HEALTH MONITORING SYSTEM WITH VENTILATOR USING ARDUINO FOR PANDEMIC

A Project Report
submitted in partial fulfillment for the award of the degree of

Bachelor of Technology
in
Electronics and Communication Engineering

Submitted by

DEEPAK VISHWAKARMA
Roll No.: 2300970319007

SHUBHAM SINGH
Roll No.: 2300970319017

VIVEK GUPTA
Roll No.: 2300970319018

DEVANSHU GUPTA
Roll No.: 2300970319009

Under the Supervision of
Dr. Rakesh Sharma
Assistant Professor

Department of Electronics and Communication Engineering

(B. Tech ECE – Accredited by NBA),

Galgotias College of Engineering and Technology, Greater Noida.

(Affiliated to Dr.A.P.J Abdul Kalam Technical University, Lucknow)

December,2025

CERTIFICATE

This is to certify that the project report entitled **"HEALTH MONITORING SYSTEM WITH VENTILATOR USING ARDUINO FOR PANDEMIC "**submitted by **DEEPAK VISHWAKARMA (2300970319007), SHUBHAM SINGH (2300970319017), VIVEK GUPTA (23009703190018) And DEVANSHU GUPTA (2300970319009)** students of **B.Tech (Electronics and Communication Engineering)**, Galgotias College of Engineering and Technology, Greater Noida, is a bona-fide record of project work carried out under my supervision in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** from **Dr. A. P. J. Abdul Kalam Technical University, Lucknow**.

The work presented in this report has not been submitted earlier for the award of any degree or diploma.

Supervisor
(Signature)
Dr. RAKESH SHARMA
Assistant Professor
Department of ECE, GCET

Head of the Department
(Signature)

Date: _____
Place: Greater Noida

DECLARATION

We hereby declare that this project report titled **"Integrated Multi-Band Concentric Ring Antenna with Improved Spectrum Response for Modern Wireless Devices"** is an original work carried out by us under the supervision of **Dr. RAKESH SHARMA**, Department of Electronics and Communication Engineering, Galgotias College of Engineering and Technology, Greater Noida.

This work has not been submitted to any other university or institution for the award of any degree or diploma.

DEEPAK VISHWAKARMA
Roll No.: 2300970319007

SHUBHAM SINGH
Roll No.: 2300970319017

VIVEK GUPTA
Roll No.: 2300970319018

DEVANSHU GUPTA
Roll No.: 2300970319009

Date: _____
Place: Greater Noida

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to **Galgotias College of Engineering and Technology, Greater Noida**, for providing me with the opportunity to carry out this final year project.

I express my deepest thanks to my project supervisor **Dr. RAKESH SHARMA**, Assistant Professor, Department of Electronics and Communication Engineering, for his constant guidance, encouragement, and technical support throughout the duration of this project. His valuable suggestions and motivation played a crucial role in the successful completion of this work.

I am also thankful to the **Head of the Department**, faculty members of the ECE Department, and laboratory staff for providing necessary facilities and support.

I extend my heartfelt gratitude to my parents and friends for their continuous encouragement and moral support.

Finally, I thank everyone who directly or indirectly contributed to the successful completion of this project.

ABSTRACT

Human lungs use the reverse pressure generated by contraction motion of the diaphragm to suck in air for breathing. A contradictory motion is used by a ventilator to inflate the lungs by pumping type motion.

A ventilator mechanism must be able to deliver in the range of 10 – 30 breaths per minute, with the ability to adjust rising increments in sets of 2. Along with this the ventilator must have the ability to adjust the air volume pushed into lungs in each breath. The last but now the least is the setting to adjust the time duration for inhalation to exhalation ratio.

Apart from this the ventilator must be able to monitor the patients blood oxygen level and exhaled lung pressure to avoid over/under air pressure simultaneously. The ventilator we here design and develop using arduino encompasses all these requirements to develop a reliable yet affordable DIY ventilator to help in times of pandemic.

We here use a silicon ventilator bag coupled driven by DC motors with 2 side push mechanism to push the ventilator bag. We use toggle switch for switching and a variable pot to adjust the breath length and the BPM value for the patient.

Our system makes use of blood oxygen sensor along with sensitive pressure sensor to monitor the necessary vitals of the patient and display on a mini screen. Also an emergency buzzer alert is fitted in the system to sound an alert as soon as any anomaly is detected.

The entire system is driven by arduino controller to achieve desired results and to assist patients in COVID pandemic and other emergency situations.

TABLE OF CONTENT

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

Introduction to Embedded Systems

What is an Embedded System?

An embedded system is a specialized electronic system that integrates hardware and software to perform a **dedicated task**. Unlike general-purpose computers, embedded systems are designed to execute a **specific function** efficiently and reliably. These systems are typically built around a **microcontroller or microprocessor** and operate within real-time constraints.

Embedded systems may work independently or interact with users, other systems, or networks. They are used to monitor and control physical parameters such as temperature, pressure, speed, and voltage in a wide variety of environments. Since they are produced for mass markets, embedded systems are designed to be **cost-effective, compact, and power-efficient**.

An embedded system is **not**:

- A general-purpose computer used mainly for data processing
- A desktop or laptop software-based system
- A conventional business or scientific computing application

Classification of Embedded Systems

1. High-End Embedded Systems

High-end embedded systems use **32-bit or 64-bit processors** and usually run a full-fledged **operating system** such as embedded Linux or Android. These systems support complex applications and advanced user interfaces.
Examples: Smartphones, tablets, personal digital assistants (PDAs), multimedia devices.

2. Low-End Embedded Systems

Low-end embedded systems are based on **8-bit or 16-bit microcontrollers** and use minimal or no operating system. Their hardware and software are specifically tailored for a single function.
Examples: Washing machines, microwave ovens, digital clocks, basic industrial controllers

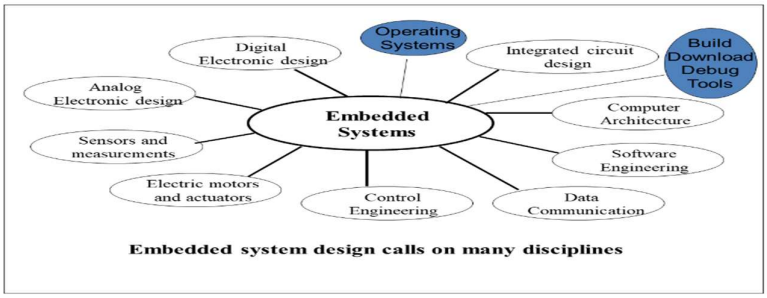


Figure 2(a): Embedded system design calls

EMBEDDED SYSTEM DESIGN CYCLE

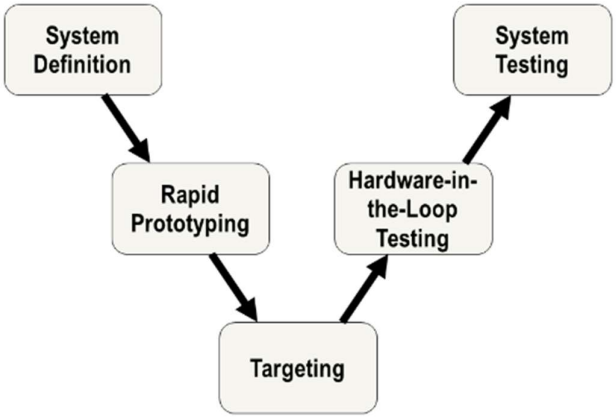


Figure:2(b) "V Diagram"

Characteristics of Embedded Systems

An embedded system is a **dedicated computing unit** that is built into a larger product and is not perceived as a standalone computer. It performs a **specific task** as part of a complete system.

Developing software for embedded systems presents several unique challenges beyond those found in general application programming:

- **High Throughput:** The system may be required to process a large volume of data within a very short time.
- **Fast Response:** Embedded systems often need to respond immediately to external events.
- **Testability:** Testing embedded software is difficult due to limited access to specialized hardware tools.
- **Debugging Difficulty:** Since most embedded systems lack keyboards, displays, or storage devices, identifying software errors becomes challenging.
- **High Reliability:** The system must function correctly under all conditions without human assistance.
- **Limited Memory:** Embedded systems have restricted memory capacity, so software and data must be optimized to fit available space.
- **Program Loading:** Special programming tools are needed to install software into embedded devices.
- **Low Power Consumption:** Battery-powered embedded systems must be designed to minimize energy usage.
- **Processor Load:** Tasks that require extensive CPU time can interfere with time-critical operations.
- **Cost Constraints:** Hardware costs must be kept low, often requiring software to run on minimal resources.

Typically, embedded systems consist of a **microcontroller or microprocessor**, memory units, and sometimes communication interfaces such as serial ports or network connections. Most systems do not include traditional input/output devices like keyboards, monitors, or disk drives.

Applications of Embedded Systems

Embedded systems are widely used in various domains, including:

1. Defense, military, and aerospace systems
2. Communication and networking devices
3. Industrial automation and process control
4. Management of complex system operations
5. Faster product development cycles
6. Real-time handling of continuously increasing data
7. Smart and autonomous sensor-based systems

Classification of Embedded Systems

Real-Time Systems (RTS)

A real-time system is one that must **respond to an event within a fixed time limit**. In such systems, correctness depends not only on the result but also on **timely execution**. Any response delivered after the deadline is considered incorrect.

Types of Real-Time Systems

1. Hard Real-Time Systems

Hard real-time systems require **strict adherence to deadlines**. Even a minor delay can lead to catastrophic consequences.

Examples:

- Nuclear reactor control systems
- Cardiac pacemakers

2. Soft Real-Time Systems

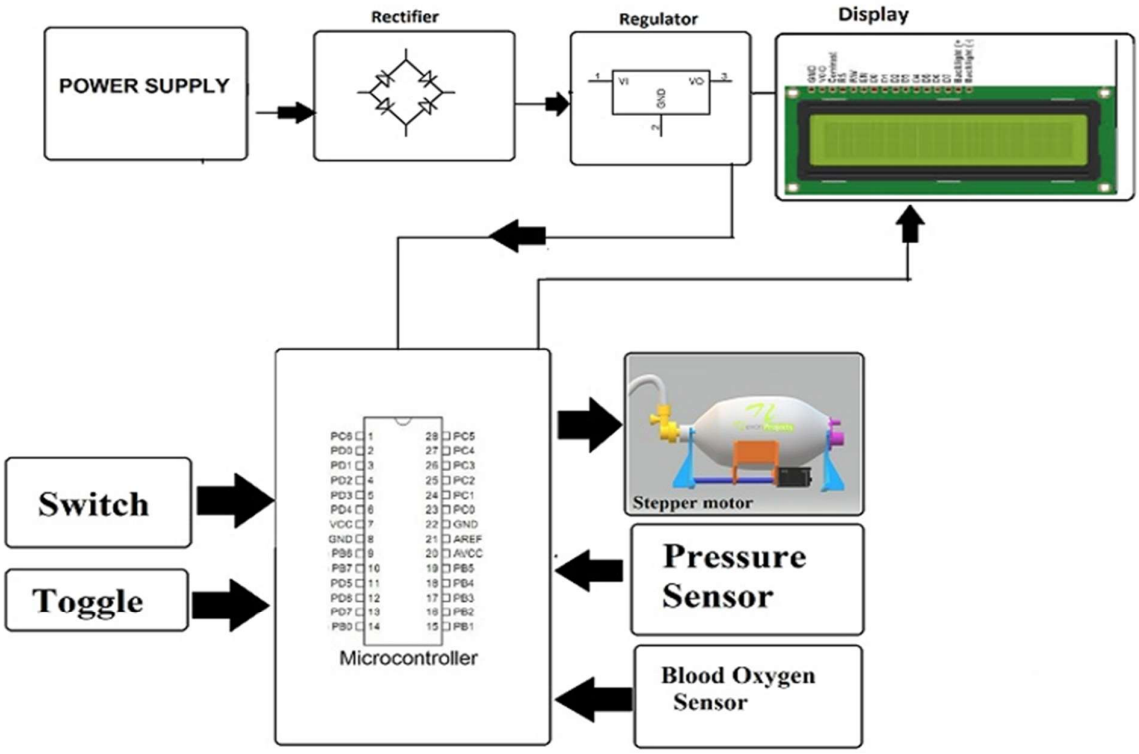
Soft real-time systems allow **limited delays**, but performance degradation may occur if deadlines are missed. Occasional lateness is acceptable without causing system failure.

Examples:

- Railway reservation systems (minor delays do not invalidate data)

CHAPTER 2
SYSTEM DESIGN AND BLOCK DIAGRAM

PROJECT BLOCK DIAGRAM



CHAPTER 3

HARDWARE REQUIREMENTS AND DESCRIPTION

HARDWARE COMPONENT:

- Atmega Controller
- Blood Oxygen Sensor
- Pressure Sensor
- Servo Motor
- Breather Mask
- Valves & Joints
- Air Breather Bag
- Push Rods
- Connector Rods
- Gear Mechanism
- Plastic Enclosure
- LCD Display IC
- Voltage Regulator IC
- Resistors
- Capacitors
- Transistors
- Cables and Connectors
- Diodes
- PCB and Breadboards
- LED
- Transformer / Adapter
- Push Buttons
- Buzzer

- IC
- IC Socket

VOLTAGE REGULATOR 7805

Features

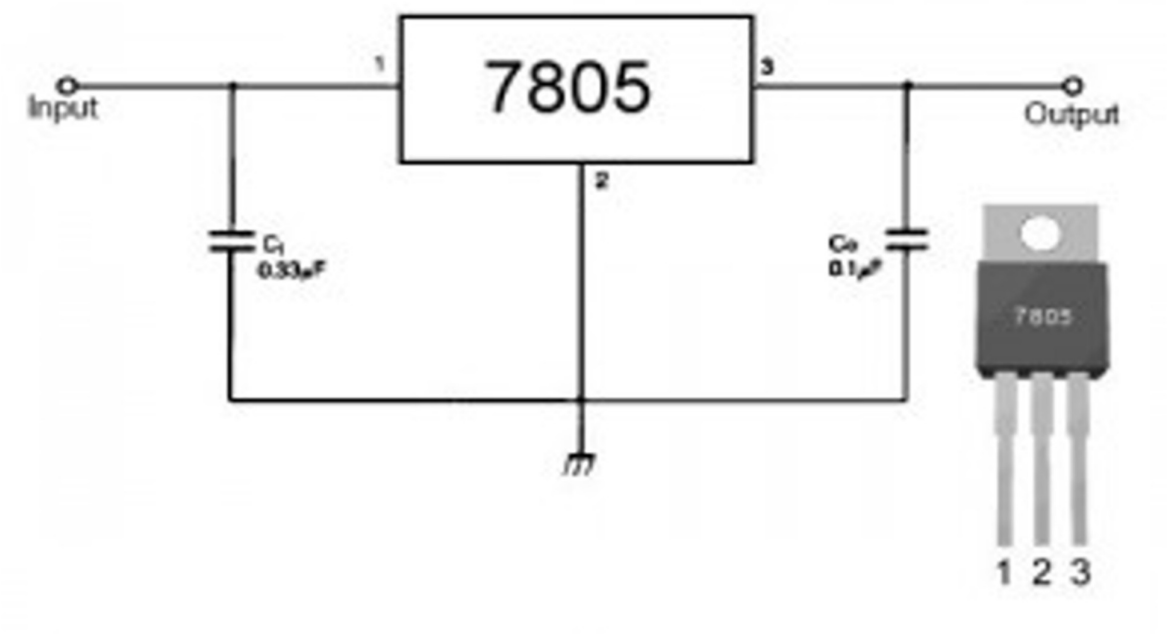
• Output Current up to 1A.

• Output Voltages of 5, 6, 8, 9, 10, 12, 15, 18, 24V.

• Thermal Overload Protection.

• Short Circuit Protection.

• Output Transistor Safe Operating Area Protection.



Description

The LM78XX and LM78XXA are **three-terminal positive voltage regulator ICs** designed to provide stable and fixed DC output voltages. These regulators are available in **TO-220 and D-PAK packages**, which makes them suitable for a wide variety of electronic applications.

Each regulator includes **built-in protection features** such as current limiting, thermal shutdown, and safe operating area control. These internal safety mechanisms protect the device from damage caused by overheating, overcurrent, or short-circuit conditions, ensuring reliable operation.

When proper heat sinking is used, the LM78XX series is capable of supplying **output currents greater than 1 ampere**. Although these ICs are mainly intended for fixed voltage regulation, they can also be combined with external components to achieve **adjustable output voltages and controlled current levels**.

frequency coverage, and suitability for modern wireless applications, including sub-6 GHz and millimeter-wave communication systems.

Rectifier

A rectifier is an electronic circuit used to convert **alternating current (AC)**, which changes direction periodically, into **direct current (DC)** that flows in a single direction. This conversion process is known as **rectification**. Rectifiers are commonly used in **power supply units** and in **signal detection circuits** such as radio receivers.

Rectifiers can be constructed using various components, including **semiconductor diodes**, vacuum tube diodes, or other specialized devices. In a typical power supply, the AC voltage obtained from the transformer is applied to the rectifier circuit, which converts it into **pulsating DC**.

Rectifier circuits are generally classified as **half-wave rectifiers** and **full-wave rectifiers**. In this project, a **bridge rectifier** is employed because it provides **full-wave rectification**, better output stability, and improved efficiency.

During the **positive half cycle** of the AC input, one pair of diodes conducts while the remaining diodes remain reverse biased. During the **negative half cycle**, the other pair of diodes conducts. In both cases, current flows through the load in the same direction, producing a continuous pulsating DC output.

Filter

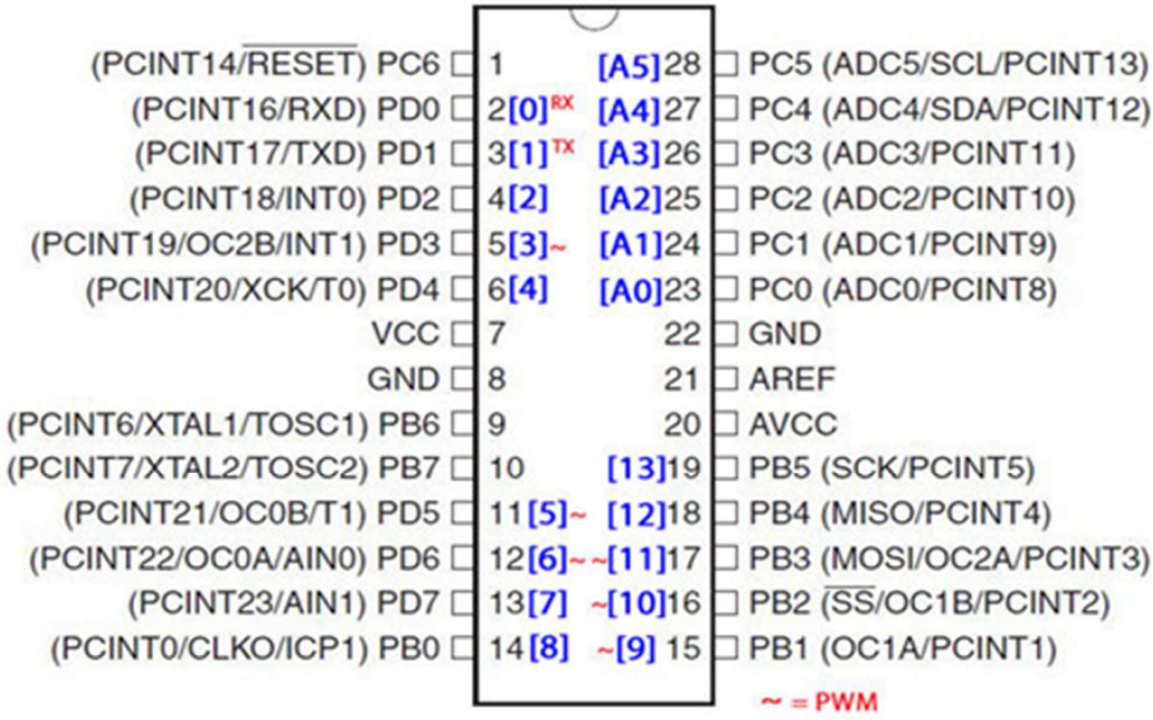
In this project, a **capacitor-based filter** is used to reduce the ripple content present in the rectifier output and to obtain a smoother **DC voltage**. The capacitor stores electrical energy during the peak of the rectified waveform and releases it when the voltage starts to fall, thereby minimizing fluctuations in the output.

As long as the **input AC supply** and the **load conditions** remain constant, the filtered DC output also remains nearly constant. However, any variation in the supply voltage or load current results in a corresponding change in the DC output. To maintain a stable and fixed voltage under such conditions, a **voltage regulator** is connected at the output stage.

The **capacitor filter** is the simplest and most basic form of power supply filtering. Due to its limitations, it is mainly used in applications where the **load current is low** and precise ripple control is not essential. Such filters are commonly applied in **high-voltage, low-current** power supplies, including circuits used for cathode-ray tubes and similar electronic devices.

The operation of this filter can be understood by observing the **charging and discharging behavior** of the capacitor across the rectified waveform, which helps in smoothing the pulsating DC output.

ATMEGA328



Introduction

The **Atmel ATmega328P** is a high-performance **8-bit microcontroller** based on the **AVR RISC architecture**. It offers **32 KB of Flash memory** and is capable of executing most instructions in a **single clock cycle**, achieving a throughput of up to **20 MIPS at 20 MHz**. The ATMEGA328-PU is available in a **28-pin PDIP package**, making it suitable for use with standard **28-pin AVR development boards**.

Unlike general-purpose computers, which are designed to perform multiple tasks such as data processing, internet browsing, and multimedia storage, **microcontrollers are built for dedicated applications**. They are commonly used in control-oriented tasks, such as automatically switching electrical devices ON or OFF based on sensor inputs like temperature or pressure.

Several microcontroller families are widely used depending on application requirements, including **8051, AVR, and PIC** microcontrollers. This project focuses on the **AVR family**, specifically the ATmega328P, due to its efficiency, low power consumption, and rich peripheral support.

Key Features of ATmega328P

1. Core Architecture

Feature	**Specification**
Architecture	8-bit AVR Advanced RISC
Instruction Set	131 instructions (most single-cycle execution)
Maximum Throughput	Up to 20 MIPS at 20 MHz
Working Registers	32 × 8-bit
Hardware Multiplier	2-cycle multiplier

2. Memory Organization

Memory Type	Capacity
Flash (Program Memory)	32 KB
EEPROM	1 KB
SRAM	2 KB
Flash Write/Erase Cycles	10,000
EEPROM Write/Erase Cycles	100,000
Data Retention	20 years at 85°C / 100 years at 25°C

Additional Memory Features	Description
Boot Loader Support	Optional boot loader with lock bits
In-System Programming	Supported via boot loader (ISP)
Read-While-Write	True read-while-write operation
Program Security	Programming lock bits available

3. Timers, Counters, and Analog Features

Feature	Specification
8-bit Timers/Counters	2 (with prescaler and compare modes)
16-bit Timer/Counter	1 (with prescaler, compare, and capture modes)
Real-Time Counter	With independent oscillator
ADC	10-bit, 6-channel
PWM Channels	6
Internal Temperature Sensor	Available

4. Communication Interfaces

Interface	Description
USART	Programmable serial communication
SPI	Master/Slave SPI interface
I2C	Philips-compatible (TWI)

5. System and Power Management Features

Feature	Description
Watchdog Timer	Programmable with internal oscillator
Analog Comparator	Internal
Interrupts	External interrupts and pin-change interrupts
Oscillator	Internal calibrated oscillator
Reset Features	Power-on reset and programmable brown-out detection

6. Power-Saving Modes

Sleep Mode	Description
Idle	CPU stopped, peripherals active
ADC Noise Reduction	Minimizes ADC noise

Power Save	Low power with RTC
Power Down	Lowest power consumption
Standby	Fast wake-up
Extended Standby	RTC active with low power

---

7. I/O, Package, and Electrical Characteristics

Parameter	Specification
Programmable I/O Pins	23
Package Type	28-pin PDIP
Operating Voltage	1.8 V – 5.5 V
Operating Temperature	-40°C to 85°C

---

8. Speed Grades

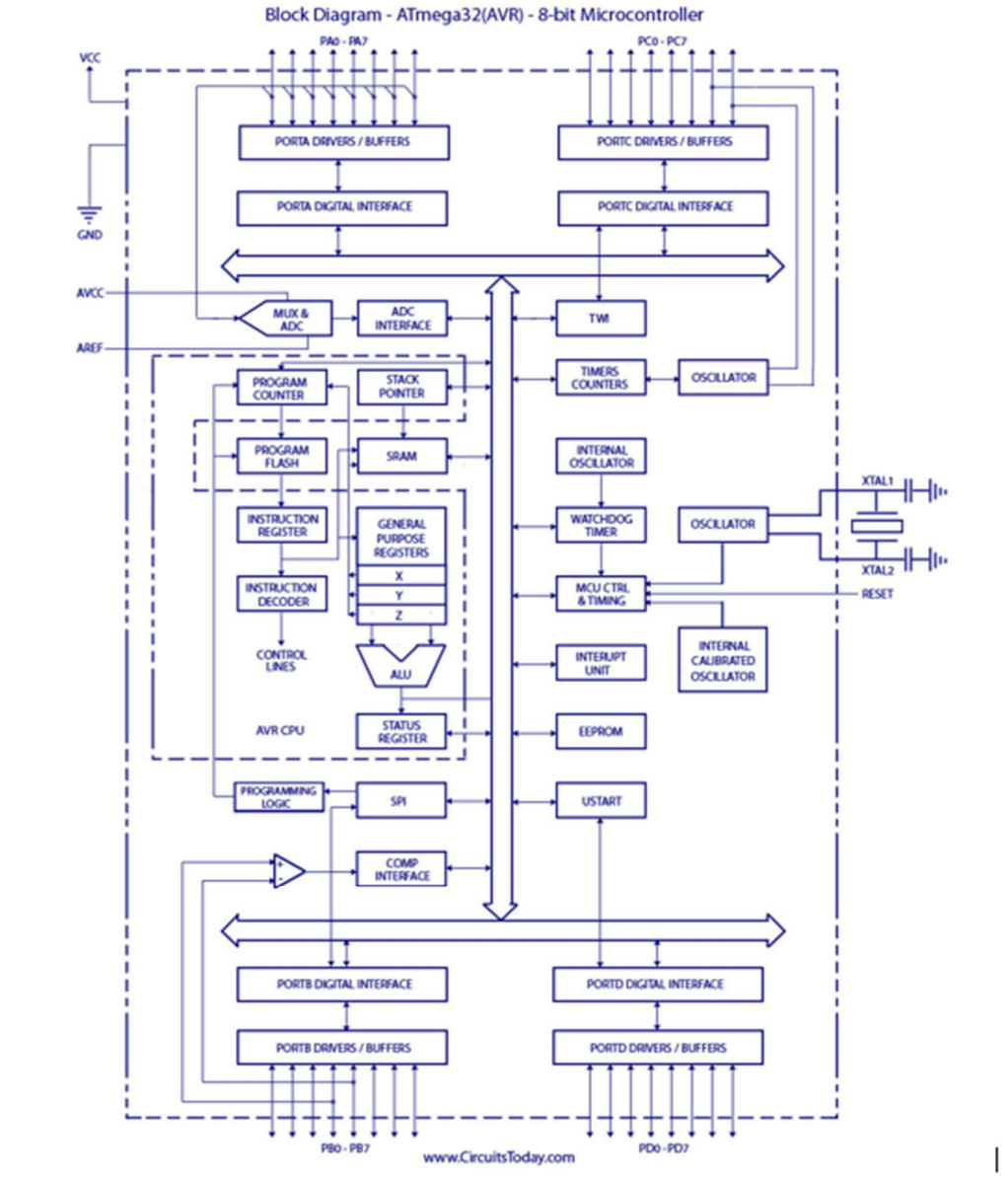
Frequency	Operating Voltage
0 – 4 MHz	1.8 – 5.5 V
0 – 10 MHz	2.7 – 5.5 V
0 – 20 MHz	4.5 – 5.5 V

---

9. Low Power Consumption (at 1.8 V, 1 MHz, 25°C)

Mode	Current Consumption
Active Mode	0.3 mA
Power-Down Mode	0.1 µA
Power-Save Mode	0.8 µA (including 32 kHz RTC)

BLOCK DIAGRAM ATMEGA32

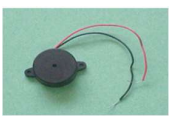


Buzzer

1. Introduction

A **buzzer**, also known as a **beeper**, is an electronic audio signaling device used to generate sound alerts. Buzzers can be **mechanical, electromechanical, or piezoelectric** in nature. They are widely used in electronic systems for indications such as alarms, timers, warnings, and user feedback.

Buzzers are commonly found in devices where simple sound signals are required, such as alert systems, confirmation signals for key presses, mouse clicks, and other user interface operations.



[21]

2. Types of Buzzers

a) Mechanical Buzzer

A mechanical buzzer operates purely on mechanical principles without the use of electronic components. A common example is the **joy buzzer**, which produces sound through mechanical vibration. These buzzers are simple in construction but are rarely used in modern electronic applications due to their limited control and reliability.

b) Electromechanical Buzzer

Electromechanical buzzers were among the earliest types of audio signaling devices. They function on a mechanism similar to an **electric bell**, but without a metal gong. In this type, an electromagnet repeatedly makes and breaks the circuit, causing vibrations that generate sound.

In some designs, a relay interrupts its own operating current, resulting in a continuous buzzing sound. These buzzers were often mounted on walls or ceilings to use the surface as a **sounding board**. The term *"buzzer"* originated from the harsh buzzing noise produced by these electromechanical devices.

c) Piezoelectric Buzzer

A piezoelectric buzzer uses a **piezoelectric crystal** that vibrates when an electrical signal is applied. It is driven by an oscillating electronic circuit or an audio signal source, often using a piezoelectric amplifier.

Piezoelectric buzzers are compact, energy-efficient, and highly reliable, making them the most commonly used type in modern electronic systems. The sounds produced may include **beeps, clicks, or ringing tones**, which are frequently used to indicate events such as button presses or system alerts.

LED (Light Emitting Diode)

1. Introduction

A **Light Emitting Diode (LED)** is a semiconductor device that emits light when an electric current flows through it. LEDs are widely used as **indicator lamps** in electronic equipment and are increasingly employed in **lighting applications** due to their efficiency and durability.

When an LED is **forward biased**, electrons combine with holes inside the semiconductor material. During this recombination process, energy is released in the form of **light (photons)**. This phenomenon is known as **electroluminescence**.

The **color of light** emitted by an LED depends on the **energy band gap** of the semiconductor material used. LEDs are generally very small in size (often less than 1 mm^2), and optical lenses are sometimes integrated to control and focus the direction of light output.

2. Advantages of LEDs

LEDs offer several benefits compared to traditional incandescent light sources:

- Low power consumption
- Long operational life
- Compact size
- High durability and reliability
- Fast switching speed
- Better resistance to shock and vibration

Because of these advantages, LEDs are increasingly replacing conventional lighting sources in many applications.

Types of LED'S

Fig 4.8(a): Types of LED

Electronic Symbol:

Fig 4.8(b): symbol of LED

White LEDs

Recent developments have led to the availability of **bright white LEDs**, which can effectively compete with incandescent bulbs for general lighting purposes. Although white LEDs are relatively more expensive than traditional lamps, they consume significantly **less current** and provide a **focused and efficient light beam**.

When operated within their specified ratings, white LEDs are highly reliable and require minimal maintenance. Their energy efficiency makes them suitable for both domestic and industrial lighting applications

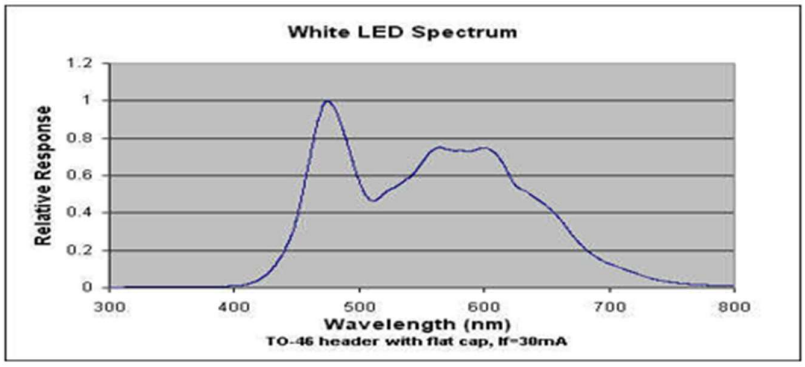


Fig 4.8(c): White LED spectrum

1N4007 Diode

. A **diode** is an electronic component used to convert **alternating current (AC)** into **direct current (DC)**. Diodes are commonly employed in **half-wave** and **full-wave rectifier** circuits.

While selecting a diode, the following three parameters must be considered:

1. Maximum forward current rating
2. Maximum reverse voltage rating
3. Maximum forward voltage drop

The **1N4007** diode belongs to the 1N400X series of rectifier diodes and is widely used in power supply circuits

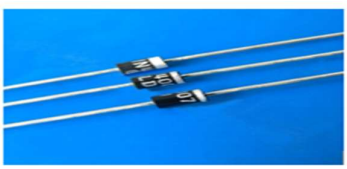


Fig: 1N4007 diodes

Key Characteristics of 1N400X Series

Diode Number	Maximum Reverse Voltage	Maximum Forward Current
1N4001 – 1N4007	50 V to 1000 V (varies by model)	1 A

All diodes in the 1N400X series have a **forward current rating of 1 ampere** but differ in their **reverse voltage capacity**. A diode with a **higher voltage rating** can replace a lower-rated diode, but a lower-rated diode must **not** be used in place of a higher-rated one.

For example, a **1N4007** diode can replace a **1N4002**, but a 1N4002 should not be used in place of a 1N4007.

Equivalent diodes manufactured by **BEL** include **BY125** (equivalent to 1N4001–1N4003), **BY126** (equivalent to 1N4004–1N4006), and **BY127** (equivalent to 1N4007).

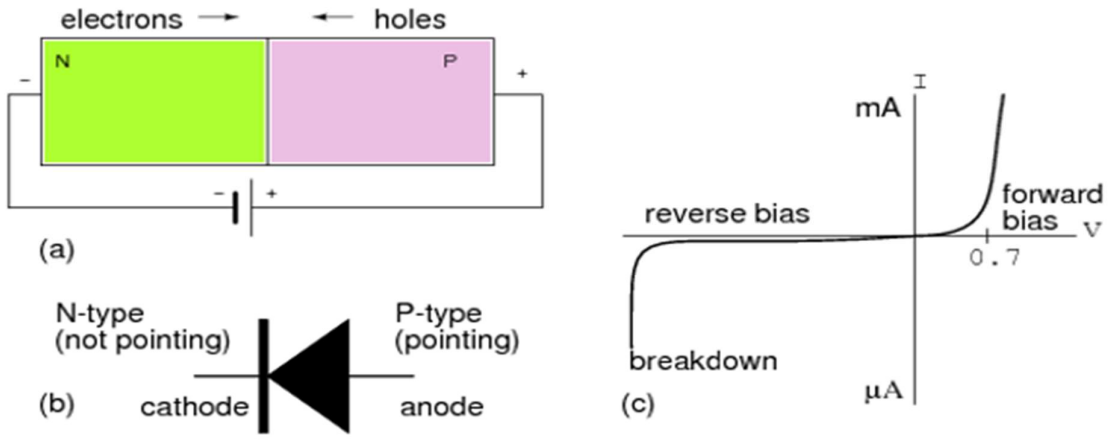


Fig:PN Junction diode

Resistor

A **resistor** is a **two-terminal passive electronic component** that restricts the flow of electric current in a circuit by creating a voltage drop across its terminals. It operates according to **Ohm's Law**, which states:

$V=IR$

Resistors are widely used in electronic and electrical circuits for **current control, voltage division, and power dissipation**. They are manufactured using materials such as **carbon compounds, metal films, and high-resistance alloy wires** like nickel-chromium.

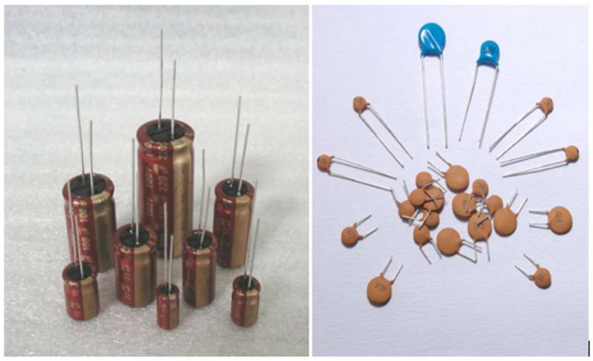
The main properties of a resistor include its **resistance value** and **power rating**. Other characteristics include **temperature coefficient, noise, and inductance**. Resistors are designed in different sizes so that they can safely dissipate heat without damage. They are commonly used in **printed circuits, hybrid circuits, and integrated circuits**.

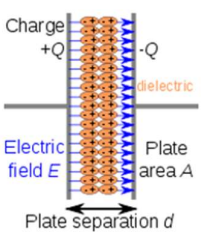
Capacitor

A **capacitor** is a **passive electronic component** used to **store electrical energy** in the form of an electric field. It consists of **two conducting plates** separated by an **insulating material called a dielectric**. When a voltage is applied across the plates, electric charge accumulates and energy is stored.

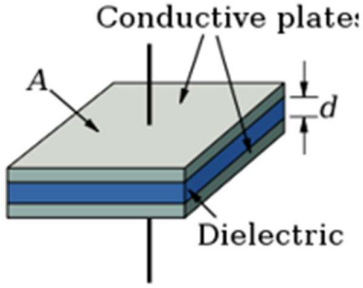
The ability of a capacitor to store charge is known as **capacitance**, which is measured in **farads (F)**. Capacitance depends on the **area of the plates**, the **distance between them**, and the **type of dielectric used**. In practical capacitors, small leakage current, resistance, and voltage breakdown limits are present.

Capacitors are widely used in electronic circuits for **filtering**, **smoothing power supply outputs**, **blocking DC while passing AC**, **signal coupling**, and **frequency tuning** in resonant circuits.





Parallel plate model



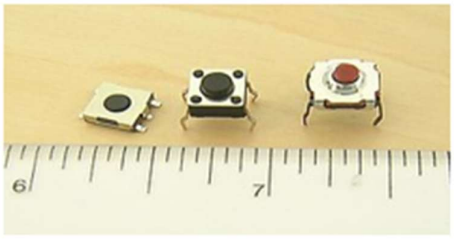
Dielectric is placed between two conducting plates, each of area A and with a separation of d. The simplest capacitor consists of two parallel conductive plates separated by a dielectric with permittivity ε (such as air). The model may also be used to make qualitative predictions for other device geometries. The plates are considered to extend uniformly over an area A and a charge density ±ρ = ±Q/A exists on their surface. Assuming that the width of the plates is much greater than their separation d, the electric field near the centre of the device will be uniform with the magnitude E = ρ/ε. The voltage is defined as the line integral of the electric field between the plates

$$V = \int_0^d E\mathrm{d}z = \int_0^d \frac{\rho}{\varepsilon}\mathrm{d}z = \frac{\rho d}{\varepsilon} = \frac{Qd}{\varepsilon A}.$$

Solving this for C = Q/V reveals that capacitance increases with area and decreases with separation

$$C = \frac{\varepsilon A}{d}.$$

PUSH BUTTONS



A **push button** is a simple **electromechanical switch** used to control the operation of a machine or system. It is usually made of **plastic or metal** and is designed to be easily pressed by a finger or hand. Most push buttons use a **spring mechanism** that returns the button to its original position after release.

Uses and Features

Push buttons are widely used in **industrial, commercial, and electronic applications**. In many systems, buttons are mechanically or electrically interlocked so that pressing one button (such as STOP) automatically releases another (such as START). This arrangement improves operational safety.

Push buttons are often **color-coded** to indicate their function:

- **Red**: Stop or emergency operation
- **Green**: Start or run operation

Emergency stop buttons usually have a **large mushroom-shaped head** for quick and easy operation, even when gloves are worn. In many industrial setups, push buttons include a **pilot light** to provide visual feedback, indicating whether an action has been successfully activated.

Push buttons are essential control elements in machines, panels, and embedded systems due to their **simplicity, reliability, and safety**

LIQUID CRYSTAL DISPLAY (LCD)

Description:
This is the example for the Parallel Port. This example doesn't use the Bi-directional feature found on newer ports, thus it should work with most, if not all Parallel Ports. It however doesn't show the use of the Status Port as an input for a 16 Character x 2 Line LCD Module to the Parallel Port. These LCD Modules are very common these days, and are quite simple to work with, as all the logic required running them is on board.

Pros:

- Very compact and light
- Low power consumption
- No geometric distortion
- Little or no flicker depending on backlight technology
- Not affected by screen burn-in
- No high voltage or other hazards present during repair/service
- Can be made in almost any size or shape
- No theoretical resolution limit

LCD Background:

Frequently, an 8051 program must interact with the outside world using input and output devices that communicate directly with a human being. One of the most common devices attached to an 8051 is an LCD display. Some of the most common LCDs connected to the 8051 are 16x2 and 20x2 displays. This means 16 characters per line by 2 lines and 20 characters per line by 2 lines, respectively.

Fortunately, a very popular standard exists which allows us to communicate with the vast majority of LCDs regardless of their manufacturer. The standard is referred to as HD44780U, which refers to the controller chip which receives data from an external source (in this case, the 8051) and communicates directly with the LCD

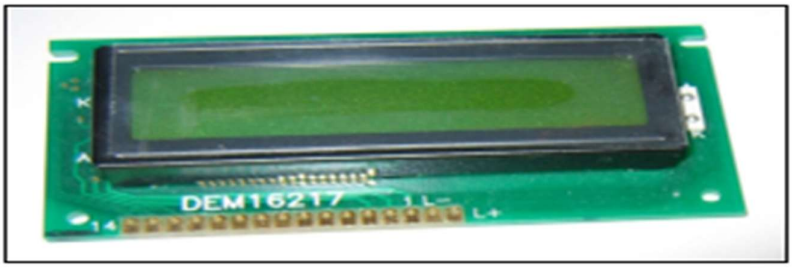
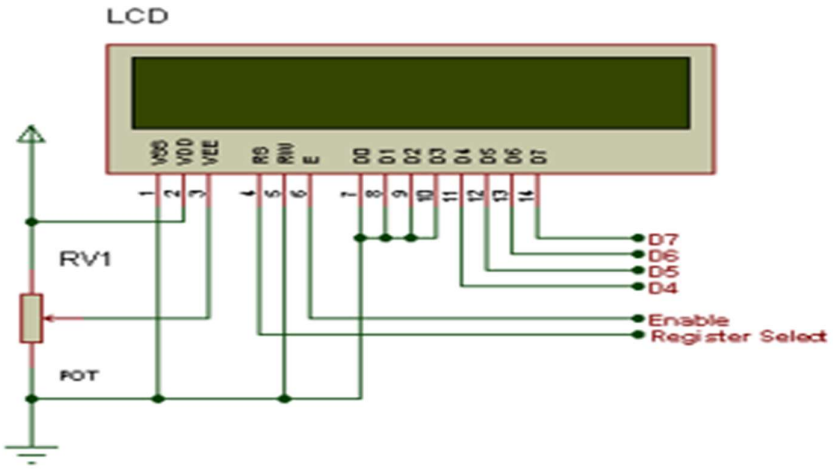


FIG 4.10: LCD



MAX30100 Pulse Oximeter Heart Rate Sensor Module

Heart Rate click carries Maxim's MAX30100 integrated pulse oximetry and a heart-rate sensor. It's an optical sensor that derives its readings from emitting two wavelengths of light from two LEDs – a red and an infrared one – then measuring the absorbance of pulsing blood through a photodetector. This particular LED colour combination is optimized for reading the data through the tip of one's finger.

The signal is processed by a low-noise analog signal processing unit and communicated to the target MCU through the mikroBUS I2C interface. Developers of end-user applications should note that the readings can be negatively impacted by excess motion and changes in temperature. Also, too much pressure can constrict capillary blood flow and therefore diminish the reliability of the data. A programmable INT pin is also available. The operates at the 3.3V power supply.

Main Applications:

1. Medical Monitoring Devices
2. Fitness Assistant Devices
3. Wearable Devices

Pressure Sensor with Analog and Digital Output

AMS 5812 is a series of high precision pressure sensors combining a ratiometric analog voltage output with a digital I2C output. They are calibrated and compensated in a temperature range of -25 ... 85 °C. The PCB- mountable sensors are available in a ceramic DIP package in variants with tube fittings or for o-ring sealing.

Features

- Calibrated and temperature compensated pressure sensor with analog and digital output (can be used simultaneously)
- Variants for all types of pressure: absolute pressure sensor, relative pressure sensor, differential pressure sensor, bidirectional differential pressure sensor
- Analog output: ratiometric, amplifed, 0.5 to 4.5 V
- Digital output for pressure and temperature data via I2C interface
- Resolution of the analog output: 11 bit
- Resolution of the digital output: 14 bit

- Supply voltage range 4.75 to 5.25 V (3.3 V variant available on request)
- High accuracy at room temperature, 0.5 %FSO for standard pressure ranges
- Low total error in the compensated temperature range of -25 ... 85 °C
- High long-term stability (maximum 0.5 %FSO/a)
- High speed measurements (typical response time 1 ms or sampling rate 1 kHz)
- Programmable I2C address
- Compact, ceramic DIL package (Length x width: 15.24 mm x 15.24 mm)
- Replacement / substitute product for SM5852 / SM5812 (Cross reference list AMS 5812 - SM5852 / SM5812)
- USB Starter Kit available for easy first operation, testing / measuring and adjustment of the I2C address
- Arduino Kits for Arduino Uno und Arduino Nano available
- RoHS conform

Typical Applications

- Static and dynamic pressure measurement
- Barometric pressure measurement
- Vacuum monitoring
- Gas flow
- Fluid level measurement
- Medical
- instrumentation:
  respirators, spirometers, CPAP, sleep diagnostics, anaesthesia, insufflators, pneumatic compression
- Heating, ventilation and air conditioning (HVAC)

Arduino Uno

Arduino Uno is the **most popular and widely used Arduino board**. When people refer to "Arduino," they usually mean the Uno. It is based on the **Atmel ATmega328 microcontroller** and is ideal for beginners.

The Arduino board is **not just a microcontroller**; it is a **complete prototyping board** that simplifies programming and hardware interfacing. It is **low-cost**, **USB-compatible**, and **easy to use**.

Legacy Arduino Boards

Before the Uno, some **legacy versions** were used:

- **Arduino NG**
- **Diecimila**
- **Duemilanove**

These boards used the **ATmega168 microcontroller** (less powerful than Uno's ATmega328) and had limitations like **manual power selection** and **reset requirements** that the Uno has improved.

Other Arduino Boards

- **Arduino Mega 2560**
  An advanced version of Arduino Uno, with **256 KB memory**, **54 digital I/O pins**, and more **analog & PWM pins**. Suitable for large and complex projects.
- **Arduino Mega ADK**
  A specialized Mega board designed for **interfacing with Android smartphones**.
- **Arduino LilyPad**
  Designed for **wearable and e-textile applications**. It can be sewn onto fabric and connected to other components using **conductive thread**.

Key Features of Arduino Uno

- Open-source hardware and software
- Easy USB interface (virtual serial port)
- Built-in voltage regulation (5V & 3.3V)
- ATmega328 microcontroller
- 16 MHz clock speed
- 32 KB flash memory
- 13 digital I/O pins and 6 analog pins
- ICSP connector for direct programming
- On-board LED on Pin 13 for debugging
- Reset button for restarting programs

STEPPER DRIVER

TB6600 Arduino Stepper Motor Driver is an easy-to-use professional stepper motor driver, which could control a two-phase stepping motor. It is compatible with Arduino and other microcontrollers that can output a 5V digital pulse signal.

TB6600Arduino stepper motor driver has a wide range of power input, 9-42VDC power supply. And it is able to output 4A peak current, which is enough for the most of stepper motors. The stepper driver supports speed and direction control.

You can set its micro step and output current with 6 DIP switch. There are 7 kinds of micro steps (1, 2 / A, 2 / B, 4, 8, 16, 32) and 8 kinds of current control (0.5A, 1A, 1.5A, 2A, 2.5A, 2.8A, 3.0A, 3.5A) in all. And all signal terminals adopt high-speed optocoupler isolation, enhancing its anti-high-frequency interference ability. As a professional device, it is able to drive 57, 42-type two-phase, four-phase, hybrid stepper motor.

TB6600 Stepper Motor Driver Controller 4A 9~42V TTL 16 Micro-Step CNC 1 Axis Upgraded Version of the 42/57/86 Stepper Motor segment increased to 32 segments, suitable for high subdivision applications. Suitable for step motor :57, 42, 86Type 4 phase 2 phase (4 line 6 line 8).

Features :

H bridge bipolar constant current drive.

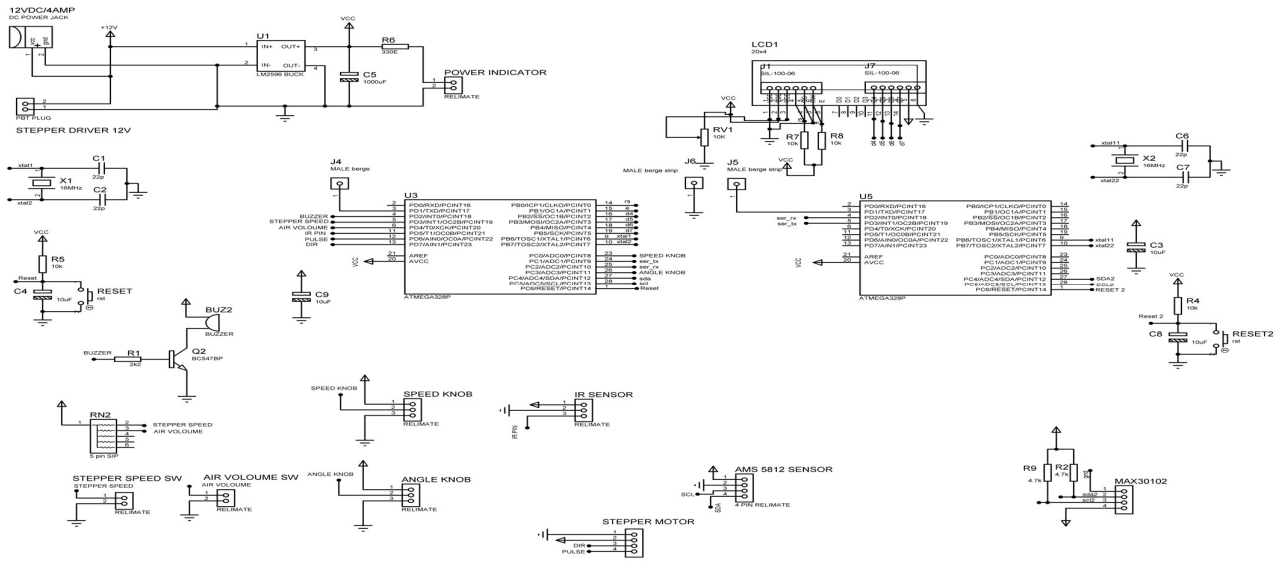
The biggest 4.0 A Eight kinds of output current are optional.

The biggest 32 To subdivide 6 kinds of subdivision modes selectable

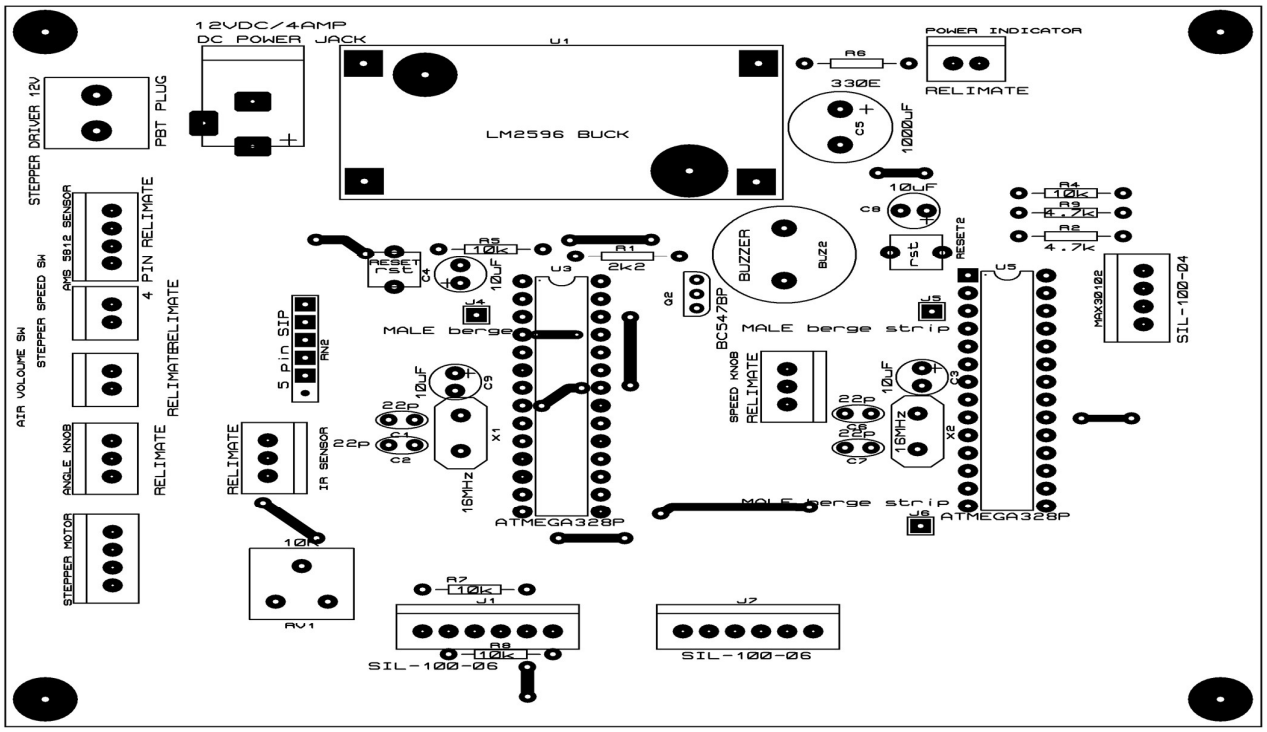
CHAPTER 4

SOFTWARE DESIGN AND CODING

SCHEMATIC DIAGRAM:



PCB LAYOUT :



CODING

```
// include the library code:#include <MsTimer2.h>
#include <LiquidCrystal.h>
#include <MsTimer2.h>
#include <Wire.h>
#include<mwc_stepper.h>
#include <SoftwareSerial.h>
SoftwareSerial atmega_master(A2, A1); // RX, TX----------rx of controller

// AMS5812 I2C address is 0x78(120)
//#define Addr 0x78
volatile int Addr = 0x78;
volatile bool key_val_1 = 1;
volatile bool key_val_2 = 1;
volatile bool exhaled_pressure_bit = 0;
volatile float psi;
#define EN_PIN 0
#define DIR_PIN 7
#define STEP_PIN 6
#define buzzer 2
#define ir_pin 5

const int rs = 8, en = 9, d4 = 10, d5 = 11, d6 = 12, d7 = 13;

bool do_other_processes_bit = 1;
//bool stepper_finished = 0;
int menu = 1;
int RPM = 200;
int RPM_2 = 30;
int step_angle = 5700;
int step_angle_2 = 500;
int clk_counter = 0;
int clk_counter_2 = 0;
//unsigned int data[4];
```

```
volatile unsigned int data[4];
volatile float pressure = 0.00;
volatile float new_pessure_reading = 0.00;

long nutral_disp_pressure_time;

String  receive_data = "00";
#define adjust_speed_knob A0
#define adjust_angle_knob A3
#define set_speed_key 3
#define set_angle_key 4

#define PULSE 3000
#define ClOCKWISE 0
#define OTHERWISE 1

MWCSTEPPER nema23(EN_PIN, DIR_PIN, STEP_PIN);

LiquidCrystal lcd(rs, en, d4, d5, d6, d7);

void setup() {
  // set up the LCD's number of columns and rows:
  // Initialise I2C communication as MASTER
  Wire.begin();
  lcd.begin(20, 4);
  lcd.setCursor(2, 0);
  lcd.print("DIY Ventilator");
  delay(1000);
  lcd.clear();
  lcd.setCursor(2, 1);
  lcd.print("Initilizing.....");
  pinMode(buzzer, OUTPUT);
  digitalWrite(buzzer, LOW);
  call_buzzer_beep();
```

```
  start_up_screen();
  nema23.init();
//  motor_home_position();                    //SET MOOTOR TO DEFAULT ANGLE
  Serial.begin(9600);
  Serial.println("            POWER ON");
  delay(100);

  MsTimer2::set(200, timer2_isr); // 500ms period
  MsTimer2::start();

  atmega_master.begin(9600);
  delay(2000);
}

void loop()
{
  receive_parameters();
  stepper_motor_control();

  if ( millis() - nutral_disp_pressure_time > 10000)
  {
    if (pressure == 0.00)
    {
      new_pessure_reading = 0.00;
      nutral_disp_pressure_time = millis();
    }
  }
}

void receive_parameters()
{
  atmega_master.listen();
  while (atmega_master.available() <= 0 );
  Serial.println("Waiting for incoming data");
  char inByte = atmega_master.read();
```

```
  Serial.write(inByte);
  if (inByte == '*')
  {
    receive_data = atmega_master.readStringUntil('#');
    Serial.print("data=");
    Serial.println(receive_data);
  }
  else
  {
    receive_data = "00";
  }
  atmega_master.end();
}

void motor_home_position()
{
  nema23.set(0, RPM, PULSE);
  nema23.run();
}

void start_up_screen()
{
  lcd.clear();
  lcd.setCursor(2, 0);
  lcd.print("DIY Ventilator");
  lcd.setCursor(3, 2);
  lcd.print("PRESS ANY KEY");//pressure_value
  lcd.setCursor(0, 3);
  lcd.print("TO START VENTILATOR");//pressure_value
  while (1)
  {
    key_val_1 = digitalRead(set_speed_key);
    key_val_2 = digitalRead(set_angle_key);
    if ((key_val_1 == 0) || (key_val_2 == 0))
    {
```

```
      call_buzzer_beep();
      delay(1000);
      break;
    }
  }
}

void call_buzzer_beep()
{
  digitalWrite(buzzer, HIGH);
  delay(300);
  digitalWrite(buzzer, LOW);
  delay(50);
}
void measure_air_pressure()
{
  // Request 4 bytes of data
  Wire.requestFrom(Addr, 4);
  // Read 4 bytes of data
  // pressure msb, pressure lsb, temp msb, temp lsb
  if (Wire.available() == 4)
  {
    data[0] = Wire.read();
    data[1] = Wire.read();
    data[2] = Wire.read();
    data[3] = Wire.read();
  }

  // Convert the data
  data[0] = (data[0] & 0xFF);
  data[0] = ((data[0] & 0xFF) * 256);
  data[1] = (data[1] & 0xFF);

  pressure = (data[0] + (data[1]));
  pressure = ((pressure - 3277.0) / ((26214.0) / 1.5));
```

```
 psi = pressure;
 pressure = 6894.76 * pressure;       //converting psi to pascal
 if (pressure < 35)
 {
  pressure = 0;
 }
 pressure = pressure * 0.0101972 * 2.8;            //converting pascal to cmH2O/////2.8 for adjusting visual reading
 Serial.println(pressure);

 if (pressure > 0.00)
 {
  new_pessure_reading =  pressure;
 }
 //  if (pressure < 0)
 // {
 //   pressure = pressure * (-1);
 // }
}
void measure_air_pressure()
{
 // Request 4 bytes of data
 Wire.requestFrom(Addr, 4);
 // Read 4 bytes of data
 // pressure msb, pressure lsb, temp msb, temp lsb
 if (Wire.available() == 4)
 {
  data[0] = Wire.read();
  data[1] = Wire.read();
  data[2] = Wire.read();
  data[3] = Wire.read();
 }

 // Convert the data
 data[0] = (data[0] & 0xFF);
 data[0] = ((data[0] & 0xFF) * 256);
```

```
  data[1] = (data[1] & 0xFF);

  pressure = (data[0] + (data[1]));
  pressure = ((pressure - 3277.0) / ((26214.0) / 1.5));
  psi = pressure;
  pressure = 6894.76 * pressure;        //converting psi to pascal
  if (pressure < 35)
  {
    pressure = 0;
  }
  pressure = pressure * 0.0101972 * 2.8;              //converting pascal to cmH2O/////2.8 for adjusting visual reading
  Serial.println(pressure);

  if (pressure > 0.00)
  {
    new_pessure_reading =  pressure;
  }
  //  if (pressure < 0)
  // {
  //   pressure = pressure * (-1);
  // }
}
void key_scanning()
{
  key_val_1 = digitalRead(set_speed_key);
  if (key_val_1 == 0)
  {
    MsTimer2::stop();
    lcd.setCursor(0, 3);
    lcd.blink();
    lcd.print("-->");
    delay(200);
    lcd.noBlink();
    delay(200);
```

```
menu = 2;

digitalWrite(buzzer, HIGH);
delay(100);
digitalWrite(buzzer, LOW);
delay(100);
delay(200);

while (menu == 2 )
{
 display_parameters();
 lcd.setCursor(0, 3);
 lcd.blink();
 lcd.print("-->");
 delay(200);
 lcd.noBlink();
 delay(200);
 int read_RPM = analogRead(adjust_speed_knob);              //50 to 160 value found

 RPM = map(read_RPM, 0, 1024, 160, 250);
 RPM_2 = map(read_RPM, 0, 1024, 9, 32);
 if (RPM_2 > 30)
 {
  RPM_2 = 30;
 }
 else if (RPM_2 < 10)
 {
  RPM_2 = 10;
 }
 key_val_1 = digitalRead(set_speed_key);
 if (key_val_1 == 0)
 {
  digitalWrite(buzzer, HIGH);
  delay(100);
  digitalWrite(buzzer, LOW);
```

```
      delay(100);
      menu = 1;
      //       delay(2000);
      MsTimer2::start();
      break;
    }
  }
}
key_val_2 = digitalRead(set_angle_key);
if (key_val_2 == 0)
{
  MsTimer2::stop();
  lcd.setCursor(11, 3);
  lcd.print("-->");
  lcd.blink();
  delay(200);
  lcd.noBlink();
  delay(200);

  menu = 3;

  digitalWrite(buzzer, HIGH);
  delay(100);
  digitalWrite(buzzer, LOW);
  delay(100);
  delay(200);

  while (menu == 3 )
  {
    display_parameters();
    lcd.setCursor(11, 3);
    lcd.print("-->");
    lcd.blink();
    delay(200);
    lcd.noBlink();
```

```
    delay(200);
    //      RPM = analogRead(adjust_speed_knob);              //50 to 160 value found
    int read_step_angle = analogRead(adjust_angle_knob);

    //      RPM = map(RPM, 0, 1024, 10, 165);

    step_angle = map(read_step_angle, 0, 1024, 0,5700);      //  800/360 * 70 degree == 155
    step_angle_2 = map(read_step_angle, 0, 1024, 8, 503);    //  800 is for full 360 degree
    if (step_angle_2 > 500)
    {
      step_angle_2 = 500;
    }
    else if (step_angle_2 < 10)
    {
      step_angle_2 = 10;
    }
    //      step_angle_2 = step_angle;
    key_val_2 = digitalRead(set_angle_key);
    if (key_val_2 == 0)
    {
      digitalWrite(buzzer, HIGH);
      delay(100);
      digitalWrite(buzzer, LOW);
      delay(100);
      menu = 1;
      //       delay(2000);
      MsTimer2::start();
      break;
    }
  }
}
else
{
  menu = 1;
}
```

```
}
void stepper_motor_control()
{
  nema23.set(ClOCKWISE, RPM, PULSE);
  for (size_t i = 0; i < step_angle; i++)
  {
    nema23.run();
    exhaled_pressure_bit = 0;
  }
  delay(250);

  exhaled_pressure_bit = 1;
  nema23.set(OTHERWISE, RPM, PULSE);
  for (size_t i = 0; i < step_angle; i++)
  {
    nema23.run();
  }
  delay(150);
}

//void stepper_motor_control()
//{
//  nema23.set(OTHERWISE, RPM, PULSE);
//  for (size_t i = 0; i < step_angle; i++)
//  {
//    nema23.run();
//    exhaled_pressure_bit = 0;
//  }
//  delay(250);
//
//  exhaled_pressure_bit = 1;
//  nema23.set(ClOCKWISE, RPM, PULSE);
//  for (size_t i = 0; i < step_angle; i++)
//  {
//    nema23.run();
```

```
// }
//  delay(250);
//}
void timer2_isr()
{
  if (menu == 1)
  {
    display_parameters();
    sei();                                ///important fundction to be called for using i2cinside interrupt functions
    if (exhaled_pressure_bit == 1)
    {
      measure_air_pressure();
    }
  }
  key_scanning();
}
#include "MAX30100_PulseOximeter.h"
#include <SoftwareSerial.h>
SoftwareSerial atmega_slave(3, 2);
#define REPORTING_PERIOD_MS     1000

PulseOximeter pox;
uint32_t tsLastReport = 0;
int heart_rate;
int spo2;
void onBeatDetected()
{
  Serial.println("Beat!");
}

void setup()
{
  Serial.begin(9600);
  atmega_slave.begin(9600);
```

```
  Serial.print("Initializing pulse oximeter..");
  if (!pox.begin())
  {
    Serial.println("FAILED");
    //    for (;;);
  } else {
    Serial.println("SUCCESS");
  }
  pox.setIRLedCurrent(MAX30100_LED_CURR_7_6MA);

  pox.setOnBeatDetectedCallback(onBeatDetected);
}

void loop()
{
  // Make sure to call update as fast as possible
  pox.update();
  if (millis() - tsLastReport > REPORTING_PERIOD_MS)
  {
    heart_rate = pox.getHeartRate();
    Serial.print("Heart rate:");
    Serial.println(heart_rate);

    spo2 = pox.getSpO2();
    if (spo2 > 99)
    {
      spo2 = 99;
    }
    Serial.print("bpm / SpO2:");
    Serial.print(spo2);
    Serial.println("%");
    transmit_parameter();       //send data
    tsLastReport = millis();
  }
}
```

```
void transmit_parameter()
{
  String spo2_string;
  if (spo2 <= 0)
  {
    spo2_string = String("00");
  }
  else
  {
    spo2_string = String(spo2);
  }
  atmega_slave.listen();
  atmega_slave.print('*');
  atmega_slave.print(spo2_string);
  atmega_slave.print('#');
  atmega_slave.end();  }
```

CHAPTER 5

HARDWARE TESTING

CONTINUITY TEST:

In electronics, a continuity test is the checking of an electric circuit to see if current flows (that it is in fact a complete circuit). A continuity test is performed by placing a small voltage (wired in series with an LED or noise-producing component such as a piezoelectric speaker) across the chosen path. If electron flow is inhibited by broken conductors, damaged components, or excessive resistance, the circuit is "open".

Devices that can be used to perform continuity tests include multi meters which measure current and specialized continuity testers which are cheaper, more basic devices, generally with a simple light bulb that lights up when current flows.

An important application is the continuity test of a bundle of wires so as to find the two ends belonging to a particular one of these wires; there will be a negligible resistance between the "right" ends, and only between the "right" ends.

This test is the performed just after the hardware soldering and configuration has been completed. This test aims at finding any electrical open paths in the circuit after the soldering. Many a times, the electrical continuity in the circuit is lost due to improper soldering, wrong and rough handling of the PCB, improper usage of the soldering iron, component failures and presence of bugs in the circuit diagram. We use a multi meter to perform this test. We keep the multi meter in buzzer mode and connect the ground terminal of the multi meter to the ground. We connect both the terminals across the path that needs to be checked. If there is continuation then you will hear the beep sound.

POWER ON TEST:

This test is performed to check whether the voltage at different terminals is according to the requirement or not. We take a multi meter and put it in voltage mode. Remember that this test is performed without microcontroller. Firstly, we check the output of the transformer, whether we get the required 12 v AC voltage.

Then we apply this voltage to the power supply circuit. Note that we do this test without microcontroller because if there is any excessive voltage, this may lead to damaging the controller. We check for the input to the voltage regulator i.e., are we getting an input of 12v and an output of 5v. This 5v output is given to the microcontrollers' 40^{th} pin. Hence we check for the voltage level at 40^{th} pin. Similarly, we check for the other terminals for the required voltage. In this way we can assure that the voltage at all the terminals is as per the requirement

TEXT BOOKS REFERED

1. ATMEGA 328 Data Sheets.

WEBSITES

- www.atmel.com
- www.beyondlogic.org
- www.wikipedia.org
- www.howstuffworks.com
- www.alldatasheets.com