Pulse sensor amped 40

Fig 3.15 Pulse sensor module circuit diagram 41

Fig 4.1 MQQT connection diagram 48

Fig 4.2 MQQT dataframe 49

Fig 4.3 Dataframe CONNECT packet 51

Fig 5.1 Hardware circuit diagram 53

**LIST OF ABBREVIATIONS**

**GPIO - G**eneral **P**urpose **I**nput **O**utput

**PCB - P**rinted **C**ircuit **B**oard

**USB - U**niversal **S**erial **B**us

**SPI - S**erial **P**eripheral Interface

**HSPI - H**igh  **S**peed **P**arallel **I**nterface

**SCLK - S**erial  **C**locksignal

**MISO - M**aster **I**n **S**lave **O**ut

**MOSI - M**aster **O**ut **S**lave **I**n

**I2C - I**nter **I**ntegrated **C**ircuits

**UART- U**niversal **A**synchronous **R**eceiver **T**ransmitter

**LED - L**ight **E**mitting **D**iode

**LDO - L**ow **D**rop **O**ut

**CPU - C**entral **P**rocessing **U**nit

**SRAM - S**tatic **R**andom **A**ccess **M**emory

**ROM - R**ead **O**nly **M**emory

**SOC - S**ystem **O**n **C**hip

**Wi Fi - W**ireless **F**idelity

**MQQT- M**essage **Q**ueuing **T**elemetry **T**ransport

**ADC - A**nalogto **D**igital **C**onvertor

**RF - R**adio **F**requency

**CMOS- C**omplementry **M**etal **O**xide **S**emiconductor

**WLAN- W**ireless **L**ocal **A**rea **N**etwork

**MAC - M**edia **A**ccess **C**ontrol

**BSS - B**ase **S**tation **S**ubsystem

**RTS - R**equest **T**o **S**end

**CTS - C**lear **T**o **S**end

**ACK - Ack**nowledgement

**QOS - Q**uality **O**f **S**ervice

**SSID - S**ervice **S**et **I**dentifier

**URL - U**niform **R**esource **L**ocator

**VCC - V**oltage **C**ommon **C**ollector

**GND - G**round

**IDE - I**ntegrated **D**evelopment  **E**nvironment

**CHAPTER -1**

**INTRODUCTION**

**1.1 OVERVIEW OF THE PROJECT**

We are living in an age where tasks and systems are fusing together with the power of IOT to have a more efficient system of working and to execute jobs quickly! With all the power at our finger tips this is what we have come up with.The Internet of Things (IoT) shall be able to incorporate transparently and seamlessly a large number of different systems, while providing data for millions of people to use and capitalize. Building a general architecture for the IoT is hence a very complex task, mainly because of the extremely large variety of devices, link layer technologies, and services that may be involved in such a system.

Storage tanks of a variety of shapes and sizes and can be found in all sorts of industries. While we typically associate storage tanks with refineries and other petrochemical-related facilities, you will see them at nearly every type of production facility. They may hold diesel fuel for a company vehicles, purified water for food processing, plastic pellet for injection molding,petroleum power plants or chemicals used to clean or to treat products and equipment.

Although storage tanks are common and their design fairly simple , they can quickly become complicated when it’s time to perform maintenance on them. Perfuming work inside or on storage tanks can expose workers to flammable, explosive , and toxic chemicals. The interiors of tanks are usually considered to be confined places, with all challenges and complexities that creates . But safety in and around tanks is a serious business, especially given the number of fatalities that have resulted. According to the chemical safety and Hazard Investigation Board,more than 60 workers have died since 1990 while performing hotworks and cleaning projects on storage tanks and since 1982,the National Institute for occupational Safety and Health has recorded 160 fatalities every year related to confined places.

**1.2 THE PROBLEM**:

Industries have their standards and protocols for dealing with workers safety in or around the chemical holding storage tanks. Storage tanks needs contiuous monitoring for the better production yield. So it requires proper shutdowns inorder to check for cleaning, cracks,leaks if needed.

When maintenance is required the tanks are emptied and (inlets ,outlets,air lines,cooling water lines,pumps ) are isolated and disconnected. All parameters like (atmospheric oxygen, toxic fumes) were tested before workers involve in that area and vessel entry permit is provide to workers once checked.PPE and oxygen and gas detectors were provided in critically rare situations.A long manual checking is required during every maintenance period before workers were involved in that area.

Everytime oxygen and toxic gas levels are detected when workers come out and go inside of the storage tanks environment because toxic gas may build up and leads to accidents.So a continuous monitoring in needed rather than manually checking gas ,oxygen,temperature parameters everytime.

Some of the major accidents was occurred due to ,

* Workers carelessness
* Hot works(Welding,cutting,grinding)
* Unknown chemical reactions between the chemical

storage tanks.

* Unplanned maintenance
* Work pressure.

**1.3 AIM OF THE PROJECT:**

The aim of the project is to develop a system that provides continuous monitoring of the workers safety in and around the chemical storage tanks maintenance which uses IOT.

Various sensor like gas sensor(mq135), temperature sensor and pulse rate sensor were used in this project. This sensor data are stored into Adafruit cloud server as feed data and observer(safety officers or supervisors) can visualize the data from a remote place .When the sensed value is higher the threshold value ,the worker is alarmed with a buzzer and the remote observer can also act immediately to prevent unwanted accidents

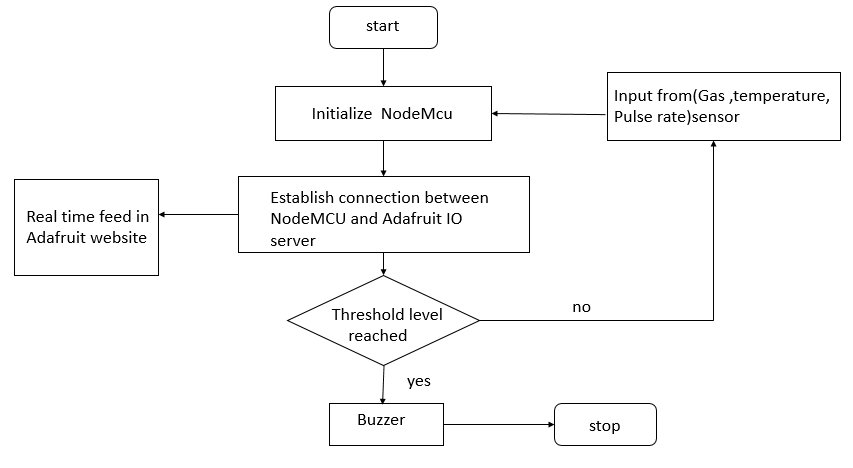
**1.4 FLOW CHART:**

Fig 1.1

The flow chart of fig.1.1 ,shows the various operation in the toxic environment supervising design.The operation begins connect the requires sensors to NodeMCU. Processing the sensor data using NodeMCU and Embedded C.The NodeMCU translate the measured sensor output into desired parameters such as temperature, gas level, pulse rate.These parameter are checked with threshold value and necessary action is taken by the NodeMCU as indicated in the flow chart.

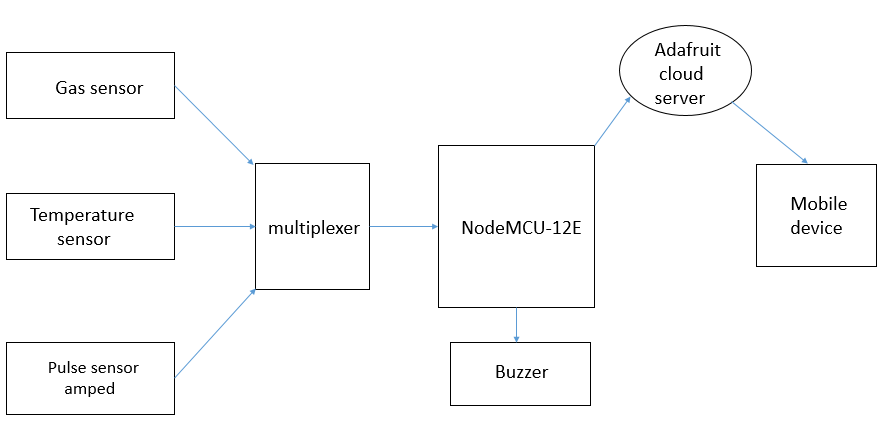
**1.5 BLOCK DIAGRAM:**

Fig 1.2

Various sensors like gas sensor (MQ135),pulse sensor, LM35 temperature are used.The output(analog signal) from this sensors are fed into multiplexer .The multiplexer multiplexes three sensors values to one analog output.It is then fed into the analog pin A0 of the NodeMCU.The microcontroller performs various process on the incoming multiplexed value and stores the sensors values wirelessly in the adafruit io cloud platform which then can plotted /visualized from a remote place by an observer(safety officer or supervisor).A buzzer is provided as output to act as alarm whenever the threshold values exceeds the sensor values .The above shown fig 1.2 represents overall working of this project.

**CHAPTER -2**

**LITERATURE SURVEY**

**2.1 IOT DEVICE FOR SEWAGE GAS MONITORING**

**AND ALERT SYSTEM**

A IOT device for sewage gas monitoring and alert system by Nitin Asthana,RidhimaBahl(2019).

This paper aims at providing smart solutions to monitor poisonous sewage gases and works on a system of live sewage level detection and monitoring.Whenever, a certain threshold is crossed, an alert is sent to the observer who is examining the conditions from a remote location.When the threshold value is lesser than the sensed values, this system alerts the sewage worker/cleaner by sending SMS and call alerts by analyzing concentrations of different toxic gases and graphing out their results for real-time monitoring thereby aiding in protection from hazardous diseases and hence serves a social cause as well. In the proposed system, sample values for sensors have been recorded and plotted on ThingSpeak analysis tool.

**2.2**  **SMART SAFETY MONITORING SYSTEM WITH**

**TWO WAY COMMUNICATION**

A smart safety monitoring system with two way communication by Sudhanshukumar,saketkumar,P.M.Tiwari,RajkumarViral.”2019 6thInternational Conference on SignalProcessing and Integrated Networks(SPIN)

In this paper ,the device presented will monitor the pulse rate of a person using a pulse oximetry sensor,the methane concentration and atmospheric oxygen concentration and provide alert to worker and exterior unit when parameters deviate from safe range.This parameters in real time will promptly alert the workers to stay safe and detect toxic gases before any harm.

**DRAWBACK:**

It doesn’t provide portability and wired connection makes complex during operations.

**2.3 TOXIC ENVIRONMENT MONITORING USING SENSORS BASED ON ARDUINO**

A toxic environment monitoring using sensors based on arduino by R.Rajalakshmi,J.Vidhya.proceeding International Conference on Systems Computation Automation andNetworking.(2019).IEEE 978-1-728115245.

This paper aimed to monitoring toxic gases in deleterious environmental conditions for safety applications based on Wi-Fi. The proposed system is to monitor harmful gases like Carbon Monoxide,Methane, Hydrogen, Flammable gases using sensors. When the criterion level go beyond the threshold limit an alert message will be send to the user using GSM.

**DRAWBACK:**

It suffers from lack of network due to limited cellphone coverage area.requires bulk power source

**CHAPTER - 3**

**HARDWARE REQUIREMENTS**

We will need the following hardware to accomplish our project.

* NodeMCU -12E
* MQ135 Gas sensor module.
* LM35 temperature sensor.
* Pulse Sensor amped.
* Multiplexer.
* Connecting wires.

**3.1 NODEMCU-12E DEVELOPMENT BOARD:**

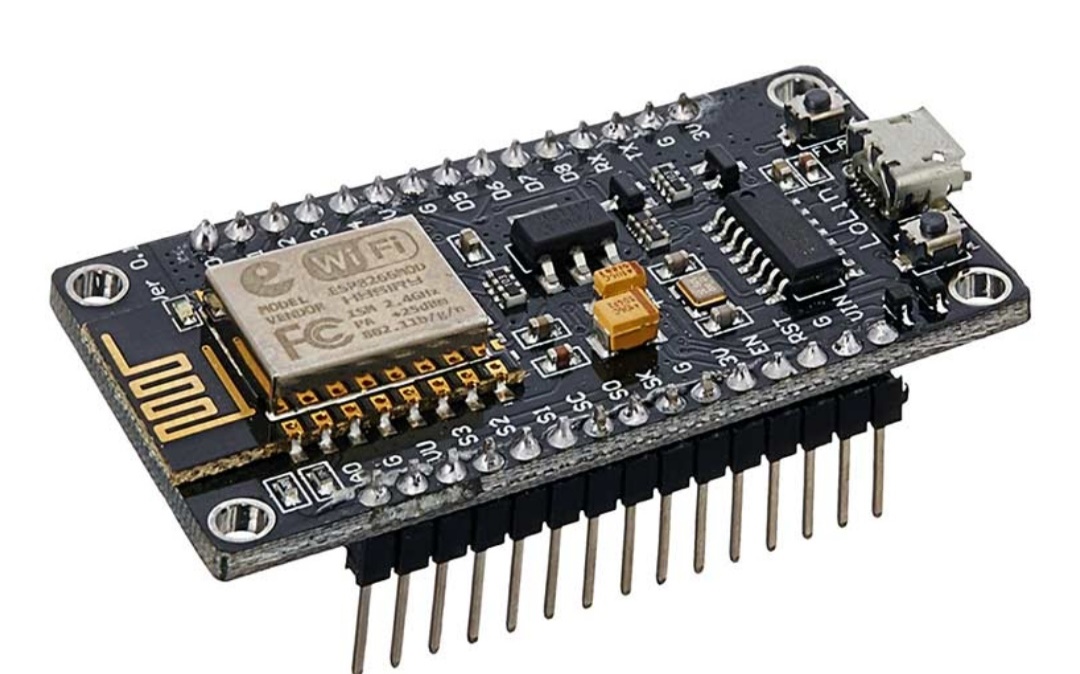
****

Fig 3.1

NodeMCU is an open-source Lua based firmware and **development board** specially targeted for IoT based Applications. It includes firmware that runs on the ESP8266 Wi-Fi SoC from Espressif Systems, and hardware which is based on the ESP-12 module. The name "NodeMCU" combines "node" and "MCU" (microcontroller unit). The term "NodeMCU" strictly speaking refers to the firmware rather than the associated development kits . The prototyping hardware typically used is a circuit board functioning as a dual in-line package(DIP) which integrates a USB controller with a smaller surface-mounted board containing the MCU and antenna. The choice of the DIP format allows for easy prototyping on breadboards . The design was initially based on the ESP-12 module of the ESP8266 , which is a Wi-Fi SoC integrated with a Tensilica Xtensa LX106 core, widely used in IoT applications. The above shown fig 3.1 shows the actual picture of the NodeMCU.

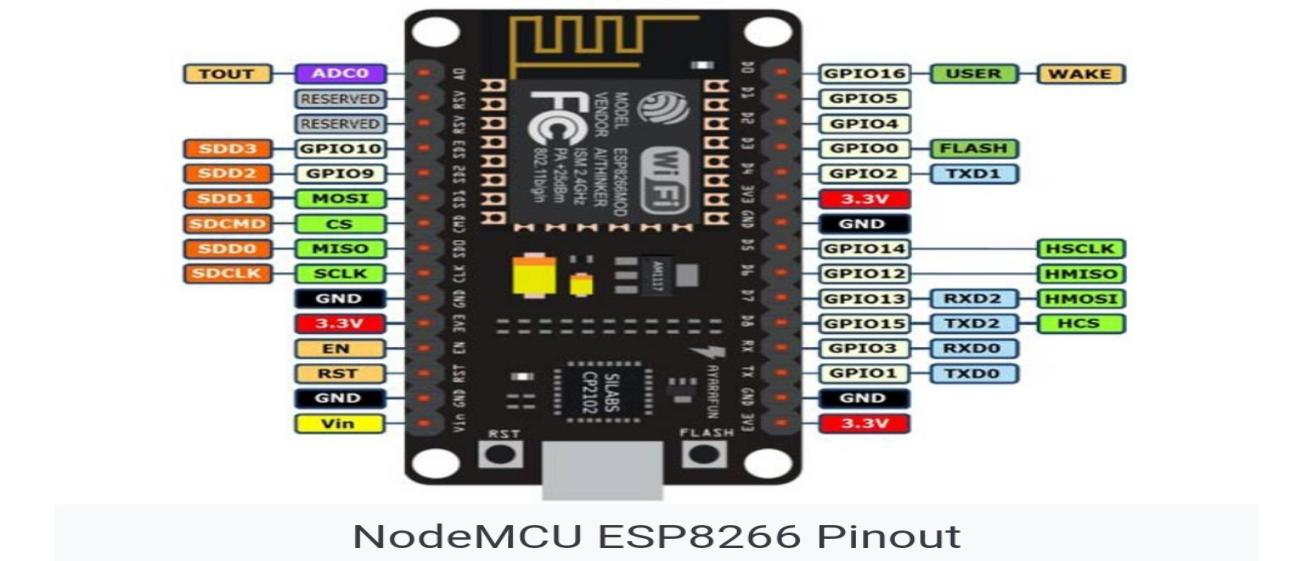
**3.1.1 NODEMCU SPECIFICATIONS**:

Fig 3.2

The above shown fig 3.2 shows the pin configuration of NodeMCU

* 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

### 3.1.2 PERIPHERALS:

#### **GPIO**

The ESP8266EX has 17 GPIO pins. But not all them available for the user as some of these are used for their alternative functions (like UART, SDIO, SPI etc.) in NodeMCU (ESP-12E Module).

#### **SPI**

There are two SPI Interfaces on ESP8266EX SoC (SPI and HSPI). Both support Master and Slave Operations. Master mode clock can be configured to 80 MHz while slave mode clock is up to 20 MHz.

* SCLK – GPIO6 (Not Available)
* MISO – GPIO7 (Not Available)
* MOSI – GPIO8 (Not Available)
* CS – GPIO11 (Not Available)
* HSPI\_CLK – GPIO14
* HSPI\_MISO – GPIO12
* HSPI\_MOSI – GPIO13
* HSPI\_CS – GPIO15

GPIO pins for SPI are multiplexed with some SDIO pins. Also, there is a 4MB SPI Flash on the ESP-12E Module connected through SPI Pins. So, you do not have access to SPI pins. You can only used HSPI pins for SPI communication.

#### **I2C**

Hardware I2C is not available in ESP8266 but is can implemented through software. GPIO4 and GPIO5 can be used as SDA and SCL as they do not have any other alternative functions.

#### **UART**

ESP8266EX has two hardware UARTs (UART0 and UART1) with baud rates up to 115200. In this, UART0 can be used for communication and it has data flow control as well. UART1 has only TX pin (its RX pin is used by SDD1) so, it can used for data logging.

* UART0 TX – GPIO3
* UART0 RX – GPIO1
* UART0 RTS – GPIO15
* UART0 CTS – GPIO13
* UART1 TX – GPIO2
* UART1 RX – GPIO8 (Not Available)

**ADDITIONAL FEATURES**

All the GPIO pins except GPIO16 support Interrupts.

There are two on-board LEDs on the NodeMCU Board. One LED is on the ESP-12E Module and is connected to GPIO2 and the other LED is on the NodeMCU Board and this is connected to GPIO16.

**3.1.3 NODEMCU INBUILT COMPONENTS:**



Fig 2.3

The above fig 3.3 shows the inbuilt components of NodeMCU

**AMS117 LDO (3.3V):**

A low-dropout regulator (LDO regulator) is a DC linear voltage regulator that can regulate the output voltage even when the supply voltage is very close to the output voltage.

The advantages of a low dropout voltage regulator over other DC to DC regulators include the absence of switching noise (as no switching takes place), smaller device size (as neither large inductors nor transformers are needed), and greater design simplicity (usually consists of a reference, an amplifier, and a pass element). The disadvantage is that, unlike switching regulators , linear DC regulators must dissipate power, and thus heat, across the regulation device in order to regulate the output voltage.

**CP2102 USB TO UART BRIDGE**:

The **CP210x** USB to UART Bridge Virtual COM Port (VCP) drivers are required for device operation as a Virtual COM Port to facilitate host communication with **CP210x** products. These devices can also interface to a host using the direct access **driver**. The fig 3.4 shows the CP2102 chip.

Fig 3.4

**UART(UNIVERSAL ASYNCHRONOUS RECIEVER TRANSMITTER):**

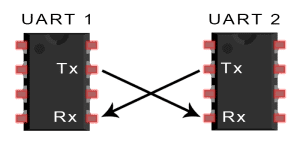
[](https://www.circuitbasics.com/wp-content/uploads/2016/01/Introduction-to-UART-Basic-Connection-Diagram.png)In UART communication, two UARTs communicate directly with each other. The transmitting UART converts parallel data from a controlling device like a CPU into serial form, transmits it in serial to the receiving UART, which then converts the serial data back into parallel data for the receiving device. Only two wires are needed to transmit data between two UARTs. Data flows from the Tx pin of the transmitting UART to the Rx pin of the receiving UART:

Fig 3.5

The above fig 3.5 shows the UART communication.UARTs transmit data asynchronously, which means there is no clock signal to synchronize the output of bits from the transmitting UART to the sampling of bits by the receiving UART. Instead of a clock signal, the transmitting UART adds start and stop bits to the data packet being transferred. These bits define the beginning and end of the data packet so the receiving UART knows when to start reading the bits.

When the receiving UART detects a start bit, it starts to read the incoming bits at a specific frequency known as the baud rate*.*Baud rate is a measure of the speed of data transfer, expressed in bits per second (bps).Both UARTs must operate at about the same baud rate. The baud rate between the transmitting and receiving UARTs can only differ by about 10% before the timing of bits gets too far off.

**DATA FRAME:**

The below shown fig3.6 shows the data frame of UART

Fig 3.6

**3.1.4 NODEMCU INTERNAL BLOCK DIAGRAM**

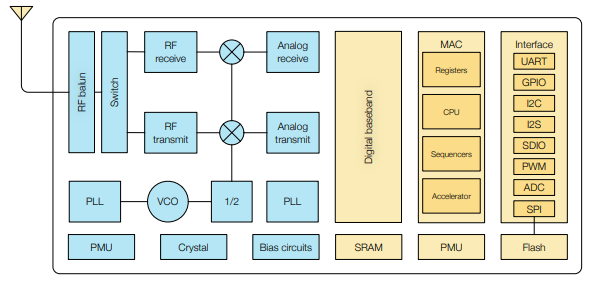


Fig 3.7

The fig 3.7 represents internal block diagram of ESP8266 chip.

**CPU:**

The ESP8266EX integrates a Tensilica L106 32-bit RISC processor, which achieves extra low power consumption and reaches a maximum clock speed of 160 MHz. The Real-Time Operating System (RTOS) and Wi-Fi stack allow 80% of the processing power to be available for user application programming and development. The CPU includes the interfaces as below:

* Programmable RAM/ROM interfaces (iBus), which can be connected with memory controller, and can also be used to visit flash.
* Data RAM interface (dBus), which can connected with memory controller.
* AHB interface which can be used to visit the register.

**MEMORY:**

ESP8266EX Wi-Fi SoC integrates memory controller and memory units including SRAM and ROM. MCU can access the memory units through iBus, dBus, and AHB interfaces. All memory units can be accessed upon request, while a memory arbiter will decide the running sequence according to the time when these requests are received by the processor.

According to our current version of SDK, SRAM space available to users is assigned as below.

* RAM size < 50 kB, that is, when ESP8266EX is working under the Station mode and connects to the router, the maximum programmable space accessible in Heap + Data section is around 50 kB.
* There is no programmable ROM in the SoC. Therefore, user program must be stored in an external SPI flash.

**EXTERNAL FLASH:**

ESP8266EX uses external SPI flash to store user programs, and supports up to 16 MB memory capacity theoretically.

The minimum flash memory of ESP8266EX is shown below:

* OTA disabled: 512 kB at least
* OTA enabled: 1 MB at least

**CLOCK:**

**HIGH FREQUENCY CLOCK:**

The high frequency clock on ESP8266EX is used to drive both transmit and receive mixers.

This clock is generated from internal crystal oscillator and external crystal. The crystal frequency ranges from 24 MHz to 52 MHz. The internal calibration inside the crystal oscillator ensures that a wide range of crystals can be used, nevertheless the quality of the crystal is still a factor to consider to have reasonable phase noise and good Wi-Fi sensitivity.

**RADIO:**

ESP8266EX radio consists of the following blocks.

* 2.4 GHz receiver
* 2.4 GHz transmitter
* High speed clock generators and crystal oscillator
* Bias and regulators
* Power management

**2.4 GHz RECEIVER:**

The 2.4 GHz receiver down-converts the RF signals to quadrature baseband signals and converts them to the digital domain with 2 high resolution high speed ADCs. To adapt to varying signal channel conditions, RF filters, automatic gain control (AGC), DC offset cancelation circuits and baseband filters are integrated within ESP8266EX.

**2.4 GHz TRANSMITTER:**

The 2.4 GHz transmitter up-converts the quadrature baseband signals to 2.4 GHz, and drives the antenna with a high-power CMOS power amplifier. The function of digital calibration further improves the linearity of the power amplifier, enabling a state of art performance of delivering +19.5 dBm average TX power for 802.11b transmission and +18 dBm for 802.11n (MSC0) transmission.

Additional calibrations are integrated to offset any imperfections of the radio, such as:

• Carrier leakage

• I/Q phase matching

• Baseband nonlinearities

These built-in calibration functions reduce the product test time and make the test equipment unnecessary.

**CLOCK GENERATOR:**

The clock generator generates quadrature 2.4 GHz clock signals for the receiver and transmitter. All components of the clock generator are integrated on the chip, including all inductors, varactors, loop filters, linear voltage regulators and dividers. The clock generator has built-in calibration and self test circuits. Quadrature clock phases and phase noise are optimized on-chip with patented calibration algorithms to ensure the best performance of the receiver and transmitter.

**Wi-Fi:**

ESP8266EX implements TCP/IP and full 802.11 b/g/n WLAN MAC protocol. It supports Basic Service Set (BSS) STA and SoftAP operations under the Distributed Control Function (DCF). Power management is handled with minimum host interaction to minimize activeduty period.

**Wi-Fi RADIO AND BASEBAND:**

The ESP8266EX Wi-Fi Radio and Baseband support the following features:

802.11 b and 802.11 g

* 802.11 n MCS0-7 in 20 MHz bandwidth
* 802.11 n 0.4 μs guard-interval
* up to 72.2 Mbps of data rate
* Receiving STBC 2 x 1
* Up to 20.5 dBm of transmitting power
* Adjustable transmitting power

**Wi-Fi MAC:**

The ESP8266EX Wi-Fi MAC applies low-level protocol functions automatically, as follows:

* 2 × virtual Wi-Fi interfaces
* Infrastructure BSS Station mode/SoftAP mode/Promiscuous mode
* Request To Send (RTS), Clear To Send (CTS) and Immediate Block ACK
* Defragmentation
* CCMP (CBC-MAC, counter mode), TKIP (MIC, RC4), WEP (RC4) and CRC
* Automatic beacon monitoring (hardware TSF)
* Dual and single antenna Bluetooth co-existence support with optional simultaneous receive (Wi-Fi/Bluetooth) capability

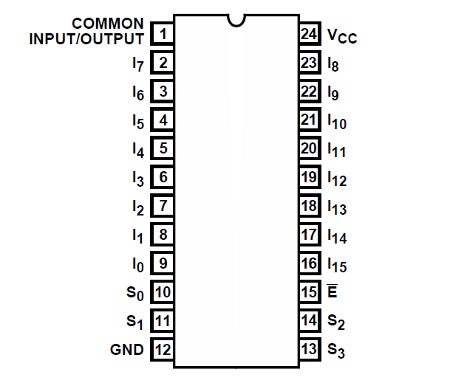
**POWER MANAGEMENT:**

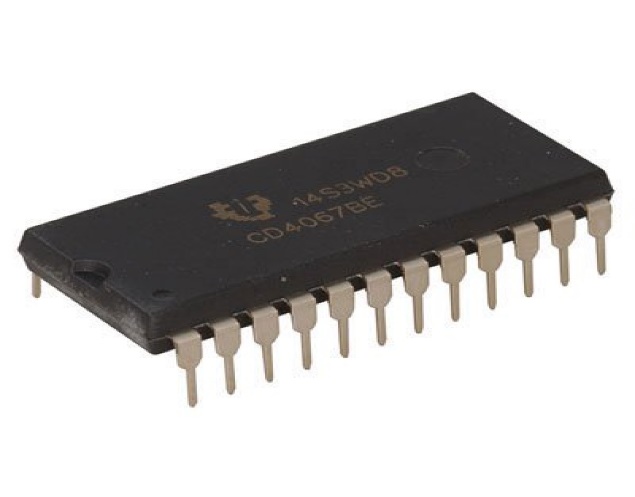
ESP8266EX is designed with advanced power management technologies and intended for mobile devices, wearable electronics and the Internet of Things applications.

The low-power architecture operates in the following modes:

* Active mode: The chip radio is powered on. The chip can receive, transmit, or listen.
* Modem-sleep mode: The CPU is operational. The Wi-Fi and radio are disabled.
* Light-sleep mode: The CPU and all peripherals are paused. Any wake-up events (MAC, host, RTC timer, or external interrupts) will wake up the chip.
* Deep-sleep mode: Only the RTC is operational and all other part of the chip are powered off.

**3.2 ANALOG 16 CHANNEL MULTIPLEXER**



  
 Fig 3.8

**74HC4067 IC:**

* It is an 16 channel analog multiplexer and demultiplexer that can route both analog and digital signals in both directions
* 1 of 16 channels can be routed to 1 output or 1 input can be routed 1 of 16 output channels
* Fig 3.8 shows the multiplexer picture.

**PROCESS:**

* Four address lines [s0-s3] select one of the 16 channels and connects it to the input/output pin[sig]
* It uses binary addressing, so address 0000 is channel 0, address 1111 is channel 15
* When a channel is ON, it has a resistance of about 70 ohms which allows signals to flow both ways
* With a 5v power supply, we measured about 60 ohms
* Maximum current is 25mA through any of the channels
* There is an enable pin[EN] that is active LOW and defaults to that value
* When LOW, the device enables the channel selected by the address lines
* If EN is pulled HIGH,all channels are disabled

**PIN CONFIGURATION:**

* 1 \*8 Header location
* SIG = signal input/output.This will usually connect to an analog input or digital i/o on the microcontroller
* S3= binary address bit 3. The address bits connect to 4 digital outout pins on the microcontroller to select channel
* S2= binary address bit 2
* S1= binary address bit 1
* S0= binary address bit 0
* EN=enable.Internally pulled LOW to enable by default.can be pulled HIGH to disable all channels
* VCC=2 to 6V.Usually connected to 5V or 3.3V to match the microcontroller power
* GND = ground, must be common with the microcontroller
* 1\*16 header location
* C15 = channel 15
* C0 = channel 0

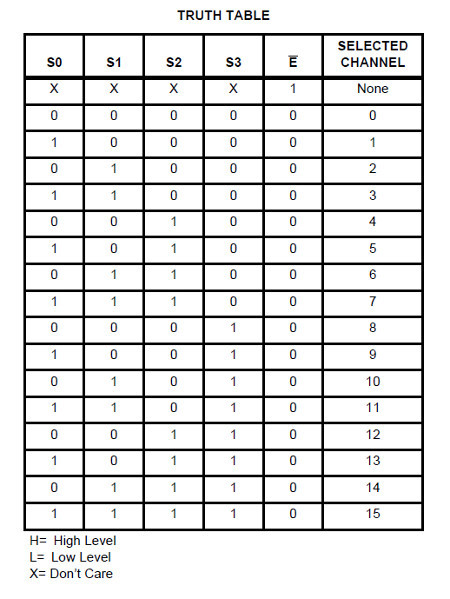
**TRUTH TABLE**:

Fig 3.9

The above shown fig 3.9 shows the truth table of multiplexer

**3.3** **MQ135 AIR QUALITY SENSOR:**

**Fig 3.10**

**SPECIFICATIONS:**

* The air quality sensor is also an MQ135 sensor for detecting venomous gases that are present in the air in homes and offices
* The air quality sensor detects ammonia,nitrogen oxide , smoke, CO2, and other harmful gases
* It has small potentiometer that permits adjustment of load resistance of the sensor circuit
* 5v power supply is used for air quality sensor
* Fig 3.10 shows the MQ135 sensor

**MQ135 GAS SENSOR AND ITS TYPES:**

* MQ135 alcohol sensor is a sno2 with a lower connectivity of clean air
* When the target explosive gas exists ,then the sensor conductivity increases more increasing more along with gas concentration rising levels
* By using simple electronic circuits, it converts the charge of conductivity to correspond output signal of gas concentration
* The MQ135 gas sensor has high sensitivity in ammonia sulfide, benzene steam, smoke, and in other harmful gas
* There are different types of alcohol sensors like MQ-2,MQ-3,MQ-4,MQ-5,MQ-6 etc

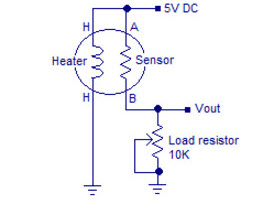
**CIRCUIT DIAGRAM**

Fig 3.11

* It consists of tin oxide[sno2],perspective layer inside aluminium oxide microtubes and a heating element inside tubular casting
* The end face of sensor is enclosed by a stainless steel net and the backside holds the connection terminals
* Ethyl alcohol present in the breath is oxidized into acetic acid passing through the heating element
* With ethyl alcohol cascade on the tin dioxide sensing layer,the resistance decreases
* By using the external load resistance the resistance variation is converted into a suitable voltage variation
* The fig 3.11 shows the internal circuit of MQ135

**CHARACTERISTICS:**

* It has long life and low cost
* It is a simple drive circuit

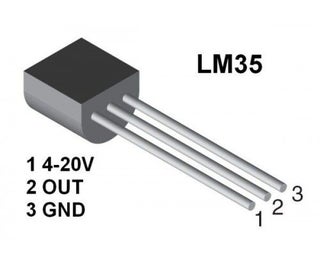
 **3.4** **LM35 TEMPERATURE SENSOR:**

Fig 3.12

* LM35 is a precision integrated circuit temperature sensor, whose output voltage varies,based on the temperature around it
* It is a small and cheap ic which can be interfaced with any microcontroller that has ADC function or any development platform like arduino
* It can be easily integrated and which can be used to measure temperature anywhere between -55c to 150c
* Fig 3.12 shows lm35 pin diagram

**PIN CONFIGURATION:**

* VCC- voltage is 5v
* Analog out- can range from -1v to 6v
* Ground – connected to ground

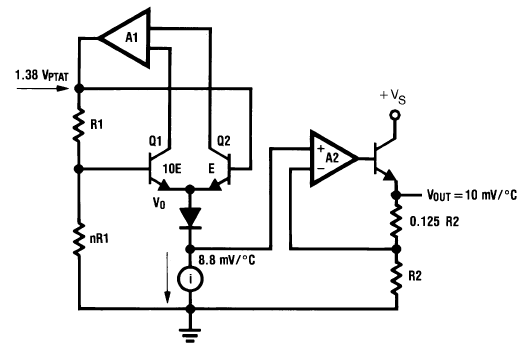
**INTERNAL CIRCUIT DIAGRAM**

Fig 3.13

* The first opamp A1 is configured as an accurate temperature sensor through feedback loop formed by a couple of BJTs configured as a current mirror
* The current mirror ensures a perfectly linear and a stabilized rate of temperature detection
* The sensed temperature is produced at the emitter side of the current mirror a rate of 8.8mv per degree Celsius
* The output is applied to buffer stage using another opamp A2 which is a configured as a high impedance voltage follower
* This A2 stage acts as a buffer to reinforce the temperature to voltage converter and presents it at the final output pin of the IC via another high impedance BJT stage configured as an emitter follower
* The final output thus becomes highly isolated from the actual temperature sensor stage and delivers highly accurate temperature sensing response
* The fig 3.13 shows the internal circuitry of lm35

**FEATURES**

* Minimum and maximum input voltage is 35v and-2v respectively.
* Can measure temperature ranging from -55c to 150c
* Output voltage is directly propotional to temperature there will be a rise of 10mv for every 1c rise in temperature
* 0.5c accuracy
* Drain current is less than 60uA
* Low cost temperature sensor
* Small and hence suitable for remote applications
* Available in TO-92,To-220,To-CAN and SOIC package

**HOW TO USE LM35:**

* Power the IC by applying a regulated voltage like+5v to the input pin and connected the ground pin to the ground of the circuit
* The voltage can be converted to temperature using below formulae

Vout = 10mv/c \* T

Where

* Vout is the LM35 output voltage
* T is the temperature

**APPLICATIONS:**

* Measuring temperature of a particular environment
* Providing thermal shutdown for a circuit/component
* Monitoring battery temperature
* Measuring temperatures for HVAC applications

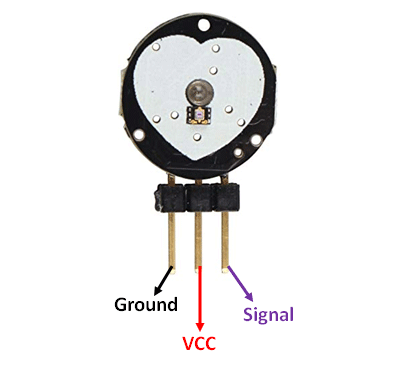
**3.5 PULSE RATE SENSOR AMPED :**

Fig 3.14

* The pulse sensor amped is a plug and play heart rate sensor for arduino
* It essentially combines a simple optical heart rate sensor with amplification and noise cancellation circuitry making it fast and easy to get reliable pulse readings
* Also, it sips power with just 4ma current draw at 5v so its great for mobile applications
* It is easy to get reliable pulse readings
* Fig 3.14 shows the pulse sensor amped pin diagram.

**PIN CONFIGURATION:**

* Ground – connected to ground
* VCC – connect to 5v or 3.3v
* Signal – pulsating output signal

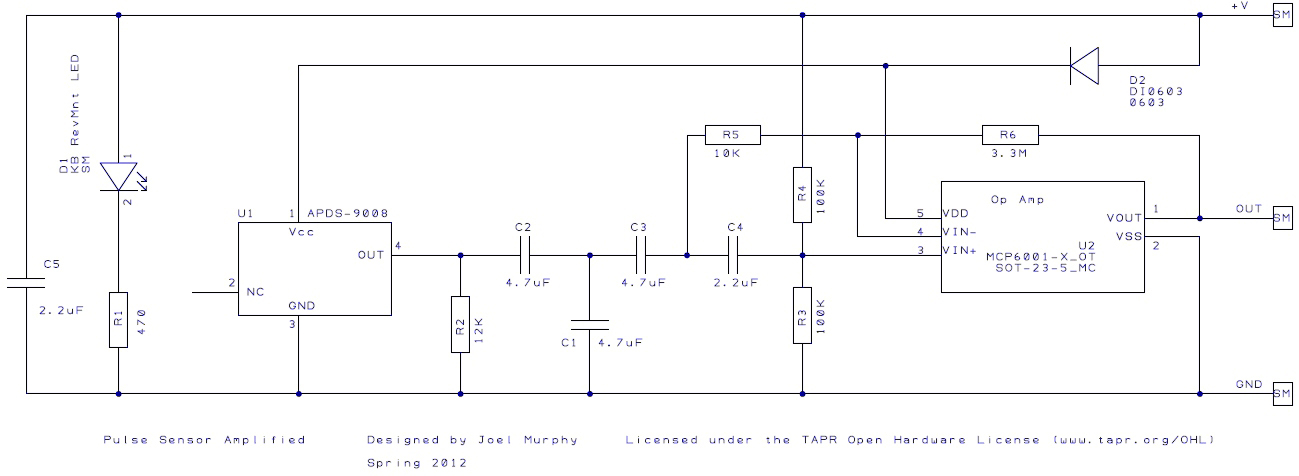
**CIRCUITDIAGRAM:** 

Fig 3.15

**WORKING:**

* This sensor has two surfaces, on the first surface ,the LIGHT EMITTING DIODE and ambient light sensor is connected
* Similarly , on the second surface , the circuit is connected which is accounting for the noise cancellation and amplifications
* The LED is located above a vein in a human body like ear tip or finger tip,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 the heart rates also
* Fig 3.15 shows the circuit diagram diagram of pulse sensor amped.

**SPECIFICATIONS:**

* This is a heart beat detecting and biometric pulse rate sensor
* Its diameter is 0.625
* Its thickness is 0.125
* The operating voltage is ranges +5v or +3.3v
* This is a plug and play type sensor
* The current utilization is 4mA
* Includes the circuits like amplification and noise cancellation
* This pulse sensor is not approved by the FDA or medical. So it is used in student level projects,not for the commercial purpose in health issues applications

**HOW TO USE PULSE SENSOR ARDUINO:**

* This sensor used in straight forward,however connecting it in the correct way matters
* Because all types of electronics components are directly exposed to the sensor
* So it is mandatory to envelop this sensor by using hot glue, vinyl strip otherwise other types of non-conductive materials
* This sensor can be used by connecting it to the arduino board
* Once the sensor is connected to the development board such as arduino,then we can use the readily accessible arduino code to make things easier

**APPLICATIONS:**

* 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

**CHAPTER - 4**

**SOFTWARE REQUIREMENTS**

**4.1 ARDUINO IDE:**

The Arduino Integrated Development Environment - or Arduino Software (IDE) - contains a text editor for writing code, a message area, a text console, a toolbar with buttons for common functions and a series of menus. It connects to the Arduino and Genuino hardware to upload programs and communicate with them.

### 4.1.1 CODING IN ARDUINO IDE FOR NODEMCU:

Note: When you use the NodeMCU with the Arduino IDE, it will write directly to the firmware, of NodeMCU erasing the original firmware, So if you want back the Lua SDK, use the “flasher” to re-install the firmware. You can download the flasher from their Github page https://github.com/nodemcu/nodemcu-flasher.

**Step 1**: Connect the NodeMCU with your PC or laptop with a micro USB cable.

**Step 2**: Download and install the drivers, You can download the driver for Mac, Linux or windows from this link https://github.com/nodemcu/nodemcu-devkit/tree/master/Drivers.

**Step 3**: Open Your Arduino IDE, then open **preference** from the **file** menu, thencopythislink  http://arduino.esp8266.com/stable/package\_esp8266com\_

index.json  to additional board manager URLs, as shown below in the screenshot, then click ok.

**Step 4:** Installing Board, Open board manager from tools -> board -> board manager.and search from "nodemcu"

Then select the latest version from the dropdown menu and click install and restart the Arduino IDE.

If everything is installed properly then you should be able to see the newly installed boards under tools -> board menu.

**3.1.2 TESTING**

Now let's test our setup by running a blink sketch in our NodeMCU.

**Step 1:**Open the example blink program from the "example for NodeMCU 1.0" section inside the example menu,

**Step 2:** Connect the NodeMCU with your computer using the micro USB cable.

**4.2 ADAFRUIT IO CLOUD SERVER**

Adafruit IO is a system that makes data useful. Our focus is on ease of use, and allowing simple data connections with little programming required

Adafruit IO is a platform designed to display, respond, and interact with your project's data. We also keep your data private (data feeds are private by default) and secure (we will never sell or give this data away to another company) for you.

**PROTOCOL USED:**

**MQQT(Message queuing Telemetry Transport)** is a publish/subscribe protocol that allows edge-of-network devices to publish to a broker. Clients connect to this broker, which then mediates communication between the two devices. When another client publishes a message on a subscribed topic, the broker forwards the message to any client that has subscribed.

The MQTT protocol defines two types of network entities: a message broker and a number of clients. An MQTT broker is a server that receives all messages from the clients and then routes the messages to the appropriate destination clients. An MQTT client is any device (from a micro controller up to a fully-fledged server) that runs an MQTT library and connects to an MQTT broker over a network.

Information is organized in a hierarchy of topics. When a publisher has a new item of data to distribute, it sends a control message with the data to the connected broker. The broker then distributes the information to any clients that have subscribed to that topic. The publisher does not need to have any data on the number or locations of subscribers, and subscribers, in turn, do not have to be configured with any data about the publishers.

If a broker receives a message on a topic for which there are no current subscribers, the broker discards the message unless the publisher of the message designated the message as a retained message. A retained message is a normal MQTT message with the retained flag set to true. The broker stores the last retained message and the corresponding QoS for the selected topic. Each client that subscribes to a topic pattern that matches the topic of the retained message receives the retained message immediately after they subscribe. The broker stores only one retained message per topic.This allows new subscribers to a topic to receive the most current value rather than waiting for the next update from a publisher.

When a publishing client first connects to the broker, it can set up a default message to be sent to subscribers if the broker detects that the publishing client has unexpectedly disconnected from the broker. Clients only interact with a broker, but a system may contain several broker servers that exchange data based on their current subscribers' topics.

A minimal MQTT control message can be as little as two bytes of data. A control message can carry nearly 256 megabytes of data if needed. There are fourteen defined message types used to connect and disconnect a client from a broker, to publish data, to acknowledge receipt of data, and to supervise the connection between client and server.MQTT relies on the TCP protocol for data transmission. A variant, MQTT-SN, is used over other transports such as UDP or Bluetooth.

MQTT sends connection credentials in plain text format and does not include any measures for security or authentication. This can be provided by using TLS to encrypt and protect the transferred information against interception, modification or forgery.The default unencrypted MQTT port is 1883. The encrypted port is 8883.

**MQTT BROKER**

The MQTT broker is software running on a computer (running on-premises or in the cloud), and could be self-built or hosted by a third party. It is available in both open source and proprietary implementations.

The broker acts as a post office, MQTT doesn’t use the address of the intended recipient but uses the subject line called “Topic”, and anyone who wants a copy of that message will subscribe to that topic. Multiple clients can receive the message from a single broker (one to many capability). Similarly, multiple publishers can publish topics to a single subscriber (many to one).

Each client can both produce and receive data by both publishing and subscribing, i.e. the devices can publish sensor data and still be able to receive the configuration information or control commands (MQTT is a bi-directional communication protocol). This helps in both sharing data, managing and controlling devices.

With MQTT broker architecture, the devices and application becomes decoupled and more secure. MQTT uses Transport Layer Security (TLS) encryption with user name, password protected connections, and optional certifications that requires clients to provide a certificate file that matches with the server’s. The clients are unaware of each others' IP address.

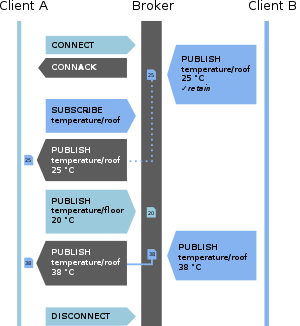
In case of a single source of failure, broker software and clients have an automatic handover to Redundant/automatic backup broker. The backup broker can also be set up to share the load of clients across multiple servers onsite, cloud, or the combination of both.

The broker can support both standard MQTT and MQTT for compliant specifications such as Sparkplug, can be done with same server, same time and with same levels of security.

The broker can store the data in the form of retained messages (need to subscribe with database client) so that new subscribers to the topic can get the last value straight away.

The broker also keeps track of all the session’s information as the devices goes on and off called “persistent sessions”.

The main advantages of MQTT broker are:

* Eliminates vulnerable and insecure client connections
* Can easily scale from a single device to thousands
* Manages and tracks all client connection states, including security credentials and certificates
* [](https://en.wikipedia.org/wiki/File:MQTT_protocol_example_without_QoS.svg)Reduced network strain without compromising the security (cellular or satellite network)

### MQTT broker

Fig 4.1

The fig 4.1 MQQT publish/subscribe connection diagram

**MESSAGE TYPES:**

**Connect**

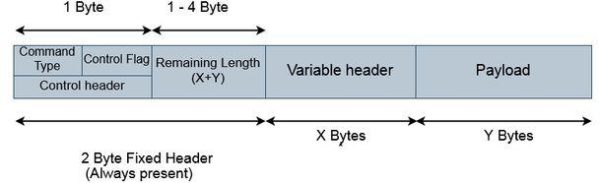
Waits for a connection to be established with the server and creates a link between the nodes.

### Disconnect

Waits for the MQTT client to finish any work it must do, and for the TCP/IP session to disconnect.

### Publish

Returns immediately to the application thread after passing the request to the MQTT client.

**4.2.1 MQQT DATA FRAME:**

The MQTT packet consists of 2-byte fixed header + a variable header and a payload. In this first 2-byte fixed header will be always present in all the packets and the other two, variable header and payload are not always present. The fig 3.2 shows data frame of MQQTprotocol

### Command type

Out of the two-byte fixed header, the first byte is the control field. This 8-bit control field Is divided into two 4 bit fields. The first 4 MSB bits are the command typefield. E.g. The value of the connect command is 1. That means for connect command the connect type field should be 1 which is 0001. For publish command the value is 3, therefore, the connect type field should be 0011.

**Control Flag bits**

The next 4 bits are the control flag bits and they are used by the publish command for the rest of the commands they are reserved and the value will be 0.  
0th bit denotes if the message that is published is retained the message.  
1st and 2nd bits are used to select the quality of service if it is 0 or 1 or 2.  
And the 3rd bit denotes if it is a duplicate message.

### Remaining Length

The second byte of the fixed header contains the remaining length which is, length of variable header + the length of the payload. Remaining length can use up to 4 bytes in which each byte uses 7 bits for the length and the MSB bit being a continuationflag.if the continuation flag bit of a byte is 1, it means the next byte is also part of the remaining length. And if the continuation flag bit is 0, it means that byte is the last one of the remaining length.

E.g. if the variable header length is 10 and the length of the payload is 20, the remaining length should be 30.

### Variable header

A variable header is not present in all the MQTT packets. Some MQTT commands or messages use this field to provide additional information or flags and they vary depending on the packet type. A packet identifier is common in most of the packets types. We will discuss in detail the variable header for the CONNECT packet below.

### Payload

In the end, the packet may contain a payload. Even the payload is optional and varies with the type of packet. This field usually  
contains the data which is being sent.

E.g. For CONNECT packet the payload is client ID and ‘username and password’ if they are present. And for PUBLISH packet, it is the message to be published.

### CONNECT packet

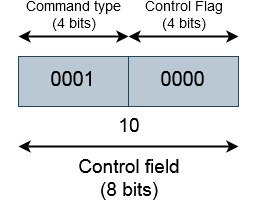
The first byte of the connect packet will be 10. Because the value of CONNECT command is 1, the first 4 MSB will be 1 and there are no flags so the next 4 bits will be 0. Fig 4.3 shows command packet of mqqt dataframe

Fig 4.3

The second byte should be the remaining length. which is the length of the variable header and length of the payload. Let us decide this length after completing the variable header and payload.  
You can see below the format of the variable header and payload for CONNECTpacket.

**4.2.2 ADAFRUIT IO SETUP**

* The first thing you will need to do is to login to Adafruit IO  and visit the **Settings**page.
* Click the **VIEW AIO KEY** button to retrieve your key.
* A window will pop up with your Adafruit IO. Keep a copy of this in a safe place. We'll need it later.

## **Creating the Digital Feed**

Next, you will need to create a feed called **Digital**. If you need help getting started with creating feeds on Adafruit IO, check out the Adafruit IO Feed Basics guide.

## **Adding the Toggle Block**

Next, add a new Toggle Block to a new or existing dashboard. Name the block whatever you would like, and set **On Text to a value of1** and **Off Text toa value of 0**. Make sure you have selected the Digitalfeed as the data source for the toggle.When you are finished editing the form, click Create Block to add the new block to the dashboard.

**CHAPTER -5**

**IMPLEMENTATION**

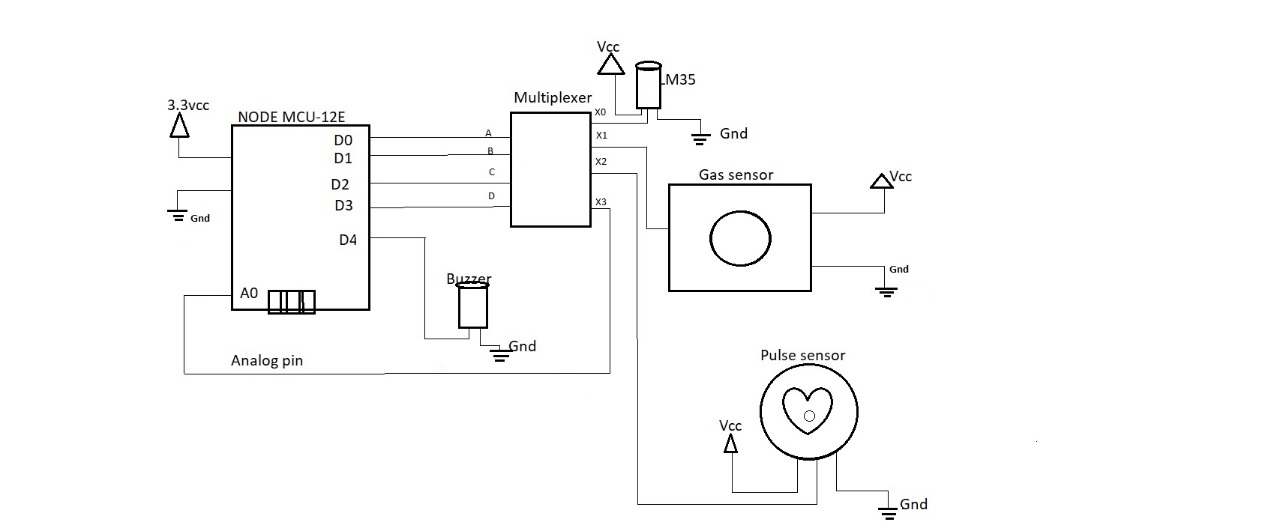
**5.1 HARDWARE IMPLEMENTATION:**

Fig 5.1

Fig 5.1 shows the hardware design of this project. NodeMCU is powered by a 3.7v li ion power supply.Connect all positive pins of the sensors to 3.3v power supply from NodeMCU.All ground pins are commonly grounded together for error free output.D0,D1,D2,D3 (GPIO) pins of nodemcu is connected to multiplexer’s A,B,C,D select pins.Connect buzzer to D4 pin of NodeMCU. X(common input/output) pin1 of multiplexer is connected to analog pin A0 of the multiplexer. The signal pin from lm35 is connected to pin 9 of multiplexer.

The signal pin from pulse sensor amped is connected to pin 8 of multiplexer.The analog signal from MQ135 gas sensor is connected to pin 7 of multiplexer.pin 15 of multiplexer is grounded through 10K ohm resistor

**5.2 SOFTWARE IMPLEMENTATIION:**

**ARDUINO IDE PROGRAM**

**#include <ESP8266WiFi.h>**

**#include "Adafruit\_MQTT.h"**

**#include "Adafruit\_MQTT\_Client.h"**

**#define S3 D0**

**#define S2 D1**

**#define S1 D2**

**#define S0 D3**

**#define analogpin A0**

**#define buzzer D4**

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* WiFi Access Point \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

**#define WLAN\_SSID "VikramSri"**

**#define WLAN\_PASS "Vikram@14"**

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Adafruit.io Setup \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

**#define AIO\_SERVER "io.adafruit.com"**

**#define AIO\_SERVERPORT 1883 // use 8883 for SSL**

**#define AIO\_USERNAME "were"**

**#define AIO\_KEY "aio\_knGZ33te6B9JQexV4qOpPD7YU6JN"**

**/\*\*\*\*\*\*\*\*\*\*\*\* Global State (you don't need to change this!) \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

**// Create an ESP8266 WiFiClient class to connect to the MQTT server.**

**WiFiClient client;**

**// or... use WiFiFlientSecure for SSL**

**//WiFiClientSecure client;**

**// Setup the MQTT client class by passing in the WiFi client and MQTT server and login details.**

**Adafruit\_MQTT\_Client mqtt(&client, AIO\_SERVER, AIO\_SERVERPORT, AIO\_USERNAME, AIO\_KEY);**

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Feeds \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

**// Setup a feed called 'photocell' for publishing.**

**// Notice MQTT paths for AIO follow the form: <username>/feeds/<feedname>**

**Adafruit\_MQTT\_Publish gas\_ppm= Adafruit\_MQTT\_Publish(&mqtt, AIO\_USERNAME "/feeds/sensor-1");**

**Adafruit\_MQTT\_Publish pulse\_rate = Adafruit\_MQTT\_Publish(&mqtt, AIO\_USERNAME "/feeds/sensor-2");**

**Adafruit\_MQTT\_Publish temperature = Adafruit\_MQTT\_Publish(&mqtt, AIO\_USERNAME "/feeds/sensor-3");**

**// Setup a feed called 'onoff' for subscribing to changes.**

**//Adafruit\_MQTT\_Subscribe onoffbutton = Adafruit\_MQTT\_Subscribe(&mqtt, AIO\_USERNAME "/feeds/onoff");**

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* SketchCode \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

**// Bug workaround for Arduino 1.6.6, it seems to need a function declaration**

**// for some reason (only affects ESP8266, likely an arduino-builder bug).**

**void MQTT\_connect();**

**void setup() {**

**Serial.begin(115200);**

**delay(10);**

**pinMode(analogpin, INPUT);**

**pinMode(S0,OUTPUT);**

**pinMode(S1,OUTPUT);**

**pinMode(S2,OUTPUT);**

**pinMode(S3,OUTPUT);**

**pinMode(buzzer,OUTPUT);**

**Serial.println(F("Adafruit MQTT demo"));**

**// Connect to WiFi access point.**

**Serial.println(); Serial.println();**

**Serial.print("Connecting to ");**

**Serial.println(WLAN\_SSID);**

**WiFi.begin(WLAN\_SSID, WLAN\_PASS);**

**while (WiFi.status() != WL\_CONNECTED) {**

**delay(500);**

**Serial.print(".");**

**}**

**Serial.println();**

**Serial.println("WiFi connected");**

**Serial.println("IP address: "); Serial.println(WiFi.localIP());**

**// Setup MQTT subscription for onoff feed.**

**//mqtt.subscribe(&onoffbutton);**

**}**

**uint32\_t x=0;**

**void loop() {**

**// Ensure the connection to the MQTT server is alive (this will make the first**

**// connection and automatically reconnect when disconnected). See the MQTT\_connect**

**// function definition further below.**

**MQTT\_connect();**

**// Now we can publish stuff!**

**Serial.print(F("\nSending Sensor's Value "));**

**digitalWrite(S0,LOW);**

**digitalWrite(S1,LOW);**

**digitalWrite(S2,LOW);**

**digitalWrite(S3,LOW);**

**Serial.print("gas ppm");Serial.println(analogRead(analogpin));**

**Serial.print("...");**

**int Value = analogRead(analogpin);**

**if (Value >150)**

**{**

**digitalWrite(buzzer, HIGH);**

**}**

**else**

**{**

**digitalWrite(buzzer, LOW);**

**}**

**if (! gas\_ppm.publish(Value)) {**

**Serial.println(F("Failed"));**

**} else {**

**Serial.println(F("OK!"));**

**}**

**digitalWrite(S0,HIGH);**

**digitalWrite(S1,LOW);**

**digitalWrite(S2,LOW);**

**digitalWrite(S3,LOW);**

**Value = analogRead(analogpin);**

**Serial.print("...");**

**Serial.print("pulse ppg ");Serial.println(----Value);**

**if (! pulse\_rate.publish(Value)) {**

**Serial.println(F("Failed"));**

**} else {**

**Serial.println(F("OK!"));**

**}**

**digitalWrite(S0,LOW);**

**digitalWrite(S1,HIGH);**

**digitalWrite(S2,LOW);**

**digitalWrite(S3,LOW);**

**//Serial.print("Sensor 3 ");Serial.println(analogRead(analogpin));**

**Serial.print("...");**

**int analogvalue=analogRead(analogpin);**

**float millivolts=(analogvalue/1024.0)\*2800;**

**float celsius=millivolts/10;**

**if (celsius >40)**

**{**

**digitalWrite(buzzer, HIGH);**

**}**

**else**

**{**

**digitalWrite(buzzer, LOW);**

**}**

**Serial.print("temperature degree celcius ");Serial.println(celsius);Serial.print("....");**

**if (! temperature.publish(celsius)) {**

**Serial.println(F("Failed"));**

**} else {**

**Serial.println(F("OK!")); }**

**delay(10000);**

**// ping the server to keep the mqtt connection alive**

**// NOT required if you are publishing once every KEEPALIVE seconds**

**/\***

**if(! mqtt.ping()) {**

**mqtt.disconnect();**

**}**

**\*/**

**}**

**// Function to connect and reconnect as necessary to the MQTT server.**

**// Should be called in the loop function and it will take care if connecting.**

**void MQTT\_connect() {**

**int8\_t ret;**

**// Stop if already connected.**

**if (mqtt.connected()) {**

**return;**

**}**

**Serial.print("Connecting to MQTT... ");**

**uint8\_t retries = 3;**

**while ((ret = mqtt.connect()) != 0) { // connect will return 0 for connected**

**Serial.println(mqtt.connectErrorString(ret));**

**Serial.println("Retrying MQTT connection in 5 seconds...");**

**mqtt.disconnect();**

**delay(5000); // wait 5 seconds**

**retries--;**

**if (retries == 0) {**

**// basically die and wait for WDT to reset me**

**while (1);**

**}**

**}**

**Serial.println("MQTT Connected!"); }**

**CHAPTER - 6**

**SYSTEM TESTING**

**6.1 TEST APPROACH:**

We will test the project in two stages: software and hardware. The software part is to be tested via the Arduino IDE, whereas the hardware part has to be tested physically. It is necessary to check whether the system is working properly or not. To check whether the readings are accurate, we will check the sensor values in the environment to be tested.

**6.2 FEATURES TO BE TESTED:**

After building the whole circuit we test it, testing procedure begins. This project should satisfy some features. Features to be tested as follows:

* The temperature sensor ,should give proper output. To check whether the output is accurate or not, the output of the sensor will be checked against a room temperature.
* The gas sensor ,should give proper output .To check whether the output

is accurate or not,the output will be checked against a gas detector.

* The pulse amped sensor output is checked with photoplethysmography signal(uses light souce to evaluate blood flow near the surface of skin).
* The NodeMCU board should show the sensors value in the serial monitor.
* The NodeMCU module should send sensors data after the specified delay. If the sensor data are reaching the cloud server , that means the Nodemcu module is working.
* The sensor data is stored in the server and can be vizualized from remote place.

**6.3 TESTING TOOLS AND ENVIRONMENT:**

* For testing of the project we require some tools, like to test Arduino program we require a software called Arduino IDE.
* Using this we can check the program that program is working properly or not.
* For hardware checking we require power supply and proper range of measurements .
* The NodeMCU should connect to the cloud serv er and should show the output.
* For this,theNodeMCU must connect first to the wifi hotspot.

**6.4 TEST CASES:**

In this section we discuss about the inputs, expected output, testing procedure.

**Inputs:**

* This project requires three inputs:
* Power supply:Power supply is the basic need of any electronic circuit. Here we use 5v dc battery to give power NodeMCU and sometimes we can give power directly from the computer.
* We can also power these circuits via two 1.5v batteries using a circuit divider.Distance,
* The sensor raw data will be the input of the NodeMCU circuit and will be gotten from the
* Gas sensor,Temperature sensor,pulse sensor amped.

**6.4.2 Expected output:**

* The expected output of this project should be a dashborad indicating different sensors values in adafruit cloud server .
* Also,it will also send the humidity and the temperature of the area. The output should also be seen on
* the serial monitor of the Arduino IDE, and also on the Cloud platform

**Testing procedure:**

* For testing first connect the circuit to the power supply is given to the Arduino using computer and it can be done by using battery. In this way the whole testing circuit is built.

**Summary of testing procedure:-**

* Connect the circuit according to the diagram
* Give power to the system.
* Vary gas,temperature,heart beat level for the sensor to give output.
* Get the output from the sensors.
* Send sensor data via the cloud server

**ADVANTAGES:**

* simple circuit and portable.
* uses only small battery for operation
* can help in continuous monitoring
* safety is ensured

**DISADVANTAGES**

* A special plastic casing is required to protect the circuitry from damage.

**CHAPTER – 7**

**RESULT AND DISCUSSION**

**HARDWARE RESULT:**

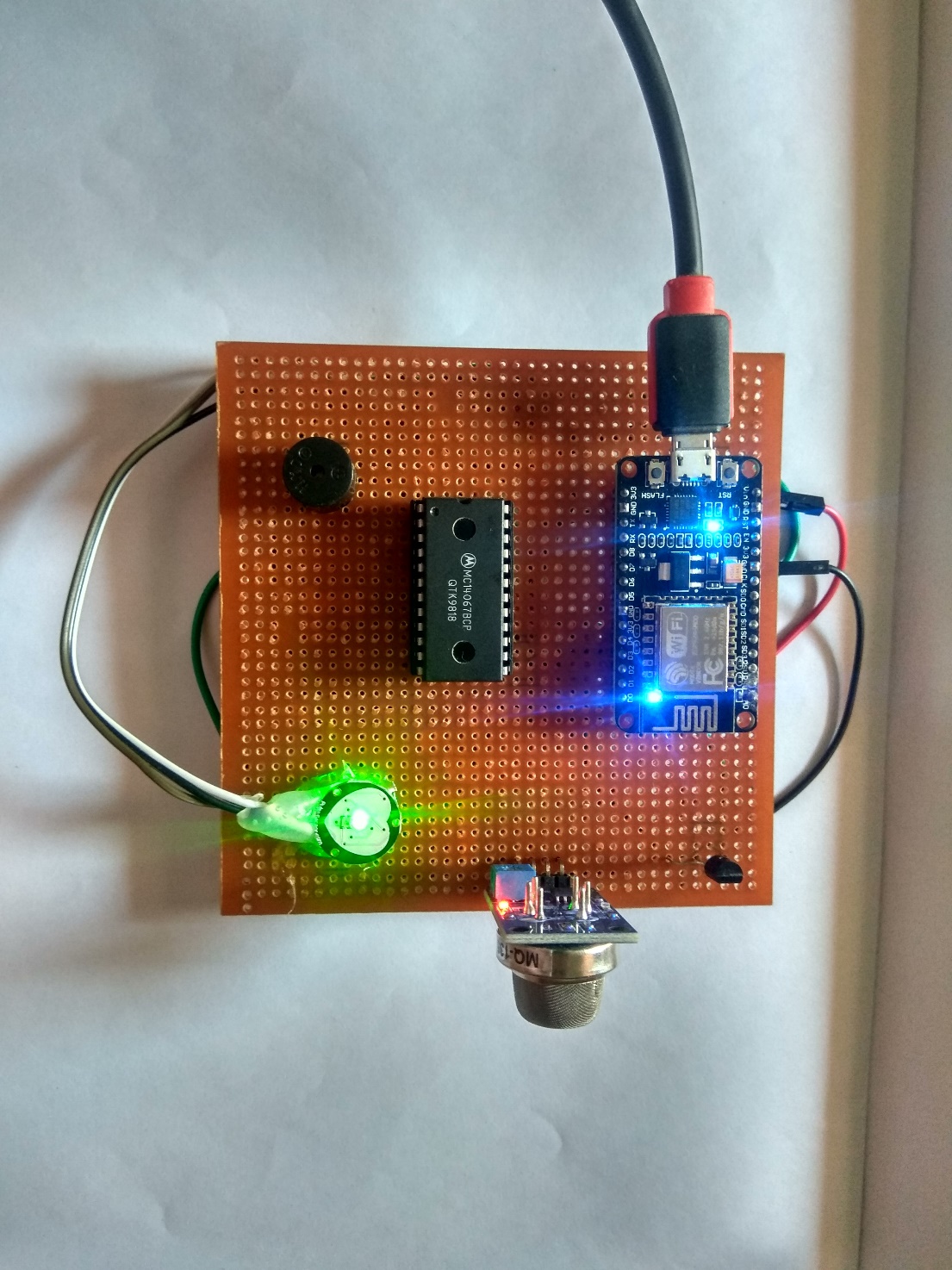
****

Fig 7.1

The above figure 7.1 represents breadboard connection of multiple sensors with NodeMCU and a buzzer for alarm

**ARDUINO SERIAL MONITOR RESULT:**

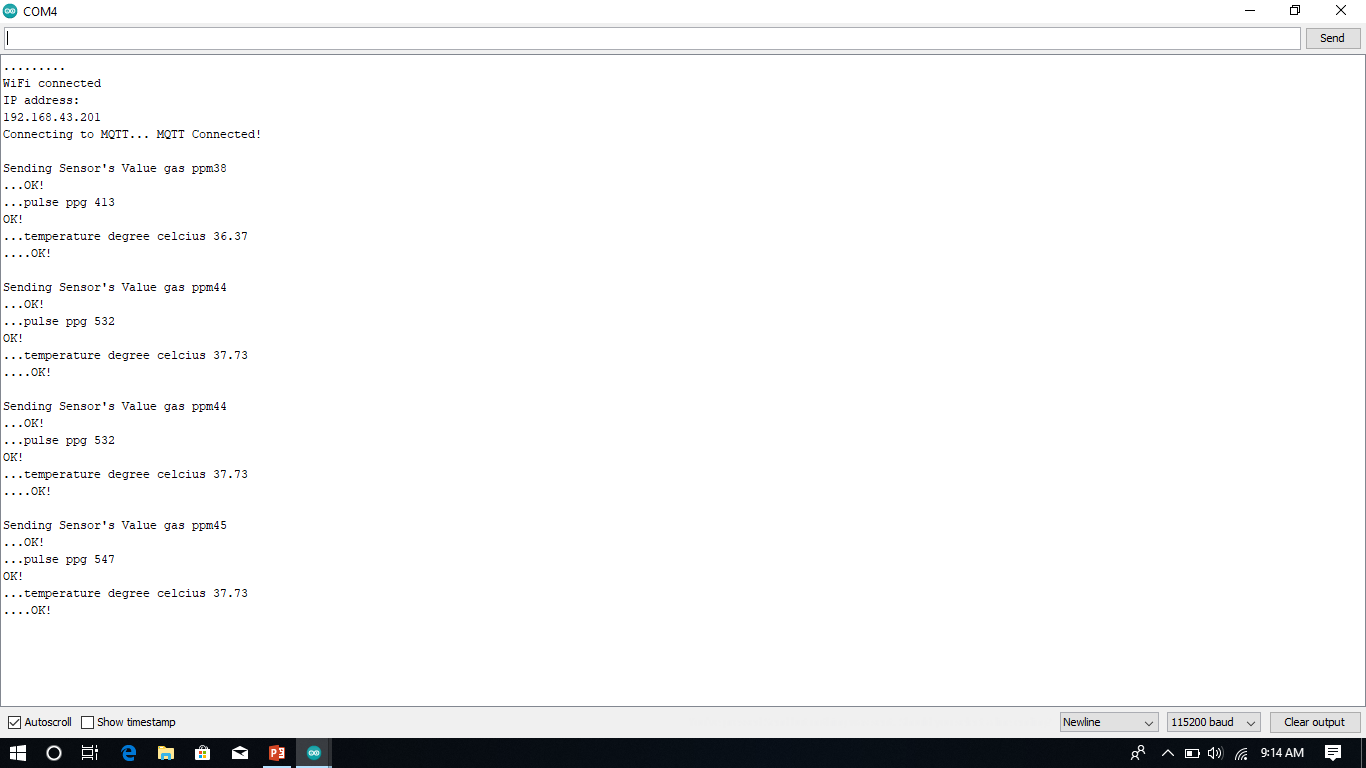
****

Fig 7.2

The above fig 7.2 represents sensors values are successfully uploading

into the Adafruit IO cloud server

**ADAFRUIT IO CLOUD SERVER RESULT:**

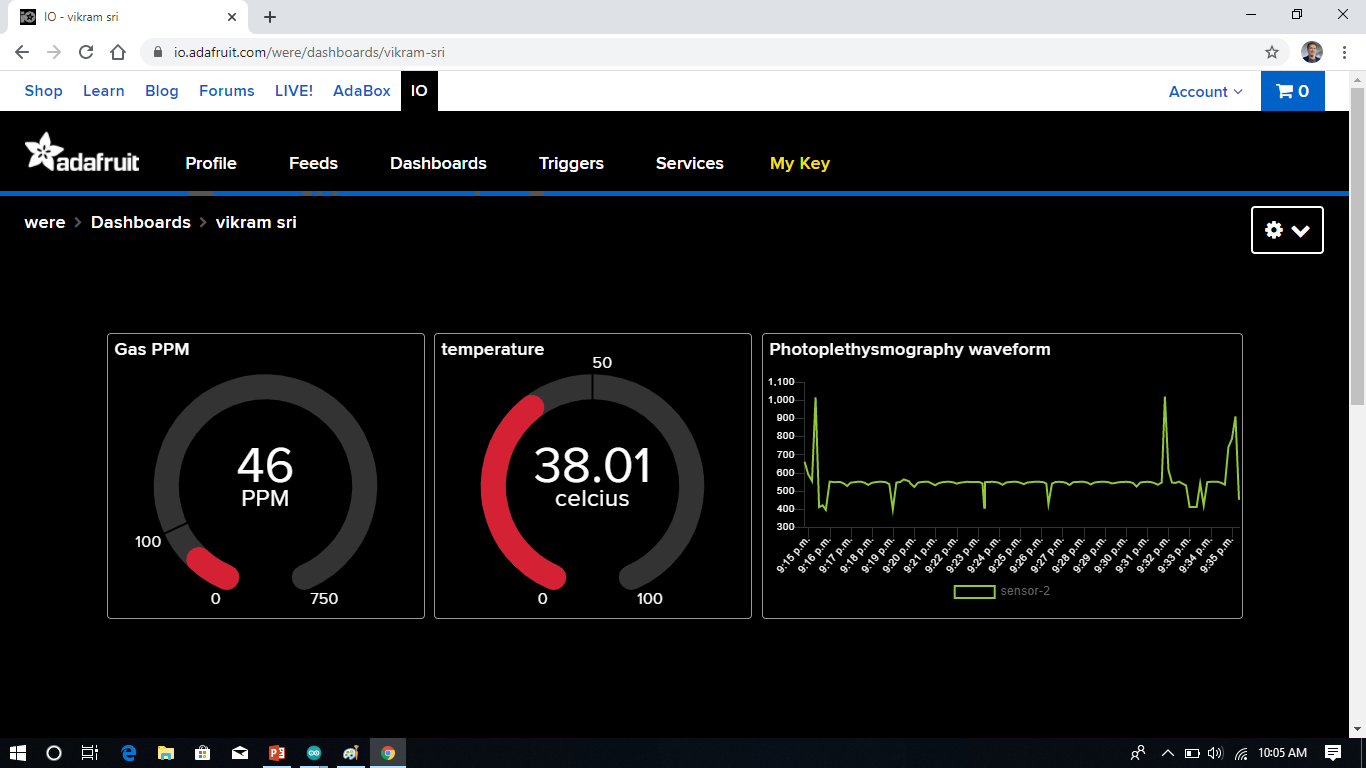
****

Fig 7.3

The above fig 7.3 represents live data of various sensor values displayed in the adafruit website that can be visualized from remote place

* First block - gas level in ppm
* Second block- body temperature in degree celcius
* Third block – photoplethymosgraphy signal of human body

X axis-time period

Y axis – pulse sensor data

**CHAPTER-8**

**CONCLUSION**

This device is designed keeping in mind , the measurement of necessary parameters,which needs to be monitored for unhindered safety of the workers in and around the chemical storage tanks .Apart from industrial safety ,this device finds major application in industries as well manholes.,gutters,deep well, sewage,etc. This IOT project can be further enhanced with the help of machine learning. This sensor data can be used for training a model and it can tested. Higher accuracy may be achieved with machine learning.

**REFERENCES**

1. PushpakumarR,RajivS.”IOT based smart drainage worker safety system”.2019 International journal of Innovative Technology and Exploring Engineering.ISSN-2278-3075,volume-8 Issue-8 (June).
2. Nitin Asthana,RidhimaBahl(2019).”IOT device for sewage gas monitoring and alert system”.
3. R.Rajalakshmi,J.Vidhya.”ToxicEnvironment Monitoring Using Sensors Based onArduino”.proceeding International Conference on Systems Computation Automation andNetworking.(2019).IEEE 978-1-7281-15245.
4. Sudhanshukumar,saketkumar,P.M.Tiwari,RajkumarViral.”Smartsafety monitoring system for sewage workers with two way communication”.2019 6thInternational Conference on SignalProcessing and Integrated Networks(SPIN) .