INDEX

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Hospital bed management is a crucial aspect of healthcare systems, especially during medical emergencies and pandemics. The lack of an efficient bed vacancy detection system leads to delays in patient admissions, mismanagement of hospital resources, and overcrowding in medical facilities. To address this issue, we propose an IoT-based automatic bed vacancy detection system that integrates IR sensors, IoT technology, and a buzzer alert mechanism to streamline hospital bed allocation and improve healthcare efficiency.

The IR sensor detects the presence or absence of a patient on a hospital bed, ensuring accurate real-time monitoring of bed occupancy status. The collected data is transmitted through IoT technology to a centralized cloud-based system, enabling hospital administrators to remotely track bed availability and update records in real-time. A buzzer alarm is incorporated to provide an immediate alert when a patient vacates a bed, allowing staff to prepare for the next patient and ensuring efficient turnover.

This system reduces manual monitoring efforts, improves hospital management efficiency, and ensures that patients can quickly find available beds. Additionally, by integrating an IoT-based web dashboard, real-time data on hospital occupancy can be accessed remotely by patients, hospital staff, and emergency responders. The proposed system provides a cost-effective, scalable, and highly accurate solution for automated hospital bed vacancy detection and management.

Keywords – IR Sensor, 16×2 LCD, Buzzer

CHAPTER – 1

INTRODUCTION TO INTERNET OF THINGS

1.1 Introduction:

The Internet of Things may be a hot topic in the industry but it's not a new concept. In the early 2000's, Kevin Ashton was laying the groundwork for what would become the Internet of Things (IOT) at MIT's Auto ID lab. Ashton was one of the pioneers who conceived this notion as he searched for ways that Proctor & Gamble could improve its business by linking RFID information to the Internet. The concept was simple but powerful. If all objects in daily life were equipped with identifiers and wireless connectivity, these objects could be communicate with each other and be managed by computers.

In a 1999 article for the RFID Journal Ashton Wrote: Today, many of these obstacles have been solved. The size and cost of wireless radios has dropped tremendously. IPv6 allows us to assign a communications address to billions of devices. Electronics companies are building Wi-Fi and cellular wireless connectivity into a wide range of devices.

ABI Research estimates over five billion wireless chips will ship in 2013.2 Mobile data coverage has improved significantly with many networks offering broadband speeds. While not perfect, battery technology has improved and solar recharging has been built into numerous devices. There will be billions of objects connecting to the network with the next several years. For example, Cisco's Internet of Things Group (IOTG) predicts there will be over 50 billion connected devices by 2020.IoTdescribes a system where items in the physical world, and sensors within or attached to these items, are connected to the Internet via wireless and wired Internet connections.

These sensors can use various types of local area connections such as RFID, NFC, Wi-Fi, Bluetooth, and Zig bee. Sensors can also have wide area connectivity such as GSM, GPRS,3G, and LTE.

1.1.1 Connect both Inanimate and Living Things:

Early trials and deployments of Internet of Things networks began with connecting industrial equipment. Today, the vision of lot has expanded to connect everything from industrial equipment to everyday objects. The types of items range from gas turbines to automobiles to utility meters.

It can also include living organisms such as plants, farm animals and people. For example, the Cow Tracking Project in Essex uses data collected from radio positioning tags to monitor cows for illness and track behavior in the herd. Wearable computing and digital health devices, such as Nike+ Fuel band and Fit bit, are examples of how people are connecting in the Internet of Things landscape. Cisco has expanded the definition of IoT to the Internet of Everything (IoE), which includes people, places, objects and things.

1.1.2 Use Sensors for Data Collection:

The physical objects that are being connected will possess one or more sensors.
Each sensor will monitor a specific condition such as location, vibration, motion and temperature. In IoT, these sensors will connect to each other and to systems that can understand or present information from the sensor's data feeds. These sensors will provide new information to a company's systems and to people.

1.1.3 Change What Types of Item Communicate Over an IP Network:

In the past, people communicated with people and with machines. Imagine I fall of your equipment had the ability to communicate. What would it tell you? IoT enabled objects will share information about their condition and the surrounding environment with people, software systems and other machines. Going forward, everything will have a digital identity and connectivity, which means you can identify, track and communicate with objects. IoT data differs from traditional computing. These attributes present opportunities to collect a wide range of data but also provide challenges in terms of designing the appropriate data networking and security.

1.1.4 What It Means For Your Business:

IoT impacts every business. Mobile and the Internet of Things will change the types of devices that connect into a company's systems. It will improve public safety, transportation and healthcare with better information and faster communications of this information. While there are many ways that the Internet of Things could impact society and business, there are at least three major benefits of IOT that will impact every business, which include: communication, control and cost savings.

1.2 The Three Causes of IoT:

1.2.1 Communication:

IoT communicates information to people and systems, such as state and health of equipment (e.g. it's on or off, charged, full or empty) and data from sensors that can monitor a person's vital signs. In most cases, we didn't have access to this information before or it was collected manually and infrequently. For example, an IOT-enabled HVAC system can report if its air filter is clean and functioning properly.

Almost every company has a class of assets it could track. GPS-enabled assets can communicate their current location and movement. Location is important for items that move, such as trucks, but it's also applicable for locating items and people within an organization. In the healthcare industry, IoT can help a hospital track the location of everything from wheelchairs to cardiac de fibrillators to surgeons.

1.2.2 Control and Automation:

In a connected world, a business will have visibility into a device's condition. In many cases, a business or consumer will also be able to remotely control a device. For example, a business can remotely turn on or shut down a specific piece of equipment or adjust the temperature in a climate-controlled environment. Meanwhile, a consumer can use loT to unlock their car or start the washing machine. Once a performance baseline has been established, a process can send alerts for anomalies and possibly deliver an automated

response. For example, if the brake pads on a truck are about to fail, it can prompt the company to take the vehicle out of service and automatically schedule maintenance.

1.2.3 Cost Savings:

Many companies will adopt IoT to save money. Measurement provides actual performance data and equipment health, instead of just estimates. Businesses, particularly industrial companies, lose money when equipment fails. With new sensor information, IoT can help a company save money by minimizing equipment failure and allowing the business to perform planned maintenance.

1.3 How To Get Started:

These are just a few examples of how lot can help a business save money, automate processes and gain new insight into the business. To reap the benefits IoT can provide, a business should address at least the following four items:

1.3.1 Define what you'd like to learn from sensors:

Over the next three years, a majority of the devices purchased will have sensors and many existing items can be retrofitted with sensors. This will produce a wide range of new data sources for people and systems to use to improve their lives and existing business processes. Within a business setting, IT must define what types of information can be obtained from these sensors and work with business leaders to define which business processes can be improved with this new IOT information.

For example, sensor data that highlights anomalies in equipment vibration can be used to predict and avoid equipment failure.

1.3.2 Build an IOT Network and Security Foundation:

Many industrial IoT deployments have used proprietary networks. Instead of building proprietary networks, IT should connect IoT devices with standards-based IP networks. An IP- based network will help businesses deliver the performance, reliability and

interoperability that are required to support global lot networks and connections with partner ecosystems. The proliferation of connected sensors and equipment provides new security concerns. As IT embraces lot, it needs to ensure it has built safeguards into the solution including security procedures such as hardware and authentication structures will also need to be updated to support "things" as well as people.

1.3.3 Collect as much Data as Possible:

Businesses that don't plan carefully for lot will be overwhelmed with the volume and variety of data that IOT will generate. While each sensor may only produce a small amount of data, a company will be collecting data from thousands to millions of sensors. Firms must build a data collection and analytics strategy that supports this new torrent of information in a scalable and cost effective manner. Bigdata technology, such as Hadoop and NoSQL, can give companies the ability to rapidly collect, store and analyze large volumes of disparate IoT data. A company should collect any data that is relevant to existing processes. If possible and cost-effective, a company should also collect additional data that will enable the business to answer new questions in the future.

1.3.4 Review the Size and Scale of IoT Providers:

IoT is a complicated landscape with numerous categories and many vendors within each category. The four main categories of an IoT solution are: a sensor(s) and radio(s) that often sits in the machine, a M2M device-management platform, a solution delivery platform and apps that enable IoT devices to report or act on data. While there are many vendors, no single vendor offers a complete solution without building partnerships. As a firm begins its IOT voyage, IT and line of business executives should build a cross-functional team to evaluate strategic partners. The team should evaluate the financial position of the vendors, industry.

1.4 Advantages:

Here are some advantages of IOT:

- **Data:** The more the information, the easier it is to make the right decision. Knowing what to get from the grocery while you are out, without having to check on your own, not only saves time but is convenient as well.
- **Tracking:** The computers keep a track both on the quality and the viability of things at home. Knowing the expiration date of products before one consumes them improves safety and quality of life. Also, you will never run out of anything when you need it at the last moment.
- **Time:** The amount of time saved in monitoring and the number of trips done otherwise would be tremendous.
- **Money:** The financial aspect is the best advantage. This technology could replace humans who are in charge of monitoring and maintaining supplies.

1.5 The Internet of Things Applications:

Smart home. Smart Home clearly stands out, ranking as highest Internet of Things application on all measured channels.

- Wearables.
- Smart City.
- Smart grids.
- Industrial internet.
- Connected car.
- Connected Health (Digital health/Telehealth/Telemedicine).
- Smart retail.

CHAPTER – 2

PROJECT OVERVIEW

2.1 Introduction to Project:

Nowadays, the Internet of Things (IoT) is a significant factor in modern human society. It is affecting every aspect of human life and its environment. [4] The digitization of the healthcare industry has significantly elevated its status as one of the top sectors. The convergence of the digital realm and Internet of Things (IOT) technologies yields a significant influence. The Internet of Things (IoT) refers to a system of interconnected physical devices that facilitate global communication and data exchange without the need for direct human intervention.

The Internet of Things (IoT) employs a combination of electronic actuators, software, sensors, tools, and machines to facilitate the transfer of data from one location to another via the internet. [2] The challenges are faced in locating unoccupied beds and decentralizing details of bed occupants in hospitals. There is not a single system in the market that is solving this problem at the administration level or at the public level. So, to solve these kinds of challenges, we are going to propose a system that will solve this problem at the administration level in hospitals and for normal humans at the public level. This research paper proposes to develop an IOT-based automatic bed vacancy detection system.

That will help in solving the above-described problem. [3] During the pandemic of COVID-19, there was a lack of a system to verify the availability of beds equipped with the necessary infrastructure for the patients in question. This system is useful for normal days and in emergencies.

2.1.1 Existing System:

In the current hospital management system, bed allocation is mostly manual, requiring hospital staff to physically check bed occupancy status and update records accordingly. This process is time-consuming, error-prone, and

inefficient, especially during peak hospital hours or medical emergencies. Due to the lack of real-time automated monitoring, many hospitals struggle with overcrowding, delayed patient admissions, and miscommunication about bed availability.

Some hospitals use traditional pressure sensors or manual registration systems for bed monitoring, but these methods lack IoT-based real-time tracking and do not provide automated alerts when a bed becomes available. Additionally, during critical situations like pandemics or natural disasters, manual bed management results in delays in patient care, further worsening healthcare service delivery.

The absence of an automated buzzer alert system also makes it difficult for hospital staff to quickly identify vacant beds and prepare them for new patients, leading to inefficient hospital operations. There is a clear need for an intelligent, automated solution that eliminates manual intervention and ensures real-time monitoring of hospital beds.

2.1.2 Proposed System:

The proposed IoT-based automatic bed vacancy detection system integrates IR sensors, IoT connectivity, and a buzzer to provide real-time bed occupancy monitoring and automated alerts. The IR sensor detects whether a patient is occupying a bed or if it has been vacated. Once a patient leaves a bed, the system triggers a buzzer alert, notifying hospital staff for immediate action, such as bed sanitization and preparation for the next patient.

The IoT module transmits the bed status to a cloud-based hospital management system, allowing real-time updates that can be accessed by hospital administrators, doctors, and emergency responders via a web dashboard or mobile application. This reduces the need for manual verification and ensures that hospital staff can efficiently allocate beds to incoming patients.

Key features of the proposed system include:

- IR Sensor-based detection of bed occupancy and vacancy.
- IoT-enabled cloud connectivity for real-time remote monitoring.

- Buzzer alert mechanism to notify staff immediately after a bed is vacated.
- Automated web dashboard updates to display hospital bed availability.
- Scalable and cost-effective solution for modern healthcare facilities.

This system significantly enhances hospital efficiency, reduces patient waiting times, and improves healthcare management by providing accurate, real-time bed tracking.

2.2 Block Diagram:

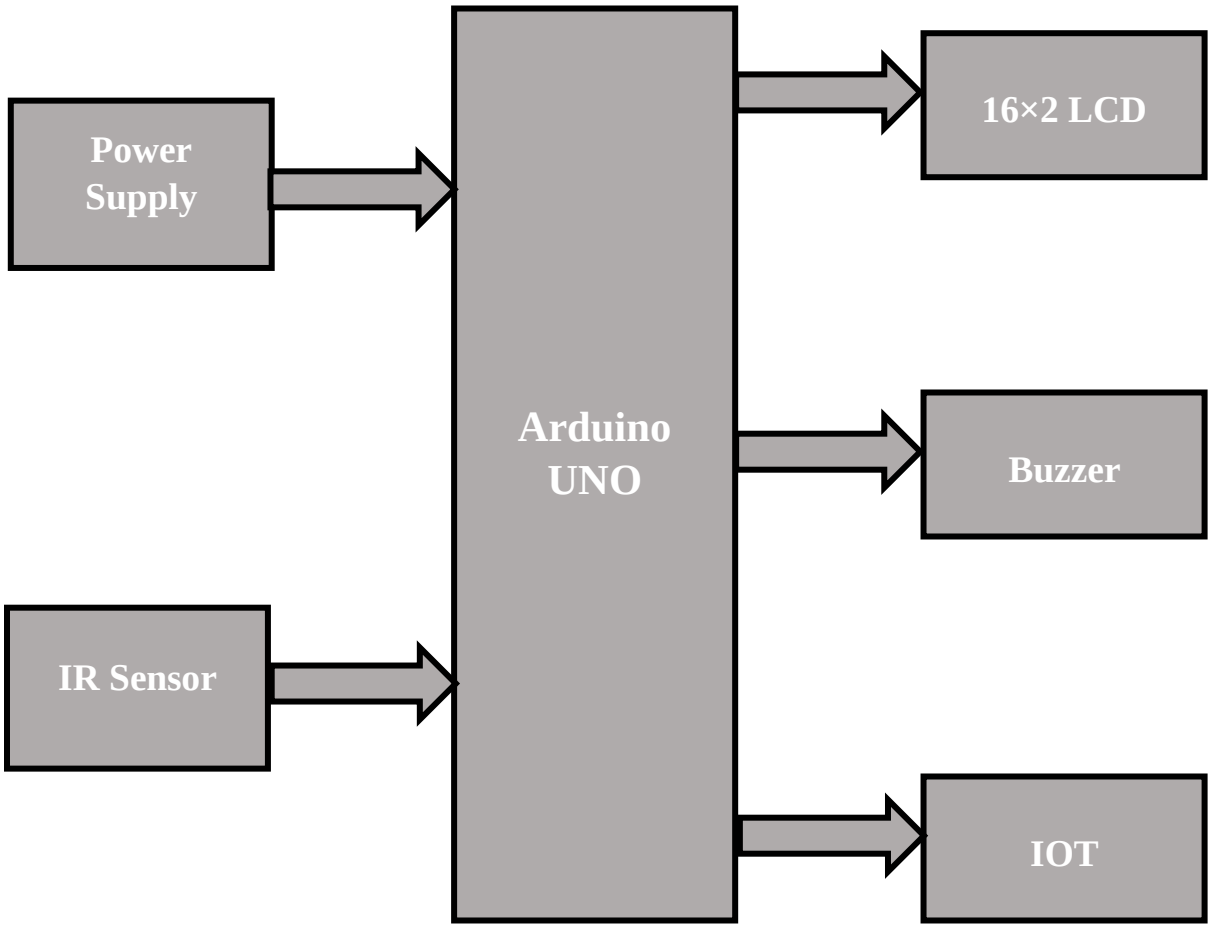

Fig 2.1: Block diagram of IoT-based automatic bed vacancy detection system

CHAPTER – 3

HARDWARE DESCRIPTION

3.1 Arduino UNO :

3.1.1 Introduction to Microcontroller:

Microcontroller as the name suggest, a small controller. They are like single chip computers that are often embedded into other systems to function as processing/controlling unit. For example, the control you are using probably has microcontrollers inside that do decoding and other controlling functions. They are also used in automobiles, washing machines, microwaves ovens, toys, etc., where automation is needed.

3.1.2 Arduino UNO Microcontroller:

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 oscillator, 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 or battery to get started.

The Uno differs from all preceding boards in that It does not use the FTDI USB-to-serial driver chip. Instead, it features the Atmega8U2 programmed as a USB-to-serial converter. “Uno” means “One” in Italian and is named to mark the upcoming release of Arduino 1.0. The Uno and version 1.0 will be the reference versions of Arduino, moving forward. The Uno is the latest in a series of USB Arduino boards, and the reference model for the Arduino platform; for a comparison with previous versions, see the index of Arduino boards.

The Arduino Uno can be powered via the USB connection or with an external power supply. The power source is selected automatically. External (non-USB) power can come either from an AC-to-DC adapter (wall-wart) or battery. The adapter can be connected by plugging a 2.1mm center-positive plug into the board’s power jack. Leads from a battery can be inserted in the Gnd and Vin pin headers of the POWER connector.

The board can operate on an external supply of 6 to 20 volts. If supplied with less than 7V, however, the 5Vpin may supply less than five volts and the board may be unstable. If using more than 12V, the voltage regulator may overheat and damage the board. The recommended range is 7 to 12 volts.

The power pins are as follows:

- **VIN.** The input voltage to the Arduino board when it's using an external power source (as opposed to5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.·
- **5V. The** regulated power supply used to power the microcontroller and other components on the board. This can come either from VIN via an on-board regulator, or be supplied by USB or another regulated 5V supply.
- **3.3V.**A 3.3 volt supply generated by the on-board regulator. Maximum current draw is 50 mA.
- **GND.** Ground pins

3.1.2.1 Memory:

The Atmega328 has 32 KB of flash memory for storing code (of which 0,5 KB is used for the bootloader); It has also 2 KB of SRAM and 1 KB of EEPROM (which can be read and written with the EEPROM library)..

3.1.2.2 Input and Output:

Each of the 14 digital pins on the Uno can be used as an input or output, using pinMode(), digitalWrite(), and digitalRead() functions. They operate at 5 volts. Each pin can provide or receive a maximum of 40 mA and has an internal pull-up resistor (disconnected by default) of 20-50 kOhms. In addition, some pins have specialized functions:

- **Serial: 0 (RX) and 1 (TX).** Used to receive (RX) and transmit (TX) TTL serial data. These pins are connected to the corresponding pins of the Atmega8U2 USB-to-TTL Serial chip.

- **External Interrupts: 2 and 3.** These pins can be configured to trigger an interrupt on a low value, arising or falling edge, or a change in value. See the attach Interrupt() function for details.
- **PWM: 3, 5, 6, 9, 10, and 11.** Provide 8-bit PWM output with the analogWrite() function.
- **SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK).** These pins support SPI communication, which although provided by the underlying hardware, is not currently included in the Arduino language.
- **LED: 13.** There is a built-in LED connected to digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.
- The Uno has 6 analog inputs, each of which provide 10 bits of resolution (i.e. 1024 different values). By default they measure from ground to 5 volts, though is it possible to change the upper end of their range using the AREF pin and the analogReference() function. Additionally, some pins have specialized functionality:
- **I2C: 4 (SDA) and 5 (SCL).** Support I2C (TWI) communication using the Wire library.

  There are a couple of other pins on the board:
- **AREF.** Reference voltage for the analog inputs. Used with analogReference().
- **Reset.** Bring this line LOW to reset the microcontroller. Typically used to add a reset button to shields which block the one on the board.

3.1.2.3 Communication:

The Arduino Uno has a number of facilities for communicating with a computer, another Arduino, or other microcontrollers. The Atmega328 provides UART TTL (5V) serial communication, which is available on digital pins 0 (RX) and 1 (TX). An Atmega8U2 on the board channels this serial communication over USB and appears as a virtual com port to software on the computer. The '8U2 firmware uses the standard USBCOM drivers, and no external driver is needed. However, on Windows, an *.inf file is required. The Arduino software includes a serial monitor which allows simple textual data to be sent to and from

the Arduino board. The RX and TX LEDs on the board will flash when data is being transmitted via the USB-to-serial chip and USB connection to the computer (but not for serial communication on pins 0 and 1). A Software Serial library allows for serial communication on any of the Uno's digital pins. The Atmega328 also support I2C (TWI) and SPI communication. The Arduino software includes a Wire library to simplify use of the I2C bus

3.1.3 Arduino UNO Board:

The Arduino Uno is a microcontroller board based on the Atmega328. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator, 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 or battery to get started.

The Uno differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it features the Atmega16U2 (Atmega8U2 up to version R2) programmed as a USB-to-serial converters

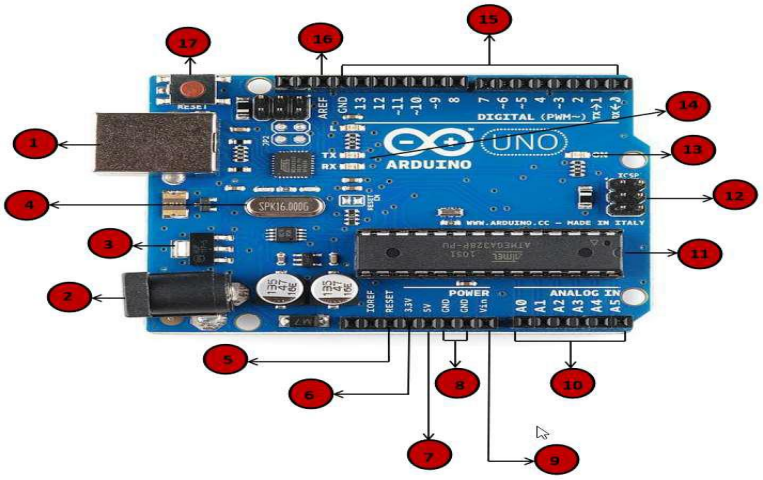

Fig 3.1: Arduino Uno board

3.1.3.1 Technical Specifications:

Table 3.1: Arduino UNO specifications

Features	Specifications
Microcontroller	ATmega328
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	14(of which 6 provide PWM output)
Analog Input Pins	6
DC Current pere I/O pin	40 mA
DC Current for 3.3v pin	50 mA
Flash Memory	32 KB(ATmega328) of which 0.5KB used by boot loader
SRAM	2 KB(ATmega328)
EEPROM	1 KB(ATmega328)
Clock Speed	16 MHz

The Arduino Uno can be powered via the USB connection or with an external power supply. The power source is selected automatically. External (non-USB) power can come either from an AC-to-DC adapter (wall-wart) or battery. The adapter can be connected by plugging a 2.1mm center-positive plug into the board's power jack. Leads from a battery can be inserted in the Gnd and Vin pin headers of the POWER connector. The board can operate on an external supply of 6 to 20 volts. If supplied with less than 7V, however, the 5V pin may supply less than five volts and the board may be unstable. If using more than 12V, the voltage regulator may overheat and damage the board. The recommended range is 7 to 12 volts.

- **USB Interface:**

  Arduino board can be powered by using the USB cable from your computer. All you need to do is connect the USB cable to the USB connection

- **External power supply:**

Arduino boards can be powered directly from the AC mains power supply by connecting it to the power supply (Barrel Jack).

- **Voltage Regulator:**

  The function of the voltage regulator is to control the voltage given to the Arduino board and stabilize the DC voltages used by the processor and other elements.

- **Crystal Oscillator:**

  The crystal oscillator helps Arduino in dealing with time issues. How does Arduino calculate time? The answer is, by using the crystal oscillator. The number printed on top of the Arduino crystal is 16.000H9H. It tells us that the frequency is 16,000,000 Hertz or 16 MHz

- **Arduino Reset:**

  It can reset your Arduino board, i.e., start your program from the beginning. It can reset the UNO board in two ways. First, by using the reset button (17) on the board. Second, you can connect an external reset button to the Arduino pin labelled RESET (5).

- **Pins (3.3, 5, GND, Vin):**
  - 3.3V (6): Supply 3.3 output volt
  - 5V (7): Supply 5 output volt
  - Most of the components used with Arduino board works fine with 3.3 volt and 5 volt.
  - GND (8)(Ground): There are several GND pins on the Arduino, any of which can be used to ground your circuit.
  - Vin (9): This pin also can be used to power the Arduino board from an external power source, like AC mains power supply.

- **Analog pins:**

  The Arduino UNO board has five analog input pins A0 through A5. These pins can read the signal from an analog sensor like the humidity sensor or temperature sensor and convert it into a digital value that can be read by the microprocessor.

- **Main microcontroller:**
- Each Arduino board has its own microcontroller (11). You can assume it as the brain of your board. The main IC (integrated circuit) on the Arduino is slightly

different from board to board. The microcontrollers are usually of the ATMEL Company. You must know what IC your board has before loading up a new program from the Arduino IDE. This information is available on the top of the IC. For more details about the IC construction and functions, you can refer to the data sheet.

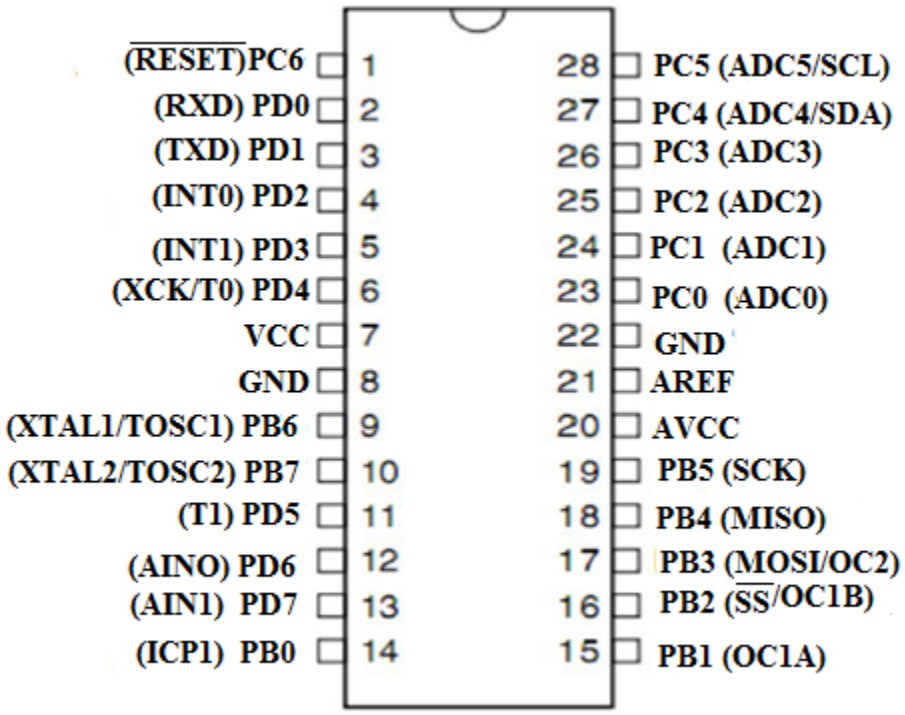

Fig 3.2: Pin diagram

The Atmega8U2 programmed as a USB-to-serial converter. “Uno” means “One” in Italian and is named to mark the upcoming release of Arduino 1.0. The Uno and version 1.0 will be the reference versions of Arduino, moving forward. The Uno is the latest in a series of USB Arduino boards, and the reference model for the Arduino platform; for a comparison with previous versions, see the index of Arduino boards

3.1.3.2 Pin Description:

- **VCC:** Digital Supply Voltage.
- **GND:** Ground.
- **Port B (PB[7:0]) 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.

- **Port C (PC[5:0]):** Port C is a 7-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The PC[5:0] output buffers have symmetrical drive characteristics with both high sink and source capability.

- **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. Shorter pulses are not guaranteed to generate a Reset.

- **Port D (PD[7:0]):** 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.

- **AVCC:** AVCC is the supply voltage pin for the A/D Converter, PC[3:0], and PE[3:2]. 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 PC[6:4] use digital supply voltage, VCC.

- **ADC [7:6] (TQFP and VFQFN Package Only):** In the TQFP and VFQFN package, ADC[7:6] serve as analog inputs to the A/D converter. These pins are powered from the analog supply and serve as 10-bit ADC channels.

- **12. ICSP pin:** Mostly, ICSP (12) is an AVR, a tiny programming header for the Arduino consisting of MOSI, MISO, SCK, RESET, VCC, and GND. It is often referred to as an SPI (Serial Peripheral Interface), which could be considered as an "expansion" of the output. Actually, you are slaving the output device to the master of the SPI bus

- **15. Digital I / O:** The Arduino UNO board has 14 digital I/O pins (15) (of which 6 provide PWM (Pulse Width Modulation) output. These pins can be configured to work as input digital pins to read logic values (0 or 1) or as digital output pins to drive different modules like LEDs, relays, etc. The pins labeled "~" can be used to generate PWM.

- **AREF:** AREF stands for Analog Reference. It is sometimes, used to set an external reference voltage (between 0 and 5 Volts) as the upper limit for the analog input pins working.

3.2 Power Supply:

Power supply is a reference to a source of electrical power. A device or system that supplies electrical or other types of energy to an output load or group of loads is called a power supply unit or PSU. The term is most commonly applied to electrical energy supplies, less often to mechanical ones, and rarely to others.

This power supply section is required to convert AC signal to DC signal and also to reduce the amplitude of the signal. The available voltage signal from the mains is 230V/50Hz which is an AC voltage, but the required is DC voltage (no frequency) with the amplitude of +5V and +12V for various applications.

In this section we have the Transformer, Bridge rectifier, are connected serially and voltage regulators for +5V and +12V (7805 and 7812) via a capacitor (1000uF) in parallel are connected parallel as shown in the circuit diagram below.

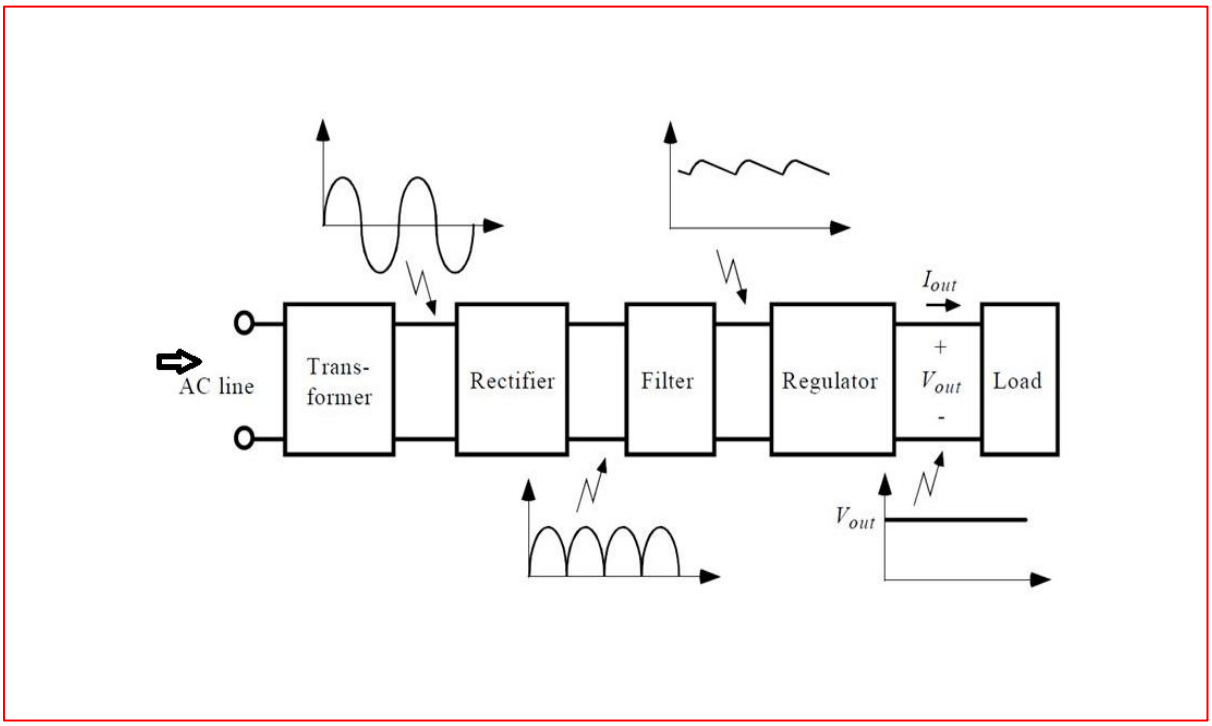

Fig 3.3: Power Supply

3.2.1 Transformer:

Transformer is a static device used to convert the voltage from one level to another level without change its frequency. There are two types of transformers

1. Step-up transformer
2. Step-down transformer

Step-up transformer converts low voltage level into high voltage level without change its frequency.

Step-down transformer converts high voltage level into low voltage level without change its frequency.

In this project we using step-down transformer which converts 230V AC to 12V AC [or] 230V AC to 5V as shown below

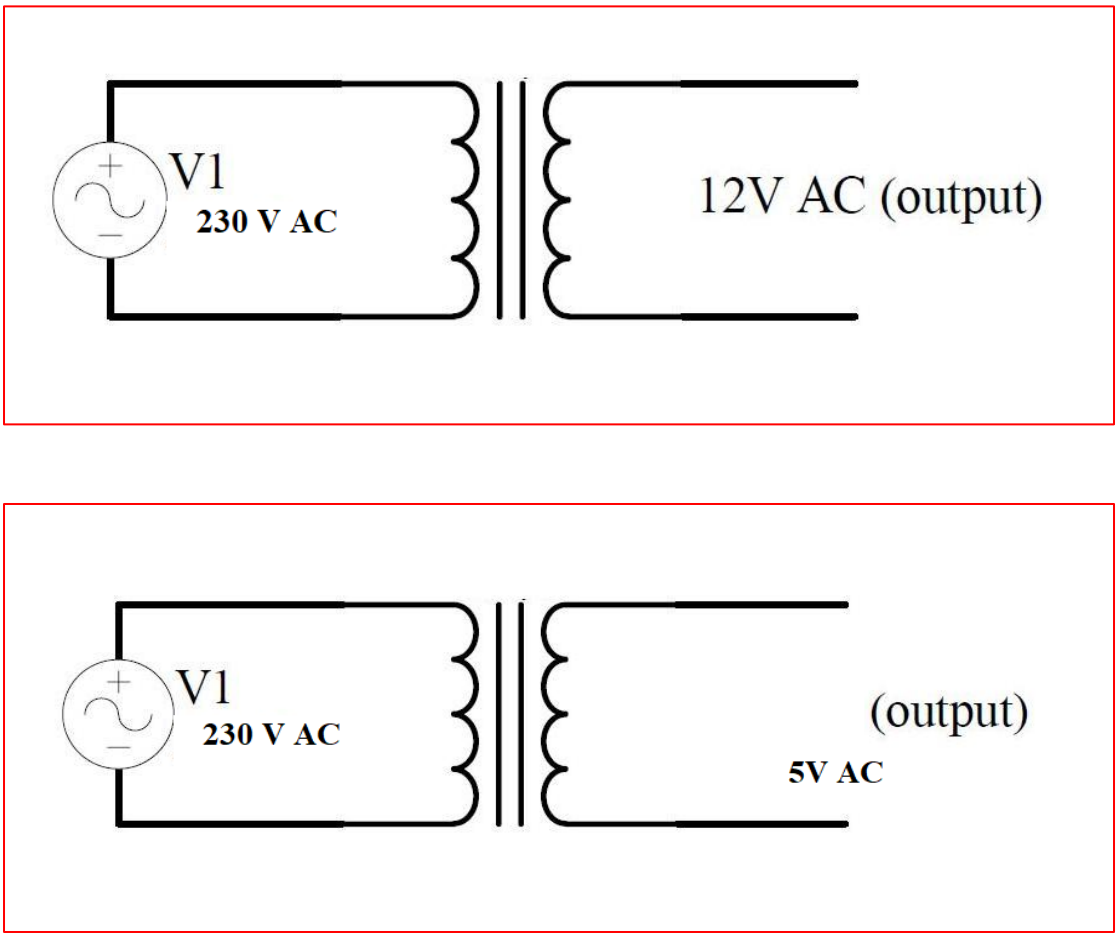

Fig 3.4: Transformers

3.2.2 Rectifier:

The purpose of a rectifier is to convert an AC waveform into a DC waveform (OR) Rectifier converts AC current or voltages into DC current or voltage. There are two different rectification circuits, known as **'half-wave'** and **'full-wave' rectifiers**. Both use components called **diodes** to convert **AC into DC**.

1. The Half-wave Rectifier

The half-wave rectifier is the simplest type of rectifier since it only uses one diode, as shown in figure.

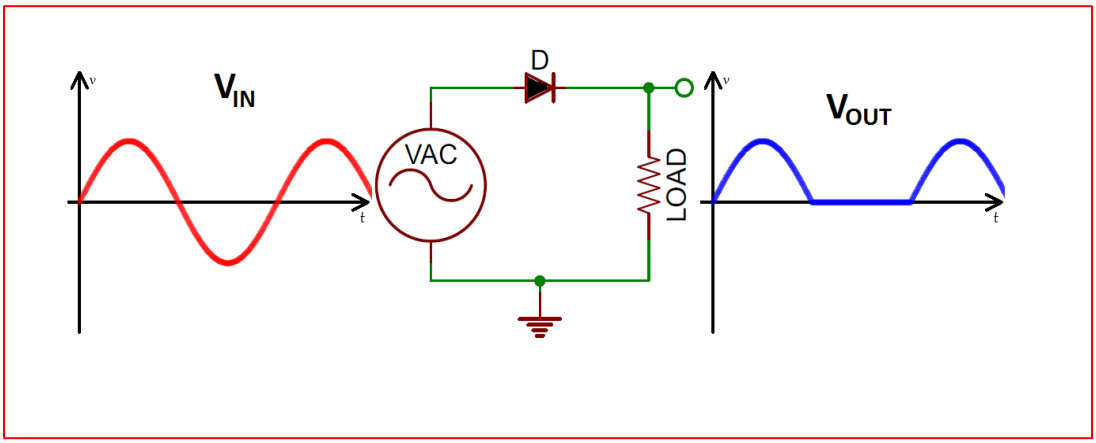

Fig 3.5: Half Wave Rectifier

Figure 3.6 shows the AC input waveform to this circuit and the resulting output. As you can see, when the AC input is positive, the diode is forward-biased and lets the current through. When the AC input is negative, the diode is reverse-biased and the diode does not let any current through, meaning the output is 0V. Because there is a 0.7V voltage loss across the diode, the peak output voltage will be 0.7V less than Vs.

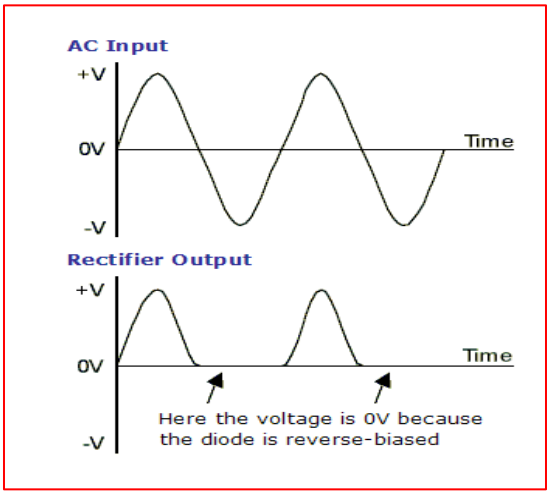

Fig 3.6: Half-Wave Rectification

While the output of the half-wave rectifier is DC (it is all positive), it would not be suitable as a power supply for a circuit. Firstly, the output voltage continually varies between 0V and Vs-0.7V, and secondly, for half the time there is no output at all.

The Full-wave Bridge Rectifier

The circuit in figure 3 addresses the second of these problems since at no time is the output voltage 0V. This time four diodes are arranged so that both the positive and negative parts of the AC waveform are converted to DC. The resulting waveform is shown in figure 4.

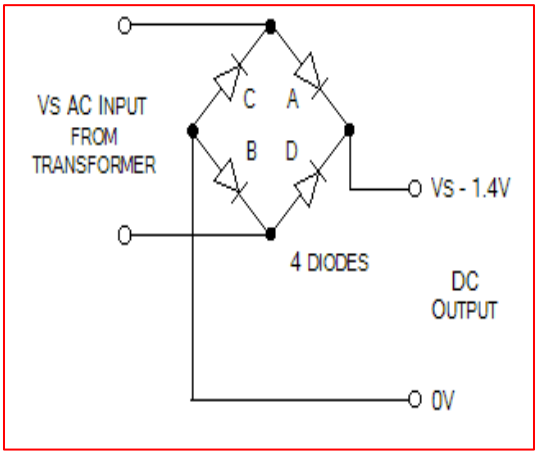

Fig 3.7: Full-Wave Rectifier

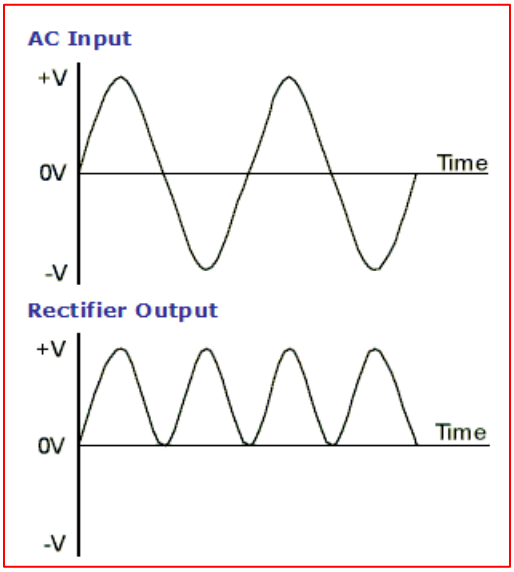

Fig 3.8: Full-Wave Rectification

When the AC input is positive, diodes A and B are forward-biased, while diodes C and D are reverse-biased. When the AC input is negative, the opposite is true – diodes C and D are forward-biased, while diodes A and B are reverse-biased.

While the full-wave rectifier is an improvement on the half-wave rectifier, its output still isn't suitable as a power supply for most circuits since the output voltage still varies between 0V and Vs-1.4V. So, if you put 12V AC in, you will 10.6V DC out.

3.2.3 Diodes:

Diodes allow electricity to flow in only one direction. The arrow of the circuit symbol shows the direction in which the current can flow. Diodes are the electrical version of a valve and early diodes were actually called valves.

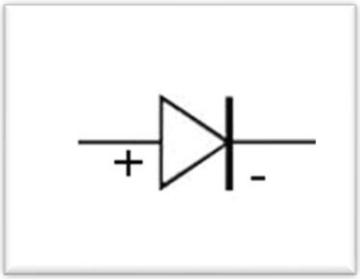

Fig 3.9: Diode Symbol

A **diode** is a device which only allows current to flow through it in one direction. In this direction, the diode is said to be 'forward-biased' and the only effect on the signal is that there will be a voltage loss of around 0.7V. In the opposite direction, the diode is said to be 'reverse-biased' and no current will flow through it.

3.3 LIQUID CRYSTAL DISPLAY:

3.3.1 Introduction to LCD:

A liquid crystal display (LCD) is a thin, flat display device made up of any number of color or monochrome pixels arrayed in front of a light source or reflector. Each pixel consists of

a column of liquid crystal molecules suspended between two transparent electrodes, and two polarizing filters, the axes of polarity of which are perpendicular to each other.

Without the liquid crystals between them, light passing through one would be blocked by the other. The liquid crystal twists the polarization of light entering one filter to allow it to pass through the other.

A 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 controller is an LCD display. Some of the most common LCDs connected to the controllers are 16X1, 16x2 and 20x2 displays. This means 16 characters per line by 1 line 16 characters per line by 2 lines and 20 characters per line by 2 lines, respectively.

Many microcontroller devices use 'smart LCD' displays to output visual information. LCD displays designed around LCD NT-C1611 module, are inexpensive, easy to use, and it is even possible to produce a readout using the 5X7 dots plus cursor of the display.

They have a standard ASCII set of characters and mathematical symbols. For an 8-bit data bus, the display requires a +5V supply plus 10 I/O lines (RS, RW, D7, D6, D5, D4, D3, D2, D1, D0).

For a 4-bit data bus it only requires the supply lines plus 6 extra lines (RS, RW, D7, D6, D5, D4). When the LCD display is not enabled, data lines are tri-state and they do not interfere with the operation of the microcontroller.

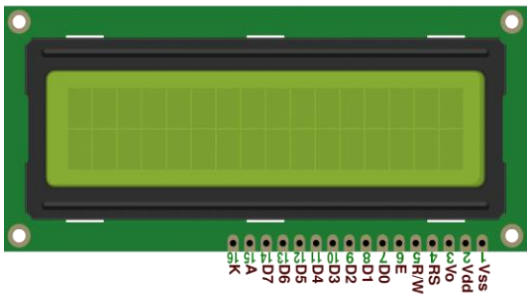

Fig 3.10: 16 X 2 LCD Display

3.3.2 General Description:

The LCD s used exclusively in watches, calculators and measuring instruments is the simple seven-segment displays, having a limited amount of numeric data. The recent advances in technology have resulted in better legibility, more information displaying capability and a wider temperature range. These have resulted in the LCD s being extensively used in telecommunications and entertainment electronics. The LCD s has even started replacing the cathode ray tubes (CRTs) used for the display of text and graphics, and also in small TV application

3.3.3 LCD Pin Diagram:

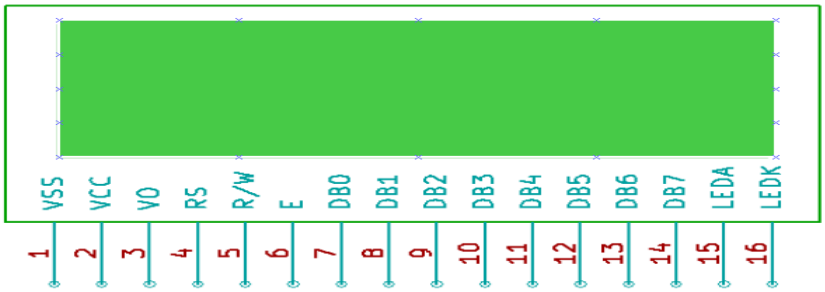

Fig 3.11: LCD Pin Diagram

Table 3.2: Pin Configuration & its Functions

PIN	SYMBOL	FUNCTION
1	VSS	Power Supply (GND)
2	VDD	Power Supply(+5V)
3	VO	Contrast Adjust
4	RS	Instruction/data register select
5	R/W	Data Bus Line
6	E	Enable Signal
7-14	DB0-DB7	Data Bus Line
15	A	Power supply for LED B/L (+)
16	K	Power Supply for LED B/L (-)

3.3.4 Control Lines:

EN:

Line is called "Enable." This control line is used to tell the LCD that you are sending it data. To send data to the LCD, your program should make sure this line is low (0) and then set the other two control lines and/or put data on the data bus. When the other lines are completely ready, bring EN high (1) and wait for the minimum amount of time required by the LCD datasheet (this varies from LCD to LCD), and end by bringing it low (0) again.

EN=0 => High Impedance

EN=1=> Low Impedance

RS:

Line is the "Register Select" line. When RS is low (0), the data is to be treated as a command or special instruction (such as clear screen, position cursor, etc.).When RS is high (1), the data being sent is text data which should be displayed on the screen.

RS = 0 => Command

RS = 1 => Data

`

RW:

Line is the "Read/Write" control line. When RW is low (0), the information on the data bus is being written to the LCD. When RW is high (1), the program is effectively querying (or reading) the LCD. Only one instruction ("Get LCD status") is a read command. All others are write commands, so RW will almost always be low.

RW = 0 => 0 Writing data to LCD

RW = 1 => 1 Reading data from LCD

3.3.5 Contrast Control:

To have a clear view of the characters on the LCD, contrast should be adjusted. To adjust the contrast, the voltage should be varied. For this, a preset is used which can behave like a variable voltage device. As the voltage of this preset is varied, the contrast of the LCD can be adjusted.

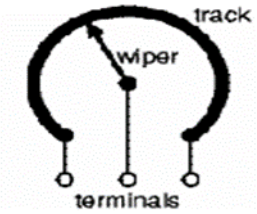

Fig 3.12: Variable Resistor

3.3.6 Potentiometer:

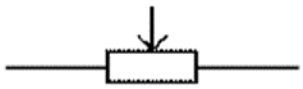

Fig 3.13: Potentiometer symbol

Variable resistors used as potentiometers have all **three terminals** connected. This arrangement is normally used to **vary voltage**, for example to set the switching point of a circuit with a sensor, or control the volume (loudness) in an amplifier circuit. If the terminals at the ends of the track are connected across the power supply, then the wiper terminal will provide a voltage which can be varied from zero up to the maximum of the supply.

3.3.7: Presets:

These are miniature versions of the standard variable resistor. They are designed to be mounted directly onto the circuit board and adjusted only when the circuit is built. For example, to set the frequency of an alarm tone or the sensitivity of a light-sensitive circuit, a small screwdriver or similar tool is required to adjust presets. Presets are much cheaper

than standard variable resistors so they are sometimes used in projects where a standard variable resistor would normally be used.

Multiturn presets are used where very precise adjustments must be made. The screw must be turned many times (10+) to move the slider from one end of the track to the other, giving very fine control.

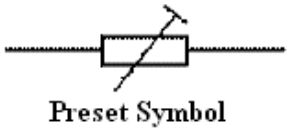

Fig 3.14: Preset symbol

3.4 Buzzer:

A buzzer is a small yet efficient component to add sound features to our project/system. It is very small and compact 2-pin structure hence can be easily used on breadboard, per board and even on PCBs which makes this a widely used component in most electronic applications. There are two types are buzzers that are commonly available. The one shown here is a simple buzzer which when powered will make a Continuous Beep sound, the other type is called a readymade buzzer which will look bulkier than this and will produce a Beep. Beep. Beep. Sound due tothe internal oscillating circuit present inside it. But, the one shown here is most widely used because it can be customized with help of other circuits to fit easily in our application. This buzzer can be used by simply powering it using a DC power supply ranging from 4V to 9V. A simple 9V battery can also be used, but it is recommended to use a regulated +5Vor +6V DC supply.

Magnetic Transducer

Magnetic transducers contain a magnetic circuit consisting of a iron core with a wound coil and a yoke plate, a permanent magnet and a vibrating diaphragm with a movable iron piece. The diaphragm is slightly pulled towards the top of the core by the magnet's magnetic field. When a positive AC signal is applied, the current flowing through the excitation coil produces a fluctuating magnetic field, which causes the diaphragm to vibrate up and down, thus vibrating air. Resonance amplifies vibration through resonator consisting of sound hole(s) and cavity and produces a loud sound.

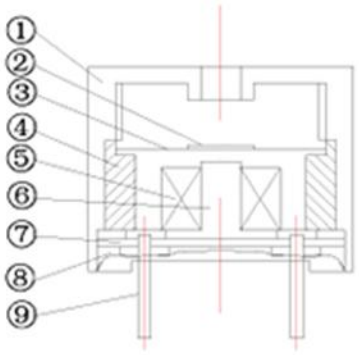

Fig 3.15: Magnetic Transducer

Magnetic Buzzer (Sounder)

Buzzers like the TMB-series are magnetic audible signal devices with built-in oscillating circuits. The construction combines an oscillation circuit unit with a detection coil, a drive coil and a magnetic transducer. Transistors, resistors, diodes and other small devices act as circuit devices for driving sound generators. With the application of voltage, current flows to the drive coil on primary side and to the detection coil on the secondary side. The amplification circuit, including the transistor and the feedback circuit, causes vibration. The oscillation current excites the coil and the unit generates an AC magnetic field corresponding to an oscillation frequency. This AC magnetic field magnetizes the yoke comprising the magnetic circuit. The oscillation from the intermittent magnetization prompts the vibration diaphragm to vibrate up and down, generating buzzer sounds through the resonator.

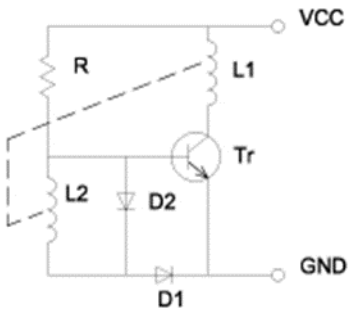

Fig 3.16: Recommended Driving Circuit for Magnetic Transducer

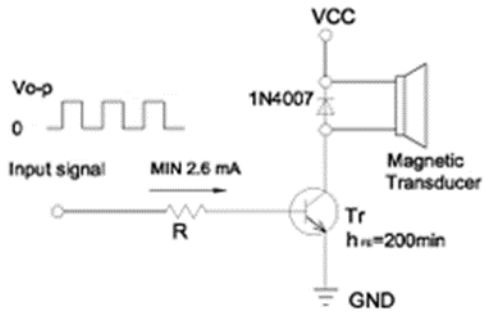

Fig 3.17: Introduction of Magnetic Buzzer (Transducer)

3.4.1 Pin Description:

- **Positive Pin**
  - Identified by (+) symbol or longer Terminal lead. Can be powered by 6 volts DC
- **Negative Pin**
  - Identified by (-) symbol or shorter Terminal lead. Typically connected to the ground of the circuit

3.4.2 Specification:

- **Rated Voltage:** 6V DC
- **Operating Voltage:** 4-8V DC
- **Rated Current**: <30mA.
- **Sound Type:** Continuous beep.
- **Resonant Frequency:** ~230 Hz
- Small and neat Sealed package
- Breadboard and Perf board friendly.

3.5 Relays:

A relay is an electrically controllable switch widely used in industrial controls, automobiles and appliances.

The relay allows the isolation of two separate sections of a system with two different voltage sources i.e., a small amount of voltage/current on one side can handle a

large amount of voltage/current on the other side but there is no chance that these two voltages mix up.

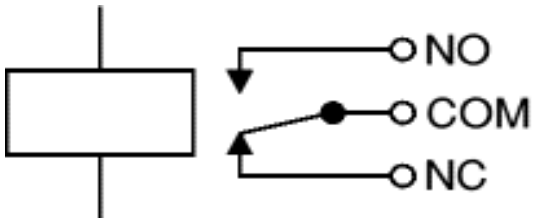

Fig 3.18: Circuit symbol of a relay

3.5.1 Operation:

When current flows through the coil, a magnetic field is created around the coil i.e., the coil is energized. This causes the armature to be attracted to the coil. The armature's contact acts like a switch and closes or opens the circuit. When the coil is not energized, a spring pulls the armature to its normal state of open or closed. There are all types of relays for all kinds of applications.

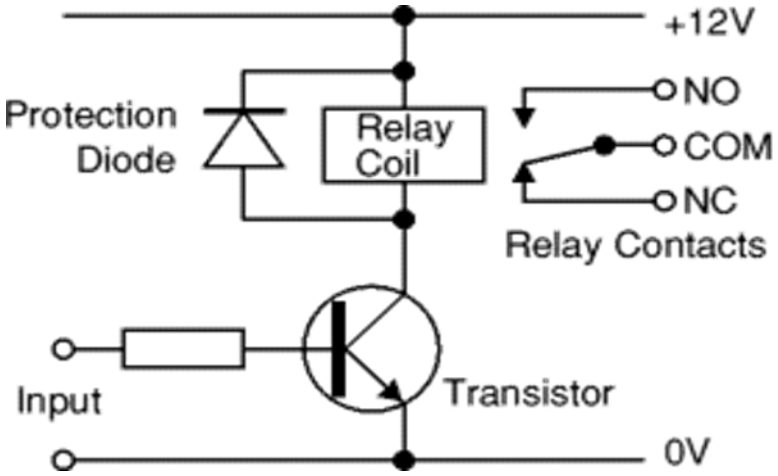

Fig 3.19: Relay Operation and use of Protection diodes

Transistors and ICs must be protected from the brief high voltage 'spike' produced when the relay coil is switched off. The above diagram shows how a signal diode (eg 1N4148) is connected across the relay coil to provide this protection. The diode is connected 'backwards' so that it will normally not conduct. Conduction occurs only when the relay coil is switched off, at this moment the current tries to flow continuously through the coil and it is safely diverted through the diode. Without the diode no current could flow

and the coil would produce a damaging high voltage 'spike' in its attempt to keep the current flowing.

In choosing a relay, the following characteristics need to be considered:

1. The contacts can be normally open (NO) or normally closed (NC). In the NC type, the contacts are closed when the coil is not energized. In the NO type, the contacts are closed when the coil is energized.

2. There can be one or more contacts. i.e., different types like SPST (single pole single throw), SPDT (single pole double throw) and DPDT (double pole double throw) relays.

3. The voltage and current required to energize the coil. The voltage can vary from a few volts to 50 volts, while the current can be from a few milliamps to 20milliamps. The relay has a minimum voltage, below which the coil will not be energized. This minimum voltage is called the “pull-in” voltage.

4. The minimum DC/AC voltage and current that can be handled by the contacts. This is in the range of a few volts to hundreds of volts, while the current can be from a few amps to 40A or more, depending on the relay.

3.5.2 Driving a Relay:

An SPDT relay consists of five pins, two for the magnetic coil, one as the common terminal and the last pins as normally connected pin and normally closed pin. When the current flows through this coil, the coil gets energized. Initially when the coil is not energized, there will be a connection between the common terminal and normally closed pin. But when the coil is energized, this connection breaks and a new connection between the common terminal and normally open pin will be established. Thus when there is an input from the microcontroller to the relay, the relay will be switched on. Thus when the relay is on, it can drive the loads connected between the common terminal and normally open pin. Therefore, the relay takes 5V from the microcontroller and drives the loads which consume high currents. Thus the relay acts as an isolation device.

Digital systems and microcontroller pins lack sufficient current to drive the relay. While the relay's coil needs around 10milli amps to be energized, the microcontroller's pin can provide a maximum of 1-2milli amps current. For this reason, a driver such as ULN2003 or a power transistor is placed in between the microcontroller and the relay. In order to operate more than one relay, ULN2003 can be connected between relay and microcontroller.

3.6 IR Sensor:

INFRARED TECHNOLOGY (IR)

Technically known as "infrared radiation", infrared light is part of the electromagnetic spectrum located just below the red portion of normal visible light – the opposite end to ultraviolet. Although invisible, infrared follows the same principles as regular light and can be reflected or pass through transparent objects, such as glass. Infrared remote controls use this invisible light as a form of communications between themselves and home theater equipment, all of which have infrared receivers positioned on the front. Essentially, each time you press a button on a remote, a small infrared diode at the front of the remote beams out pulses of light at high speed to all of your equipment. When the equipment recognizes the signal as its own, it responds to the command.

But much like a flashlight, infrared light can be focused or diffused, weak or strong. The type and number of emitters can affect the possible angles and range your remote control can be used from. Better remotes can be used up to thirty feet away and from almost any angle, while poorer remotes must be aimed carefully at the device being controlled.

The light our eyes see is but a small part of a broad spectrum of electromagnetic radiation. On the immediate high energy side of the visible spectrum lies the ultraviolet, and on the low energy side is the infrared. The portion of the infrared region most useful for analysis of organic compounds is not immediately adjacent to the visible spectrum, but is that having a wavelength range from 2,500 to 16,000 nm, with a corresponding frequency range from $1.9*10^{13}$ to $1.2*10^{14}$ Hz.(From http://hyperphysics.phy-

astr.gsu.edu/hbase/ems3.html : the frequency of infrared ranges from 0.003 - 4 x 10^{14} Hz or about 300 gigahertz to 400 terahertz.).

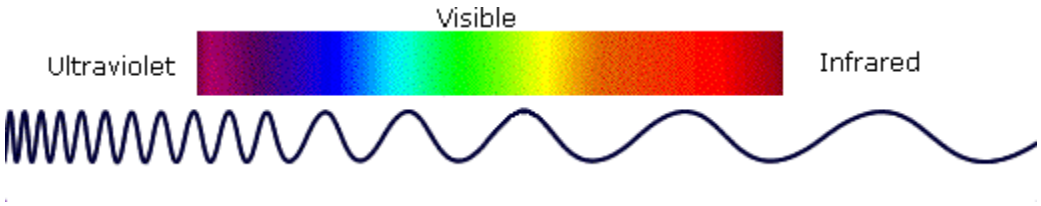

Fig 3.20 Ultraviolet Rays

Infrared imaging is used extensively for both military and civilian purposes. Military applications include target acquisition, surveillance, night vision, homing and tracking. Non-military uses include thermal efficiency analysis, remote temperature sensing, short-ranged wireless communication, spectroscopy, and weather forecasting. Infrared astronomy uses sensor-equipped telescopes to penetrate dusty regions of space, such as molecular clouds; detect cool objects such as planets, and to view highly red-shifted objects from the early days of the universe

3.6.1 IR LED QED234:

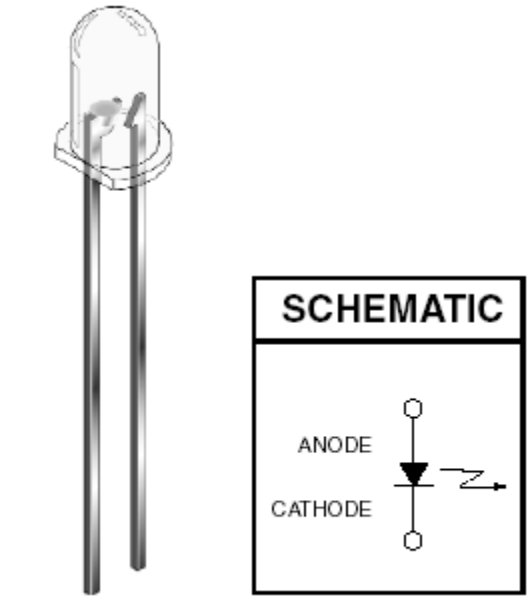

Fig 3.21 Schematic diagram

FEATURES:

• Wave length is 940 nm

• Chip material =GaAs with AlGaAs window

• Package type: T-1 3/4 (5mm lens diameter)

• Matched Photo sensor: QSD122/123/124

• Medium Emission Angle, 40°

• High Output Power

• Package material and color: Clear, untainted, plastic

• Ideal for remote control applications

3.7 Emitter/Detector Alignment:

Good alignment of the emitter and detector is important for good operation, especially if the gap is large. This can be done with a piece of string stretched between and in line with LED and phototransistor. A length of dowel or stiff wire could be used to set the alignment. Another method that can be used for longer distances is a laser pointer shone through one of the mounting holes.

For best results the height of the "beam" should be at coupler height and at an angle across the tracks. The emitter could also be mounted above the track with the phototransistor placed between the rails in locations such as hidden yards. Placing the emitter and detector at an angle would again be helpful.

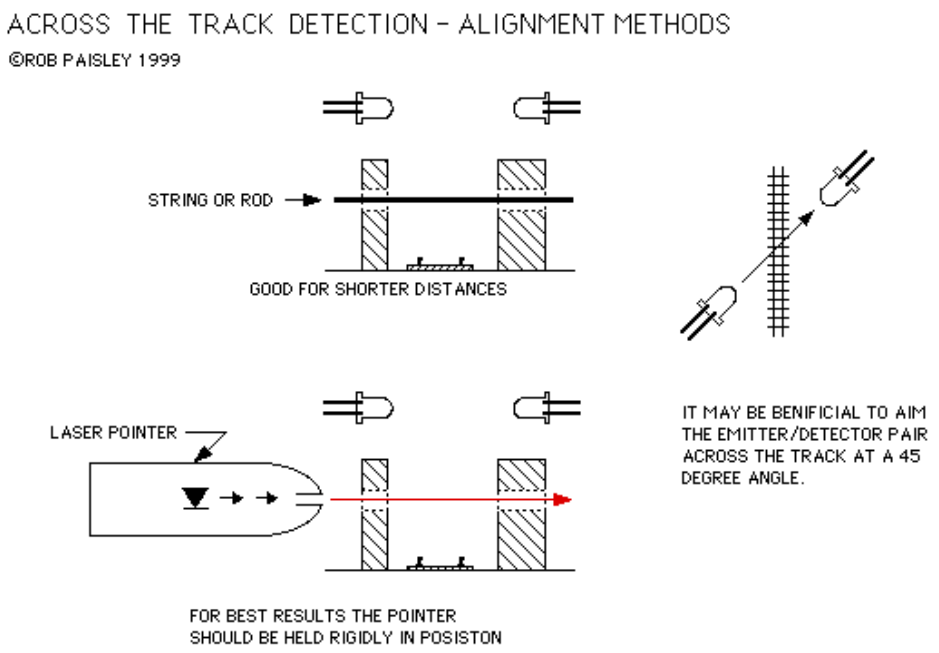

Fig 3.22 The Track Detection Aligment methods

3.7.1 Emitter/Detector Alignment Methods:

A sample infrared remote control setup:

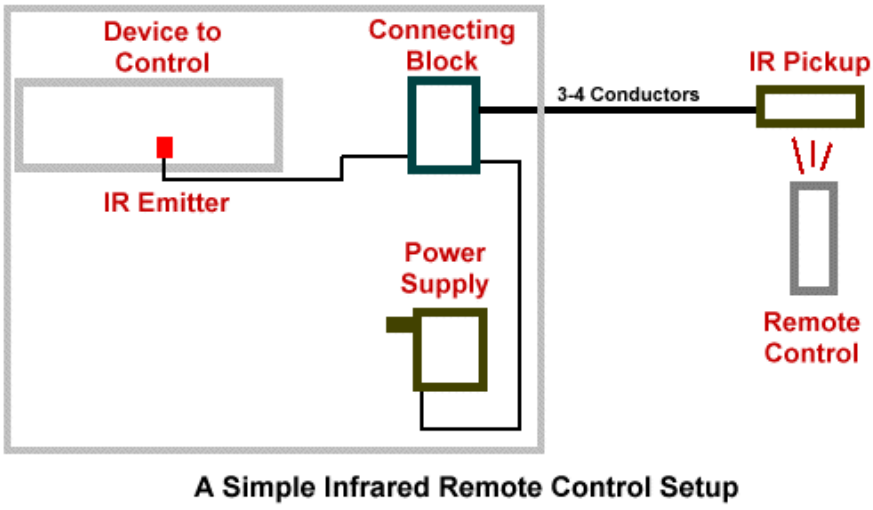

Fig 3.23 Infrared Remote Control Setup

3.7.2 Infrared Receiver (Pickup):

This device picks up the infrared signal from your remote control just like a TV or VCR. It encodes the infrared signal into a signal suitable for transmission. Receivers must be located in the room you wish to use the remote control. The wire from the receiver to the connecting block needs at least three available conductors and can be several hundred feet long. Both quad wire and category 5wire work fine. See our IR receivers here.

3.7.3 Connecting Block:

This is simply a place for all the parts to plug in or connect to. Connecting blocks are usually classified based on the number of outputs (how many IR emitters can connect to the block) Amplified connecting blocks can generally support more outputs. All connecting blocks can support many IR receivers wired in parallel. Connecting blocks are usually located near the equipment that is to be controlled, along with the power supply and emitters. See our connecting blocks here.

3.7.4 Infrared Emitters:

IR Emitters generally "stick" onto the front of the device you want to control. Therefore you need one emitter for each device. "Dual" emitters have two emitters and one plug, so they only take up one jack of the connecting block. "Blink" emitters blink visibly as well as infrared, so they are easier to troubleshoot. All emitters come with long cords and extra double-stick tape. "Blast" style emitters, where one emitter blinks into several devices, are usually less reliable but can be used when the environment is tightly controlled and

3.7.5 Applications:

- Proximity Detection
- Motion Detection
- Temperature Measurement

- Remote Controls
- Obstacle Detection
- Line Following Robots
- Item Counters
- Environmental Monitoring

3.8 ESP8266 Wi-Fi Module:

The NodeMCU ESP8266 is a versatile Wi-Fi and Bluetooth module widely used in loT (Internet of Things) projects. It's based on the ESP8266 microcontroller, offering more power and capabilities compared to its predecessor, the ESP8266. With built-in Wi-Fi and Bluetooth capabilities, it allows for easy connectivity and communication with other devices. It's programmable using the Arduino IDE and supports a variety of sensors and peripherals, making it ideal for creating smart home devices, sensor networks, and other loT applications. NodeMCU is an open-source firmware and development board based on the ESP8266 Wi-Fi module, which is a low-cost, low-power microcontroller with built-in Wi-Fi capabilities. NodeMCU is specifically designed for Internet of Things (IoT) applications and provides an easy-to-use platform for developing loT projects quickly and efficiently, NodeMCU firmware is written in Lua scripting language, making it accessible to those who are familiar with scripting languages such as JavaScript or Python. Lua is a lightweight and powerful scripting language that is well-suited for embedded systems, and it allows developers to write code forcontrolling sensors, actuators, and other peripherals connected to the NodeMCU board.NodeMCU development board features a variety of input/output (I/O) pins, including digital pins for reading and writing digital signals, analog pins for readinganalog signals, and communication pins for interfacing with other devices using protocols such as 12C, SPI, and UART. These I/O pins provide flexibility for connecting sensors, actuators, displays, and other components to the NodeMCU board. One of the main advantages of NodeMCU is its built-in Wi-Fi capabilities, which allow it to connect to the internet and communicate with other devices over a Wi-Fi network. This makes it ideal for applications such as home automation, smart agriculture, industrial monitoring, and remote sensing, where internet

3.8.1 NodeMcu Specifications & Features:

- Microcontroller: ESP32 dual-core Tensilica LX6 microcontroller
- Operating Voltage: 3.3V
- WIFI module: ESP32-S
- Input Voltage: 7-12V
- Wi-Fi version IEEE standard: 802.11b/g/n
- Frequency: 2.4 GHz
- Data Interfaces: UART/12C/SPI/DAC/ADC
- GPIO Pins: 36
- Flash Memory: 4 MB
- SRAM: 512 KB
- Clock Frequency: 240 MHz
- PCB Antenna: Onboard PCB Antenna

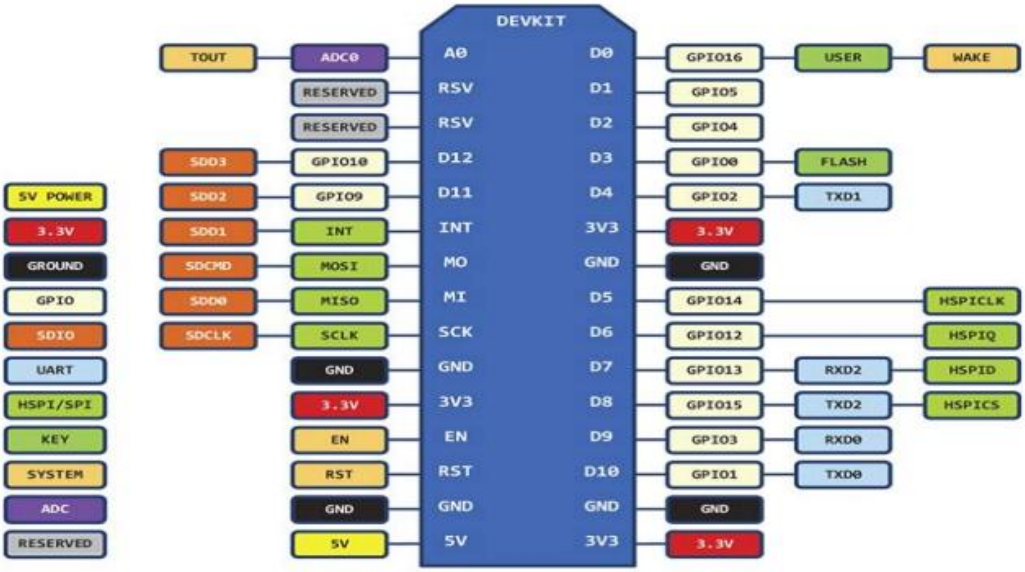

Fig 3.25: NodeMcu Pin Specification

3.8.2 Application of ESP32:

- JoT devices
- Home Automation
- Robotics
- Projects requiring multiple I/O interfaces with Wi-Fi and Bluetooth functionalities
- Audio and Music Applications
- Wearables

3.9 THING SPEAK:

ThingSpeak is an Internet of Things (IoT) platform that allows you to collect, analyze, and visualize data from various sensors or devices. It provides APIs for data logging. real-time data processing, and integration with other loT platforms. ThingSpeak is often used for applications like environmental monitoring, home automation, and industrial control systems.

3.9.1 WorkFlow of ThingSpeak:

- Sending data to ThingSpeak using HTTP or MQTT protocols from iot devices or sensors.
- ThingSpeak stores the data in the specified channel and field.
- You can then use ThingSpeak Apps to visualize, analyze, and act on the data in real-time or historically.

ThingSpeak is used for a variety of IOT applications:

1. Environmental Monitoring

2. Home Automation

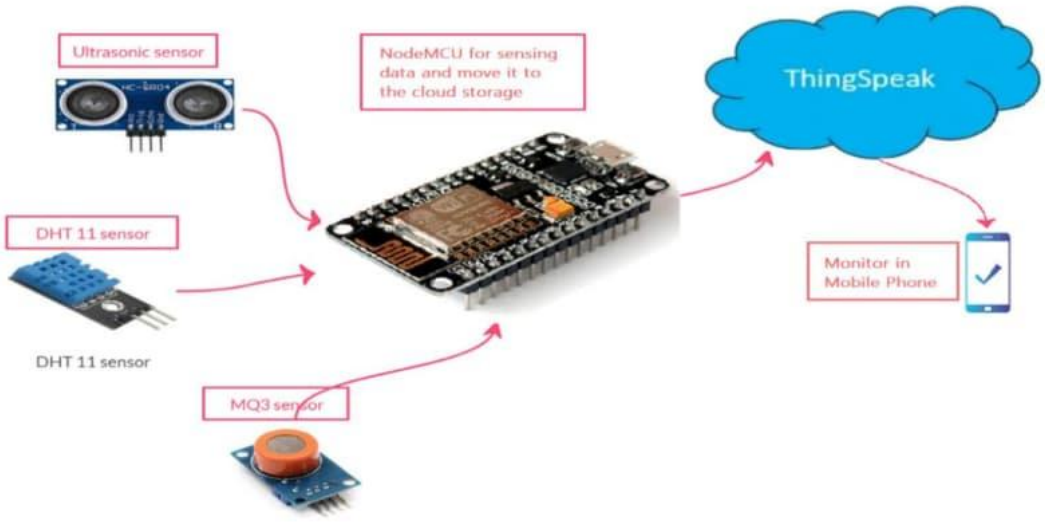

Fig 3.26: ThingSpeak

CHAPTER – 4

SOFTWARES

4.1 Introduction to Arduino IDE:

Arduino is a prototype platform (open-source) based on an easy-to-use hardware and software. It consists of a circuit board, which can be programmed (referred to as a microcontroller) and a ready-made software called Arduino IDE (Integrated Development Environment), which is used to write and upload the computer code to the physical board.

4.1.1 The key features are:

- Arduino boards are able to read analog or digital input signals from different sensors and turn it into an output such as activating a motor, turning LED on/off, connect to the cloud and many other actions.
- You can control your board functions by sending a set of instructions to the microcontroller on the board via Arduino IDE (referred to as uploading software).
- Unlike most previous programmable circuit boards, Arduino does not need an extra piece of hardware (called a programmer) in order to load a new code onto the board. You can simply use a USB cable.
- Additionally, the Arduino IDE uses a simplified version of C++, making it easier to learn to program.
- Finally, Arduino provides a standard form factor that breaks the functions of the micro-controller into a more accessible package.

After learning about the main parts of the Arduino UNO board, we are ready to learn how to set up the Arduino IDE. Once we learn this, we will be ready to upload our program on the Arduino board.

4.1.2 Arduino data types:

- Data types in C refers to an extensive system used for declaring variables or functions of different types.
- The type of a variable determines how much space it occupies in the storage and how the bit pattern stored is interpreted.
- The following table provides all the data types that you will use during Arduino programming.

- **Void:**

  The void keyword is used only in function declarations. It indicates that the function is expected to return no information to the function from which it was called.

  Example:

  ```
  Void Loop ()
  {
  // rest of the code
  }
  ```

- **Boolean:**

  A Boolean holds one of two values, true or false. Each Boolean variable occupies one byte of memory.

  Example:

  Boolean state= false; // declaration of variable with type Boolean and initialize it with false.

  Boolean state = true; // declaration of variable with type Boolean and initialize it with false.

- **Char:**

  A data type that takes up one byte of memory that stores a character value. Character literals are written in single quotes like this: 'A' and for multiple characters, strings use double quotes: "ABC". However, characters are stored as numbers. You can see the specific encoding in the ASCII chart.

This means that it is possible to do arithmetic operations on characters, in which the ASCII value of the character is used.

For example, 'A' + 1 has the value 66, since the ASCII value of the capital letter A is 65.

Example:

- Char chr_a = ‘a’ ;//declaration of variable with type char and initialize it with character a.
- Char chr_c = 97 ;//declaration of variable with type char and initialize it with character 97

- **Unsigned char:**

Unsigned char is an unsigned data type that occupies one byte of memory. The unsigned char data type encodes numbers from 0 to 255.

Example:

Unsigned Char chr_y = 121 ; // declaration of variable with type Unsigned char and initialize it with character y.

- **Byte:**

A byte stores an 8-bit unsigned number, from 0 to 255.

Example:

byte m = 25 ;//declaration of variable with type byte and initialize it with 25

- **Int:**

Integers are the primary data-type for number storage. **int** stores a 16-bit (2-byte) value. This yields a range of -32,768 to 32,767 (minimum value of -2^15 and a maximum value of (2^15) - 1).

The **int** size varies from board to board. On the Arduino Due, for example, an **int** stores a 32-bit (4-byte) value. This yields a range of -2,147,483,648 to 2,147,483,647 (minimum value of -2^31 and a maximum value of (2^31) - 1).

Example:

int counter = 32 ;// declaration of variable with type int and initialize it with 32.

- **Unsigned Int:**

  Unsigned ints (unsigned integers) are the same as int in the way that they store a 2 byte value. Instead of storing negative numbers, however, they only store positive values, yielding a useful range of 0 to 65,535 (2^16) - 1). The Due stores a 4 byte (32-bit) value, ranging from 0 to 4,294,967,295 (2^32 - 1).

  Example:

  Unsigned int counter = 60 ; // declaration of variable with type unsigned int and initialize it with 60.

- **Word:**

  On the Uno and other ATMEGA based boards, a word stores a 16-bit unsigned number. On the Due and Zero, it stores a 32-bit unsigned number.

  Example:

  word w = 1000 ;//declaration of variable with type word and initialize it with 1000.

- **Long:**

  Long variables are extended size variables for number storage, and store 32 bits (4 bytes), from 2,147,483,648 to 2,147,483,647.

  Example:

  Long velocity= 102346 ;//declaration of variable with type Long and initialize it with 102346

- **Unsigned Long:**

  Unsigned long variables are extended size variables for number storage and store 32 bits (4 bytes). Unlike standard longs, unsigned longs will not store negative numbers, making their range from 0 to 4,294,967,295 (2^32 - 1).

Example:

Unsigned Long velocity = 101006 ;// declaration of variable with type Unsigned Long and initialize it with 101006.

- **Short:**

A short is a 16-bit data-type. On all Arduinos (ATMega and ARM based), a short store a 16-bit (2-byte) value. This yields a range of -32,768 to 32,767 (minimum value of -2^15 and a maximum value of (2^15) - 1).

Example:

short val= 13 ;//declaration of variable with type short and initialize it with 13

- **Float:**

Data type for floating-point number is a number that has a decimal point. Floating-point numbers are often used to approximate the analog and continuous values because they have greater resolution than integers.

Floating-point numbers can be as large as 3.4028235E+38 and as low as 3.4028235E+38. They are stored as 32 bits (4 bytes) of information.

Example:

float num = 1.352;//declaration of variable with type float and initialize it with 1.352.

- **Double:**

On the Uno and other ATMEGA based boards, Double precision floating-point number occupies four bytes. That is, the double implementation is exactly the same as the float, with no gain in precision. On the Arduino Due, doubles have 8-byte (64 bit) precision.

Example:

double num = 45.352 ;// declaration of variable with type double and initialize it with 45.352.

4.1.3 Arduino IDE and Board Setup:

In this section, we will learn in easy steps, how to set up the Arduino IDE on our computer and prepare the board to receive the program via USB cable.

- **Step 1: Board Selection & its Cable:**

  First you must have your Arduino board (you can choose your favorite board) and a USB cable. In case you use Arduino UNO, Arduino Duemilanove, Nano, Arduino Mega2560, or Diecimila, you will need a standard USB cable (A plug to B plug), the kind you would connect to a USB printer as shown in the following image.

Fig 4.1: USB Cable

- **Step 2: Download Arduino IDE Software**

  You can get different versions of Arduino IDE from the Download page on the Arduino Official website. You must select your software, which is compatible with your operating system (Windows, IOS, or Linux). After your file download is complete, unzip the file.

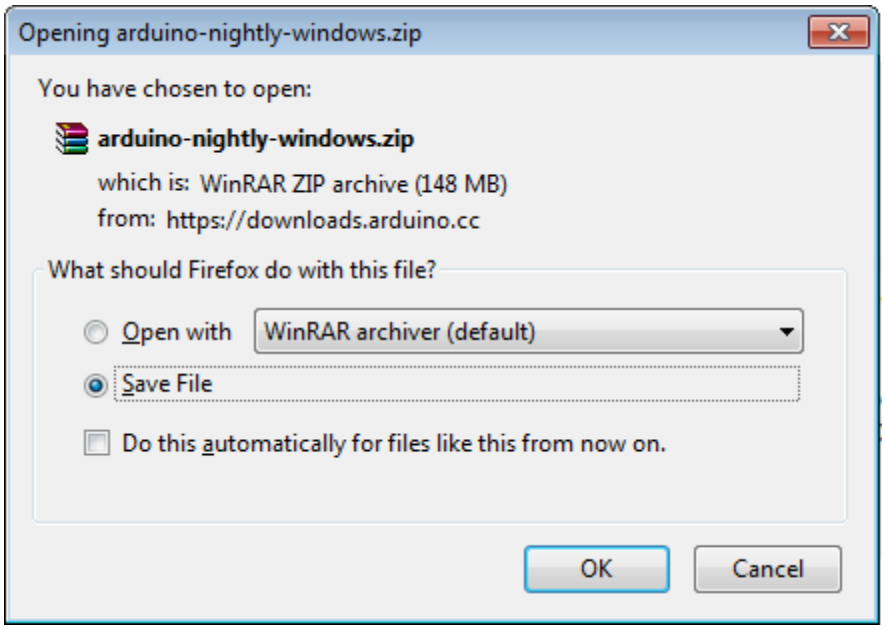

Fig 4.2: Downloading Arduino UNO Software

- **Step 3: Power up your board**

The Arduino Uno, Mega, Duemilanove and Arduino Nano automatically draw power from either, the USB connection to the computer or an external power supply. If you are using an Arduino Diecimila, you have to make sure that the board is configured to draw power from the USB connection. The power source is selected with a jumper, a small piece of plastic that fits onto two of the three pins between the USB and power jacks. Check that it is on the two pins closest to the USB port. Connect the Arduino board to your computer using the USB cable. The green power LED (labeled PWR) should glow.

- **Step 4: Launch Arduino IDE.**

After your Arduino IDE software is downloaded, you need to unzip the folder. Inside the folder, you can find the application icon with an infinity label (application.exe). Double click the icon to start the IDE.

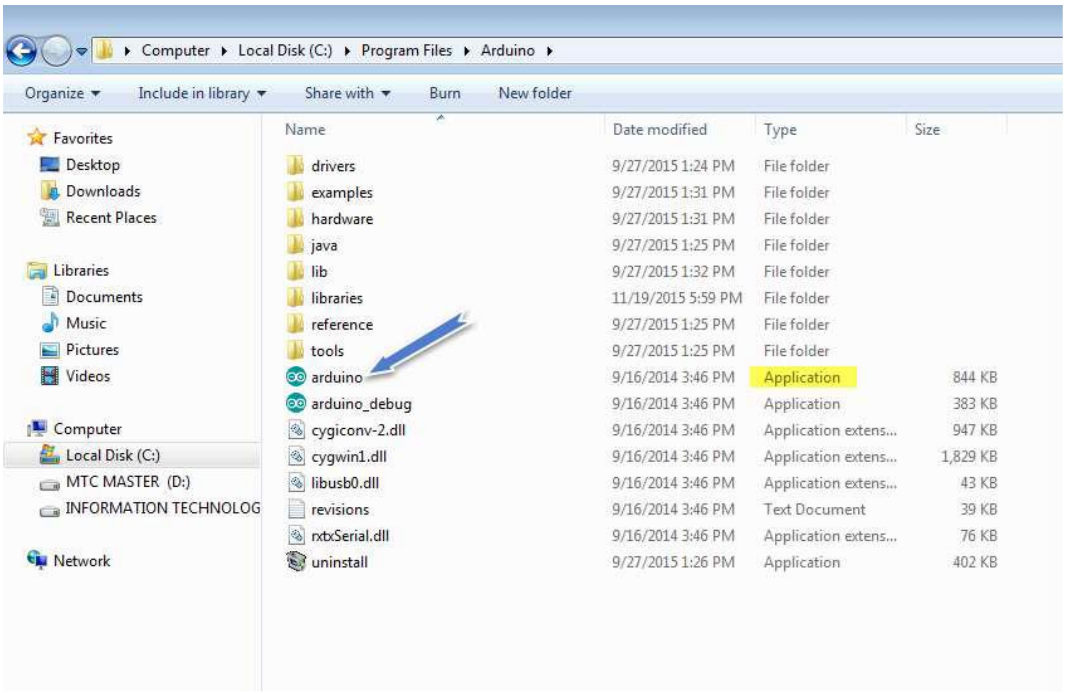

Fig 4.3: Launching Arduino IDE

- **Step 5: Open your first project.**

  Once the software starts, you have two options:

  - Create a new project.
  - Open an existing project example.

  To create a new project, select File --> New. To open

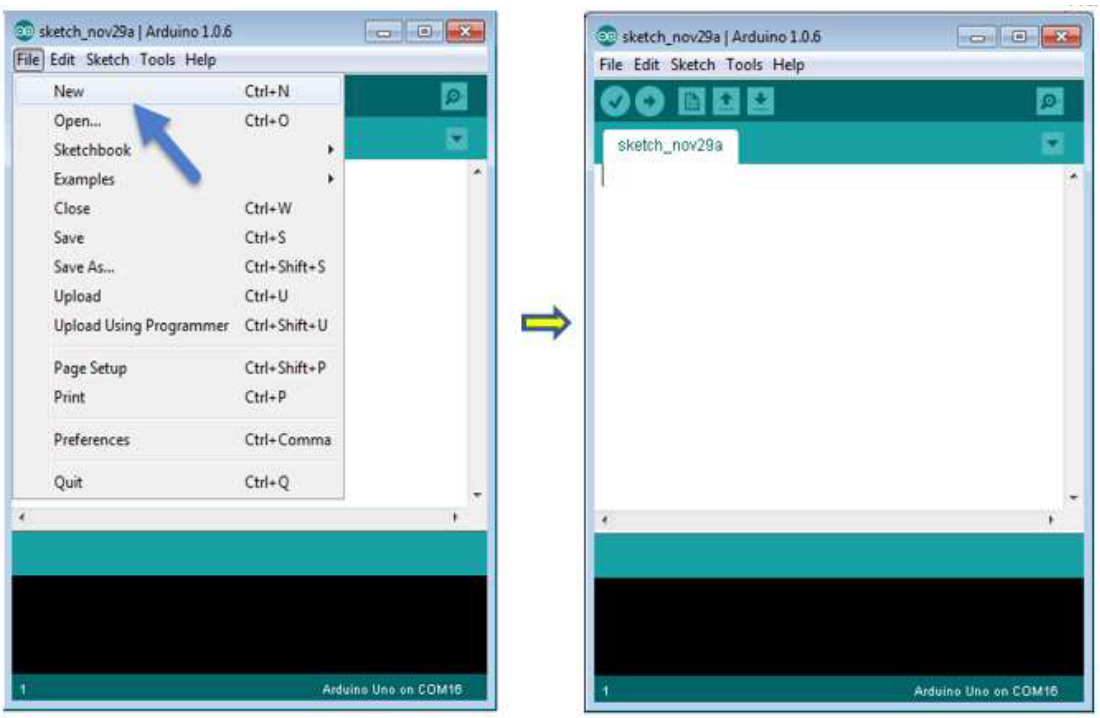

Fig 4.4: Opening New File

To open an existing project example, select File -> Example -> Basics -> Blink.

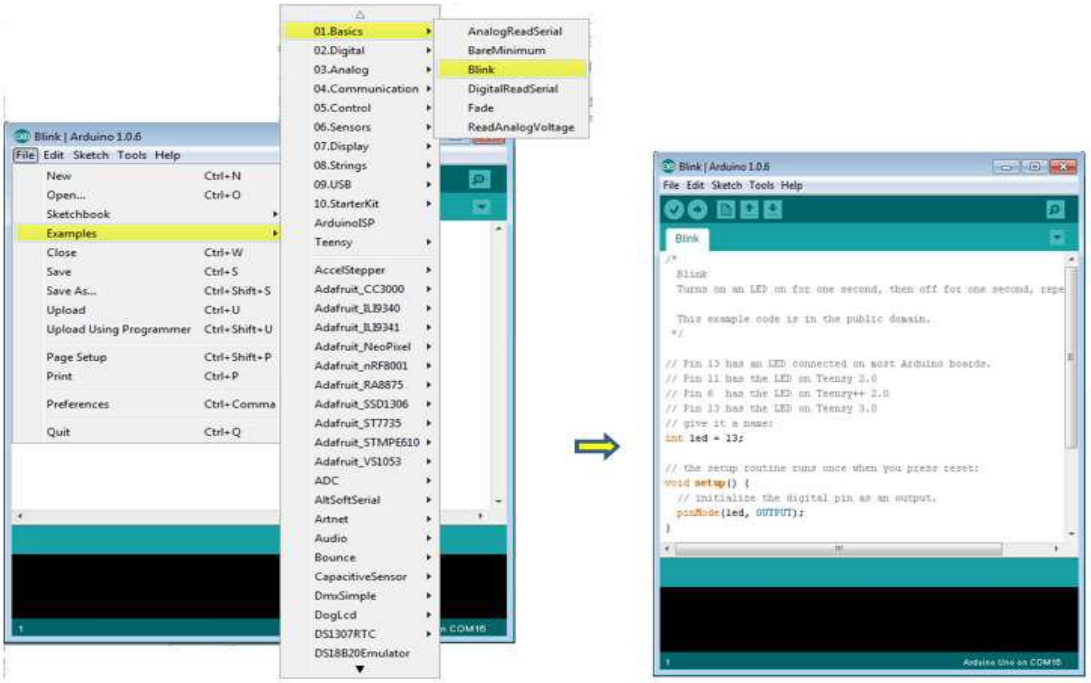

Fig 4.5: Opening Existing File

Here, we are selecting just one of the examples with the name **Blink**. It turns the LED on and off with some time delay. You can select any other example from the list.

- **Step 6: Select your Arduino board.**

To avoid any error while uploading your program to the board, you must select the correct Arduino board name, which matches with the board connected to your computer.

Go to Tools -> Board and select your board

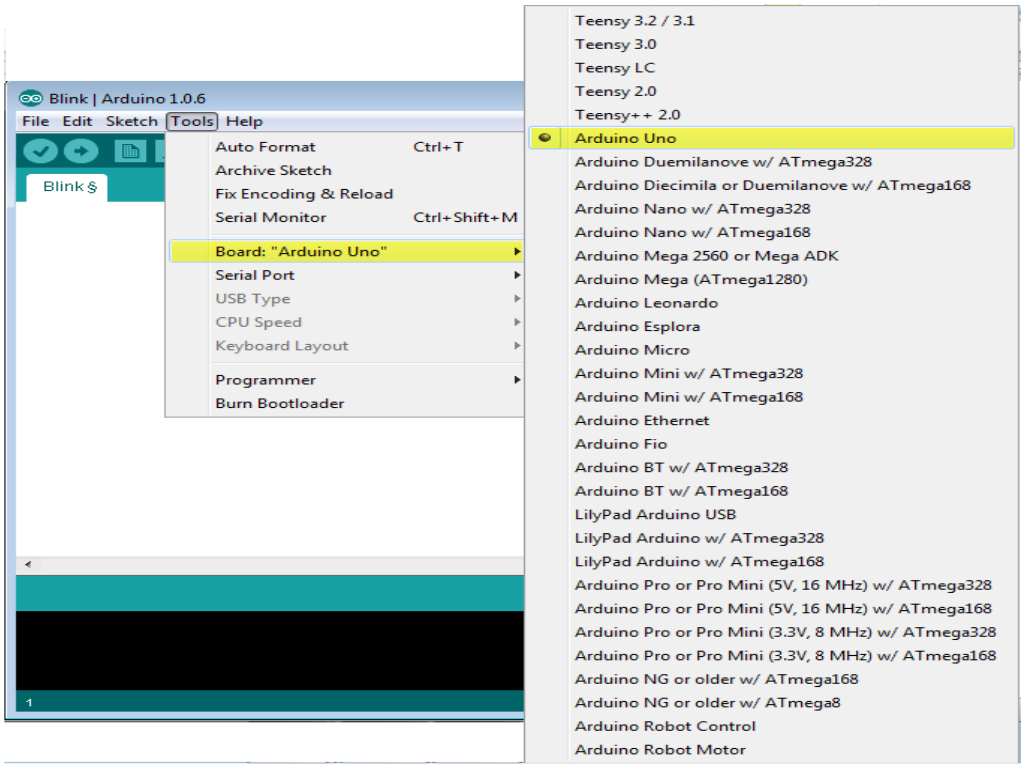

Fig 4.6: Selecting Arduino Board

Here, we have selected Arduino Uno board according to our tutorial, but you must select the name matching the board that you are using

- **Step 7: Select your Serial board**

Select the serial device of the Arduino board. Go to **Tools** ->**Serial Port** menu. This is likely to be COM3 or higher (COM1 and COM2 are usually reserved for hardware serial ports). To find out, you can disconnect your Arduino board and re-open the menu, the entry that disappears should be of the Arduino board. Reconnect the board and select that serial port.

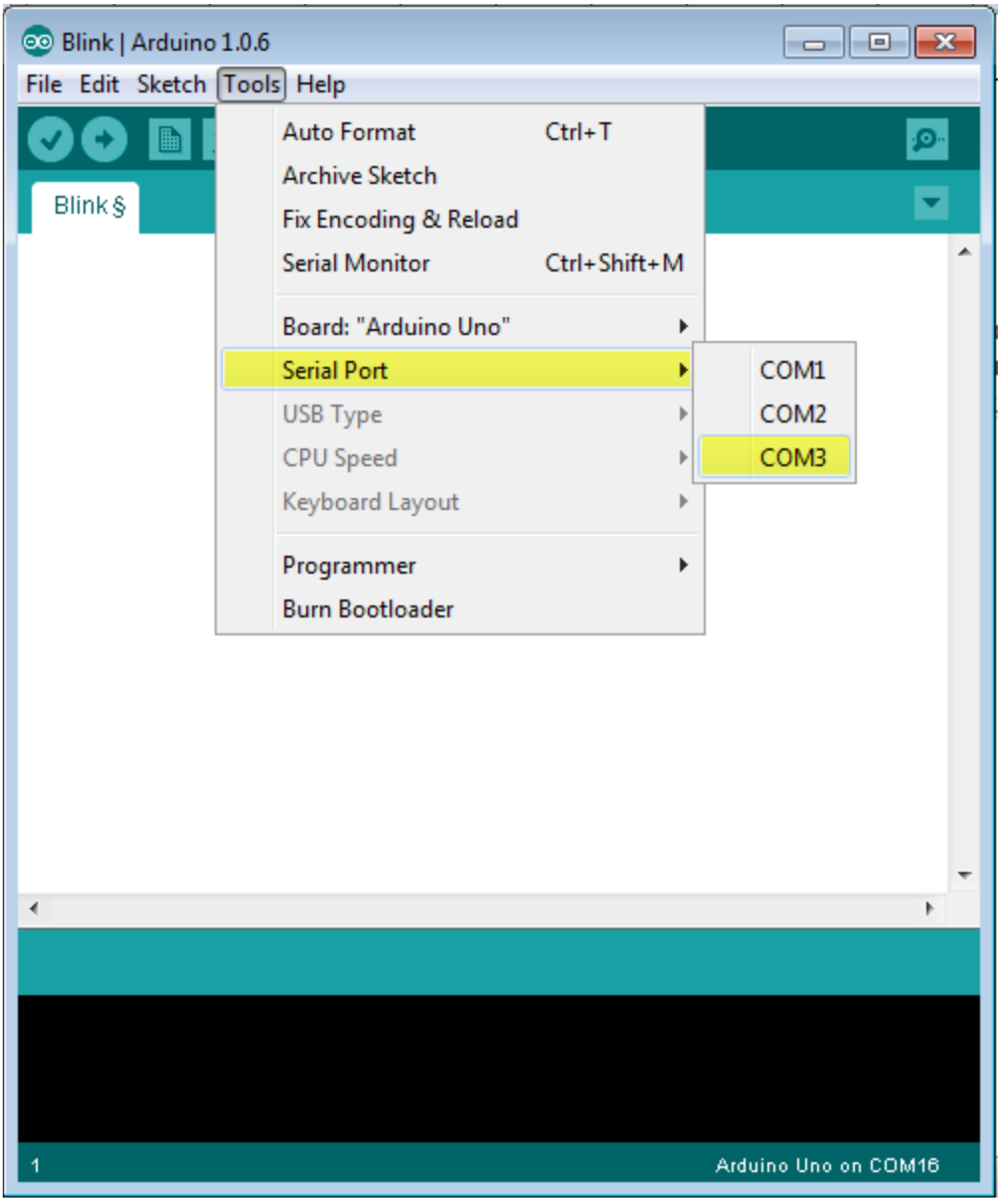

Fig 4.7: Selecting Serial Port

- **Step 8: Upload the program to your board.**

Before explaining how we can upload our program to the board, we must demonstrate the function of each symbol appearing in the Arduino IDE toolbar.

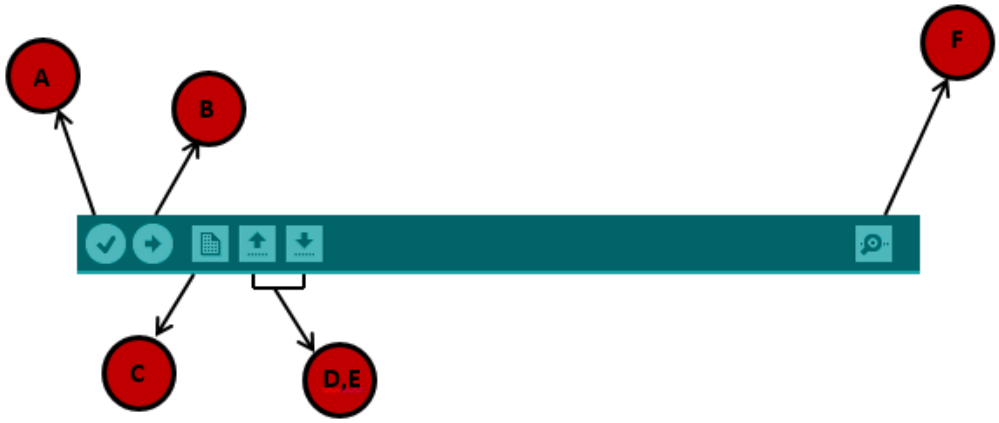

Fig 4.8: Uploading Program to the Board

A. Used to check if there is any compilation error.

B. Used to upload a program to the Arduino board.

C. Shortcut used to create a new sketch.

D. Used to directly open one of the example sketches.

E. Used to save your sketch.

F. Serial monitor used to receive serial data from the board and send the serial data to the board.

Now, simply click the "Upload" button in the environment. Wait a few seconds; you will see the RX and TX LEDs on the board, flashing. If the upload is successful, the message "Done uploading" will appear in the status bar.

Note: If you have an Arduino Mini, NG, or other board, you need to press the reset button physically on the board, immediately before clicking the upload button on the Arduino Software.

4.1.4 Arduino Programming Structure:

In this chapter, we will study in depth, the Arduino program structure and we will learn more new terminologies used in the Arduino world. The Arduino software is open-source. The source code for the Java environment is released under the GPL and the C/C++ microcontroller libraries are under the LGPL.

Sketch: The first new terminology is the Arduino program called "**sketch**".

Structure:

Arduino programs can be divided in three main parts: **Structure**, **Values** (variables and constants), and **Functions**. In this tutorial, we will learn about the Arduino software program, step by step, and how we can write the program without any syntax or compilation error.

Let us start with the **Structure**. Software structure consist of two main functions:

- Setup( ) function
- Loop( ) function

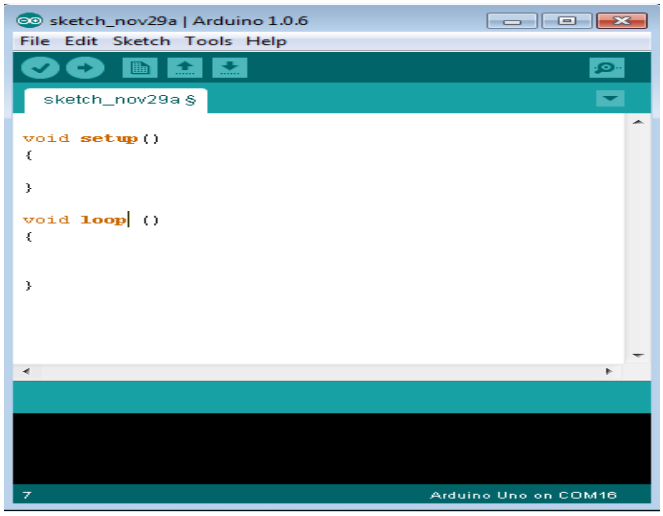

Fig 4.9: Sketch

Void setup ( )

{

}

- **PURPOSE:**

The **setup()** function is called when a sketch starts. Use it to initialize the variables, pin modes, start using libraries, etc. The setup function will only run once, after each power up or reset of the Arduino board.:

  - INPUT
  - OUTPUT
  - RETURN

Void loop() { }:

- **PURPOSE:**

After creating a **setup()** function, which initializes and sets the initial values, the **loop()** function does precisely what its name suggests, and loops secutively, allowing your program to change and respond. Use it to actively control the Arduino board.

  - INPUT
  - OUTPUT
  - RETURN

CHAPTER – 5

SOURCE CODE

```
#include <SoftwareSerial.h>
#include <LiquidCrystal.h>
// LCD Configuration
const int rs = 13, en = 12, d4 = 11, d5 = 10, d6 = 9, d7 = 8;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);
// IR Sensor and Buzzer
const int irSensor = A1;
const int buzzer = A4;
void setup() {
   Serial.begin(9600);
   Serial.println("Automatic Bed Vacancy System");
   lcd.begin(16, 2);
   lcd.clear();
   lcd.print("Automatic Bed");
   lcd.setCursor(0, 1);
   lcd.print("Vacancy Detect");
   delay(3000);
   pinMode(irSensor, INPUT);
   pinMode(buzzer, OUTPUT);
```

```
    digitalWrite(buzzer, LOW);
}
void loop() {
    int irValue = analogRead(irSensor);
    if (irValue > 300) {  // If bed is occupied
        lcd.clear();
        lcd.print("NO Bed Vacant");
        Serial.println("NO Bed is Vacant");
        digitalWrite(buzzer, LOW);
    } else {  // If bed is vacant
        lcd.clear();
        lcd.print("Bed is Vacant");
        Serial.println("Bed is Vacant");
        digitalWrite(buzzer, HIGH);
        delay(1000);
    }
    delay(1000);  // Delay to prevent flickering
}
```

Wi-Fi Code

```
#include <WiFiClientSecure.h>   // Include the HTTPS library

#include <ESP8266WiFi.h>        // Include the Wi-Fi library

#include <ESP8266WiFiMulti.h>   // Include the Wi-Fi-Multi library

#include "Arduino.h"

//#include <EMailSender.h>

ESP8266WiFiMulti wifiMulti;     // Create an instance of the ESP8266WiFiMulti class, called 'wifiMulti'

uint8_t connection_state = 0;

uint16_t reconnect_interval = 10000;

WiFiClient  client;

String data1="";

String data="9014475682";

/*EMailSender emailSend("siva9014475682@gmail.com", "SIVAPRASADSIVPRASAD");

void gmail()

{

  EMailSender::EMailMessage message;


    message.subject = "hai Sivaji"+data;

    message.message = "This is Mail From ESP8266"+data;


    EMailSender::Response resp = emailSend.send("siva9014475682@gmail.com", message);


    Serial.println("Sending status: ");


    Serial.println(resp.status);

    Serial.println(resp.code);
```

```
   Serial.println(resp.desc);
}*/
void upload()
{
const char* server4 = "api.thingspeak.com";
const char* _getLink4 =
"https://api.thingspeak.com/update?api_key=IH4ILE76C1BFEUMG&field1="; //
Thingspeak.com



 // Serial.println("data uploading");delay(1000);
  client.connect(server4,80);
 if (client.connect(server4,80))     // "184.106.153.149" or api.thingspeak.com
https://api.thingspeak.com/apps/thinghttp/send_request?api_key=CT9B331KB5PLM1G5
  {
    String getStr4 = _getLink4;
    client.print("GET "+getStr4+data1+"\n");
    client.print("HTTP/1.1\n");
    client.print("Host: api.thingspeak.com\n");
    client.print("Connection: close\n\n\n");
  }
  client.stop();


}


void readdata()
{
const char* server4 = "api.thingspeak.com";
const char* _getLink4 = " https://api.thingspeak.com/channels/2870365/fields/1/last.txt";
// Thingspeak.com
```

```
  //Serial.println("data uploading");delay(1000);
  client.connect(server4,80);
 if (client.connect(server4,80))     // "184.106.153.149" or api.thingspeak.com
https://api.thingspeak.com/apps/thinghttp/send_request?api_key=CT9B331KB5PLM1G5
  {
    String getStr4 = _getLink4;
    client.print("GET "+getStr4+"\n");
    client.print("HTTP/1.1\n");
    client.print("Host: api.thingspeak.com\n");
    client.print("Connection: close\n\n\n");
    client.available();
    data1=client.readString();delay(1000);
   Serial.println(data1);delay(1000);
   /* if((data1=="1")||(data1=="2")||(data1=="3")||(data1=="4")||(data1=="0"))
    {
   Serial.print(data1);delay(10000);
      }
     */
  }
  client.stop();
}
void setup()
{
  Serial.begin(9600);         // Start the Serial communication to send messages to the
computer
  delay(10);
  //Serial.println('\n');
```

```
 // wifiMulti.addAP("Airtel_9177121789", "air72656");   // add Wi-Fi networks you want to connect to

 //wifiMulti.addAP("manideepdrk", "manideepdrk123");

 //wifiMulti.addAP("yogeshanm", "123456789");

  wifiMulti.addAP("indhu408","123456789");

  wifiMulti.addAP("123456789", "123456789");

  wifiMulti.addAP("bed vacancy", "123456789");

  //Serial.println("Connecting ...");

  int i = 0;

  while (wifiMulti.run() != WL_CONNECTED) { // Wait for the Wi-Fi to connect: scan for Wi-Fi networks, and connect to the strongest of the networks above

    delay(250);

    Serial.print('.');

  }

  Serial.println('\n');

  Serial.print("Connected to ");

  Serial.println(WiFi.SSID());              // Tell us what network we're connected to

  Serial.print("IP address:\t");

 Serial.println(WiFi.localIP());           // Send the IP address of the ESP8266 to the computer

  Serial.println('\n');


  //readdata();

}


void loop()

{
```

```
readdata();
 while(Serial.available())
{
data1=Serial.readString();
upload();
}

/*readdata();
while(serial.available())
{
  data1=serial.readstring();
  upload();
}*/



}
```

CHAPTER – 6

ADVANTAGES & FUTURE SCOPE

6.1 Advantages:

- **Improved Efficiency**: Reduces manual effort in tracking bed availability, saving time for hospital staff and improving operations.
- **Enhanced Patient Experience**: Minimizes waiting times by providing real-time updates on bed availability, ensuring faster admissions.
- **Better Resource Utilization**: Helps optimize bed usage and plan for future needs by providing accurate data and analytics.
- **Emergency Management**: Facilitates quicker response in critical situations by identifying available beds immediately.
- **Cost Savings**: Reduces overhead costs associated with manual tracking and mismanagement of resources.
- **Scalability and Integration**

6.2 Future Scope:

- **Integration with IoT and AI**: Enhanced real-time monitoring and predictive analytics can help anticipate bed availability and patient needs2.
- **Improved Patient Care**: Systems could integrate with medical devices to monitor health data and provide alerts for emergencies.
- **Scalability**: Expansion to rural areas where healthcare infrastructure is limited, ensuring better resource allocation.
- **Smart Bed Technology**: Incorporating features like sensors for vital signs, automated adjustments, and wireless data transmission.
- **Operational Efficiency**: Streamlining hospital administration and reducing wait times for patients.

CHAPTER – 7

RESULT AND CONCLUSION

7.1 Result:

7.1.1 OFF State:

The below figure indicates the result of the system when it's in off state which means there is no power supplied. It contains Arduino UNO, Power Supply, Buzzer, Transformer, DHT-11 Sensor, Arduino base board, LCD display, IR Sensor, External power supply board.

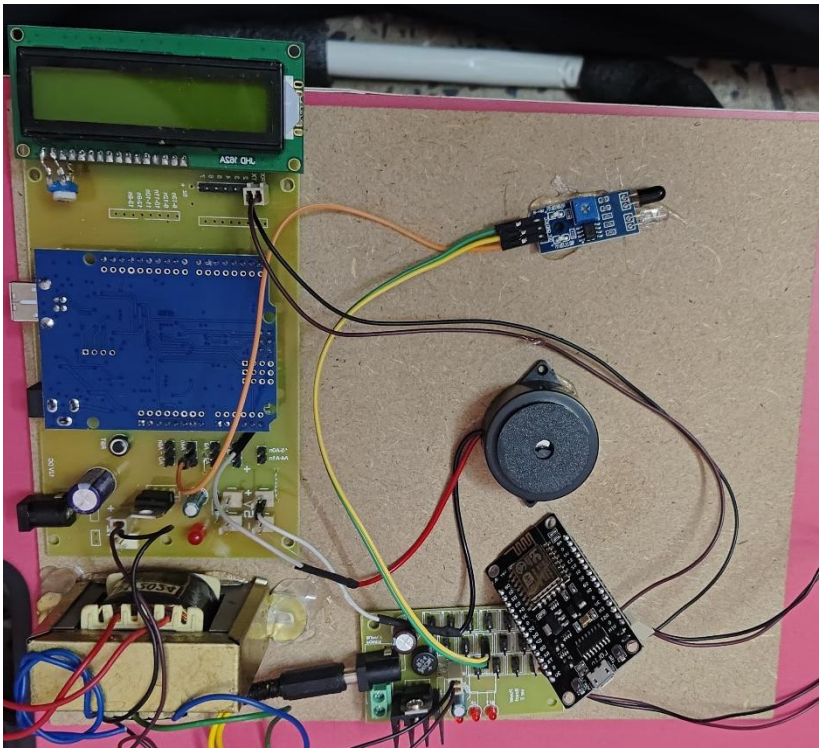

Fig 7.1: OFF State

7.1.2 ON State:

The below figure indicates the result of the system when it's in **ON State** means power is supplied to the Arduino board. It contains Arduino UNO, Power Supply, Buzzer, Transformer, Arduino base board, LCD display, IR Sensor, External power supply board.

The board makes the buzzer sound when its turned ON and the LCD displays the title of the project which is Bed Vacency

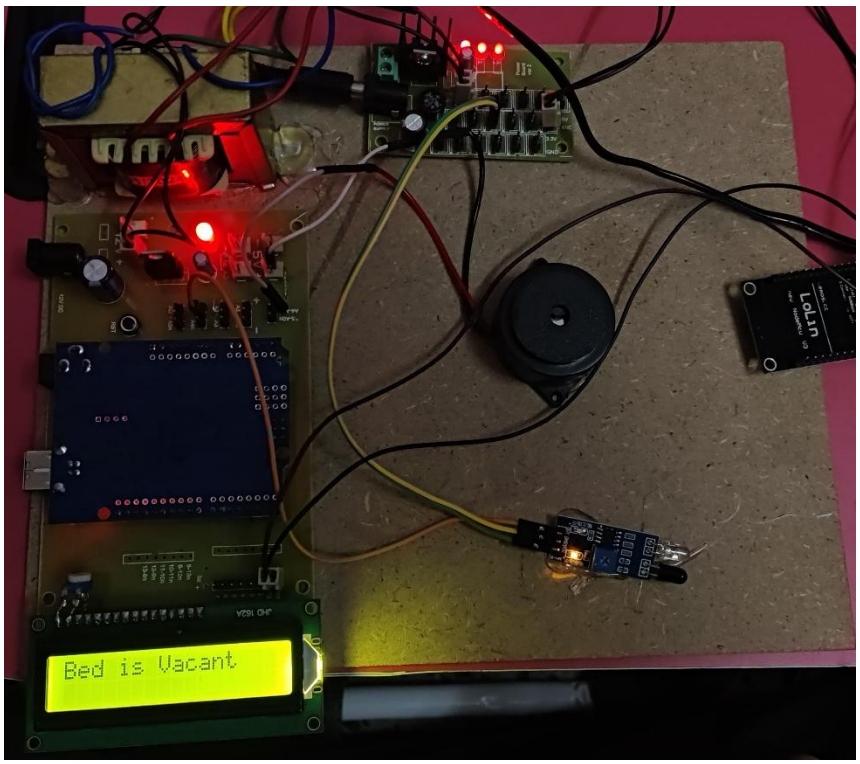

Fig 7.2: ON State 1

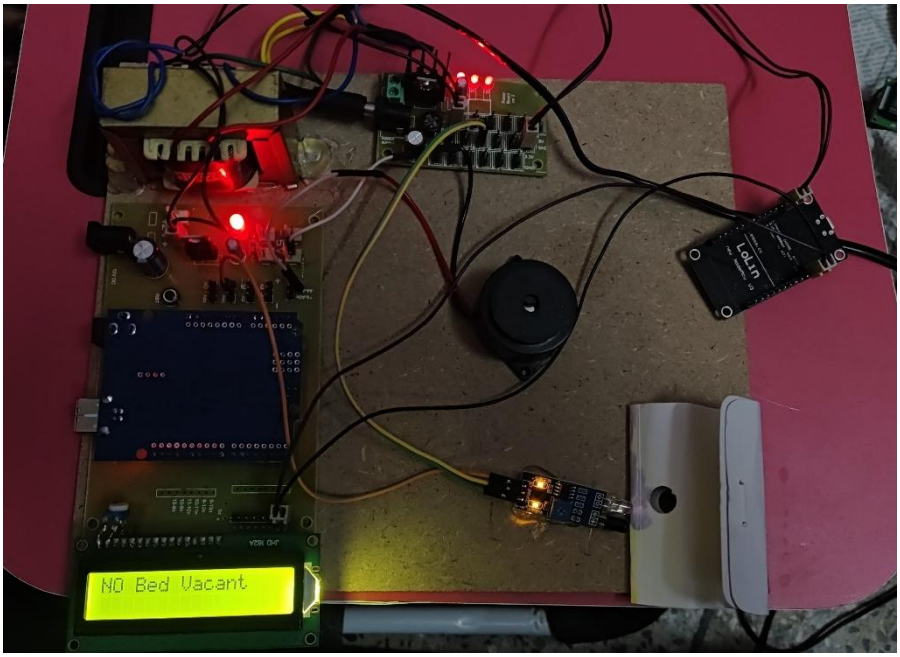

Fig 7.3: ON State 2

CONCLUSION

The IoT-based automatic bed vacancy detection system provides a smart, reliable, and real-time solution for hospital bed management. By integrating IR sensors, IoT technology, and a buzzer alert system, the system eliminates manual bed monitoring, ensuring faster patient admission, reduced workload for hospital staff, and efficient hospital resource utilization.

Unlike conventional hospital bed management methods, this system offers real-time IoT-based monitoring and automated alerts, ensuring that hospitals can respond quickly to bed availability updates. The buzzer system further enhances efficiency by providing instant notifications, allowing staff to take immediate action.

Future enhancements could include AI-based predictive bed allocation, automated hospital resource management, and integration with emergency medical response systems for better healthcare service delivery. This cost-effective, scalable, and easy-to-implement solution ensures that hospitals can manage patient admissions more efficiently, ultimately improving patient care and hospital workflow.

• REFERENCES

- • **[1]** "Smart Emergency Management with IoT for Integration of Emergency Centres and Patients" by G.N. Keshava Murthy, published in the *International Journal of Recent Technology and Engineering (IJRTE)*, Volume-8 Issue-4, November 2019.
- • **[2]** "Automatic Bed Tracking System in Hospitals using IoT" by Arjun (SRM Alumnus), published in the *International Journal of Engineering Research & Technology (IJERT)*, Volume-10 Issue-12, December 2021. This work is licensed under a Creative Commons Attribution 4.0 International License.
- • **[3]** "IoT Based Bus Seats Vacancy Unit for Roadways SMART BUS: SEAT OCCUPANCY DETECTION SYSTEM," published in the *International Journal of Research in Engineering and Science (IJRES)*, Volume-9 Issue-8, 2021, PP. 68-72.
- • **[4]** "IoT Based Automatic Seat Vacancy Detection in Travel Buses using Cloud Database" by M. Venkatesh and R. Rashia Suba Shree, published in the *International Journal of Innovative Technology and Exploring Engineering (IJITEE)*, Volume-10 Issue-8, June 2021.
- • **[5]** "Autonomous Remote Controlled Smart Bed for Hospitals" by Majumder, M., Sinha, P. J., and Vedanarayanan, V.
- • **[6]** "Internet of Things: A Survey of Enabling Technologies in Healthcare and Its Applications" by Mrinai M. Dhanvijay and Shailaja C. Patil, published in *Computer Networks*, Volume 153, 2019, pp. 113–131.
- • **[7]** "Android Application for Ticket Booking and Ticket Checking in Suburban Railways" by Subarnarekha Ghosal, Shalini Chaturvedi, and Akshay Taywade, published in the *Indian Journal of Science and Technology (IJST)*, Volume-8 Supplementary 2, January 2015, DOI: 10.17485/ijst/2015/v8iS2/60291.
- • **[8]** "IoT Based Ticket Checking System" by Kirti Dhiman and Er. C. K. Raina, published in the *International Journal of Advanced Research in Computer and Communication Engineering (IJARCCE)*, Volume-6 Issue-3, March 2017, pp. 916–919.
- • **[9]** "A Survey on Internet of Things Architectures" by P.P. Ray, published in the *Journal of King Saud University – Computer and Information Sciences*, Volume 30, 2018, pp. 291–319.
- • **[10]** "A Survey on IoT Technologies, Evolution and Architecture" by Rj.M. Gomathi, G. Hari Satya Krishnan, E. Brumancia, and Y. Mistica Dhas, presented at the *2nd International Conference on Computer, Communication, and Signal Processing (ICCCSP)*, 2018.
- • **[11]** "Designing and Testing a Laser-Based Vibratory Sensor" by Gnath, published in *IOP Science*, March 2018, Volume 28, No. 4.
- • **[12]** "IoT-Based Laser-Inscribed Sensors for Detection of Sulfate in Water Bodies" by Shan He, Shilun Feng, and colleagues, published in *IEEE Open Access*, Volume 8, December 22, 2020.