

TRANSFORMER OR GENERATOR HEALTH MONITORING REMOTELY OVER INTERNET

Submitted in partial fulfillment of the requirements for the award of

*Bachelor of Engineering Degree in
Electrical and Electronics Engineering*

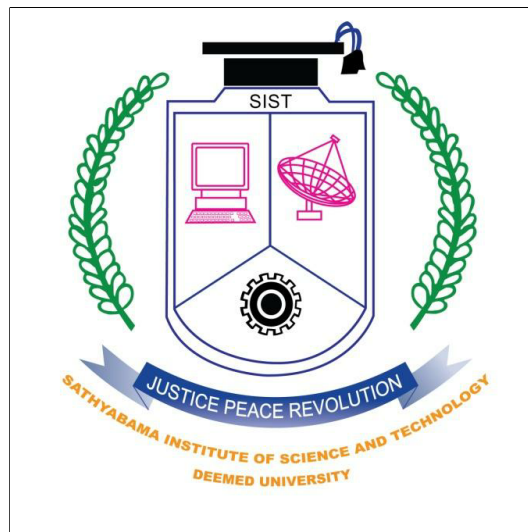
By

Fukhan Mohideen.M (39140014)

Arun Kumar P.B (39140706)

Muazim Ayas T.A (39140705)

Hrithik.S (39140703)



DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING
SCHOOL OF ELECTRICAL AND ELECTRONICS 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



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 ELECTRICAL AND ELECTRONICS ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of
Fukhan Mohideen.M (39140014),**Arun Kumar P.B** (39140706),**Muazim Ayas T.A** (39140705) and **Hrithik.S**
(39140703) who carried out the project entitled "**TRANSFORMER OR GENERATOR HEALTH
MONITORING REMOTELY OVER INTERNET**" under our supervision from November 2021 to May
2022.

DR. C. BARNABAS PAUL GLADY
Internal Guide

Dr. V. SIVACHIDAMBARANATHAN
Head of the Department

Submitted for Viva voce Examination held on _____

Name:

Signature:

Internal Examiner

External Examiner

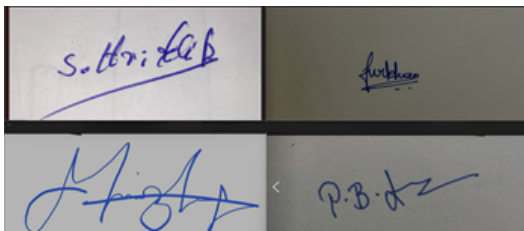
DECLARATION

We **Fukhan Mohideen.M** (39140014),**Arun Kumar P.B** (39140706),**Muazim Ayas T.A** (39140705) and **Hrithik.S** (39140703) hereby declare that the Project Report entitled **"TRANSFORMER OR GENERATOR HEALTH MONITORING REMOTELY OVER INTERNET"** done by us under the guidance of **Dr. J.BARNABAS PAUL GLADY** is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in Electrical and Electronics Engineering.

1.

2.

3.



4.

DATE:

PLACE:

SIGNATURE OF THE CANDIDATES

ACKNOWLEDGEMENT

We are pleased to acknowledge our sincere thanks to Board of Management of **SATHYABAMA** 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. Nandhita., Dean, School of Electrical and Electronics Engineering** and **Dr. V. Sivachidambaranathan, Head of the Department, Dept. of Electrical and Electronics 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. J.BARNABAS PAUL GLADY** for his valuable guidance, suggestions and constant encouragement paved way for the successful completion of our project work.

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

ABSTRACT:

Here this document presents design and implementation of a mobile embedded system to measure load currents, over voltage, transformer oil level and oil temperature. This is implemented by using on-line measuring system using Internet of Things (IOT), with single chip Arduino microcontroller and sensors. It is installed at the distribution transformer site. The output values of sensors are processed and recorded in the system memory. System

programmed with some predefined instructions to check abnormal conditions. If there is any abnormality on the system, details are automatically updated in the internet through serial communication. This Internet of Things (IOT) will help the utilities to optimally utilize transformers and identify problems before any catastrophic failure occurs. Thus online-measuring system is used to collect and analyze temperature data over time. So Transformer Health Measuring will help to identify or recognize unexpected situations before any serious failure which leads to a greater reliability and significant cost savings.

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGENO.
	ABSTRACT	V
	LIST OF FIGURES	Xi
	LIST OF PROGRAM	D
1	INTRODUCTION	
	1.1 ABSTRACT	1
	1.2 GENERAL INTRODUCTION	2
	1.3 LITERATURE REVIEW	3
	1.4 EXISTING SYSTEMS	6
	1.5 PROPOSED SYSTEM	6
2	PROJECT DESCRIPTION	
	2.1 BLOCK DIAGRAM	7
	2.2 SYSTEM REQUIREMENTS	8
	2.2.1 HARDWARE REQUIREMENTS	8
		8

2.2.2 SOFTWARE TOOLS	8
2.3 MODULES	9
2.4 MODULE DESCRIPTION	9
2.4.1 SENSOR INTERFACING	9
2.4.2 PREPARING POWER SUPPLY UNIT	9
2.4.3 MICRO-CONTROLLER PROGRAMMING	10
2.4.4 READING ANALOG DATA	10
2.4.5 TEST AND DEBUG	10
2.4.6 SUBMISSION	11
2.5 EMBEDDED SYSTEM	11
2.6 MICRO CONTROLLERS	12
2.7 INTERNET OF THINGS (IoT)	13
2.8 HARDWARE DESCRIPTION	13
2.8.1 ARDUINO	14
2.8.2 ARDUINO UNO INTERFACE WITH SENS & BUZZER	15
2.8.3 FEATURES	15
2.8.4 APPLICATIONS	15
2.8.5 ATMEGA328 IC	
ARCHITECTURE DIAGRAM	19
3.1 ARCHITECTURE DIAGRAM	20
3.2 PIN DESCRIPTION	20
3.2.1 Port B (PB7:0) XTAL1/XTAL2/TOSC1/TOSC2	

	20
3.2.2 Port C (PC5:0)	21
3.2.3 PC6/RESET	21
3.2.4 Port D (PD7:0)	21
3.2.5 AVCC	21
3.2.7 ADC7:6 (TQFP and QFN/MLF Package Only)	22
3.3 NODE MCU ESP32	23
3.3.1 Featured Solutions	24
3.3.2 Wi-Fi Key Features	24
3.3.3 BT Key Features	24
3.4 MCU and Advanced Features	25
3.4.1 CPU and Memory	25
3.4.2 Clocks and Timers	25
3.4.3 Advanced Peripheral Interfaces	26
3.4.4 Security	27
3.5 BLOCK DIAGRAM	28
3.5.1 Pin Layout	29
3.5.2 CPU	
3.5.3 Internal Memory	

4	TEMPERATURE SENSOR	30
	4.1 DESCRIPTION	
	4.2 TEMPERATURE SENSOR INTERFACE W RASPBERRY PI	
	4.3 FEATURES	
	4.4 APPLICATIONS	32
5	VOLTAGE SENSOR	33
	5.1 DESCRIPTION	
	6.2 AC VOLTAGE SENSOR	34
	6.3 DC VOLTAGE SENSOR	34
	6.4 FEATURES	35
	6.5 APPLICATIONS	36
6	CURRENT SENSOR	37
	6.1 DESCRIPTION	37
	62 CIRCUIT DIAGRAM OF CURRENT SENSOR	37
	6.3 FEATURES	
	6.4 APPLICATIONS	38
	ULTRASONIC SENSOR	39
7	8.1 DESCRIPTION	
	8.2 ULTRASONIC SENSOR INTERFACE WITH ARDUINO	40
	8.3 FEATURES	40
	8.4 APPLICATIONS	41

	POWER SUPPLY UNIT	42
8	8.1 DESCRIPTION	43
	8.2 ADAPTER (12V 1AMP)	43
	8.3 CIRCUIT DIAGRAM	44
	8.4 FEATURES	
	8.5 APPLICATIONS	45
	SOFTWARE DESCRIPTION	46
9	9.1 PROGRAMMING LANGUAGE	46
	9.2 ARDUINO IDE	46
	ARDUINO CODE	47
	10.1 ARDUINO code	48
	10.2 SERVER LINK	50
	CONCLUSION	67
	11.1 CONCLUSION	68
	11.2 REFERENCES	68
	SCREEN SHOT	

PAGE No.

LIST OF FIGURES TITLE

7

	BLOCK DIAGRAM	12
FIGURE No.fig	IOT DIAGRAM	13
2.1	ARDUINO DIAGRAM	14
2.2	ARDUINO UNO INTERFACE WITH SENSOR	
2.3	& BUZZER	16
2.4	ATmega328P IC diagram fig	19
2.5	ARCHITECTURE DIAGRAM	23
3.1	NODE MCU ESP32 circuit	28
3.2	CIRCUIT BLOCK DIAGRAM	29
3.3	Pin Layout	32
3.4	TEMPERATURE SENSOR	33
4.1	TEMPERATURE SENSOR INTERFACE	
4.2	WITH RASPBERRY PI circuit	35
	AC VOLTAGE SENSOR	36
5.1	DC VOLTAGE SENSOR	38
5.2	CURRENT SENSOR diagram	39
6.1	CIRCUIT DIAGRAM OF CURRENT	
6.2	SENSOR	41
	ULTRASONIC SENSOR	42
7.1	ULTRASONIC SENSOR INTERFACE WITH	44
7.2	ARDUINO	45
	ADAPTER (12V 1AMP)	70
8.1	CIRCUIT DIAGRAM	

8.2

SCREEN SHOT



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

**THIRD YEAR UG INTERDISCIPLINARY PROJECT PROGRESS
REPORT
FINAL REVIEW FORM (to be submitted on or before 27.04.2022)**

Batch No. : 12
Name : Fukhan Mohideen.M, Arun Kumar P.B, Muazim Ayas T.A and Hrithik.S
Department : Electrical and Electronic Engineering

Register No. : 39140014, 39140706, 39140705 and 39140703

Title of the project : TRANSFORMER OR GENERATOR HEALTH MONITORING REMOTELY OVER INTERNET

Name of the Internal Guide : Dr. J.BARNABAS PAUL GLADY

Present Status of Project Work : FULLY COMPLETED

Percentage of work completed : 100%

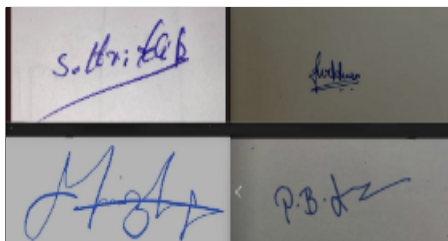
Future work to be carried out : NIL

Date: 22/04/2022

Review: FINAL

Name & Signature of Project Members

Name & Signature of the Internal guide



A large, stylized handwritten signature in blue ink, reading 'J. Barnabas', with a long horizontal line extending from the end of the signature.

CHAPTER 1

INTRODUCTION

1.1 ABSTRACT:

Here this document presents design and implementation of a mobile embedded system to measure load currents, over voltage, transformer oil level and oil temperature.

This is implemented by using on-line measuring system using Internet of Things (IOT), with single chip Arduino microcontroller and sensors. It is installed at the distribution transformer site. The output values of sensors are processed and recorded in the system memory. System programmed with some predefined instructions to check abnormal conditions. If there is any abnormality on the system, details are automatically updated in the internet through serial communication. This Internet of Things (IOT) will help the utilities to optimally utilize transformers and identify problems before any catastrophic failure occurs. Thus online-measuring system is used to collect and analyze temperature data over time. So Transformer Health Measuring will help to identify or recognize unexpected situations before any serious failure which leads to a greater reliability and significant cost savings.

1.2 GENERAL INTRODUCTION:

In recent times, the power system network is spread worldwide to transfer large amounts of power from generating plants to the end-users through transmission and distribution lines. The power transformer forms an important element in a power

transmission and distribution system. Monitoring of transformer is essential to assess the performance and to ensure its operational safety. Therefore, it is essential to monitor regularly the operational status of the loaded distribution transformer. Monitoring is the observation of transformer conditions and is of two types: offline and online. The difference between the two is that in off-line, the transformer is in the off state, and online, the transformer is in on state to measure the data . The transformer is to be protected from both internal and external faults. Among these temperature variations, oil level fall and load change require regular monitoring to safeguard the transformer. If transformers operate under healthy conditions, they have a long life; otherwise, their lives are significantly reduced. The main causes of failure in the transformer are overloading and ineffective cooling. Overloading of power transformers beyond the nameplate rating causes rise in temperature of both oil and windings. If the temperature rise of winding crosses the specified value, the insulation may get damaged. Continuous heating weakens the insulation and causes an accelerated reduction in transformer life.

1.3 LITERATURE REVIEW:

TITLE: Transformers monitoring using Arduino

AUTHOR: TEJAS PATIL,ONKAR SAKPAL ,OMKAR TUPE , SAURABH TIWARI

YEAR:2020

DESCRIPTION:

In this paper main objective is to design a protective circuit for distribution transformer, supported programmable controller to watch transformer ambient temperature, voltage level, current level and oil level by using various sensors. Here the most perspective is to beat or reduce the faults occurring in transformer because of various factor like over-voltage, over-current, temperature rise and oil-level decrement. Designing a value efficient and high performing system which is able to sense the faults and it'll display all the parameters of transformer on the application by means of WI-FI module (ESP8266) by using API technology (Application programming interface) that there's a problem within the perspective transformer in keeping with that user can check the status of the transformer

TITLE: Transformer Status Monitoring and Diagnosis Using IOT

AUTHOR:K.Nandini[1],V.Prabakaran[2],V.Prakash[3],R.Ranjithkumar[4],M.Vigneshkumar[5]

YEAR:2019

DESCRIPTION:

A recent huge interest in Machine to Machine communication is known as the Internet of Things (IOT), to allow the possibility for autonomous devices to use Internet for exchanging the data. This work presents design and execution of real time monitoring and fault detection of transformer and record key operation indicators of a dispersion transformer like load current, voltage, transformer oil and encompassing temperatures and humidity. They have to look at it continuously by using this project it can minimize working efforts and improve accuracy, stability, efficiency in this project, sensors are used to sense the main parameters of equipment such as voltage, current(over voltage, under voltage, over current) this sensed data is sent to microcontroller and this

controller checks parameter limits which further send to the IOT web server using IOT module of these data makes sure the right information is in hand to the operator and operator can make useful decisions before any catastrophic failure on basis of that data of parameters.

TITLE: Real Time Transformer Health Monitoring System using IOT

AUTHOR: D.Sarathkumar, Uvaraj.M, Kabilesh Kumar C V, Kalaiselvi A

YEAR:2018

DESCRIPTION:

The main aim of the paper is to acquire real-time data of transformer remotely over the internet falling under the category of Internet of Things (IOT). For this real-time aspect, we take one temperature sensor, one potential transformer and one current transformer for monitoring T, V, I data of the transformer and then send them to a remote location. These three analog values are taken in multiplexing mode and connected to a programmable microcontroller of 8051 families through an ADC 0808. They are then sent directly to a Wi-Fi module under TCP IP protocol to a dedicated IP that displays the data in real-time chart form in any web connected PC / Laptop for display in 3 different charts. So, This Transformer Health Measuring will help to identify or recognize unexpected situations before any serious failure which leads to a greater reliability and significant cost savings.

TITLE: Real Time Monitoring of Transformer using IOT

AUTHOR: Mrs. A. P. Khandait¹ , Swapnil Kadaskar² , Girish Thakare³

YEAR:2017

DESCRIPTION:

A recent huge interest in Machine to Machine communication is known as the Internet of Things (IOT), to allow the possibility for autonomous devices to use Internet for exchanging the data. This work presents design and execution of real time monitoring

and fault detection of transformer and record key operation indicators of a dispersion transformer like load current, voltage, transformer oil and encompassing temperatures and humidity. They have to look at it continuously by using this project it can minimize working efforts and improve accuracy, stability, efficiency in this project, sensors are used to sense the main parameters of equipment such as voltage, current(over voltage, undervoltage, over current) this sensed data is sent to microcontroller and this controller checks parameter limits which further send to the IOT web server Adafruit software using Wi-Fi module of these data makes sure the right information is in hand to the operator and operator can make useful decisions before any catastrophic failure on basis of that data of parameters.

1.4 EXISTING SYSTEMS:

1. Firing of transformer can easily occurs.
2. Not accurate.
3. Frequency interference in system.
4. Noise problem in network

1.5 PROPOSED SYSTEM:

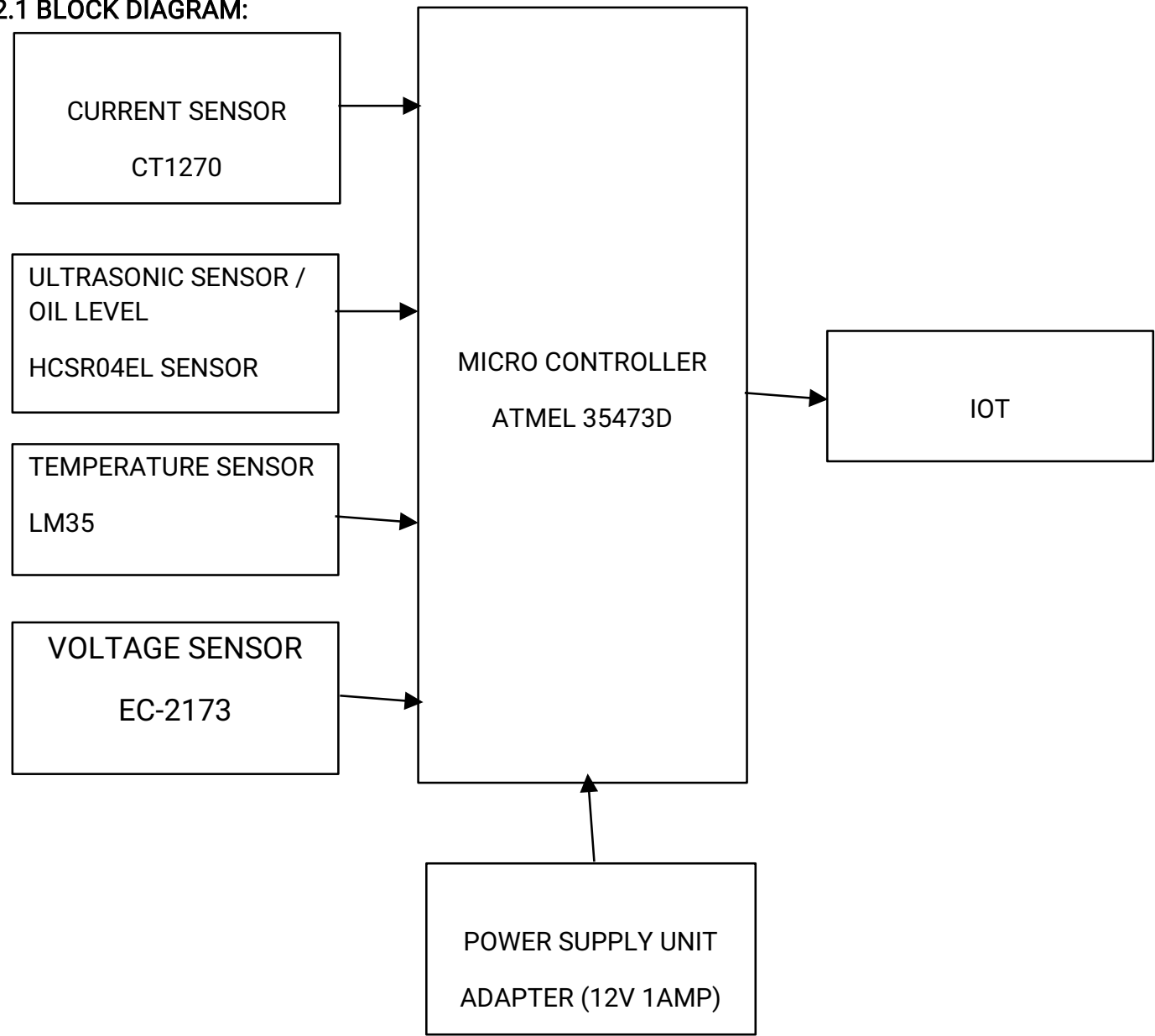
The main objective of this proposal is to acquire live data of transformer health remotely over the internet using Internet of Things (IOT) technology. We are going to monitor the transformer parameter such as temperature, current and oil level. These data will be sent over internet using MQTT protocol.

Avoiding equipment failures, reducing labor work, avoiding accidents due to faults, avoiding stealing of electricity etc. are the objectives of the system.

CHAPTER 2

TRANSFORMER OR GENERATOR HEALTH MONITORING REMOTELY OVER INTERNET

2.1 BLOCK DIAGRAM:



general block diagram fig 2.1

2.2 SYSTEM REQUIREMENTS:

2.2.1 HARDWARE REQUIREMENTS:

- MICRO CONTROLLER
- TEMPERATURE SENSOR
- CURRENT SENSOR
- ULTRASONIC SENSOR / OIL LEVEL SENSOR
- UART CABLE
- IOT
- POWER SUPPLY UNIT

2.2.2 SOFTWARE TOOLS:

- ARDUINO IDE
- EMBEDDED C

2.3 MODULES:

1. SENSOR INTERFACING
2. PREPARING POWER SUPPLY UNIT
3. MICRO-CONTROLLER PROGRAMMING
4. READING ANALOG DATA
5. TEST AND DEBUG
6. SUBMISSION

2.4 MODULE DESCRIPTION:

2.4.1 SENSOR INTERFACING:

Sensor interfacing is a mix of amplification, filtering, and other signal conditioning as well as analog-to-digital conversion. The analog-to-digital converter (ADC) may be in your microcontroller, but you will still need to make the sensor compatible with the ADC input.

2.4.2 PREPARING POWER SUPPLY UNIT:

A power supply unit (or PSU) converts mains AC to low-voltage regulated DC power for the internal components of a controller. A power supply is used to reduce the mains electricity at 240 volts AC down to something more useable, say 12 volts DC. There are two types of power supply, linear and switch mode. A linear power supply uses a transformer to reduce the voltage. The AC signal is rectified and regulated to produce a high DC voltage.

An AC adapter, AC/DC adapter, or AC/DC converter is a type of external power supply, often enclosed in a case similar to an AC plug. Adapters for battery-powered equipment may be described as chargers or rechargers (see also battery charger). AC adapters are used with electrical devices that require power but do not contain internal components to derive the required voltage and power from main power. The internal circuitry of an external power supply is very similar to the design that would be used for a built-in or internal supply.

2.4.3 MICRO-CONTROLLER PROGRAMMING:

A microcontroller is a programmable IC, capable of multiple functions depending on how it's programmed. Many different kinds of microcontrollers exist that offer a wide range of functionality. The versatility of the microcontroller is what makes it one of the most powerful tools in modern design. This guide will explain the basics of microcontrollers and how they are programmed.

2.4.4 READING ANALOG DATA:

The microcontroller of the board has a circuit inside called an analog-to-digital converter or ADC that reads this changing voltage and converts it to a number between 0 and 1023. When the shaft is turned all the way in one direction, there are 0 volts going to the pin, and the input value is 0. When the shaft is turned all the way in the opposite direction, there are 5 volts going to the pin and the input value is 1023. In between, `analog Read()` returns a number between 0 and 1023 that is proportional to the amount of voltage being applied to the pin.

2.4.5 TEST AND DEBUG:

Testing means verifying correct behavior. Testing can be done at all stages of module development: requirements analysis, interface design, algorithm design, implementation, and integration with other modules. In the following, attention will be directed at implementation testing. Implementation testing is not restricted to execution testing. An implementation can also be tested using correctness proofs, code tracing, and peer reviews, as described below.

Debugging is a cyclic activity involving execution testing and code correction. The testing that is done during debugging has a different aim than final module testing. Final module testing aims to demonstrate correctness, whereas testing during debugging is primarily aimed at locating errors. This difference has a significant effect on the choice of testing strategies.

2.4.6 SUBMISSION:

Submitting the prototype for the presentation to the faculty guide for review purpose

2.5 EMBEDDED SYSTEM:

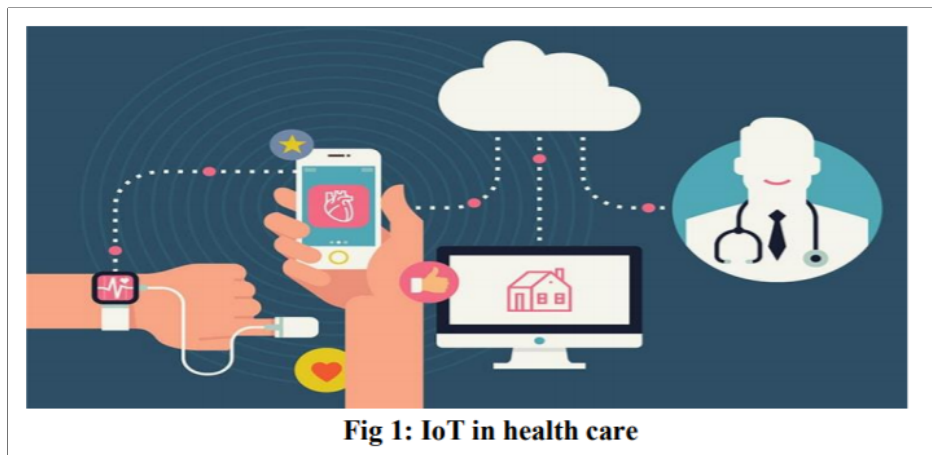
The Embedded systems are the devices which are used for a particular job. It is an omnipresent. We find them everywhere in our home, in our office, in shopping malls, in hospital in aircraft and so on. For example Fax machine, printer, mobile phone, etc. Every application has their own microprocessors and hardware to perform its own function. The processor is the heart of the embedded system. The core of the processors is called microcontrollers and the software are used. The main characteristic, however, is being keen to handle a particular task. Embedded systems are used for specific tasks so its size and cost are reduced in order to gain the function speed. Some implanted frameworks are mass created, profiting by economies of scale. Some embedded system has to run in extreme atmospheric conditions such as very high temperature and humidity. Many embedded systems work in very stumpy power. It can be driven through battery .

2.6 MICRO CONTROLLERS:

Microcontrollers are utilized in naturally controlled items and gadgets through a little PC. It is a solitary coordinated circuit. It comprises of a Processor center, memory, and programmable information/yield peripherals. It's intended for gadgets, remote controls, office machines, apparatuses, power instruments, toys and other inserted systems. Some microcontrollers may utilize 4-bit words for low consumption . They will work for the most part can hold usefulness while waiting for an occasion, for example, a catch press or other interfere with power utilization whereas dozing may well be merely Nano watts, creating a big range of them applicable for sturdy battery applications. Completely different microcontrollers could serve basic jobs, wherever they will got to act more and more sort of processor, with higher speeds and utility

2.7 INTERNET OF THINGS (IoT):

The IoT is the strategy for gadgets that encase hardware, and network, which enables these devices to fix, act together and switch information. IoT incorporates broadening Internet beneficial than standard gadgets, for example, work areas to any decision of generally stupid or non web get to material gadgets and on a day by day source objects. Inserted through innovation, these gadgets can banter and coordinate over the Internet, and they can be a little checked and restricted .



iot diagram fig 2.2

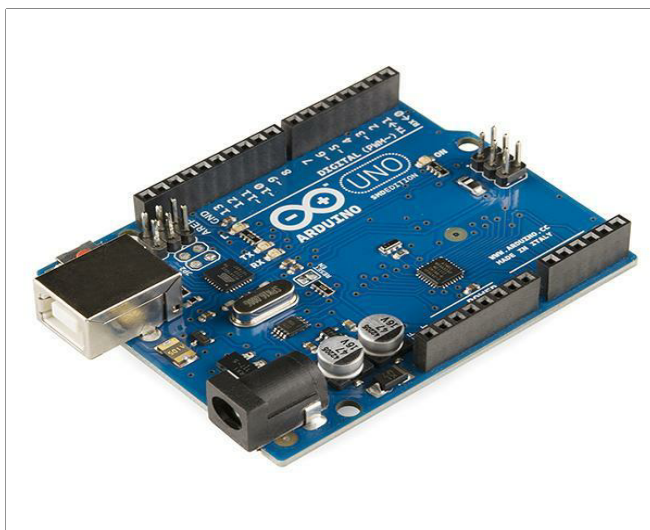
2.8 HARDWARE DESCRIPTION:

2.8.1 ARDUINO:

DESCRIPTION :

Arduino is an open source, computer hardware and software company, project, and user community that designs and manufactures Single-board microcontrollers and microcontroller kits for building digital devices and interactive objects that can sense and control objects in the physical world.

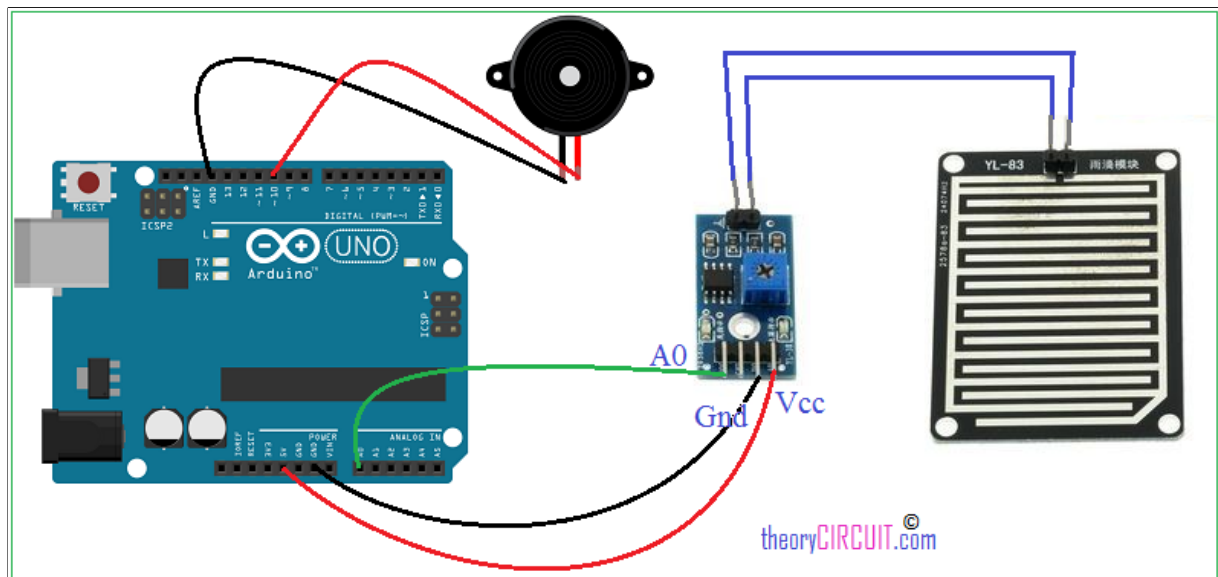
Arduino is an open-source electronics platform based on easy-to-use hardware and software. Arduino boards are able to read inputs - light on a sensor, a finger on a button, or a Twitter message - and turn it into an output - activating a motor, turning on an LED, publishing something online. The Arduino Uno is a microcontroller board based on the ATmega328 (datasheet). It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz crystal connection, a USB connection, a power jack, and a reset button.



ARDUINO diagram fig 2.3

Arduino is an open-source project that created microcontroller-based kits for building digital devices and interactive objects that can sense and control physical devices. The project is based on microcontroller board

These systems provide sets designs, produced by several vendors, using various microcontrollers. of digital and analog input/output (I/O) pins that can interface to various expansion boards (termed shields) and other circuits. The boards feature serial communication interfaces, including Universal Serial Bus (USB) on some models, for loading programs from personal computers. For programming the microcontrollers, the Arduino project provides an integrated development environment (IDE) based on a programming language named Processing, which also supports the languages C and C++.



ARDUINO UNO INTERFACE WITH SENSOR & BUZZER diagram fig 2.4

2.8.2 ARDUINO UNO INTERFACE WITH SENSOR & BUZZER:

Arduino Uno is a microcontroller board based on the ATmega328P. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz quartz crystal, a USB connection, a power jack, an ICSP header and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter. Arduino Uno has a number of facilities for communicating with a computer, another Arduino board, or other microcontrollers.

2.8.3 FEATURES:

- Microcontroller: ATmega328P
- Operating voltage: 5V
- Input voltage: 7-12V
- Flash memory: 32KB
- SRAM: 2KB
- EEPROM: 1KB

2.8.4 APPLICATIONS:

- Real time biometrics
- Robotic applications
- Academic applications

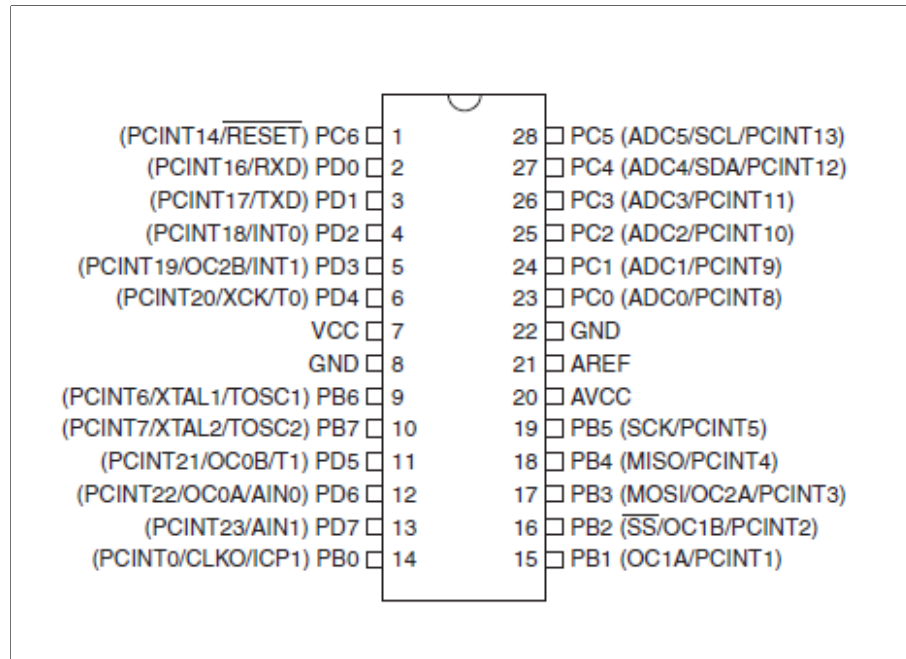
2.8.5 ATMEGA328 IC:

DISCRIPTION:

The ATmega328 is a single-chip microcontroller created by Atmel in the mega AVR family. The Atmel 8-bit AVR RISC-based microcontroller combines 32 kB ISP flash memory with read-while-write capabilities, 1 kB EEPROM, 2 kB SRAM, 23 general purpose I/O lines, 32 general purpose working registers, three flexible timer/counters with compare modes, internal and external interrupts, serial programmable USART, a byte-oriented 2-wire serial interface, SPI serial port, 6-channel 10-bit A/Dconverter (8-channels in TQFP and QFN/MLF packages), programmable watchdog timer with internal oscillator, and five software selectable power saving modes. The device operates between 1.8-5.5 volts. The device achieves throughput approaching 1 MIPS per MHz.



ATmega328P IC diagram fig 2.5

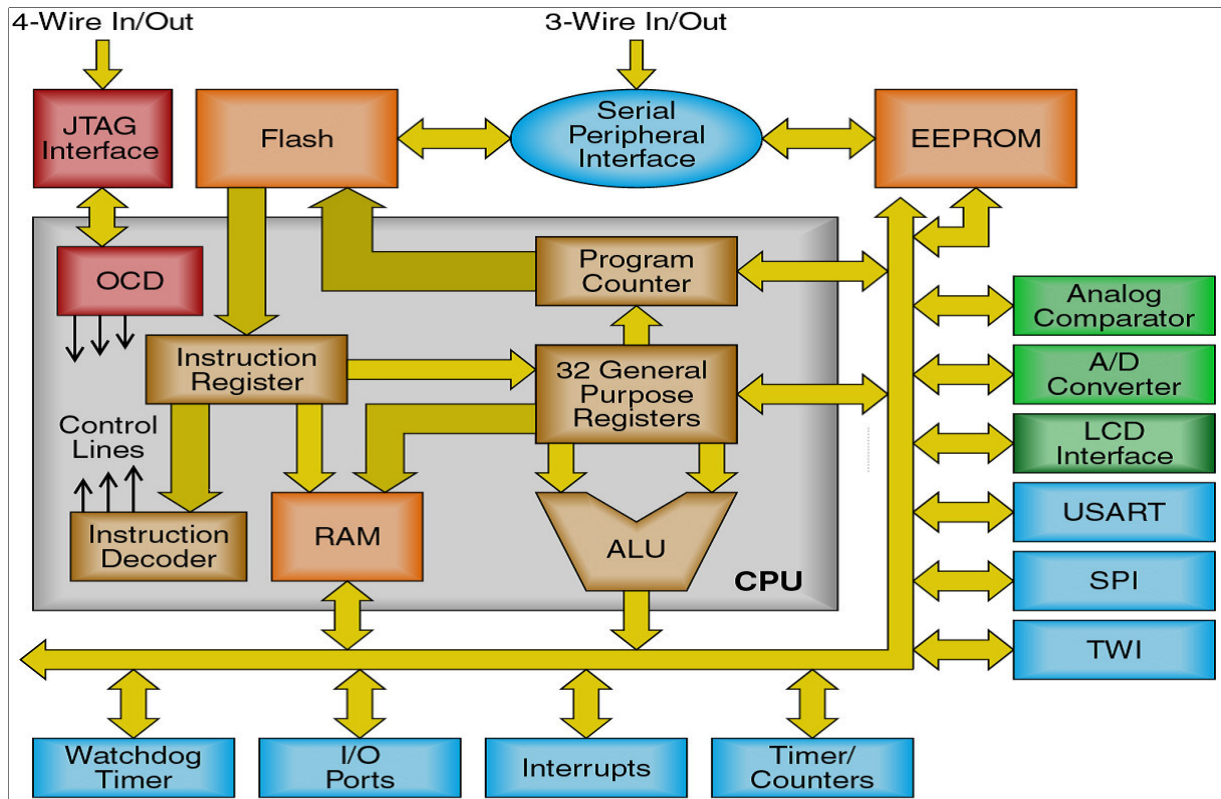


- High Performance, Low Power Atmel®AVR® 8-Bit Microcontroller Family

- Advanced RISC Architecture
- 131 Powerful Instructions
- Most Single Clock Cycle Execution
- 32 x 8 General Purpose Working Registers
- Fully Static Operation
- Up to 20 MIPS Throughput at 20MHz
- On-chip 2-cycle Multiplier
- High Endurance Non-volatile Memory Segments
- 4/8/16/32KBytes of In-System Self-Programmable Flash program memory
- 256/512/512/1KBytes EEPROM
- 512/1K/1K/2KBytes Internal SRAM
- Write/Erase Cycles: 10,000 Flash/100,000 EEPROM C(1)°C/100 years at 25° Data retention: 20 years at 85
- Optional Boot Code Section with Independent Lock Bits In-System Programming by On-chip Boot Program
- True Read-While-Write Operation
- Programming Lock for Software Security Atmel® QTouch® library support
- Capacitive touch buttons, sliders and wheels— QTouch and QMatrix® acquisition
- Up to 64 sense channels
- Peripheral Features
- Two 8-bit Timer/Counters with Separate Prescaler and Compare Mode
- One 16-bit Timer/Counter with Separate Prescaler, Compare Mode, and Capture Mode

- Real Time Counter with Separate Oscillator
- Six PWM Channels
- 8-channel 10-bit ADC in TQFP and QFN/MLF package
- Temperature Measurement
- 6-channel 10-bit ADC in PDIP Package
- Temperature Measurement Programmable Serial USART – Master/Slave SPI Serial Interface
- Byte-oriented 2-wire Serial Interface (Philips I2 C compatible)
- Programmable Watchdog Timer with Separate On-chip Oscillator
- On-chip Analog Comparator
- Interrupt and Wake-up on Pin Change

CHAPTER 3



ARCHITECTURE DIAGRAM fig 3.1

3.1 ARCHITECTURE DIAGRAM:

- Power-on Reset and Programmable Brown-out Detection
- Internal Calibrated Oscillator
- External and Internal Interrupt Sources
- Six Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, Standby, and Extended Standby

- I/O and Packages
- 23 Programmable I/O Lines
- 28-pin PDIP, 32-lead TQFP, 28-pad QFN/MLF and 32-pad QFN/MLF

- Operating Voltage: 1.8 - 5.5V
- Temperature Range: -40 °C to 85°
- Speed Grade: 0 - 4MHz@1.8 - 5.5V, 0 - 10MHz@2.7 - 5.5V, 0 - 20MHz @ 4.5 - 5.5V C°
- Power Consumption at 1MHz, 1.8V, 25
- Active Mode: 0.2Ma
- Power-down Mode: 0.1µA
- Power-save Mode: 0.75µA (Including 32kHz RTC)

3.2 PIN DISCRIPTION :

VCC Digital supply voltage

GND Ground.

3.2.1 Port B (PB7:0) XTAL1/XTAL2/TOSC1/TOSC2:

Port B is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port B output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port B pins that are externally pulled low will source current if the pull-up resistors are activated. The Port B pins are tri-stated when a reset condition becomes active, even if the clock is not running. Depending on the clock selection fuse settings, PB6 can be used as input to the inverting Oscillator amplifier and input to the internal clock operating circuit. Depending on the clock selection fuse settings, PB7 can be used as output from the inverting Oscillator amplifier. If the Internal Calibrated RC

Oscillator is used as chip clock source, PB7..6 is used as TOSC2..1 input for the Asynchronous Timer/Counter2 if the AS2 bit in ASSR is set.

3.2.2 Port C (PC5:0):

Port C is a 7-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The PC5 0 output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port C pins that are externally pulled low will source current if the pull-up resistors are activated. The Port C pins are tri-stated when a reset condition becomes active, even if the clock is not running.

3.2.3 PC6/RESET:

If the RSTDISBL Fuse is programmed, PC6 is used as an I/O pin. Note that the electrical characteristics of PC6 differ from those of the other pins of Port C. If the RSTDISBL Fuse is unprogrammed, PC6 is used as a Reset input. A low level on this pin for longer than the minimum pulse length will generate a Reset, even if the clock is not running. The minimum pulse length is given in Table 28-3 on page 308. Shorter pulses are not guaranteed to generate a Reset.

3.2.4 Port D (PD7:0):

Port D is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port D output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port D pins that are externally pulled low will source current if the pull-up resistors are activated. The Port D pins are tri-stated when a reset condition becomes active, even if the clock is not running.

3.2.5 AVCC:

AVCC is the supply voltage pin for the A/D Converter, PC3:0, and ADC7:6. It should be externally connected to VCC, even if the ADC is not used. If the ADC is used, it

should be connected to VCC through a low-pass filter. Note that PC6..4 use digital supply voltage, VCC

3.2.6 AREF:

AREF is the analog reference pin for the A/D Converter

3.2.7 ADC7:6 (TQFP and QFN/MLF Package Only):

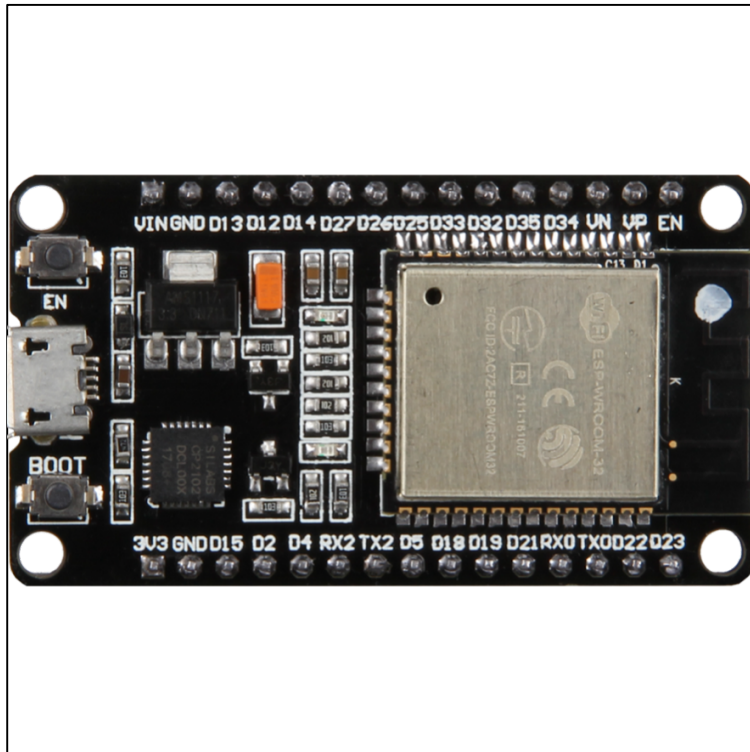
In the TQFP and QFN/MLF package, ADC7:6 serve as analog inputs to the A/D converter. These pins are powered from the analog supply and serve as 10-bit ADC channels.

3.3 NODE MCU ESP32:

DESCRIPTION:

ESP32 is a single 2.4 GHz Wi-Fi-and-Bluetooth combo chip designed with the TSMC ultra-low-power 40 nm technology. It is designed to achieve the best power and RF performance, showing robustness, versatility and reliability in a wide variety of applications and power scenarios.

The ESP32 series of chips includes ESP32-D0WD-V3, ESP32-D0WDQ6-V3, ESP32-D0WD, ESP32-D0WDQ6, ESP32-D2WD, ESP32-S0WD, and ESP32-U4WDH, among which, ESP32-D0WD-V3, ESP32-D0WDQ6-V3, and ESP32-U4WDH are based on ECO V3 wafer.



NODE MCU ESP32 circuit fig 3.2

3.3.1 Featured Solutions:

Ultra- Low- Power Solution

ESP32 is designed for mobile, wearable electronics, and Internet-of-Things (IoT) applications. It features all the state-of-the-art characteristics of low-power chips, including fine-grained clock gating, multiple power modes, and dynamic power scaling. For instance, in a low-power IoT sensor hub application scenario, ESP32 is woken up periodically and only when a specified condition is detected. Low-duty cycle is used to minimize the amount of energy that the chip expends. The output of the power amplifier is also adjustable, thus contributing to an optimal trade-off between communication range, data rate and power consumption.

Complete Integration Solution

ESP32 is a highly-integrated solution for Wi-Fi-and-Bluetooth IoT applications, with

around 20 external components. ESP32 integrates an antenna switch, RF balun, power amplifier, low-noise receive amplifier, filters, and power management modules. As such, the entire solution occupies minimal Printed Circuit Board (PCB) area. ESP32 uses CMOS for single-chip fully-integrated radio and baseband, while also integrating advanced calibration circuitries that allow the solution to remove external circuit imperfections or adjust to changes in external conditions. As such, the mass production of ESP32 solutions does not require expensive and specialized Wi-Fi testing equipment.

3.3.2 Wi-Fi Key Features:

- ✓ 802.11 b/g/n
- ✓ 802.11 n (2.4 GHz), up to 150 Mbps
- ✓ WMM
- ✓ TX/RX A-MPDU, RX A-MSDU
- ✓ Immediate Block ACK
- ✓ Defragmentation
- ✓ Automatic Beacon monitoring (hardware TSF)
- ✓ 4 × virtual Wi-Fi interfaces
- ✓ Simultaneous support for Infrastructure Station, SoftAP, and Promiscuous modes Note that when ESP32 is in Station mode, performing a scan, the SoftAP channel will be changed.
- ✓ Antenna diversity

3.3.3 BT Key Features:

- Compliant with Bluetooth v4.2 BR/EDR and BLE specifications
- Class-1, class-2 and class-3 transmitter without external power amplifier
- Enhanced Power Control

- +12 dBm transmitting power
- NZIF receiver with –94 dBm BLE sensitivity
- Adaptive Frequency Hopping (AFH)
- Standard HCI based on SDIO/SPI/UART
- High-speed UART HCI, up to 4 Mbps
- Bluetooth 4.2 BR/EDR BLE dual mode controller
- Synchronous Connection-Oriented/Extended (SCO/eSCO)
- CVSD and SBC for audio codec
- Bluetooth Piconet and Scatternet
- Multi-connections in Classic BT and BLE
- Simultaneous advertising and scanning

3.4 MCU and Advanced Features:

3.4.1 CPU and Memory:

- Xtensa® single-/dual-core 32-bit LX6 microprocessor(s), up to 600 MIPS (200 MIPS for ESP32-S0WD/ESP32-U4WDH, 400 MIPS for ESP32-D2WD)
- 448 KB ROM
- 520 KB SRAM
- 16 KB SRAM in RTC
- QSPI supports multiple flash/SRAM chips

3.4.2 Clocks and Timers:

- Internal 8 MHz oscillator with calibration
- Internal RC oscillator with calibration
- External 2 MHz ~ 60 MHz crystal oscillator (40 MHz only for Wi-Fi/BT functionality)
- External 32 kHz crystal oscillator for RTC with calibration
- Two timer groups, including 2 × 64-bit timers and 1 × main watchdog in each group
- One RTC timer
- RTC watchdog

3.4.3 Advanced Peripheral Interfaces:

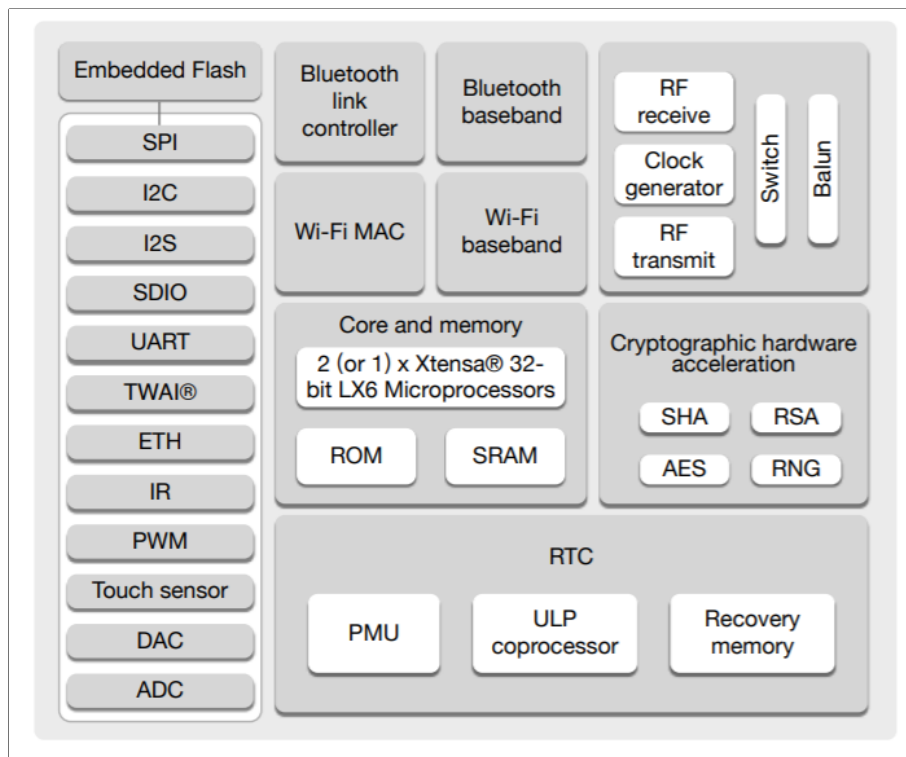
- 34 × programmable GPIOs
- 12-bit SAR ADC up to 18 channels
- 2 × 8-bit DAC
- 10 × touch sensors
- 4 × SPI
- 2 × I²S
- 2 × I²C
- 3 × UART
- 1 host (SD/eMMC/SDIO)

- 1 slave (SDIO/SPI)
- Ethernet MAC interface with dedicated DMA and IEEE 1588 support
- Two-Wire Automotive Interface (TWAI®, compatible with ISO11898-1)
- IR (TX/RX)
- Motor PWM
- LED PWM up to 16 channels
- Hall sensor

3.4.4 Security:

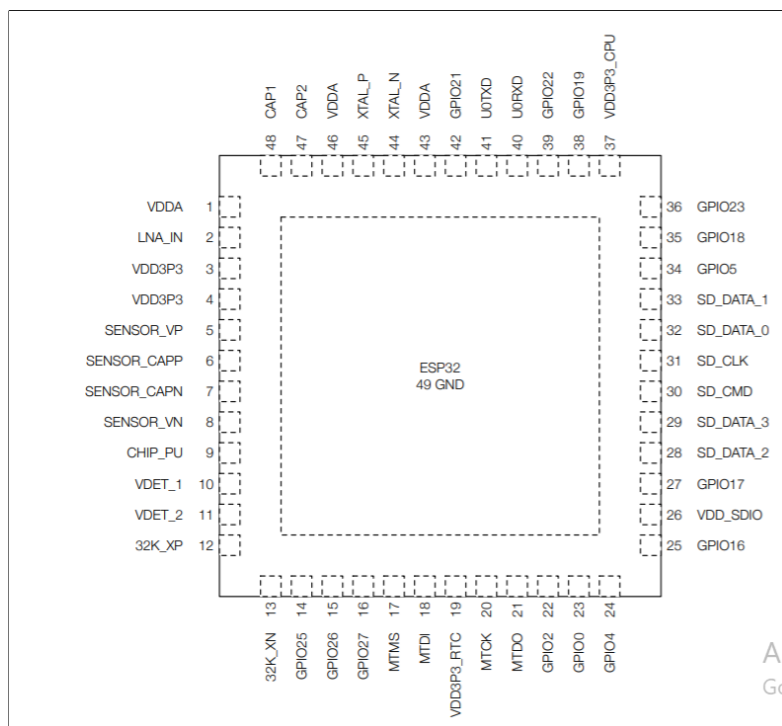
- Secure boot
- Flash encryption
- 1024-bit OTP, up to 768-bit for customers
- Cryptographic hardware acceleration: – AES – Hash (SHA-2)
 - RSA – ECC – Random Number Generator (RNG)

3.5 BLOCK DIAGRAM:



CIRCUIT BLOCK DIAGRAM fig 3.3

3.5.1 Pin Layout:



Pin Layout FIG 3.4

3.5.2 CPU:

ESP32 contains one or two low-power Xtensa® 32-bit LX6 microprocessor(s) with the following features:

- 7-stage pipeline to support the clock frequency of up to 240 MHz (160 MHz for ESP32-S0WD, ESP32-D2WD, and ESP32-U4WDH)
- 16/24-bit Instruction Set provides high code-density
- Support for Floating Point Unit
- Support for DSP instructions, such as a 32-bit multiplier, a 32-bit divider, and a 40-bit MAC
- Support for 32 interrupt vectors from about 70 interrupt sources The single-/dual-CPU interfaces include:
 - Xtensa RAM/ROM Interface for instructions and data
 - Xtensa Local Memory Interface for fast peripheral register access
 - External and internal interrupt sources
 - JTAG for debugging

3.5.3 Internal Memory:

ESP32's internal memory includes:

- 448 KB of ROM for booting and core functions
- 520 KB of on-chip SRAM for data and instructions
- 8 KB of SRAM in RTC, which is called RTC FAST Memory and can be used for data storage; it is accessed by the main CPU during RTC Boot from the Deep-sleep mode.
- 8 KB of SRAM in RTC, which is called RTC SLOW Memory and can be accessed by the

co-processor during the Deep-sleep mode.

- 1 Kbit of eFuse: 256 bits are used for the system (MAC address and chip configuration) and the remaining 768 bits are reserved for customer applications, including flash-encryption and chip-ID.

- Embedded flash

External Flash and SRAM:

ESP32 supports multiple external QSPI flash and SRAM chips. More details can be found in Chapter SPI in the ESP32 Technical Reference Manual. ESP32 also supports hardware encryption/decryption based on AES to protect developers' programs and data in flash.

ESP32 can access the external QSPI flash and SRAM through high-speed caches.

- Up to 16 MB of external flash can be mapped into CPU instruction memory space and read-only memory space simultaneously.

- When external flash is mapped into CPU instruction memory space, up to 11 MB + 248 KB can be mapped at a time. Note that if more than 3 MB + 248 KB are mapped, cache performance will be reduced due to speculative reads by the CPU.

- When external flash is mapped into read-only data memory space, up to 4 MB can be mapped at a time. 8-bit, 16-bit and 32-bit reads are supported.

- External SRAM can be mapped into CPU data memory space. SRAM up to 8 MB is supported and up to 4 MB can be mapped at a time. 8-bit, 16-bit and 32-bit reads and writes are supported.

CHAPTER 4

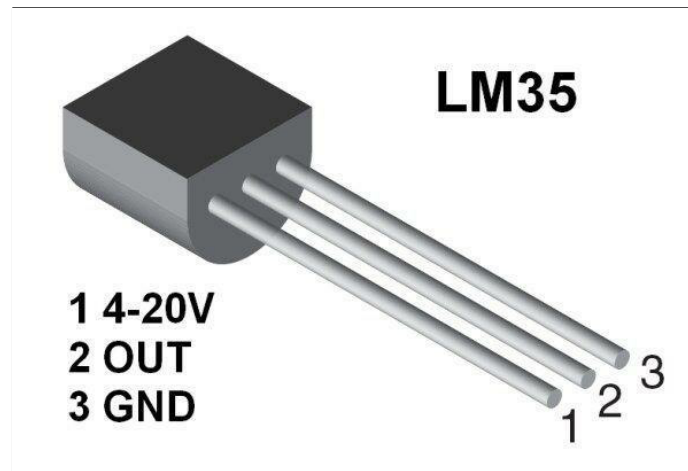
4.1 TEMPERATURE SENSOR:

DESCRIPTION:

A temperature sensor is a device, typically, a thermocouple or RTD, that provides for temperature measurement through an electrical signal. A thermocouple (T/C) is made from two dissimilar metals that generate electrical voltage in direct proportion to changes in temperature.

The LM35 series are precision integrated-circuit temperature devices with an

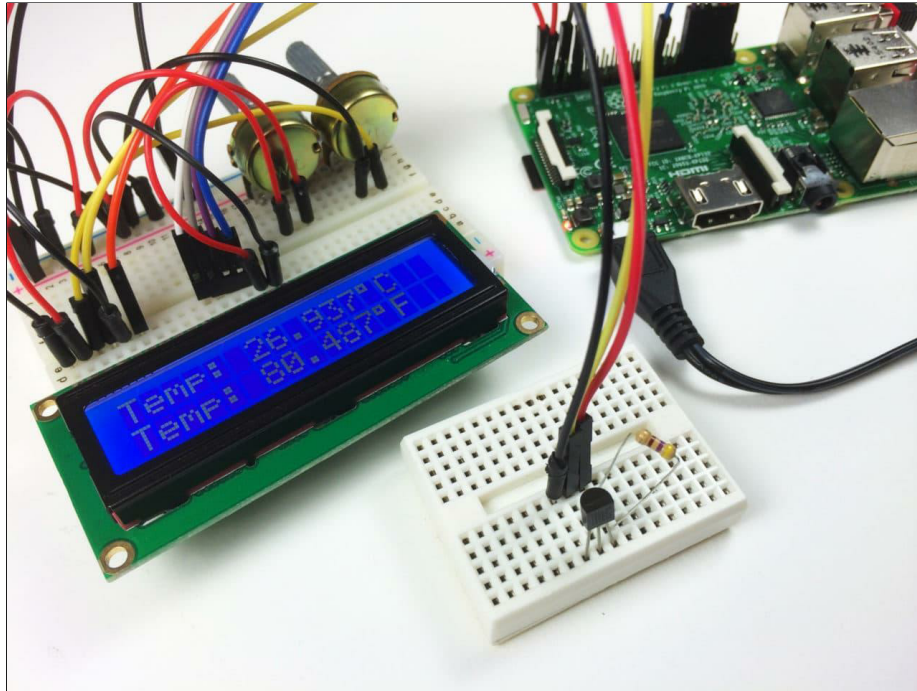
output voltage linearly-proportional to the Centigrade temperature. The LM35 device has an advantage over linear temperature sensors calibrated in Kelvin, as the user is not required to subtract a large constant voltage from the output to obtain convenient Centigrade scaling.



temperature sensor fig 4.1

The LM35 device does not require any external calibration or trimming to provide typical accuracies of $\pm\frac{1}{4}^{\circ}\text{C}$ at room temperature and $\pm\frac{3}{4}^{\circ}\text{C}$ over a full -55°C to 150°C temperature range. The low-output impedance, linear output, and precise inherent calibration of the LM35 device makes interfacing to readout or control circuitry especially easy.

LM35 is a precision IC temperature sensor with its output proportional to the temperature (in $^{\circ}\text{C}$). The sensor circuitry is sealed and therefore it is not subjected to oxidation and other processes. With LM35, temperature can be measured more accurately than with a thermistor. It also possess low self heating and does not cause more than 0.1°C temperature rise in still air.



TEMPERATURE SENSOR INTERFACE WITH RASPBERRY PI circuit fig 4.2

4.2 TEMPERATURE SENSOR INTERFACE WITH RASPBERRY PI:

The device is used with single power supplies, or with plus and minus supplies. As the LM35 device draws only 60 μA from the supply, it has very low self-heating of less than 0.1°C in still air. The LM35 device is rated to operate over a -55°C to 150°C temperature range.

4.3 FEATURES:

- Calibrated Directly in Celsius (Centigrade)
- Linear + 10-mV/°C Scale Factor
- 0.5°C Ensured Accuracy (at 25°C)
- Rated for Full -55°C to 150°C Range
- Suitable for Remote Applications

4.4 APPLICATIONS:

- Power Supplies
- Battery Management
- HVAC
- Appliances

CHAPTER 5

5.1 VOLTAGE SENSOR:

DESCRIPTION:

A voltage sensor is going to be able to determine and even monitor and measure the voltage supply. It is then able to take those measurements and turn them into a

signal that one will then be able to read. The signal will often go into a specialized electronic device for recording, but sometimes, an observer will be present to manually read the sensor output.

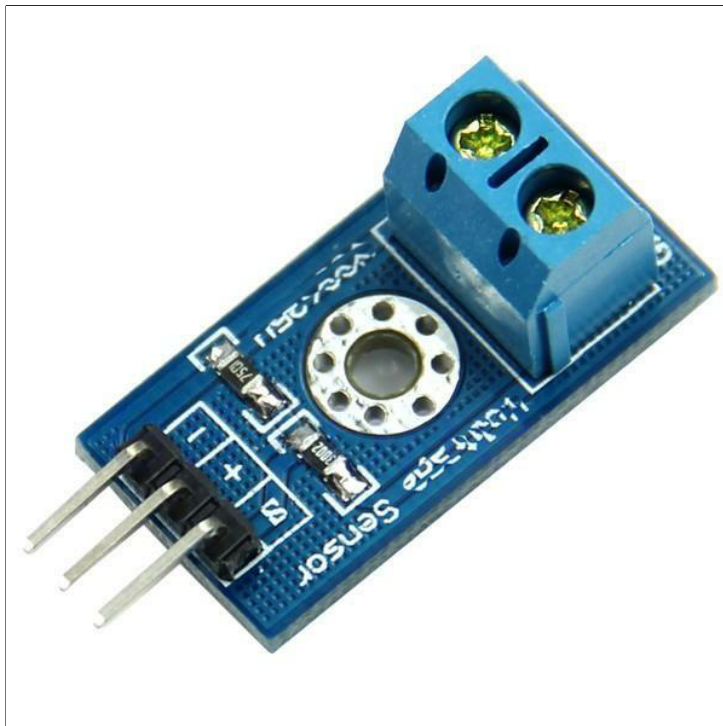


AC VOLTAGE SENSOR fig 5.1

5.2 AC VOLTAGE SENSOR:

The Voltage Sensor block represents an ideal voltage sensor, that is, a device that converts voltage measured between two points of an electrical circuit into a physical signal proportional to the voltage. Connections + and – are electrical conserving ports through which the sensor is connected to the circuit. Connection V is a physical signal port that outputs the measurement result.

AC voltage sensor works according to Magnetic Modulation and is designed for AC voltage measurement. The output signal of this sensor is proportional to the input AC voltage. It can be used for continuous ac voltage monitoring of the system.



DC VOLTAGE SENSOR fig 5.2

5.3 DC VOLTAGE SENSOR:

DC Voltage Sensors are used to measure the potential difference between the ends of an electrical component. This can be used to measure the DC voltage in the circuits. The sensor is mechanically fixed by soldering the secondary circuit pins to the PCB. The primary connection can also be integrated in the sensor. Pulsating voltage with a galvanic insulation between primary and secondary circuits. The voltage detector indicates the presence of a voltage higher than a limit. The Voltage Sensors are equipped with a micro controller that greatly improves the sensor accuracy, precision and consistency of the readings. They are supplied calibrated and the stored calibration (in Volts) is automatically loaded when the Voltage Sensor is connected.

A DC voltage sensor is going to be able to determine and even monitor and measure the voltage supply. It is then able to take those measurements and turn them into a signal that one will then be able to read. The signal will often go into a specialized electronic device for recording, but sometimes, an observer will be present to manually read the sensor output. For reasons of accuracy, if more than one Voltage Sensor is being used in a circuit, ensure that they share a common earth. DC voltage sensor works according to Magnetic Modulation and is designed for DC voltage measurement. The output signal of this sensor is proportional to the input DC voltage. It can be used for continuous dc voltage monitoring of the system.

5.4 FEATURES:

- Under-voltage, over-voltage, or voltage band models
- Powered from sensing input lines or from separate DC supply
- Available with time delays on pull-in and/or drop-out or with customized
- voltage-time trip curves

5.5 APPLICATIONS:

- Power generation unit
- Power transmission unit
- Industry and other applications

CHAPTER 6

6.1 CURRENT SENSOR:

DESCRIPTION:

A current sensor is a device that detects and converts current to an easily measured output voltage, which is proportional to the current through the measured path.

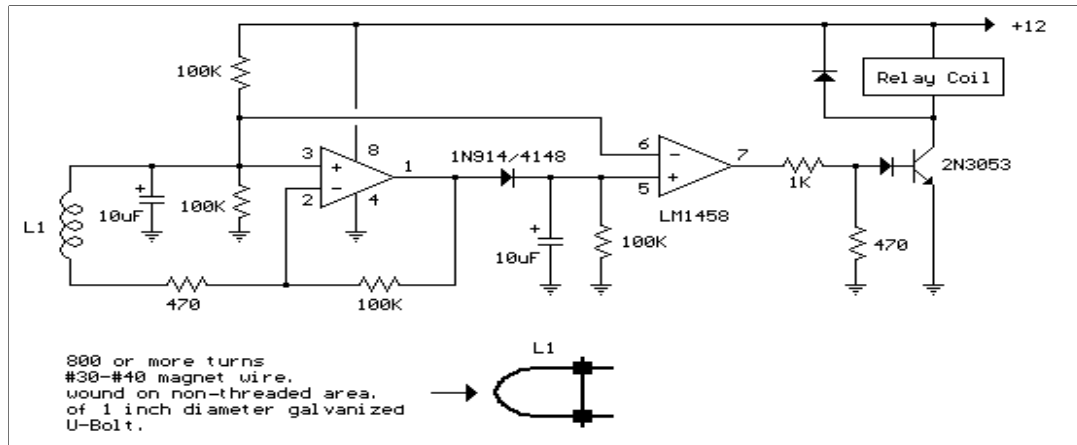
When a current flows through a wire or in a circuit, voltage drop occurs. Also, a magnetic field is generated surrounding the current carrying conductor. Both of these phenomena are made use of in the design of current sensors. Thus, there are two types of current sensing: direct and indirect. Direct sensing is based on Ohm's law, while indirect sensing is based on Faraday's and Ampere's law.



CURRENT SENSOR diagram fig 6.1

A current sensor (CT1270) is a device that detects electric current (AC or DC) in a wire, and generates a signal proportional to it. The generated signal could be analog voltage or current or even digital output.

When a current flows through a wire or in a circuit, voltage drop occurs. Also, a magnetic field is generated surrounding the current carrying conductor. Both of these phenomena are made use of in the design of current sensors. Thus, there are two types of current sensing: direct and indirect. Direct sensing is based on Ohm's law, while indirect sensing is based on Faraday's and Ampere's law. Direct Sensing involves measuring the voltage drop associated with the current passing through passive electrical components.



CIRCUIT DIAGRAM OF CURRENT SENSOR fig 6.2

6.2 CIRCUIT DIAGRAM OF CURRENT SENSOR:

Indirect Sensing involves measurement of the magnetic field surrounding a conductor through which current passes. Generated magnetic field is then used to induce proportional voltage or current which is then transformed to a form suitable for measurement and/or control system.

6.3 FEATURES:

- Supply voltage: 5v DC
- Output: analog
- Small size

- Low cost

6.4 APPLICATIONS:

- Ammeters
- Current control purposes
- DC/DC converters
- Ground fault detectors

CHAPTER7

7.1 ULTRASONIC SENSOR:

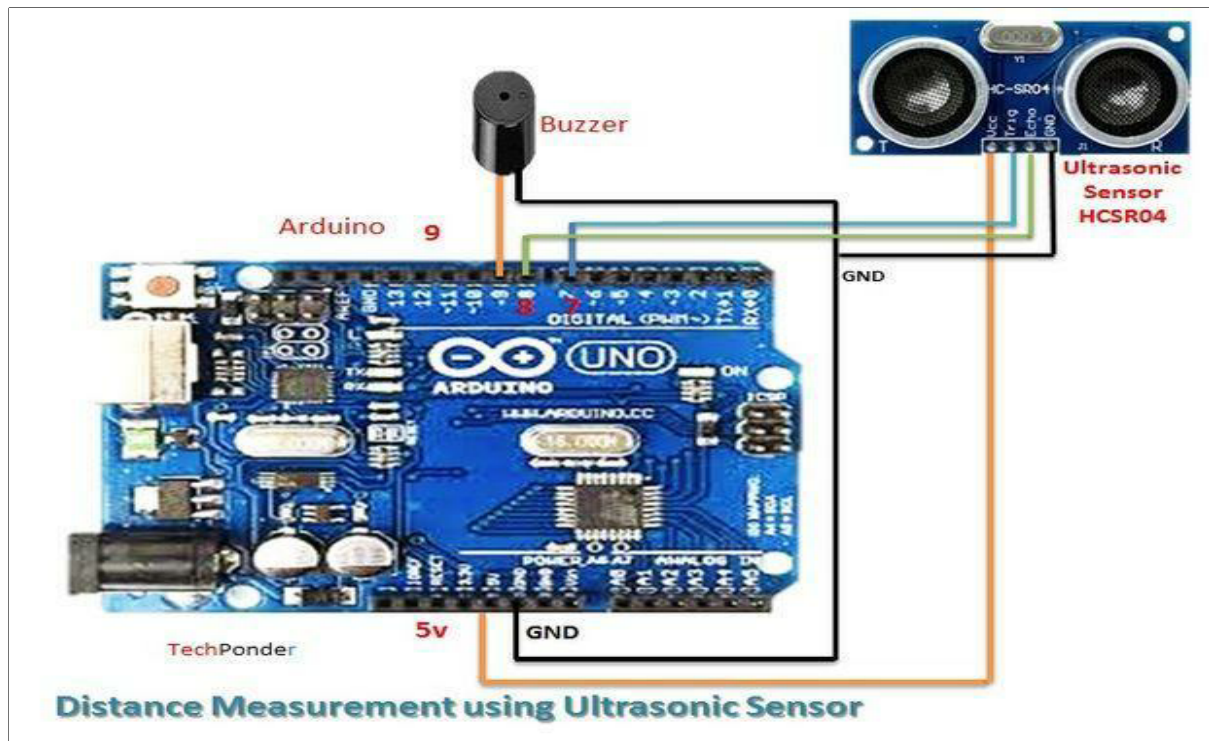
DESCRIPTION:

An Ultrasonic sensor is a device that can measure the distance to an object by using sound waves. It measures distance by sending out a sound wave at a specific frequency and listening for that sound wave to bounce back. By recording the elapsed time between the sound wave being generated and the sound wave bouncing back, it is possible to calculate the distance between the sonar sensor and the object.



ULTRASONIC SENSOR fig7.1

Ultrasonic sensor emit ultrasonic pulses, and by measuring the time of ultrasonic pulse reaches the object and back to the transducer. The sonic waves emitted by the transducer are reflected by an object and received back in the transducer. After having emitted the sound waves, the ultrasonic sensor will switch to receive mode. The time elapsed between emitting and receiving is proportional to the distance of the object from the sensor.



ULTRASONIC SENSOR INTERFACE WITH ARDUINO fig7.2

7.2 ULTRASONIC SENSOR INTERFACE WITH ARDUINO:

Ultrasonic transmitter emitted an ultrasonic wave in one direction and started timing when it launched. Ultrasonic spread in the air and would return immediately when it encountered obstacles on the way. At last the ultrasonic receiver would stop timing when it receives the reflected wave. The distance of sensor from the target object is calculated. It offers excellent non-contact range detection with high accuracy and stable

readings in an easy-to-use package. Its operation is not affected by sunlight or black material. The supply voltage to the sensor is 5VDC. The sensor has two pins namely trig and echo which is connected to the controller to give digital input.

7.3 FEATURES:

- Working Voltage: 5VDC
- Quiescent Current : 2Ma
- Working Current: 15mA
- Detecting Range: 2cm - 4.5m
- Trigger Input Pulse width: 10uS

7.4 APPLICATIONS:

- Robot navigation
- Obstacle avoidance
- Engineering measurement tools
- Industrial control system

CHAPTER 8

8.1 POWER SUPPLY UNIT:

DESCRIPTION:

A power supply unit (or PSU) converts mains AC to low-voltage regulated DC power for the internal components of a computer. Modern personal computers universally use switched-mode power supplies. Some power supplies have a manual switch for selecting input voltage, while others automatically adapt to the mains voltage.

A power supply is used to reduce the mains electricity at 240 volts AC down to something more useable, say 12 volts DC. There are two types of power supply, linear and switch mode. A linear power supply uses a transformer to reduce the voltage. The AC signal is rectified and regulated to produce a high DC voltage.

An AC adapter, AC/DC adapter, or AC/DC converter is a type of external power supply, often enclosed in a case similar to an AC plug. Adapters for battery-powered equipment may be described as chargers or rechargers (see also battery charger). AC adapters are used with electrical devices that require power but do not contain internal components to derive the required voltage and power from main power. The internal circuitry of an external power supply is very similar to the design that would be used for a built-in or internal supply.

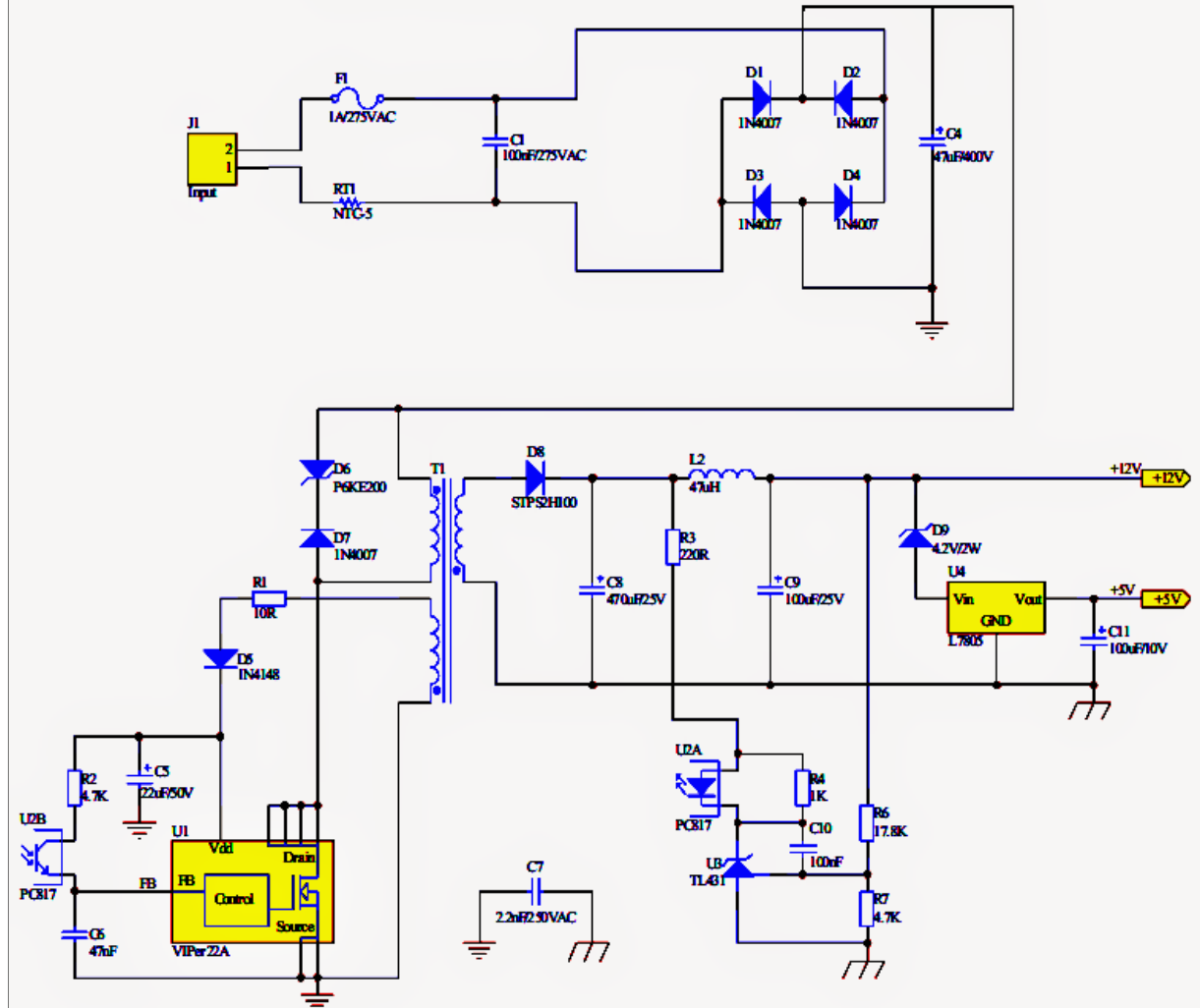


ADAPTER (12V 1AMP) fig8.1

8.2 ADAPTER (12V 1AMP):

An adapter is a device that converts attributes of one electrical device or system to those of an otherwise incompatible device or system. Some modify power or signal attributes, while others merely adapt the physical form of one electrical connector to another. In a computer, an adapter is often built into a card that can be inserted into a slot on the computer's motherboard. The card adapts information that is exchanged between the computer's microprocessor and the devices that the card supports.

12V, 1 Amp SMPS Circuit



CIRCUIT DIAGRAM fig 8.2

8.3 CIRCUIT DIAGRAM:

An electric power adapter may enable connection of a power plug, sometimes called, used in one region to a AC power socket used in another, by offering connections for the disparate contact arrangements, while not changing the voltage. An AC adapter, also called a "recharger", is a small power supply that changes household electric current from distribution voltage) to low voltage DC suitable for consumer electronics.

Some modify power or signal attributes, while others merely adapt the physical form of one electrical connector to another. For computers and related items, one kind of serial port adapter enables connections between 25-contact and nine-contact connectors, but does not affect electrical power- and signalling-related attributes

8.4 FEATURES:

- Output current:1A
- Supply voltage: 220-230VAC
- Output voltage: 12VDC
- Reduced costs
- Increased value across front-office and back-office functions
- Access to current, accurate, and consistent data
- It generates adapter metadata as WSDL files with J2CA extension.

8.5 APPLICATIONS:

- Back-end systems which need to send purchase order data to oracle applications send it to the integration service via a integration server client.
- SMPS applications.

CHAPTER 9

SOFTWARE DESCRIPTION:

9.1 PROGRAMMING LANGUAGE:

1. Embedded C is a set of language extensions for the C programming language by the C Standards Committee to address commonality issues that exist between C extensions for different embedded systems.

2. Embedded C is a set of language extensions for the C programming language by the C Standards Committee to address commonality issues that exist between C extensions for different embedded systems. Historically, embedded C programming requires nonstandard extensions to the C language in order to support exotic features such as fixed-point arithmetic, multiple distinct memory banks, and basic I/O operations

3. An embedded system is a computer system with a dedicated function within a larger mechanical or electrical system, often with real-time computing constraints. It is embedded as part of a3. An embedded system is a computer system with a dedicated function within a larger mechanical or electrical system, often with real-time computing constraints. It is embedded as part of a complete device often including hardware and mechanical parts. Embedded systems control many devices in common use today. Ninety-eight percent of all microprocessors are manufactured as components of embedded systems.

01. Examples of properties of typical embedded computers when compared with general-purpose counterparts are lowpower consumption, small size, rugged

operating ranges, and low per-unit cost. This comes at the price of limited processing resources, which make them significantly more difficult to program and to interact with. However, by building intelligence mechanisms on top of the hardware, taking advantage of possible existing sensors and the existence of a network of embedded units, one can both optimally manage available resources at the unit and network levels as well as provide augmented functions, well beyond those available. For example, intelligent techniques can be designed to manage power consumption of embedded systems.

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

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

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

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

Before uploading your sketch, you need to select the correct items from the **Tools > Board** and **Tools > Port** menus. The boards are described below. On the Mac, the serial port is probably something like `/dev/tty.usbmodem241` (for an Uno or Mega2560 or Leonardo) or `/dev/tty.usbserial-1B1` (for a Duemilanove or earlier USB board), or `/dev/tty.USA19QW1b1P1.1` (for a serial board connected with a Keyspan USB-to-Serial adapter). On Windows, it's probably COM1 or COM2 (for a serial board) or COM4, COM5, COM7, or higher (for a USB board) - to find out, you look for USB serial device in the ports section of the Windows Device Manager. On Linux, it should be `/dev/ttyACMx`, `/dev/ttyUSBx` or similar. Once you've selected the correct serial port and board, press the upload button in the toolbar or select the **Upload** item from the **Sketch** menu. Current Arduino boards will reset automatically and begin the upload. With older boards (pre-Diecimila) that lack auto-reset, you'll need to press the reset button on the board just before starting the upload. On most boards, you'll see the RX and TX LEDs blink as the sketch is uploaded. The Arduino Software (IDE) will display a message when the upload is complete, or show an error.

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

CHAPTER 10

10.1 ARDUINO code:

```
#define echoPin 5
```

```
#define trigPin 6
```

```
long duration;
```

```
int distance, temp, Current;
```

```
void setup() {
```

```
    pinMode(trigPin, OUTPUT);
```

```
    pinMode(echoPin, INPUT);
```

```
    pinMode(2, OUTPUT);
```

```
    Serial.begin(9600);
```

```
    digitalWrite(2, LOW);
```

```
}
```

```
void loop() {
```

```
    temp = analogRead(A0) / 6.32258064516129;
```

```
float current = 0;
for (int i = 0; i < 1000; i++)
{
    current = current + (.0264 * analogRead(A1) - 13.51) / 1000;
    delay(1);
}
Current = current * 100;
digitalWrite(trigPin, LOW);
delayMicroseconds(2);
digitalWrite(trigPin, HIGH);
delayMicroseconds(10);
digitalWrite(trigPin, LOW);
duration = pulseIn(echoPin, HIGH);
distance = duration * 0.034 / 2;
Serial.print("T");
Serial.print(temp);
Serial.print("C");
Serial.print(Current);
Serial.print("D");
Serial.print(distance);
Serial.println("#");
delay(500);
}
```

Node mcu

```
#include <ESP8266WiFi.h>
```

```
#include <ESP8266HTTPClient.h>
```

```
#include <WiFiClient.h>
```

```
#include <SoftwareSerial.h>
```

```
const char* ssid    = "iot";
```

```
const char* password = "12345678";
```

```
const char* serverName = "http://iotcloud22.in/transformer_health/post_value.php";
```

```
WiFiClient client;
```

```
HTTPClient http;
```

```
int Temp,Distance,Current;
```

```
int len, h;

int cnt = 0;

String inString = "";


#define rx D7

#define tx D8

SoftwareSerial NodeSerial (rx, tx);


void setup()
{

    Serial.begin(9600);
    NodeSerial.begin(9600);
    WiFi.begin(ssid, password);
    Serial.println("Connecting");
    while (WiFi.status() != WL_CONNECTED) {
        Serial.print(".");
        delay(500);
    }
    Serial.println("");
    Serial.print("Connected to WiFi network with IP Address: ");
    Serial.println(WiFi.localIP());

}
```

```

void loop ()
{

    while (NodeSerial.available() > 0) {

        char c = NodeSerial.read();
        Serial.print(c);
        if (c == '\n' || c == '\r') {
            if (cnt >= 2) {

                status_filter(inString);
                status_filter1(inString);
                status_filter2(inString);
                status_filter3(inString);
                status_filter4(inString);
                status_filter5(inString);
                status_filter6(inString);
                status_filter7(inString);
                status_filter8(inString);
                // status_filter9(inString);
                Serial.println(inString);
                inString = "";
                cnt = 0;
            }
            else

```

```

{
    inString = "";
    cnt = 0;
}
}
else
{
    inString += c;
    cnt++;
}
}
}

void sending_to_db()
{
    if (WiFi.status() == WL_CONNECTED)
    {
        http.begin(client, serverName);
        http.addHeader("Content-Type", "application/x-www-form-urlencoded");

        String httpRequestData = "&value1=" + String(Temp) + "&value2=" + String(Distance) +
"&value3=" + String(Current) + "";

        // Serial.print("httpRequestData: ");
        // Serial.println(httpRequestData);

        int httpResponseCode = http.POST(httpRequestData);

        if (httpResponseCode > 0) {

```

```
    Serial.print("HTTP Response code: ");
    Serial.println(httpResponseCode);
}
else {
    Serial.print("Error code: ");
    Serial.println(httpResponseCode);
}
http.end();
}
else {
    Serial.println("WiFi Disconnected");
}
delay(1000);

}
```

```
void status_filter(String buff)
{
    len = buff.length();
    //Serial.println(len);
    if (buff[0] == 'T')
        if (buff[3] == 'C')
            if (buff[7] == 'D')
                if (buff[11] == '#')
```

```
//T00C000D000#  
//012345678901  
{  
  
    String temp = buff.substring(1, 3);  
    String current = buff.substring(4, 7);  
    String distance = buff.substring(8, 11);  
  
    Temp = temp.toInt();  
    Current = current.toInt();  
    Distance = distance.toInt();  
  
    Serial.print("TEMPERATURE = "); Serial.println(Temp);  
    Serial.print("DISTANCE = "); Serial.println(Distance);  
    Serial.print("CURRENT = "); Serial.println(Current);  
  
    sending_to_db();  
}  
}
```

```
void status_filter1(String buff)
```

```
{  
    len = buff.length();  
    //Serial.println(len);  
    if (buff[0] == 'T')  
        if (buff[3] == 'C')  
            if (buff[7] == 'D')  
                if (buff[10] == '#')  
  
                    //T00C000D00#  
                    //01234567890  
                    {  
  
                        String temp = buff.substring(1, 3);  
                        String current = buff.substring(4, 7);  
                        String distance = buff.substring(8, 10);  
  
                        Temp = temp.toInt();  
                        Current = current.toInt();  
                        Distance = distance.toInt();  
  
                        Serial.print("TEMPERATURE = "); Serial.println(Temp);  
                        Serial.print("DISTANCE = "); Serial.println(Distance);  
                        Serial.print("CURRENT = "); Serial.println(Current);  
  
                        sending_to_db();  
                    }  
                }  
            }  
        }  
    }  
}
```

```
    }  
}
```

```
void status_filter2(String buff)  
{  
    len = buff.length();  
    //Serial.println(len);  
    if (buff[0] == 'T')  
        if (buff[3] == 'C')  
            if (buff[7] == 'D')  
                if (buff[9] == '#')  
  
                    //T00C000D0#  
                    //0123456789  
  
                    {  
  
                        String temp = buff.substring(1, 3);  
                        String current = buff.substring(4, 7);  
                        String distance = buff.substring(8, 9);  
  
                        Temp = temp.toInt();  
                        Current = current.toInt();  
                        Distance = distance.toInt();
```

```
    Serial.print("TEMPERATURE = "); Serial.println(Temp);

    Serial.print("DISTANCE = "); Serial.println(Distance);

    Serial.print("CURRENT = "); Serial.println(Current);

    sending_to_db();
}
}
```

```
void status_filter3(String buff)
{
    len = buff.length();
    //Serial.println(len);
    if (buff[0] == 'T')
        if (buff[3] == 'C')
            if (buff[6] == 'D')
                if (buff[10] == '#')

                    //T00C00D000#

                    //01234567890
}
```

```
String temp = buff.substring(1, 3);

String current = buff.substring(4, 6);

String distance = buff.substring(7, 10);


Temp = temp.toInt();

Current = current.toInt();

Distance = distance.toInt();


Serial.print("TEMPERATURE = "); Serial.println(Temp);

Serial.print("DISTANCE = "); Serial.println(Distance);

Serial.print("CURRENT = "); Serial.println(Current);


    sending_to_db();
}
}
```

```
void status_filter4(String buff)
{
    len = buff.length();
    //Serial.println(len);
    if (buff[0] == 'T')
        if (buff[3] == 'C')
            if (buff[6] == 'D')
```

```
if (buff[9] == '#')

    //T00C00D00#
    //0123456789
    {

        String temp = buff.substring(1, 3);
        String current = buff.substring(4, 6);
        String distance = buff.substring(7, 9);

        Temp = temp.toInt();
        Current = current.toInt();
        Distance = distance.toInt();

        Serial.print("TEMPERATURE = "); Serial.println(Temp);
        Serial.print("DISTANCE = "); Serial.println(Distance);
        Serial.print("CURRENT = "); Serial.println(Current);

        sending_to_db();
    }
}
```

```
void status_filter5(String buff)
```

```
{  
    len = buff.length();  
    //Serial.println(len);  
    if (buff[0] == 'T')  
        if (buff[3] == 'C')  
            if (buff[6] == 'D')  
                if (buff[8] == '#')  
  
                    //T00C00D0#  
                    //012345678  
                    {  
  
                        String temp = buff.substring(1, 3);  
                        String current = buff.substring(4, 6);  
                        String distance = buff.substring(7, 8);  
  
                        Temp = temp.toInt();  
                        Current = current.toInt();  
                        Distance = distance.toInt();  
  
                        Serial.print("TEMPERATURE = "); Serial.println(Temp);  
                        Serial.print("DISTANCE = "); Serial.println(Distance);  
                        Serial.print("CURRENT = "); Serial.println(Current);  
  
                        sending_to_db();  
                    }  
                }  
            }  
        }  
    }  
}
```

```
    }  
}
```

```
void status_filter6(String buff)  
{  
    len = buff.length();  
    //Serial.println(len);  
    if (buff[0] == 'T')  
        if (buff[3] == 'C')  
            if (buff[5] == 'D')  
                if (buff[9] == '#')  
  
                    //T00C0D000#  
                    //0123456789  
                    {  
  
                        String temp = buff.substring(1, 3);  
                        String current = buff.substring(4, 5);  
                        String distance = buff.substring(6, 9);  
  
                        Temp = temp.toInt();  
                        Current = current.toInt();
```

```

        Distance = distance.toInt();

        Serial.print("TEMPERATURE = "); Serial.println(Temp);
        Serial.print("DISTANCE = "); Serial.println(Distance);
        Serial.print("CURRENT = "); Serial.println(Current);

        sending_to_db();
    }
}

```

```

void status_filter7(String buff)
{
    len = buff.length();
    //Serial.println(len);
    if (buff[0] == 'T')
        if (buff[3] == 'C')
            if (buff[5] == 'D')
                if (buff[8] == '#')

                    //T00C0D00#
                    //012345678

```

```
{

    String temp = buff.substring(1, 3);

    String current = buff.substring(4, 5);

    String distance = buff.substring(6, 8);


    Temp = temp.toInt();

    Current = current.toInt();

    Distance = distance.toInt();


    Serial.print("TEMPERATURE = "); Serial.println(Temp);

    Serial.print("DISTANCE = "); Serial.println(Distance);

    Serial.print("CURRENT = "); Serial.println(Current);


    sending_to_db();

}
```

```
void status_filter8(String buff)

{

    len = buff.length();

    //Serial.println(len);
```

```
if (buff[0] == 'T')
  if (buff[3] == 'C')
    if (buff[5] == 'D')
      if (buff[7] == '#')

        //T00C0D0#
        //0123456789
        {

          String temp = buff.substring(1, 3);
          String current = buff.substring(4, 5);
          String distance = buff.substring(6, 7);

          Temp = temp.toInt();
          Current = current.toInt();
          Distance = distance.toInt();

          Serial.print("TEMPERATURE = "); Serial.println(Temp);
          Serial.print("DISTANCE = "); Serial.println(Distance);
          Serial.print("CURRENT = "); Serial.println(Current);

          sending_to_db();
        }
}
```

10.2:

http://iotcloud22.in/transformer_health

CHAPTER 11

12.1 CONCLUSION:

In this work, the development and implementation of realtime monitoring and control systems for transformer protection and recording have been presented. This system is designed to protect the distribution transformer from overloading, overheating, and other abnormalities. An IOT system is employed to monitor and control the operational parameters of the transformer. The IOT system is located close to the transformer base, and the above parameters are sensed and then transmitted to the centralized web server. Thus, the data is utilized to know the transformer condition on a real-time basis and are stored within a server-based database for future analysis and immediate protective precautions.

12.2 REFERENCES:

- [1] Kalyani More, Ashwini Khaire, Sudhir Khalkar, and P.G Salunke, XBEE Based Transformer Protection and Oil Testing, International Journal of Scientific Research Engineering & Technology, 4(3), March 2015.
- [2] Rohith R Pawar, Deosarkar S.B, Health condition monitoring system for distribution transformer using Internet of Things (IoT), International conference on computing methodologies and communication, Erode India, February 2018.
- [3] Satya Kumar Behera, Ravi Masand, and S.P. Shukla, A Review of Transformer

Protection by Using PLC System, International Journal of Digital Application & Contemporary research, 3(2), 2014.

[4] Pathak A.K, Kolhe A.N, Gagare J.T, and Khemnar S.M, GSM Based Distribution Transformer Monitoring And Controlling System,

[5] International Journal of Advance Research Innovative Ideas in Education, 2(2), 2016.

[6] Yoon, J.T, Youn, B.D, Park, K.M, Lee, W.R, Vibration-based robust health diagnostics for mechanical failure modes of power transformers, In proceeding of the IEEE Conference on Prognostics and Health Management, (2013) 1-5.

[7] Cristina Ciulavu, Elena Helerea, Power transformer incipient faults monitoring, Annals of the University of Craiova, Electrical Engineering series, No. 32, 2008.

[8] S. Gokula Krishnan, T Gokul, J Abinaya, M Kanniga, S Balaji, GSM Based Transformer Fault Monitoring System, International Research Journal of Engineering and Technology, 6(3), 2019.

[9] Prajakta Ashok dangat, et al., Power Transformer Fault Monitoring and Alert System, Resincap International Journal of Science & Engineering 1(3), 2017.

[10] U.V. Patil, Kathe Mohan, Harkal Saurabh, Warhade Nilesh, Transformer Health Condition Monitoring Using GSM Technology, IJARIIIE, 2(2), 2016.

[11] Vishwanath R et al., A New Approach to monitoring Condition of Transformers incipient fault diagnosis based on GSM & XBEE, International Journal of Engineering Development and Research, 3(2), 2015.

[12] Van Harmelen, F., Ten Teije, A., Approximations in diagnosis: motivations and techniques, SARA, 1995.

[13] Technical manual – Power Transformer maintenance and acceptance testing, 1998.

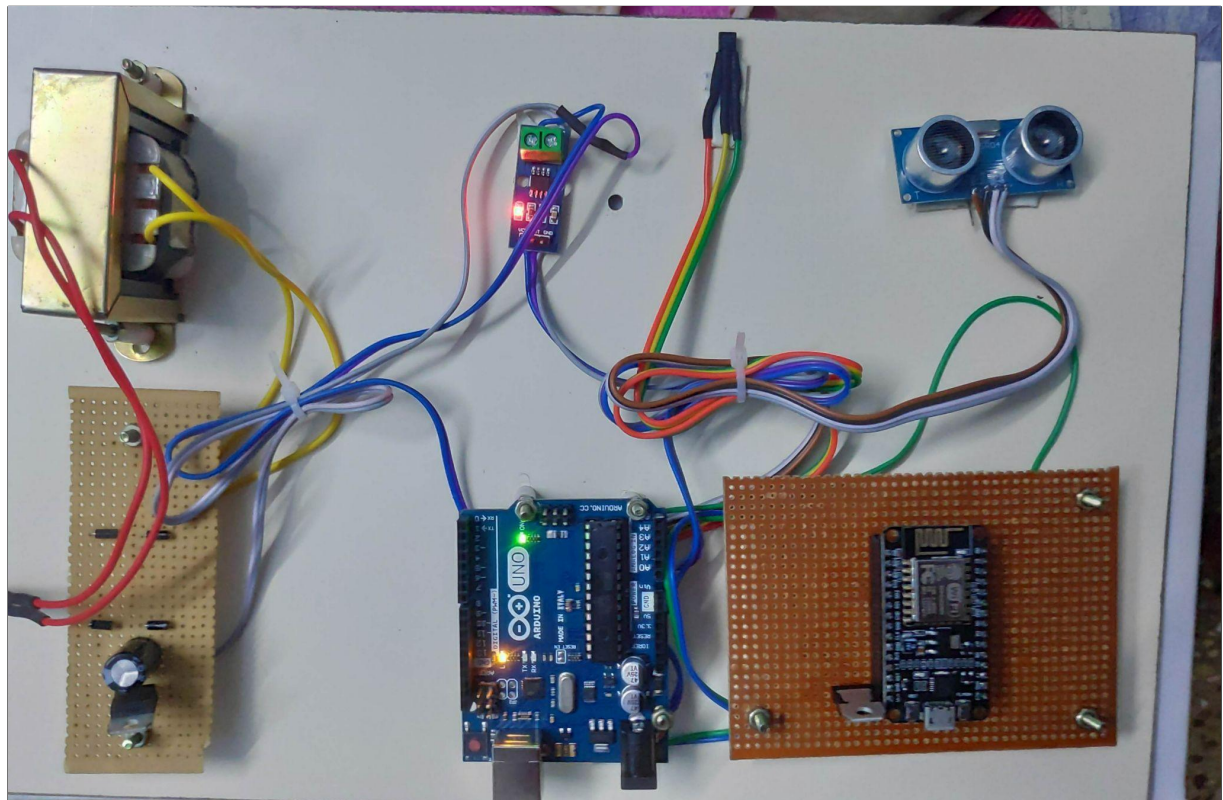
[14] IEEE PES Transformers Committee, IEEE Guide for Failure Investigation, Documentation, and Analysis for Power Transformers and Shunt Reactors, IEEE

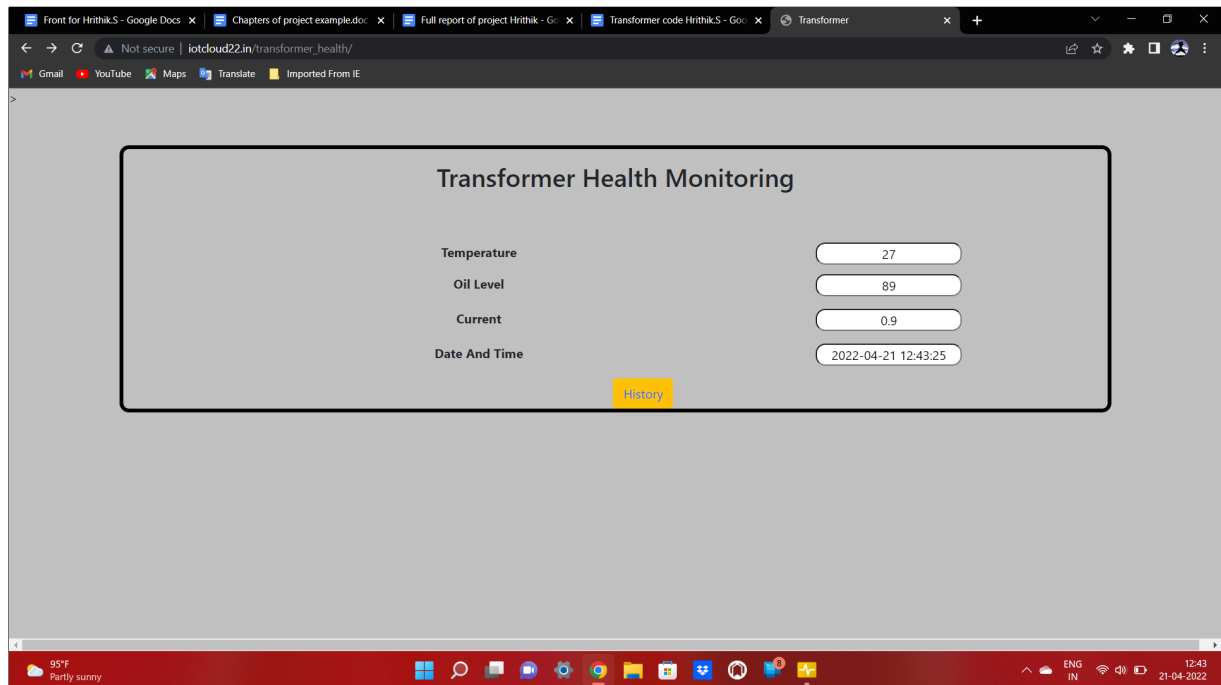
Standards Board.

[15] T.Asha Rakshana, R.Palselvam,S.Priya, K.Genga Devi, IOT Based Solar Panel Fault Monitoring And Control By Using Wi-Fi Modem, SSRG International Journal of Electrical and Electronics Engineering 7(1) (2020): 16-21.

[16] Lance R. Lewand, Condition Assessment of Transformers – Analysis of oil Data and its Quality, Neta World: The official publication of the International Electrical Testing Association, winter 2005-2006.

SCREEN SHOT:





Front for Hrithik.S - Google DocsChapters of project example.docFull report of project Hrithik - GTransformer code Hrithik.S - GoSensor Details

Not secure | iotcloud22.in/transformer_health/sensor_fetchdata.php

GmailYouTubeMapsTranslateImported From IE

Transformer Health Monitoring Sensors Data

S.no	Temperature	Oil_Level	Current	Date And Time
1	27	88	0.9	2022-04-21 12:44:14
2	27	89	0.9	2022-04-21 12:44:12
3	27	89	0.9	2022-04-21 12:44:10
4	27	89	0.9	2022-04-21 12:44:09
5	27	89	0.9	2022-04-21 12:44:07
6	27	88	0.9	2022-04-21 12:44:05
7	27	89	0.9	2022-04-21 12:44:03
8	27	89	1	2022-04-21 12:44:02
9	27	89	0.9	2022-04-21 12:44:00
10	27	89	0.9	2022-04-21 12:43:58
11	27	89	0.9	2022-04-21 12:43:57
12	28	89	0.9	2022-04-21 12:43:55
13	27	89	0.9	2022-04-21 12:43:53

95°F
Partly sunny