

# **IOT-BASED COVID-19 SUSPECT SMART ENTRANCE MONITORING SYTEM**

## **A PROJECT REPORT**

*Submitted in partial fulfillment of the requirements  
for the award of the degree of*

## **BACHELOR OF TECHNOLOGY**

in

## **COMPUTER SCIENCE & ENGINEERING-INTERNET OF THINGS**

SUBMITTED BY

**G. DEEPTHI** – 204N1A3513

**D. SAI MANOJ** – 204N1A3510

**U. SURENDRA KUMAR REDDY** – 204N1A3551

**G. MURALI** – 204N1A3506

Under the Guidance of

***Mr. A . SRINIVASA RAO***

*Assistant Professor,  
Department of CSE*



**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING -  
INTENET OF THINGS**

**VISVODAYA ENGINEERING COLLEGE  
KAVALI- 524 201, NELLORE DISTRICT, AP**

**Jawaharlal Nehru Technological University, Anantapur, AP, India**

**APRIL 2024**

# **IOT-BASED COVID-19 SUSPECT SMART ENTRANCE MONITORING SYTEM**

## **A PROJECT REPORT**

*Submitted in partial fulfillment of the requirements  
for the award of the degree of*

## **BACHELOR OF TECHNOLOGY**

in

## **COMPUTER SCIENCE & ENGINEERING-INTERNET OF THINGS**

**SUBMITTED BY**

**G. DEEPTHI** – 204N1A3513

**D. SAI MANOJ** – 204N1A3510

**U. SURENDRA KUMAR REDDY** – 204N1A3551

**G. MURALI** – 204N1A3506

Under the Guidance of

***Mr. A . SRINIVASA RAO***

*Assistant Professor,  
Department of CSE*



**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING -  
INTENET OF THINGS**

**VISVODAYA ENGINEERING COLLEGE  
KAVALI- 524 201, NELLORE DISTRICT, AP**

**Jawaharlal Nehru Technological University, Anantapur, AP, India**

**APRIL 2024**

**VISVODAYA ENGINEERING COLLEGE**  
**KAVALI**

**BONAFIDE CERTIFICATE**

Certified that this project report “**IOT-BASED COVID-19 SUSPECT SMART ENTRANCE MONITORING SYSTEM**” is the bonafide work done by “**G.DEEPTHI(204N1A3513), D.SAI MANOJ(204N1A3510), U.SURENDRA KUMAR REDDY(204N1A3551), G.MURALI(204N1A3506)**”, who carried out the project under my guidance during the year 2023-24, towards partial fulfillment of the requirements of the Degree of Bachelor of Technology in Computer Science & Engineering- Internet of Things, from Jawaharlal Nehru Technological University, Anantapur. The results embodied in this report have not been submitted to any other University for the award of any degree.

**Mr. A. Srinivasa Rao**  
Assistant Professor  
**PROJECT GUIDE**  
DEPARTMENT OF C.S.E.

**Dr. D.Surjan Chandra Reddy**  
Professor  
**HEAD OF THE DEPARTMENT**  
DEPARTMENT OF C.S.E

External Viva voce conducted on \_\_\_\_\_

**Internal Examiner**

**External Examiner**

**VISVODAYA ENGINEERING COLLEGE**  
**KAVALI**

**CERTIFICATE OF AUTHENTICATION**

We solemnly declare that this project report “**IOT-BASED COVID-19 SUSPECT SMART ENTRANCE MONITORING SYSTEM**” is the bonafide work done purely by us, carried out under the supervision of **Mr.A.SRINIVASA RAO**, towards partial fulfillment of the requirements of the Degree of **BACHELOR OF TECCHNOLOGY in Computer Science & Engineering-Internet of Things** from Jawaharlal Nehru Technological University, Anantapur during the year 2023-24.

It is further certified that this work has not been submitted, either in part of in full, to any other department of the Jawaharlal Nehru Technological University, or any other University, institution or elsewhere, or for publication in any form.

**DATE:**

**Signature of the Students**

1. G. DEEPTHI (204N1A3513)

2. D. SAI MANOJ (204N1A3510)

3. U. SURENDRA KUMAR REDDY  
(204N1A3551)

4. G. MURALI (204N1A3506)

## **ACKNOWLEDGEMENT**

We express my heartfelt gratitude towards institution, **VISVODAYA ENGINEERING COLLEGE, KAVALI**, for giving us an opportunity for the successful completion of our degree.

We express great sense of gratitude and indebtedness to our beloved Chairman **Mr. D. Vidyadhara Kumar Reddy** and Academic In-Charge **Dr. D. Prathyusha Reddy** for promoting excellent academic environment in the course.

It gives us immense pleasure to express a gratitude to our Director **Wg. Cdr. I.P.C. Reddy, (Retd.)** and Principal **Dr. B. Dattatreya Sarma**, who gave me an opportunity in completing this course.

We deeply express profound gratitude and wholehearted thanks to beloved Head of the Department, **Mr. D. Srujan Chandra Reddy**, Head & Associate Professor, Department of Computer Science and Engineering, who provided us with necessary facilities, guidance and endless encouragement, which helped me a lot in completing the project with in time.

We express a great sense of gratitude and indebtedness to our beloved guide **Mr. A . Srinivasa Rao** Associate Professor, for his inspiring guidance and his Encouragement throughout the course and project.

We also express gratitude to lecturers and lab coordinators, non-teaching staff and friends who directly or indirectly helped me in completing the project successfully and providing me with suitable suggestions and guidance throughout.

## **ABSTRACT**

The COVID-19 is very infectious, and it spreads super-fast by contacting other COVID infected people. The idea is to create a system to stop entering in the COVID suspected to our surroundings like corporate companies. The aim of this project is to detect COVID-19 suspect people and stop them from spreading the disease to other people. By monitoring the temperature, pulse and heartbeat of the person. we can suspect the people who are infected with covid-19. It has a Temperature Sensor which measures the body temperature and also has Pulse and heartrate sensor which measures the heart rate and oxygen levels of a person. In case of high body temperature & low oxygen levels we can block the person from entering our place by displaying a message in the LCD. If all the values are in the specified range, it will give access automatically by displaying on monitor ,then the door will be opened by using a servomotor and the device will detect the movement of the person can be entered or not into the place by using an IR Sensor. The most advantage of our system is cost-effective, good performance and easy to implement and maintain.

# TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	<b>Introduction</b>	<b>1</b>
1	1.1 Embedded System	1
	1.1.2 Characteristics Of Embedded System	2
	1.1.3 Overview Of Embedded System Architecture	3
	1.1.3.1 Central processing unit	4
	1.1.3.2 Memory	4
	1.1.3.3 Input Devices	5
	1.1.3.4 Output Devices	5
	1.1.3.5 Communication Interface	5
	1.1.3.6 Application Specific Circuitry	5
	1.2 classification Of Embedded System	5
	1.2.1 applications	7
	1.3 Micro Controller	7
	1.4 Existing Systems	8
	1.4.1 Disadvantages of Existing System	8
	1.5 Proposed System	9
2	<b>Components and Description</b>	<b>9</b>
	2.1 Arduino UNO	9
	2.1.1 Features of Arduino UNO Board	10
	2.1.2 Arduino UNO Pin Configuration	11
	2.1.3 Power Supply	11
	2.1.4 Input & Output	11
	2.1.5 Memory	12
	2.1.6 Communication	12
	2.1.7 Pin Description	13
	2.1.8 How to use Arduino Board	14
	2.1.8.1 Communication	15
	2.1.8.2 Arduino UNO to ATmega328 Pin Mapping	15
	2.1.8.3 Applications	16
	2.2 NodeMCU	16
	2.2.1 NodeMCU Specifications	17
	2.2.2 NodeMCU Pinout & Functions Explained	20
	2.2.3 USB to Serial Converter-CP2102 OR CH340G	22
	2.2.4 NodeMCU Compatibility with Arduino IDE	23

	2.3 DS18B20 Digital Temperature Sensor	25
	2.3.1 Pin Description	26
	2.4 16x2 I2C LCD Display	26
	2.4.1 General Description	28
	2.4.2 Product Description	28
	2.4.3 features	29
	2.4.4 Applications	29
	2.4.5 Advantages	29
	2.4.6 Disadvantages	30
	2.5 Interfacing of 16x2 I2C LCD with Arduino	30
	2.6 IR Sensor	31
	2.6.1 Types of IR Sensor	32
	2.6.1.1 Active IR Sensor	32
	2.6.1.2 Passive IR Sensor	32
	2.7 MAX30100 Pulse Oximeter	33
	2.7.1 Spo2 Subsystem	34
	2.7.2 MAX30100 Pulse Oximeter Sensor	34
	Technical Specification	
	2.7.3 Interfacing MAX30100 Pulse Oximeter Sensor with Arduino	35
	2.8 Displaying MAX30100 Spo2 & BPM Value On LCD Display	36
	2.9 One Channel Relay Module Specifications	36
	2.10 Servomotor	38
	2.10.1 Servos	39
	2.10.2 SG90 Micro Servo	40
	2.10.3 Servo Library	41
	2.11 IOT(Internet Of Things)	42
	2.11.1 How IOT Works	42
	2.12 Arduino Software Description	43
	2.12.1 IDE	44
	2.12.2 Configuring Simulator	44
	2.12.3 Compilation	46
	2.12.4 Debug	47
	2.13 Blynk App	49
	2.13.1 Create a Blynk Project	50
	2.13.2 Add Widgets to the Project	
3	<b>Code</b>	52
	3.1 Arduino Code	52
	3.2 NodeMCU Code	57
4	<b>Flow Chart</b>	60



5	<b>Block Diagram</b>	<b>61</b>
	<b>Conclusion &amp; Future Enhancement</b>	
6	6.1 Conclusion	61
	6.2 Future Enhancement	62
	<b>References</b>	<b>63</b>
7		

## **LIST OF FIGURES:**

<b>FIGURE NO.</b>	<b>FIGURE NAME</b>	<b>PAGE NO.</b>
1.1	Example of Embedded System	<b>1</b>
1.2	Building block of Hardware of Embedded System	<b>4</b>
1.3	Arduino uno	<b>10</b>
1.4	Arduino Uno Board	<b>11</b>
1.5	Atmega328 pin	<b>16</b>
1.6	16x2 LCD Display	<b>29</b>
1.7	Interfacing of 16x2 I2C LCD with Arduino Mega	<b>30</b>
1.8	IR Sensor	<b>31</b>
1.9	Principle of IR Sensor	<b>32</b>
1.10	One Relay Module	<b>37</b>

# 1.INTRODUCTION:

## 1.1 EMBEDDED SYSTEM

An Embedded system is a special-purpose system in which the computer is completely encapsulated by or dedicated to the device or system it controls. Unlike a general-purpose computer, such as a personal computer, an embedded system performs one or a few predefined tasks, usually with very specific requirements. Since the system is dedicated to specific tasks, design engineers can optimize it, reducing the size and cost of the product.

Personal digital assistants (PDAs) or handheld computers are generally considered embedded devices because of the nature of their hardware design, even though they are more expandable in software terms. With the introduction of the OQO Model 2 with the Windows XP operating system and ports such as a USB port both features usually belong to "general purpose computers".

Physically, embedded systems range from portable devices such as digital watches and MP3 players, to large stationary installations like traffic lights, factory controllers, or the systems controlling nuclear power plants.

In terms of complexity embedded systems can range from very simple with a single microcontroller chip, to very complex with multiple units, peripherals and networks mounted inside a large chassis or enclosure



**Fig 1.1: Examples of embedded system**

Avionics, such as inertial guidance systems, flight control hardware/software and other integrated systems in aircraft and missiles.

- Cellular telephones and telephone switches.
- Engine controllers and antilock brake controllers for automobiles
- Home automation products, such as thermostats, air conditioners, sprinklers, and security monitoring systems

- Handheld calculators
- Handheld computers
- Household appliances, including microwave ovens, washing machines, television sets, DVD players and recorders
- Medical equipment
- Personal digital assistant
- Videogame consoles
- Computer peripherals such as routers and printers.
- Industrial controllers for remote machine operation.

### **1.1.2 CHARACTERISTICS OF EMBEDDED SYSTEM**

- An embedded system is any computer system hidden inside a product other than a computer.
- They will encounter a number of difficulties when writing embedded system software in addition to those we encounter when we write applications
  - Throughput – Our system may need to handle a lot of data in a short period of time.
  - Response–Our system may need to react to events quickly
  - Testability–Setting up equipment to test embedded software can be difficult
  - Debugability–Without a screen or a keyboard, finding out what the software is doing wrong (other than not working) is a troublesome problem
  - Reliability – embedded systems must be able to handle any situation without human intervention
  - Memory space – Memory is limited on embedded systems, and you must make the software and the data fit into whatever memory exists
  - Program installation – you will need special tools to get your software into embedded systems
  - Power consumption – Portable systems must run on battery power, and the software in these systems must conserve power

- Processor hogs – computing that requires large amounts of CPU time can complicate the response problem
- Cost – Reducing the cost of the hardware is a concern in many embedded system projects; software often operates on hardware that is barely adequate for the job.
- Embedded systems have a microprocessor/ microcontroller and a memory. Some have a serial port or a network connection. They usually do not have keyboards, screens or disk drives.

### **1.1.3 OVERVIEW OF EMBEDDED SYSTEM ARCHITECTURE**

Every embedded system consists of custom-built hardware built around a Central Processing Unit (CPU). This hardware also contains memory chips onto which software is loaded. The software residing on the memory chip is also called the ‘firmware’.

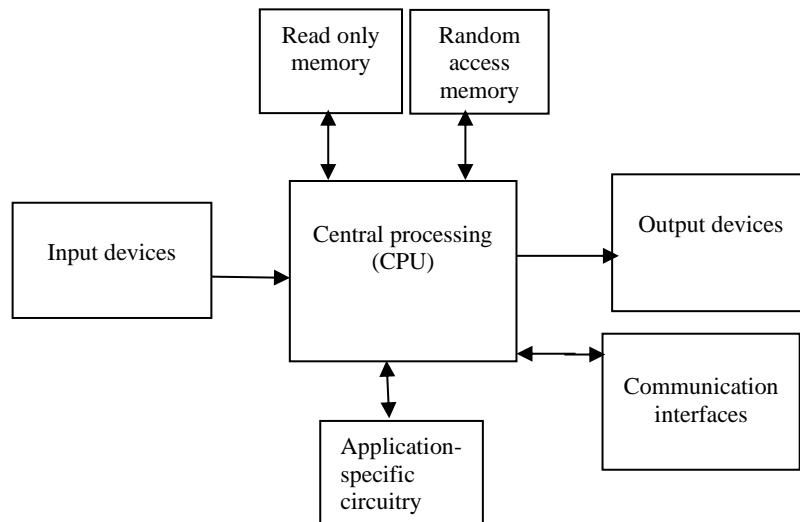
The embedded system architecture can be represented as a layered architecture as shown in Fig 1.2

The operating system runs above the hardware, and the application software runs above the operating system. The same architecture is applicable to any computer including a desktop computer. However, there are significant differences. It is not compulsory to have an operating system in every embedded system. For small appliances such as remote control units, air conditioners, toys etc., there is no need for an operating system and you can write only the software specific to that application. For applications involving complex processing, it is advisable to have an operating system. In such a case, you need to integrate the application software with the operating system and then transfer the entire software on to the memory chip. Once the software is transferred to the memory chip, the software will continue to run for a long time you don’t need to reload new software.

The details of the various building blocks of the hardware of an embedded system. As shown in Fig.1.2 the building blocks are;

- Central Processing Unit (CPU)
- Memory (Read-only Memory and (Random Access Memory)
- Input Devices

- Output devices
- Communication interfaces
- Application – specific circuitry.



**Fig1.2: Building blocks of hardware of an embedded system**

### **1.1.3.1 CENTRAL PROCESSING UNIT (CPU)**

The Central Processing Unit (processor, in short) can be any of the following: microcontroller, microprocessor or Digital Signal Processor (DSP). A micro-controller is a low-cost processor. Its main attraction is that on the chip itself, there will be many other components such as memory, serial communication interface, analog-to digital converter etc. So, for small applications, a micro-controller is the best choice as the number of external components required will be very less. On the other hand, microprocessors are more powerful, but you need to use many external components with them. DSP is used mainly for applications in which signal processing is involved such as audio and video processing.

### **1.1.3.2 MEMORY**

The memory is categorized as Random Access Memory (RAM) and Read Only Memory (ROM). The contents of the RAM will be erased if power is switched off to the chip, whereas ROM retains the contents even if the power is switched off. So, the firmware is stored in the ROM. When power is switched on, the processor reads the ROM; the program is executed.

### **1.1.3.3 INPUT DEVICES**

Unlike the desktops, the input devices to an embedded system have very limited capability. There will be no keyboard or a mouse, and hence interacting with the embedded system is no easy task. Many embedded systems will have a small keypad- you press one key to give a specific command. A keypad may be used to input only the digits. Many embedded systems used in process control do not have any input device for user interaction; they take inputs from sensors or transducers and produce electrical signals that are in turn fed to other systems.

### **1.1.3.4 OUTPUT DEVICES**

The output devices of the embedded systems also have very limited capability. Some embedded systems will have a few Light Emitting Diodes (LEDs) to indicate the health status of the system modules, or for visual indication of alarms. A small Liquid Crystal Display (LCD) may also be used to display some important parameters.

### **1.1.3.5 COMMUNICATION INTERFACES**

The embedded systems may need to, interact with other embedded systems as they may have to transmit data to a desktop. To facilitate this, the embedded systems are provided with one or a few communication interfaces such as RS232, RS422, RS485, Universal Serial Bus (USB), and IEEE 1394, Ethernet etc.

### **1.1.3.6 APPLICATION-SPECIFIC CIRCUITRY**

Sensors, transducers, special processing and control circuitry may be required for an embedded system, depending on its application. This circuitry interacts with the processor to carry out the necessary work. The entire hardware has to be given power supply either through the 230 volts' main supply or through a battery. The hardware has to be designed in such a way that the power consumption is minimized.

## **1.2 CLASSIFICATION OF EMBEDDED SYSTEMS**

Embedded systems are classified into four categories based on their performance and functional requirements:

- Standalone embedded systems
- Real time embedded systems

- Networked embedded systems
- Mobile embedded systems

Embedded Systems are classified into three types based on the performance of the microcontroller such as

- Small scale embedded systems
- Medium scale embedded systems
- Sophisticated embedded systems

#### ➤ **Stand Alone Embedded Systems**

Standalone embedded systems do not require a host system like a computer, it works by itself. It takes the input from the input ports either analog or digital and processes, calculates and converts the data and gives the resulting data through the connected device-Which either controls, drives or displays the connected devices.

Examples for the standalone embedded systems are mp3 players, digital cameras, video game consoles, microwave ovens and temperature measurement systems.

#### ➤ **Real Time Embedded Systems**

A real time embedded system is defined as, a system which gives a required o/p in a particular time. These types of embedded systems follow the time deadlines for completion of a task. Real time embedded systems are classified into two types such as soft and hard real time systems.

#### ➤ **Networked Embedded Systems**

These types of embedded systems are related to a network to access the resources. The connected network can be LAN, WAN or the internet. The connection can be any wired or wireless. This type of embedded system is the fastest growing area in embedded system applications. The embedded web server is a type of system wherein all embedded devices are connected to a web server and accessed and controlled by a web browser. Example for the LAN networked embedded system is a home security system wherein all sensors are connected and run on the protocol TCP/IP

#### ➤ **Mobile Embedded Systems**



Mobile embedded systems are used in portable embedded devices like cell phones, mobiles, digital cameras, mp3 players and personal digital assistants, etc. The basic limitation of these devices is the other resources and limitation of memory.

#### ➤ **Small Scale Embedded Systems**

These types of embedded systems are designed with a single 8 or 16-bit microcontroller that may even be activated by a battery. For developing embedded software for small scale embedded systems, the main programming tools are an editor, assembler, cross assembler and integrated development environment (IDE).

#### ➤ **Medium Scale Embedded Systems**

These types of embedded systems design with a single or 16 or 32 bit microcontroller, RISCs or DSPs. These types of embedded systems have both hardware and software complexities. For developing embedded software for medium scale embedded systems, the main programming tools are C, C++, JAVA, Visual C++, RTOS, debugger, source code engineering tool, simulator and IDE.

#### ➤ **Sophisticated Embedded Systems**

These types of embedded systems have enormous hardware and software complexities that may need ASIPs, IPs, PLAs, scalable or configurable processors. They are used for cutting-edge applications that need hardware and software Co-design and components which have to assemble in the final system.

### **1.2.1 APPLICATIONS**

1. Military and aerospace embedded software applications
2. Communication Applications
3. Industrial automation and process control software
4. Mastering the complexity of applications.
5. Reduction of product design time.
6. Real time processing of ever increasing amounts of data.
7. Intelligent, autonomous sensors.

### **1.3 MICRO CONTROLLER**

A microcontroller (MCU for *microcontroller unit* or UC for  $\mu$ -controller) is a

small [computer](#) on a single [integrated circuit](#). In modern terminology, it is similar to, but less sophisticated than, a [System on a chip](#) (SoC); an SoC may include a microcontroller as one of its components. A microcontroller contains one or more [CPUs](#)([processor cores](#)) along with [memory](#) and programmable [input/output](#) peripherals. Program memory in the form of [ferroelectric RAM](#), [NOR flash](#) or [OTP ROM](#) is also often included on chip, as well as a small amount of [RAM](#). Microcontrollers are designed for [embedded](#) applications, in contrast to the [microprocessors](#) used in [personal computers](#) or other general purpose applications consisting of various discrete chips. Microcontrollers are used in [automatically controlled](#) products and devices, such as automobile engine control systems, implantable medical devices, remote controls, office machines, appliances, power tools, toys and other [embedded systems](#). By reducing the size and cost compared to a design that uses a separate [microprocessor](#), memory, and input/output devices, microcontrollers make it economical to digitally control even more devices and processes. [Mixed signal](#) microcontrollers are common, integrating analog components needed to control non-digital electronic systems. In the context of the [internet of things](#), microcontrollers are an economical and popular means of [data collection](#), [sensing](#) and [actuating](#) the physical world as [edge devices](#).

## **1.4 EXISTING SYSTEM**

The coronavirus spreads mainly from person to person. A person infected with coronavirus — even one with no symptoms — may emit aerosols when they talk or breathe. Aerosols are infectious viral particles that can float or drift around in the air for up to three hours. Another person can breathe in these aerosols and become infected with the coronavirus.

When people are in close contact with one another, droplets that are produced when an infected person coughs or sneezes may land in the mouths or noses of people who are nearby, or possibly be inhaled into their lungs.

The risk of spread from contact with contaminated surfaces or objects is considered to be extremely low. According to the CDC, each contact with a contaminated surface has less than a 1 in 10,000 chance of causing an infection.

### **1.4.1 Disadvantages of Existing System:**

- Lost of lives
- Fast spreading of covid.

## **1.5 PROPOSED SYSTEM**

- The COVID-19 is very infectious, and it spreads super-fast by contacting other COVID infected people. Even though we are taking all the precautions the disease is spreading rapidly worldwide. The idea is to create a system to stop entering in the COVID suspected to our surroundings like our home, colleges, public malls etc.
- The aim of this work is to detect COVID-19 suspect people and stop them from spreading the disease to other people. The person can sanitize his hands using contactless hand sanitizer and then the door will be unlocked and the person can enter into our place.
- The most advantage of our system is cost-effective, good performance and easy to implement on any campus or office and also in our homes. By implementing this we can reduce the risk of contacting Covid-19 and save our lives.

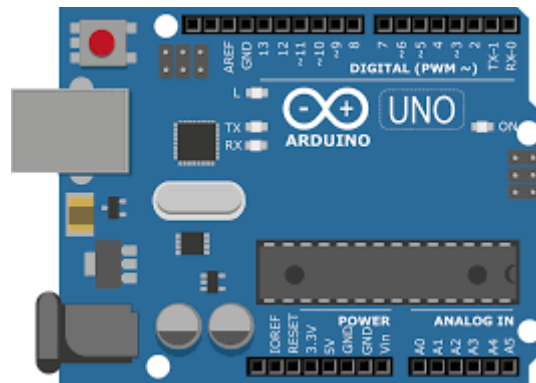
## **2. COMPONENTS AND DESCRIPTION**

### **2.1 ARDUNIO UNO**

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 Arduno, 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 **ATmega328** is one kind of single-chip microcontroller formed with Atmel within the **megaAVR family**. The architecture of this Arduino Uno is a customized Harvard architecture with 8 bit **RISC processor** core. **Uno** include Arduino Pro Mini, Arduino Nano, Arduino Due, Arduino Mega, and Arduino Leonardo.



**Fig 1.3 Ardunio uno**

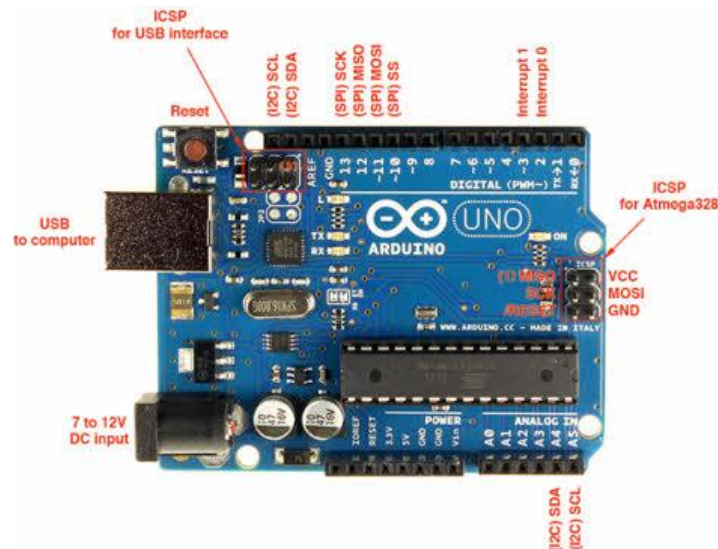
### **2.1.1 Features of Arduino Uno Board**

The **features of Arduino Uno ATmega328** includes the following.

- The operating voltage is 5V
- The recommended input voltage will range from 7v to 12V
- The input voltage ranges from 6v to 20V
- Digital input/output pins are 14
- Analog i/p pins are 6
- DC Current for each input/output pin is 40 mA
- DC Current for 3.3V Pin is 50 mA
- Flash Memory is 32 KB
- SRAM is 2 KB
- EEPROM is 1 KB
- CLK Speed is 16 MHz

### 2.1.2 Arduino Uno Pin Diagram

The Arduino Uno board can be built with power pins, analog pins, ATmega328, ICSP header, Reset button, [power LED](#), digital pins, test led 13, TX/RX pins, USB interface, an external [power supply](#). The **Arduino UNO board description** is discussed below.



**Fig 1.4: Arduino Uno Board**

### 2.1.3 Power Supply

The **Arduino Uno power supply** can be done with the help of a USB cable or an external power supply. The external power supplies mainly include AC to DC adapter otherwise a battery. The adapter can be connected to the Arduino Uno by plugging into the power jack of the Arduino board. Similarly, **the battery** leads can be connected to the Vin pin and the GND pin of the POWER connector. The suggested voltage range will be 7 volts to 12 volts.

### 2.1.4 Input & Output

The 14 digital pins on the Arduino Uno can be used as input & output with the help of the functions like `pinMode()`, `digitalWrite()`, & `Digital Read()`.

**Pin1 (TX) & Pin0 (RX) (Serial):** This pin is used to transmit & receive TTL serial data, and these are connected to the ATmega8U2 USB to TTL Serial chip equivalent pins.

**Pin 2 & Pin 3 (External Interrupts):** External pins can be connected to activate an interrupt over a low value, change in value.

**Pins 3, 5, 6, 9, 10, & 11 (PWM):** This pin gives 8-bit PWM o/p by the function of `analogWrite()`. **SPI Pins (Pin-10 (SS), Pin-11 (MOSI), Pin-12 (MISO), Pin-13 (SCK):** These pins maintain SPI-communication, even though offered by the fundamental hardware, is not presently included within the Arduino language.

**Pin-13(LED):** The inbuilt LED can be connected to pin-13 (digital pin). As the HIGH-value pin, the light emitting diode is activated, whenever the pin is LOW.

**Pin-4 (SDA) & Pin-5 (SCL) (I2C):** It supports TWI-communication with the help of the Wire library.

**AREF (Reference Voltage):** The reference voltage is for the analog i/ps with `analogReference()`.

**Reset Pin:** This pin is used for reset (RST) the microcontroller.

### 2.1.5 Memory

The memory of this Atmega328 Arduino microcontroller includes flash memory-32 KB for storing code, SRAM-2 KB EEPROM-1 KB.

### 2.1.6 Communication

The Arduino Uno ATmega328 offers UART TTL-**serial communication**, and it is accessible on digital pins like TX (1) and RX (0). The software of an Arduino has a serial monitor that permits easy data. There are two LEDs on the board like RX & TX which will blink whenever data is being broadcasted through the USB.

A SoftwareSerial library permits for serial communication on Arduino Uno digital pins and the ATmega328P supports TWI (I2C) as well as **SPI-communication**. The Arduino software contains a wired library for simplifying the utilization of the I2C bus.

### 2.1.7. PIN DESCRIPTION

Pin Category	Pin Name	Details
Power	Vin, 3.3V, 5V, GND	Vin: Input voltage to Arduino when using an external power source. 5V: Regulated power supply used to power microcontroller and other components on the board. 3.3V: 3.3V supply generated by on-board voltage regulator. Maximum current draw is 50mA. GND: ground pins.
Reset	Reset	Resets the microcontroller.
Analog Pins	A0 – A5	Used to provide analog input in the range of 0-5V
Input/Output Pins	Digital Pins 0 - 13	Can be used as input or output pins.
Serial	0(Rx), 1(Tx)	Used to receive and transmit TTL serial data.
External Interrupts	2, 3	To trigger an interrupt.
PWM	3, 5, 6, 9, 11	Provides 8-bit PWM output.
SPI	10 (SS), 11 (MOSI), 12 (MISO) and 13 (SCK)	Used for SPI communication.
Inbuilt LED	13	To turn on the inbuilt LED.

TWI	A4 (SDA), A5 (SCA)	Used for TWI communication.
AREF	AREF	To provide reference voltage for input voltage.

### 2.1.8 HOW TO USE ARDUINO BOARD

The 14 digital input/output pins can be used as input or output pins by using pinMode(), digitalRead() and digitalWrite() functions in arduino programming. Each pin operate at 5V and can provide or receive a maximum of 40mA current, and has an internal pull-up resistor of 20-50 KOhms which are disconnected by default. Out of these 14 pins, some pins have specific functions as listed below:

- **Serial Pins 0 (Rx) and 1 (Tx):** Rx and Tx pins are used to receive and transmit TTL serial data. They are connected with the corresponding ATmega328P USB to TTL serial chip.
- **External Interrupt Pins 2 and 3:** These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value.
- **PWM Pins 3, 5, 6, 9 and 11:** These pins provide an 8-bit PWM output by using analogWrite() function.
- **SPI Pins 10 (SS), 11 (MOSI), 12 (MISO) and 13 (SCK):** These pins are used for SPI communication.
- **In-built LED Pin 13:** This pin is connected with an built-in LED, when pin 13 is HIGH – LED is on and when pin 13 is LOW, its off.

Along with 14 Digital pins, there are 6 analog input pins, each of which provide 10 bits of resolution, i.e. 1024 different values. They measure from 0 to 5 volts but this limit can be increased by using AREF pin with analog Reference() function.

- Analog pin 4 (SDA) and pin 5 (SCA) also used for TWI communication using Wire library.



Arduino Uno has a couple of other pins as explained below:

- **AREF:** Used to provide reference voltage for analog inputs with `analogReference()` function.
- **Reset Pin:** Making this pin LOW, resets the microcontroller.

#### 2.1.8.1 COMMUNICATION

Arduino can be used to communicate with a computer, another Arduino board or other microcontrollers. The ATmega328P microcontroller provides UART TTL (5V) serial communication which can be done using digital pin 0 (Rx) and digital pin 1 (Tx). An ATmega16U2 on the board channels this serial communication over USB and appears as a virtual com port to software on the computer. The ATmega16U2 firmware uses the standard USB COM drivers, and no external driver is needed. However, on Windows, a .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. There are two RX and TX LEDs on the arduino board which will flash when data is being transmitted via the USB-to-serial chip and USB connection to the computer (not for serial communication on pins 0 and 1). A SoftwareSerial library allows for serial communication on any of the Uno's digital pins. The ATmega328P also supports I2C (TWI) and SPI communication. The Arduino software includes a Wire library to simplify use of the I2C bus.

#### 2.1.8.2 Arduino Uno to ATmega328 Pin Mapping

When ATmega328 chip is used in place of Arduino Uno, or vice versa, the image below shows the pin mapping between the two.

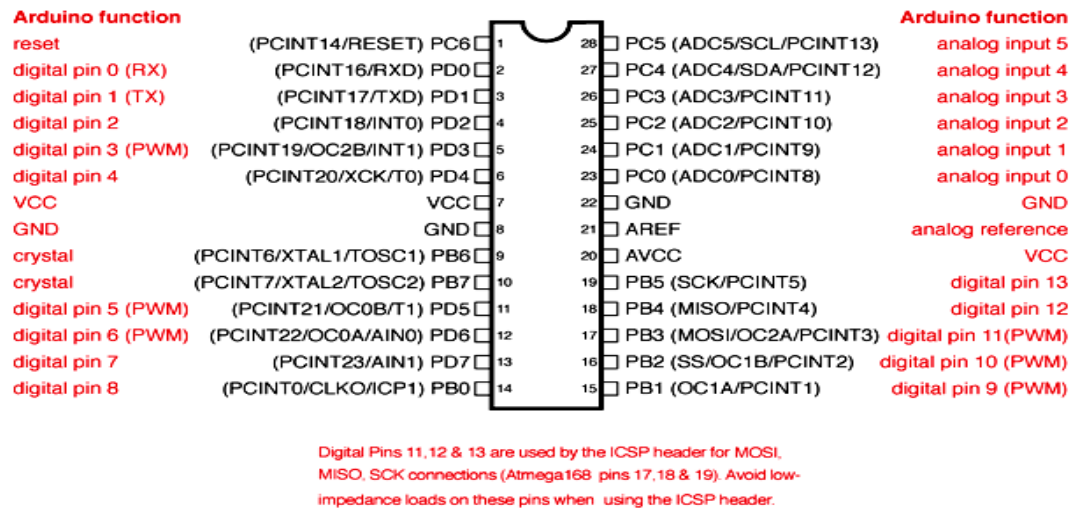


Fig 1.5 : AT mega 328 pin

### 2.1.8.3 Applications

- Prototyping of Electronics Products and Systems
- Multiple DIY Projects.
- Easy to use for beginner level DIYers and makers.

Projects requiring Multiple I/O interfaces and communications

## 2.2 NODEMCU



:

TheNodeMCU (Node *Micro*Controller *Unit*) is an open-source software and hardware development environment built around an inexpensive System-on-a-Chip (SoC) called the ESP8266. The ESP8266, designed and manufactured by Espressif Systems, contains

the crucial elements of a computer: CPU, RAM, networking (WiFi), and even a modern operating system and SDK. That makes it an excellent choice for Internet of Things (IoT) projects of all kinds.

However, as a chip, the ESP8266 is also hard to access and use. You must solder wires, with the appropriate analog voltage, to its pins for the simplest tasks such as powering it on or sending a keystroke to the “computer” on the chip. You also have to program it in low-level machine instructions that can be interpreted by the chip hardware. This level of integration is not a problem using the ESP8266 as an embedded controller chip in mass-produced electronics. It is a huge burden for hobbyists, hackers, or students who want to experiment with it in their own IoT projects.

But, what about Arduino? The Arduino project created an open-source hardware design and software SDK for their versatile IoT controller. Similar to NodeMCU, the Arduino hardware is a microcontroller board with a USB connector, LED lights, and standard data pins. It also defines standard interfaces to interact with sensors or other boards. But unlike NodeMCU, the Arduino board can have different types of CPU chips (typically an ARM or Intel x86 chip) with memory chips, and a variety of programming environments. There is an Arduino reference design for the ESP8266 chip as well. However, the flexibility of Arduino also means significant variations across different vendors. For example, most Arduino boards do not have WiFi capabilities, and some even have a serial data port instead of a USB port.

### **2.2.1 NodeMCU Specifications**

The NodeMCU is available in various package styles. Common to all the designs is the base ESP8266 core. Designs based on the architecture have maintained the standard 30-pin layout. Some designs use the more common narrow (0.9”) footprint, while others use a wide (1.1”) footprint – an important consideration to be aware of.

The most common models of the NodeMCU are the Amica (based on the standard narrow pin-spacing) and the LoLin which has the wider pin spacing and larger board. The open-

source design of the base ESP8266 enables the market to design new variants of the NodeMCU continually.

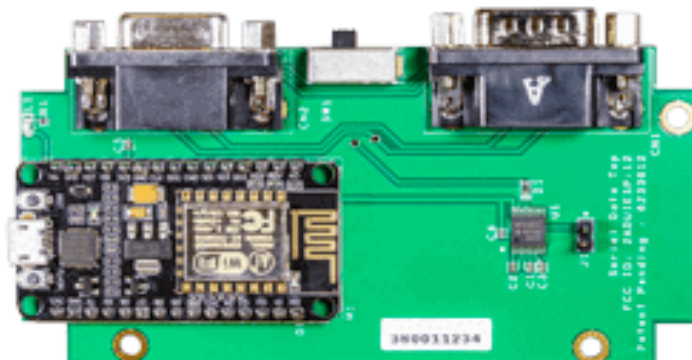
### **Official Amica NodeMCU**



Amica NodeMCU measures 49mm x 26mm with a standard pin space of 0.1" between pins and 0.9" between rows.

The Amica NodeMCU is approximately 25% smaller in size than a closely compatible LoLin style NodeMCU.

### **Official Amica NodeMCU on Carrier Board**



Amico NodeMCU mounted to a 102mm x 51mm carrier board with dual DB-09 male/female connectors.

FEATURES	NODEMCU
Microcontroller	ESP-8266 32-bit
NodeMCU Model	Amica
NodeMCU Size	49mm x 26mm
Carrier Board Size	n/a
Pin Spacing	<b>0.9" (22.86mm)</b>
Clock Speed	80 MHz
USB to Serial	CP2102
USB Connector	Micro USB
Operating Voltage	3.3V
Input Voltage	4.5V-10V
Flash Memory/SRAM	4 MB / 64 KB
Digital I/O Pins	11
Analog In Pins	1
ADC Range	0-3.3V
UART/SPI/I2C	1 / 1 / 1

## FEATURES

## NODEMCU

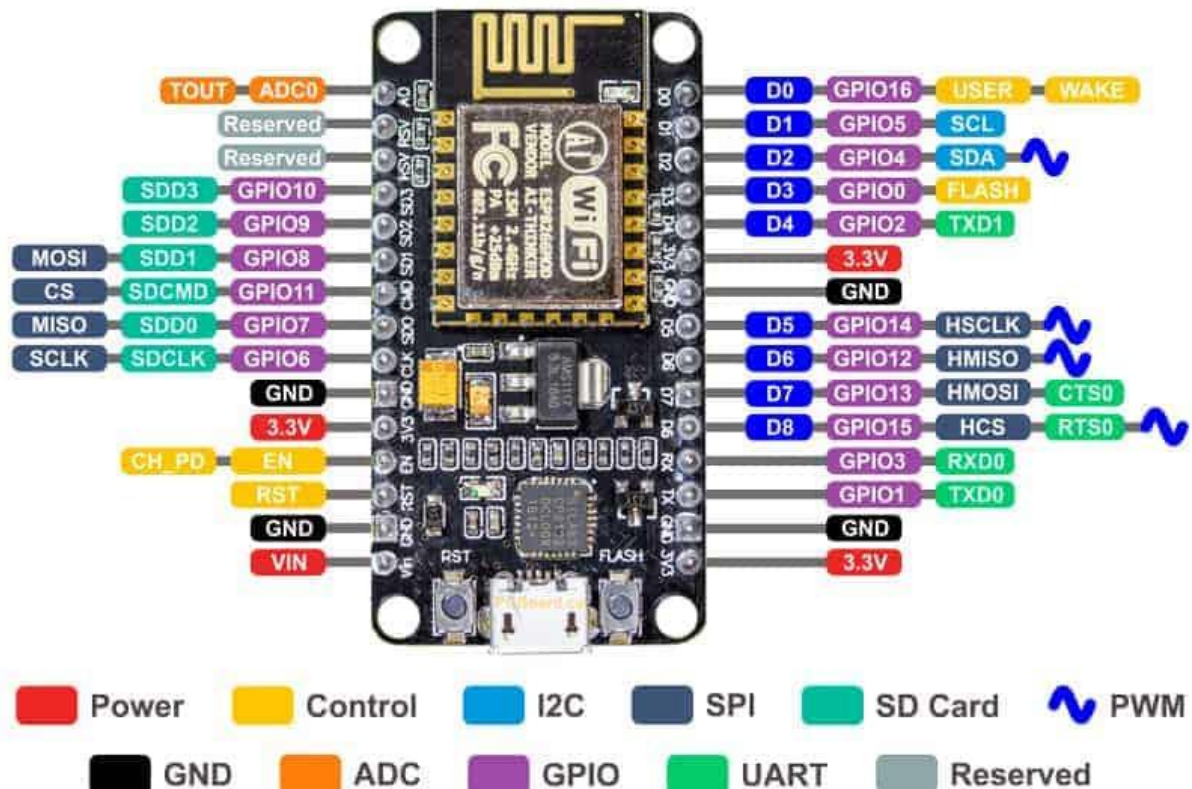
WiFi Built-In

802.11 b/g/n

Temperature Range

-40C - 125C

### 2.2.2 NodeMCU Pinout and Functions Explained :-



- **Power Pins** There are four power pins. **VIN** pin and three **3.3V** pins.
- **VIN** can be used to directly supply the NodeMCU/ESP8266 and its peripherals. Power delivered on **VIN** is regulated through the onboard regulator on the NodeMCU module – you can also supply 5V regulated to the **VIN** pin
- **3.3V** pins are the output of the onboard voltage regulator and can be used to supply power to external components.
- **GND** are the ground pins of NodeMCU/ESP8266
- **I2C Pins** are used to connect I2C sensors and peripherals. Both I2C Master and I2C Slave are supported. I2C interface functionality can be realized programmatically, and the clock frequency is 100 kHz at a maximum. It should be




noted that I2C clock frequency should be higher than the slowest clock frequency of the slave device.

- **GPIO Pins** NodeMCU/ESP8266 has 17 GPIO pins which can be assigned to functions such as I2C, I2S, UART, PWM, IR Remote Control, LED Light and Button programmatically. Each digital enabled GPIO can be configured to internal pull-up or pull-down, or set to high impedance. When configured as an input, it can also be set to edge-trigger or level-trigger to generate CPU interrupts.
- **ADC Channel** The NodeMCU is embedded with a 10-bit precision SAR ADC. The two functions can be implemented using ADC. Testing power supply voltage of VDD3P3 pin and testing input voltage of TOUT pin. However, they cannot be implemented at the same time.
- **UART Pins** NodeMCU/ESP8266 has 2 UART interfaces (UART0 and UART1) which provide asynchronous communication (RS232 and RS485), and can communicate at up to 4.5 Mbps. UART0 (TXD0, RXD0, RST0 & CTS0 pins) can be used for communication. However, UART1 (TXD1 pin) features only data transmit signal so, it is usually used for printing log.
- **SPI Pins** NodeMCU/ESP8266 features two SPIs (SPI and HSPI) in slave and master modes. These SPIs also support the following general-purpose SPI features:
  - 4 timing modes of the SPI format transfer
  - Up to 80 MHz and the divided clocks of 80 MHz
  - Up to 64-Byte FIFO
- **Pins** NodeMCU/ESP8266 features Secure Digital Input/Output Interface (SDIO) which is used to directly interface SD cards. 4-bit 25 MHz SDIO v1.1 and 4-bit 50 MHz SDIO v2.0 are supported.
- **PWM Pins** The board has 4 channels of Pulse Width Modulation (PWM). The PWM output can be implemented programmatically and used for driving digital motors and LEDs. PWM frequency range is adjustable from 1000  $\mu$ s to 10000  $\mu$ s (100 Hz and 1 kHz).

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

- **EN:** The ESP8266 chip is enabled when EN pin is pulled HIGH. When pulled LOW the chip works at minimum power.
- **RST:** RST pin is used to reset the ESP8266 chip.
- **WAKE:** Wake pin is used to wake the chip from deep-sleep.

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

- **EN:** The ESP8266 chip is enabled when EN pin is pulled HIGH. When pulled LOW the chip works at minimum power.
- **RST:** RST pin is used to reset the ESP8266 chip.
- **WAKE:** Wake pin is used to wake the chip from deep-sleep.

### 2.2.3 USB to Serial Converter – CP2102 or CH340G

Incorporated into each NodeMCU is a USB to Serial Converter. The official design is based on the CP2102 chipset and offers the best compatibility. Genuine boards use the CP2102 chipset including the officially licensed Amica NodeMCU modules. The other common USB to Serial Converter used is the CH340G which is common on the lower-priced modules including the LoLin units. Other designs may use drivers including the FTDI chipset, but those designs are rare.

Depending on the Operating System you are using with the NodeMCU, the appropriate driver must be installed. Generally, Windows 10 immediately recognizes the CP2102 chipset while the CH340G may require separate installation.



- Drivers for the CP2102 are available for **download** from the [Silicon Labs support site](https://www.silabs.com/Support%20/Software/Drivers). Drivers constantly evolve and ensuring and installing the most recent version in your development environment minimum issues. Drivers are



available for Windows, Mac, Linux, and Android. *We also have a local copy of the CP2102 drivers (v10.1.8) available locally for [download](#). You are always best to visit the original manufacturer to ensure you are receiving the most recent versions of the driver.*

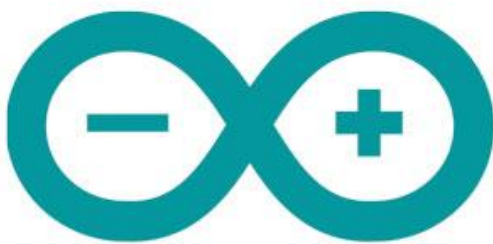


- WCH maintain and update the drivers for the CH340G on a regular basis. Versions of the driver are also available for Windows, Mac, Linux, and Android. Visit their [Driver Download](#) page. We also have a local copy of the CH340G drivers (version 3.5) available locally for [download](#). *You are always best to visit the original manufacturer to ensure you are receiving the most recent versions of the driver.*

We have experienced situations where both CP2102 and CH340G devices have not functioned or been recognized as expected. The solution was as simple as uninstalling the old driver and installing the most recent version.

#### 2.2.4 NodeMCU COMPATIBILITY WITH ARDUINO IDE:

# ARDUINO + NodeMCU



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

The **NodeMCU IoT Experimenter** is a versatile prototyping platform for use with a variety of the most popular NodeMCU modules including our NodeMCU Carrier Board. Great to use for IoT projects, advanced or straightforward interfacing, and as a prototyping platform. The NodeMCU, with its versatility, including its ability to be programmed and used from the Arduino IDE, makes it along with this prototyping board the perfect experimenter's solution.

The **NodeMCU IoT Experimenter** measures 5 5/16" x 4.5" (135mm x 115mm) with a solder mask on each side, plated holes along with a high-contrast silk-screen labeling component and prototyping positions.

Features of the board include a mounting socket area to accept either wide 1.1" or narrow 0.9" pitch NodeMCU modules. This includes the Amica NodeMCU carrier board (narrow pin spacing) to compatible variants such as the LoLin NodeMCU models. Power can be provided directly to your NodeMCU module through its built-in USB interface. Alternately, power can be supplied to the **IoT Experimenter** board which has provisions for an integrated regulated power supply module.

The board offers over 1,000 plated-through holes on the prototype surface, mounting for eight status indicator LEDs along with dropping resistors and a power indicator LED.

The **NodeMCU IoT Experimenter** is a versatile prototyping platform for use with a variety of the most popular NodeMCU modules including our NodeMCU Carrier Board. Great to use for IoT projects, advanced or straightforward interfacing, and as a prototyping platform. The NodeMCU, with its versatility, including its ability to be

programmed and used from the Arduino IDE, makes it along with this prototyping board the perfect experimenter's solution.

The **NodeMCU IoT Experimenter** measures 5 5/16" x 4.5" (135mm x 115mm) with a solder mask on each side, plated holes along with a high-contrast silk-screen labeling component and prototyping positions.

Features of the board include a mounting socket area to accept either wide 1.1" or narrow 0.9" pitch NodeMCU modules. This includes the Amica NodeMCU carrier board (narrow pin spacing) to compatible variants such as the LoLin NodeMCU models. Power can be provided directly to your NodeMCU module through its built-in USB interface. Alternately, power can be supplied to the **IoT Experimenter** board which has provisions for an integrated regulated power supply module.

The board offers over 1,000 plated-through holes on the prototype surface, mounting for eight status indicator LEDs along with dropping resistors and a power indicator LED. The prototyping area offers power busbars for the Ground (**G**), +3.3V (**3V**) power rail, and a third rail **X**. The third rail can be used for external voltages such as a 5V rail.

Interfacing to the NodeMCU is through a series of headers that extend each pin of the NodeMCU to rows of four headers. Each port is labeled to identify matching pins f

## **2.3 DS18B20 DIGITAL TEMPERATURE SENSOR:-**

In this tutorial, we will use pre-wired and waterproof version of the DS18B20 digital temperature sensor. This sensor is handy when you want to measure temperature of the place which is far away, or in wet condition. Because the sensor is digital, we don't get signal degradation over a long distance. This 1-wire digital temperature sensor is fairly precise (i.e + or – 0.5-degree Celsius).



### 2.3.1 PIN DESCRIPTION

The DS18B20 waterproof digital temperature sensor has 3 pins (2-power supply pin 1-data pin)

- VCC – Red
- GND – Black
- DATA – Yellow or White
- Connect “GND” of DS18B20 sensor to “GND” of evive board
- We will also use 4.6 k ohm resistor as a pull-up resistor between “VCC” and “DATA” pin as shown below in the circuit diagram
- Connect one end of the resistor to “VCC” and another end to “DATA” wires of the temperature sensor
- Connect “VCC” of the temperature sensor to “VCC” of evive board
- Connect “DATA” pin of the temperature sensor to pin number 2 of evive board

### 2.4 16x2 I2C LCD DISPLAY

A **liquid-crystal display (LCD)** is a flat-panel display or other electronically modulated optical device that uses the light-modulating properties of liquid crystals combined with polarizers. Liquid crystals do not emit light directly, instead

using a backlight or reflector to produce images in color or monochrome. LCDs are available to display arbitrary images (as in a general-purpose computer display) or fixed images with low information content, which can be displayed or hidden, such as preset words, digits, and seven-segment displays, as in a digital clock. They use the same basic technology, except that arbitrary images are made from a matrix of small pixels, while other displays have larger elements. LCDs can either be normally on (positive) or off (negative), depending on the polarizer arrangement. For example, a character positive LCD with a backlight will have black lettering on a background that is the color of the backlight, and a character negative LCD will have a black background with the letters being of the same color as the backlight. Optical filters are added to white on blue LCDs to give them their characteristic appearance.

LCDs are used in a wide range of applications, including LCD televisions, computer monitors, instrument panels, aircraft cockpit displays, and indoor and outdoor signage. Small LCD screens are common in portable consumer devices such as digital cameras, watches, calculators, and mobile telephones, including smartphones. LCD screens are also used on consumer electronics products such as DVD players, video game devices and clocks. LCD screens have replaced heavy, bulky cathode ray tube (CRT) displays in nearly all applications. LCD screens are available in a wider range of screen sizes than CRT and plasma displays, with LCD screens available in sizes ranging from tiny digital watches to very large television receivers.

LCDs are slowly being replaced by OLEDs, which can be easily made into different shapes, and have a lower response time, wider color gamut, virtually infinite color contrast and viewing angles, lower weight for a given display size and a slimmer profile (because OLEDs use a single glass or plastic panel whereas LCDs use two glass panels; the thickness of the panels increases with size but the increase is more noticeable on LCDs) and potentially lower power consumption (as the display is only "on" where needed and there is no backlight). OLEDs, however, are more expensive for a given display size due to the very expensive electroluminescent materials or phosphors that they use. Also due to the use of phosphors, OLEDs suffer from screen burn-in and there is currently no way to recycle OLED displays, whereas LCD panels can be recycled,

although the technology required recycling LCDs is not yet widespread. Attempts to increase the lifespan of LCDs are quantum dot displays, which offer similar performance to an OLED display, but the quantum dot sheet that gives these displays their characteristics cannot yet be recycled.

Since LCD screens do not use phosphors, they rarely suffer image burn-in when a static image is displayed on a screen for a long time, e.g., the table frame for an airline flight schedule on an indoor sign. LCDs are, however, susceptible to image persistence. The LCD screen is more energy-efficient and can be disposed of more safely than a CRT can. Its low electrical power consumption enables it to be used in battery-powered electronic equipment more efficiently than a CRT.

#### **2.4.1 GENERAL DESCRIPTION**

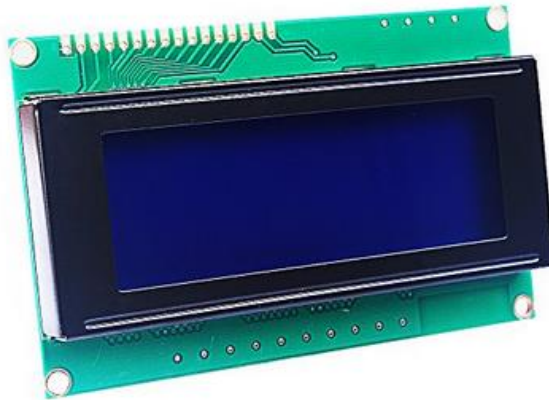
A liquid-crystal display (LCD) is a flat panel display, electronic visual display, or video display that uses the light modulating properties of liquid crystals. Liquid crystals do not emit light directly. Here, in this we're going to use a monochromatic 16x2 alphanumeric LCD. 16x2 means that 20 characters can be displayed in each of the 4 rows of the 16x2 LCD, thus a total of 80 characters can be displayed at any instance of time. LCD accepts two types of signals, one is data, and another is control. These signals are recognized by the LCD module from status of the RS pin. Now data can be read also from the LCD display, by pulling the R/W pin high. As soon as the E pin is pulsed, LCD display reads data at the falling edge of the pulse and executes it, same for the case of transmission.

#### **2.4.2 PRODUCT DESCRIPTION**

This is a basic 20 character by 4 line display. Utilizes the extremely common HD44780 parallel interface chipset. Interface code is freely available. You will need ~11 general I/O pins to interface to this LCD screen. Includes LED backlight. 20 characters wide, 4 rows character LCD module, SPLC780C controller, 6800 4/8-bit parallel interface, single led backlight with yellow green colour included can be dimmed easily with a resistor or PWM, STN-LCD positive, dark blue text on the yellow green colour,

wide operating temperature range, ROHS compliant, built in character set supports English/Japanese text.

The full character set. Optional 3.3v or 5v power supply and optional pin header connection it's easily controlled by MCU such as 8051, PIC, AVR, ARDUINO, ARM and Raspberry PI. IT can be used in any embedded systems, industrial device, and security, medical and hand-held equipment.



**Fig 1.6: 16x2 LCD display**

### **2.4.3 FEATURES**

- 5x8 dots
- Built-in controller (S6A0069 or equivalent)
- +5V power supply
- 1/16 duty cycle
- LED Backlight

### **2.4.4 APPLICATIONS**

- Monitoring

### **2.4.5 ADVANTAGES**

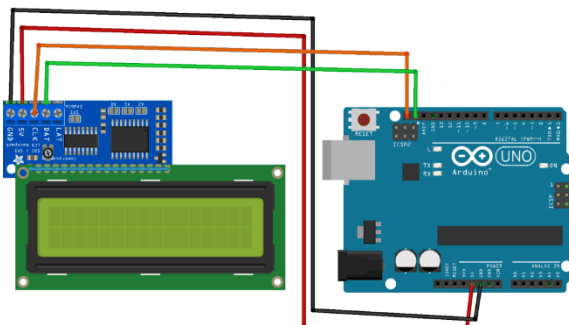
- Low power consumption
- Little heat emitted during operation
- No geometric distortion.

- Usually no refresh-rate flicker, because the LCD pixels hold their state between refreshes
- Unaffected by magnetic fields, including the Earth's
- Can be made with very narrow frame borders, allowing multiple LCD screens to be arrayed side-by-side to make up what looks like one big screen.

### 2.2.12 DISADVANTAGES

- Limited viewing angle in some older or cheaper monitors, causing colour, saturation, contrast and brightness to vary with user position, even within the intended viewing angle.
- As of 2012, most implementations of LCD backlighting use pulse-width modulation (PWM) to dim the display, which makes the screen flicker more acutely
- Dead may occur during manufacturing or after a period of use, dead one will always remain black.
- Loss of contrast in high temperature environments.

## 2.5 INTERFACING OF 16x2 I2C LCD WITH ARDUINO MEGA:



**Fig 1.7: Interfacing of 16x2 I2C LCD with Arduino Mega**



## 2.6 IR SENSOR:

IR sensor is an electronic device that emits the light in order to sense some object of the surroundings. An [IR sensor](#) can measure the heat of an object as well as detects the motion. Usually, in the **infrared spectrum**, all the objects radiate some form of thermal radiation. These types of radiations are invisible to our eyes, but infrared sensor can detect these radiations.



**Fig1.8: IR Sensor**

The emitter is simply an IR LED ([Light Emitting Diode](#)) and the detector is simply an IR photodiode. Photodiode is sensitive to IR light of the same wavelength which is emitted by the IR LED. When IR light falls on the photodiode, the resistances and the output voltages will change in proportion to the magnitude of the IR light received.

There are five basic elements used in a typical infrared detection system: an infrared source, a transmission medium, optical component, infrared detectors or receivers and signal processing. Infrared lasers and Infrared LED's of specific wavelength used as infrared sources.

The three main types of media used for infrared transmission are vacuum, atmosphere and optical fibers. Optical components are used to focus the infrared radiation or to limit the spectral response.

### 2.6.1 TYPES OF IR SENSOR:

There are two types of IR sensors are available and they are,

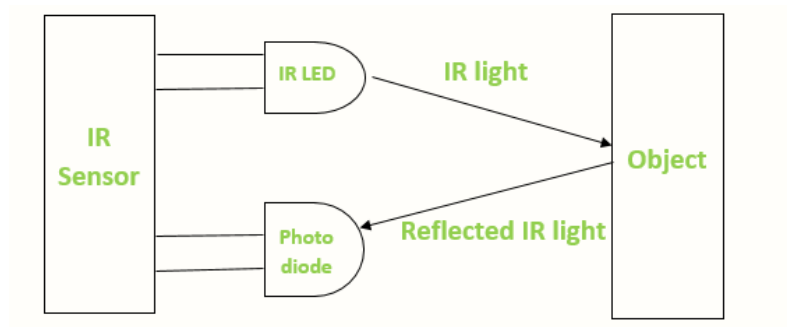
- Active Infrared Sensor
- Passive Infrared Sensor

#### 2.6.1.1 ACTIVE IR SENSOR:

Active infrared sensors consist of two elements: infrared source and infrared detector. Infrared sources include the LED or infrared [laser diode](#). Infrared detectors include photodiodes or phototransistors. The energy emitted by the infrared source is reflected by an object and falls on the infrared detector.

#### 2.6.1.2 PASSIVE IR SENSOR:

Passive infrared [sensors](#) are basically Infrared detectors. Passive infrared sensors do not use any infrared source and detector. They are of two types: quantum and thermal. Thermal infrared sensors use infrared energy as the source of heat. **Thermocouples**, pyroelectric detectors and bolometers are the common types of thermal infrared detectors. Quantum type infrared sensors offer higher detection performance. It is faster than thermal type infrared detectors. The photo sensitivity of quantum type detectors is wavelength dependent.



**Fig1.9 Principle of IR Sensor**

When the IR transmitter emits radiation, it reaches the object and some of the radiation reflects back to the IR receiver. Based on the intensity of the reception by the IR receiver, the output of the [sensor](#) defines.

## 2.7 MAX30100 PULSE OXIMETER:-

The MAX30100 is an integrated pulse Oximetry and heartrate monitor sensor solution. It combines two LEDs, a photodetector, optimized optics, and low-noise analog signal processing to detect pulse Oximetry and heart-rate signals



There is another type of the Max30100 pulse Oximeter as you can see in the picture below,

This type of the Oximeter has the sensors and electronic components all on the same side, moreover there are many complaints about this type of the sensor. I recommend

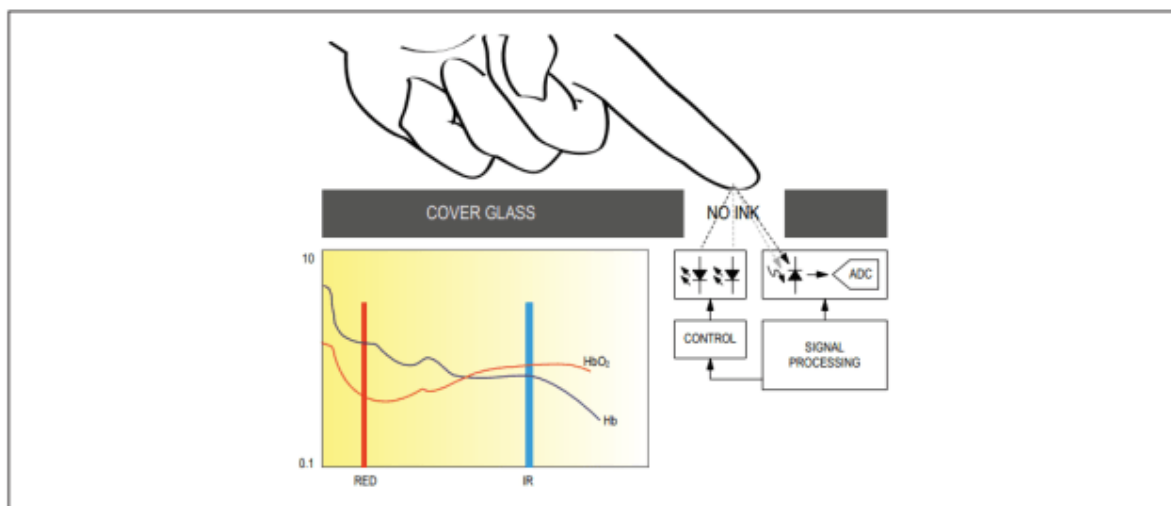
you should purchase the one I am using in this tutorial. Because, while using this sensor you don't need to remove the resistors or solder any extra wires.

### 2.7.1 SpO2 SUBSYSTEM

The SpO2 subsystem in the MAX30100 is composed of ambient light cancellation (ALC), 16-bit sigma delta ADC, and proprietary discrete time filter.

The SpO2 ADC is a continuous time oversampling sigma delta converter with up to 16-bit resolution. The ADC out-put data rate can be programmed from 50Hz to 1kHz. The MAX30100 includes a proprietary discrete time filter to reject 50Hz/60Hz interference and low-frequency residual ambient noise.

#### System Block Diagram



As per the system block diagram which is available in the datasheet, it clearly shows that there should be small distance between the sensor and finger.

### 2.7.2 MAX30100 PULSE OXIMETER SENSOR TECHNICAL SPECIFICATION:

The MAX30100 operates from 1.8V and 3.3V power supplies.

#### Applications

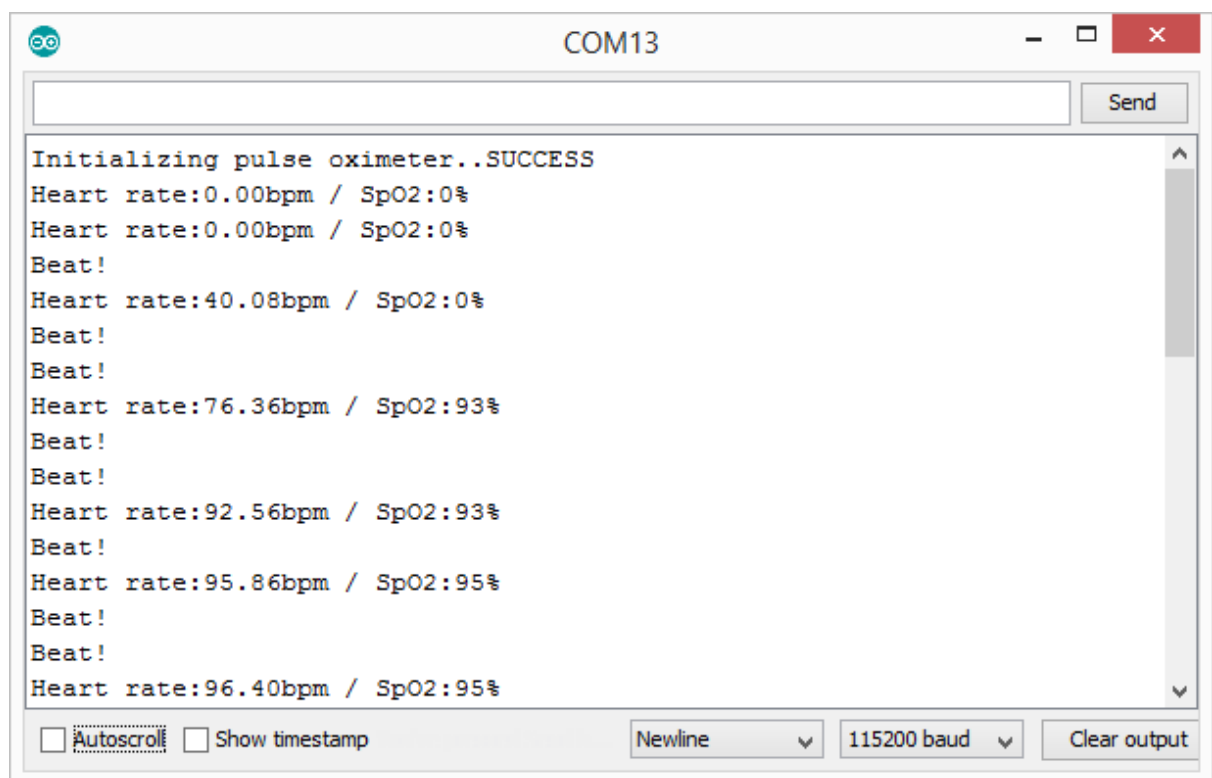
- Wearable Devices
- Fitness Assistant Devices
- Medical Monitoring Devices

### 2.7.3 INTERFACING MAX30100 PULSE OXIMETER SENSOR WITH ARDUINO

Now let us interface MAX30100 Pulse Oximeter Sensor with Arduino and display the value in serial monitor. The circuit diagram and connection is very simple. You can follow the same.

Connect the Vin pin of MAX30100 to Arduino **5V** or **3.3V** pin, GND to GND. Connect the I2C Pin of MAX30100, i.e **SCL & SDA** to **A5 & A4** of Arduino.

After uploading the code, open the serial monitor to see the values as shown in the image. Initially the the BPM & SpO2 value appears as incorrect value but soon you can observe the correct stable reading.





---

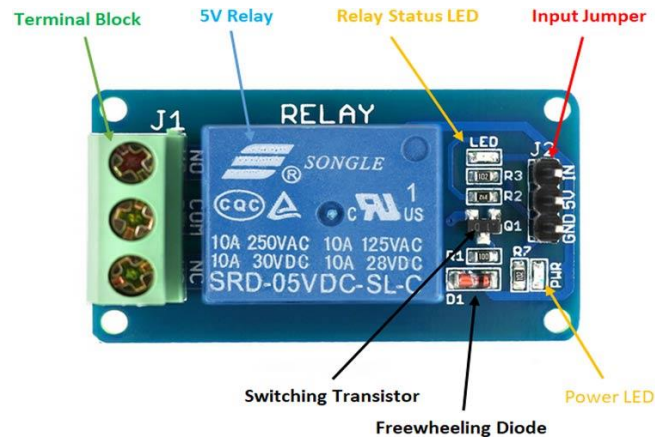
## 2.8 DISPLAYING MAX30100 SpO2 & BPM VALUE ON LCD DISPLAY

Now let us use the **16X2 LCD Display** to see the value of **BPM & SpO2** instead of Serial Monitor. Assemble the circuit as per the circuit diagram below.

Connect the Vin pin of MAX30100 to Arduino 5V or 3.3V pin, GND to GND. Connect the I2C Pin, SCL & SDA of MAX30100 to A5 & A4 of Arduino. Similarly connect the LCD pin 1, 5, 16 to GND of Arduino and 2, 15 to 5V VCC. Similarly connect LCD pin 4, 6, 11, 12, 13, 14 to Arduino pin 13, 12, 11, 10, 9, 8. Use 10K Potentiometer at pin 3 of LCD to adjust the contrast of LCD.

## 2.9 ONE CHANNEL 5V RELAY

The 1 Channel 5V Relay Module provides a single relay that can be controlled by any 5V digital output from your microcontroller. The relay is accessible using screw terminals and can handle up to 2A of current. A handy LED indicates the status of the relay. This module provides a standard 3 pin Signal/Voltage/Ground male header and a 4 pin "Grove" connector.



***Fig 1.10 One Relay Module***

The single-channel relay module is much more than just a plain [relay](#), it contains components that make switching and connection easier and act as indicators to show if the module is powered and if the relay is active.

First is the screw terminal block. This is the part of the module that is in contact with mains so a reliable connection is needed. Adding screw terminals makes it easier to connect thick mains cables, which might be difficult to solder directly. The three connections on the terminal block are connected to the normally open, normally closed, and common terminals of the relay.

The second is the relay itself, which, in this case, is a blue plastic case. Lots of information can be gleaned from the markings on the relay itself. The part number of the relay on the bottom says “05VDC”, which means that the relay coil is activated at 5V minimum – any voltage lower than this will not be able to reliably close the contacts of the relay. There are also voltage and current markings, which represent the maximum voltage and current, the relay can switch. For example, the top left marking says “10A 250VAC”, which means the relay can switch a maximum load of 10A when connected to a 250V mains circuit. The bottom left rating says “10A 30VDC”, meaning the relay can switch a maximum current of 10A DC before the contacts get damaged.

The 'relay status LED' turns on whenever the relay is active and provides an indication of current flowing through the relay coil.



The input jumper is used to supply power to the relay coil and LEDs. The jumper also has the input pin, which when pulled high activates the relay.

The switching transistor takes an input that cannot supply enough current to directly drive the relay coil and amplifies it using the supply voltage to drive the relay coil. This way, the input can be driven from a microcontroller or sensor output. The freewheeling diode prevents voltage spikes when the relay is switched off

The power LED is connected to  $V_{CC}$  and turns on whenever the module is powered.

### 2.9.1 SINGLE-CHANNEL RELAY MODULE SPECIFICATIONS

- Supply voltage – 3.75V to 6V
- Quiescent current: 2mA
- Current when the relay is active: ~70mA
- Relay maximum contact voltage – 250VAC or 30VDC
- Relay maximum current – 10A

### 2.10 SERVO MOTOR

A servomotor (or servo motor) is a [rotary actuator](#) or [linear actuator](#) that allows for precise control of angular or linear position, velocity and acceleration.<sup>[1]</sup> It consists of a suitable motor coupled to a sensor for position feedback. It also requires a relatively sophisticated controller, often a dedicated module designed specifically for use with servomotors.





Servomotors are generally used as a high-performance alternative to the [stepper motor](#). Stepper motors have some inherent ability to control position, as they have built-in output steps. This often allows them to be used as an open-loop position control, without any feedback encoder, as their drive signal specifies the number of steps of movement to rotate, but for this the controller needs to 'know' the position of the stepper motor on power up. Therefore, on first power up, the controller will have to activate the stepper motor and turn it to a known position, e.g. until it activates an end limit switch. This can be observed when switching on an [inkjet printer](#); the controller will move the ink jet carrier to the extreme left and right to establish the end positions. A servomotor will immediately turn to whatever angle the controller instructs it to, regardless of the initial position at power up.

The lack of feedback of a stepper motor limits its performance, as the stepper motor can only drive a load that is well within its capacity, otherwise missed steps under load may lead to positioning errors and the system may have to be restarted or recalibrated. The encoder and controller of a servomotor are an additional cost, but they optimise the performance of the overall system (for all of speed, power and accuracy) relative to the capacity of the basic motor. With larger systems, where a powerful motor represents an increasing proportion of the system cost, servomotors have the advantage.

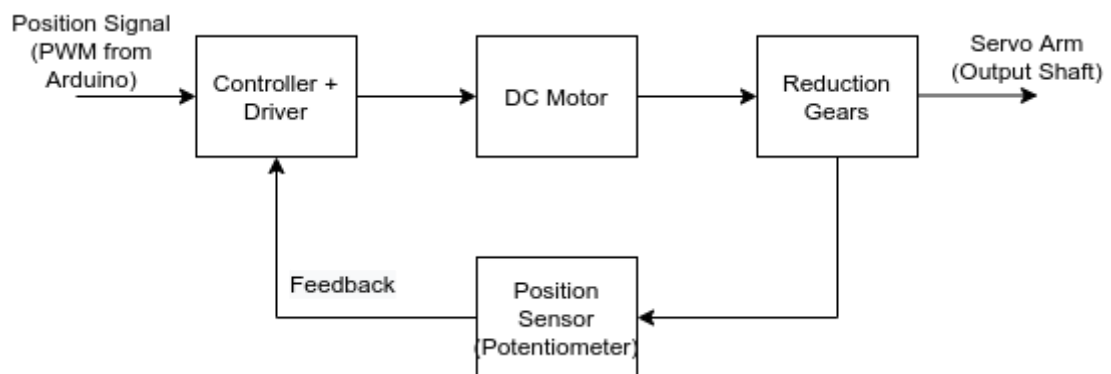
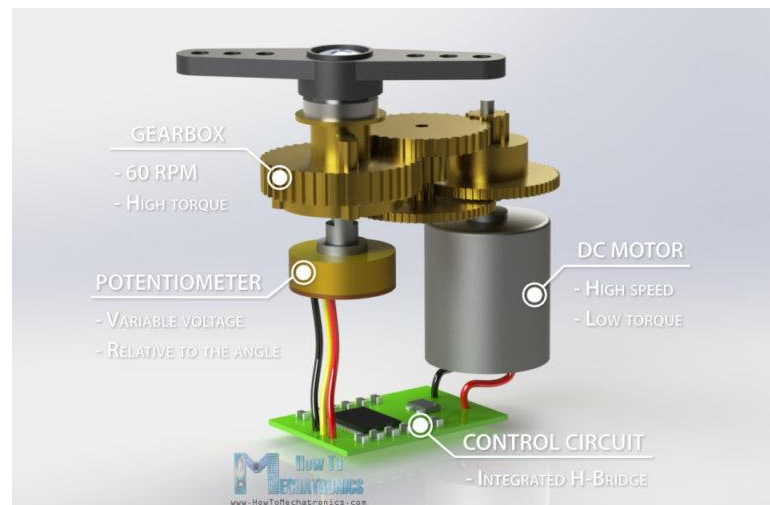
There has been increasing popularity in closed loop stepper motors in recent years. [\[citation needed\]](#) They act like servomotors but have some differences in their software control to get smooth motion. The main benefit of a closed loop stepper motor is its relatively low cost. There is also no need to tune the [PID controller](#) on a closed loop stepper system. [\[5\]](#)

Many applications, such as [laser cutting](#) machines, may be offered in two ranges, the low-priced range using stepper motors.

### 2.10.1 SERVOS

- Standard servos are used to maintain a certain angle, not meant for full rotation
- Uses closed-loop feedback control

- Servos are used in applications requiring high torque, accurate rotation within a limited angle such as Robotic arms, valve control, rudder control etc.



### 2.10.2 SG90 MICRO SERVO



- SG90 is a servo motor which operates based on PWM control signals
- The servo maintains a certain angle (position) based on the width of the pulse fed in through a signal input

- Some technical specifications
- Weight: 9 g
- Dimension: 22.2 x 11.8 x 31 mm approx.
- Stall torque: 1.8 kgf·cm
- Operating speed: 0.1 s/60 degree
- Operating voltage: 4.8 V (~5V)
- PWM frequency = 50Hz
- Pin configuration: Yellow / Light Orange / White (Signal), Red / Dark Orange (+5V), Brown/Black (Ground)

Note : The pulse width in the image does not correspond to SG90, for illustration of the concept only.

Image courtesy: Stefan Tauner

544 – 1500 – 2400 us

0° – 90° – 180°

### 2.10.3 SERVO LIBRARY

- This library allows an Arduino board to control servo motors
- Standard servos allow the shaft to be positioned at various angles, usually between 0° and 180°
- Any digital pin on UNO can be used, not necessarily those supporting PWM. However, note that using Servo library disables analogWrite() functionality on pins 9 and 10
- `attach(int)` - attach a servo to an I/O pin, e.g., `servo.attach(pin)`, `servo.attach(pin, min, max)`
- `servo`: a variable of type Servo, `pin`: pin number, default values: min = 544 us, max = 2400 us

- `write(int)` - write a value to the servo to control its shaft accordingly
- `detach()` - stop an attached Servo from pulsing its I/O pin.

## 2.11 IOT (INTERNET OF THINGS)

The internet of things, or IoT, is a system of interrelated computing devices, mechanical and digital machines, objects, animals or people that are provided with unique identifiers ([UIDs](#)) and the ability to transfer data over a network without requiring human-to-human or human-to-computer interaction.

A [thing](#) in the internet of things can be a person with a heart monitor implant, a farm animal with a [biochip transponder](#), an automobile that has built-in [sensors](#) to alert the driver when tire pressure is low or any other natural or man-made object that can be assigned an Internet Protocol (IP) address and is able to transfer data over a network.

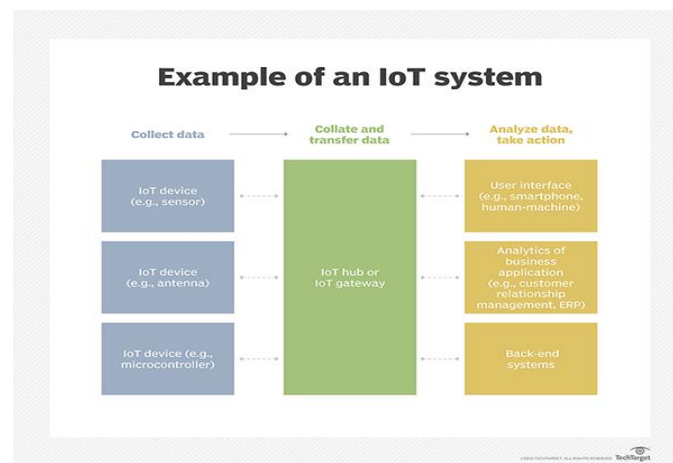
Increasingly, organizations in [a variety of industries are using IoT to operate more efficiently](#), better understand customers to deliver enhanced customer service, improve decision-making and increase the value of the business.

### 2.11.1 HOW IOT WORKS

An IoT ecosystem consists of web-enabled smart devices that use embedded systems, such as processors, sensors and communication hardware, to collect, send and act on data they acquire from their environments. [IoT devices](#) share the sensor data they collect by connecting to an [IoT gateway](#) or other edge device where data is either sent to the cloud to be analyzed or analyzed locally. Sometimes, these devices communicate with other related devices and act on the information they get from one another. The devices do most of the work without human intervention, although people can interact with the devices -- for instance, to set them up, give them instructions or access the data.

The connectivity, networking and communication protocols used with these web-enabled devices largely depend on the specific IoT applications deployed.

IoT can also make use of artificial intelligence (AI) and machine learning to aid in making data collecting processes easier and more dynamic.



## 2.12 ARDUINO SOFTWARE DESCRIPTION:

**Arduino** is an open source hardware and software company project and user community that designs and manufactures single board microcontroller and kits for building digital devices and interactive objects that can sense and control both physically and digitally. Its products are licensed under the [GNU Lesser General Public License](#) (LGPL) or the [GNU General Public License](#) (GPL), emitting the manufacture of Arduino boards and software distribution by anyone. Arduino boards are available commercially in preassembled form or as [do-it-yourself](#) (DIY) kits.

Arduino board designs use a variety of microprocessors and controllers. The boards are equipped with sets of digital and analog [input/output](#) (I/O) pins that may be interfaced to various expansion boards or [breadboards](#) (*shields*) and other circuits. The boards feature serial communications interfaces, including [Universal Serial Bus](#) (USB) on some models, which are also used for loading programs from personal computers. The microcontrollers are typically programmed using a dialect of features from the programming languages [C](#) and [C++](#). In addition to using traditional [compiler toolchains](#), the Arduino project provides an [integrated development environment](#) (IDE) based on the [Processing](#) language project.

A program for Arduino hardware may be written in any [programming](#)

[language](#) with compilers that produce binary machine code for the target processor. Atmel provides a development environment for their 8-bit [AVR](#) and 32-bit [ARM Cortex-M](#) based microcontrollers: AVR Studio (older) and Atmel Studio (newer).

### 2.12.1 IDE:

The Arduino [integrated development environment](#) (IDE) is a [cross-platform](#) application (for [Windows](#), [macOS](#), [Linux](#)) that is written in the programming language [Java](#). It originated from the IDE for the languages [Processing](#) and [Wiring](#). It includes a code editor with features such as text cutting and pasting, searching and replacing text, automatic indenting, [brace matching](#), and [syntax highlighting](#), and provides simple *one-click* mechanisms to compile and upload programs to an Arduino board. It also contains a message area, a text console, a toolbar with buttons for common functions and a hierarchy of operation menus. The source code for the IDE is released under the [GNU General Public License](#), version.

The Arduino IDE supports the languages [C](#) and [C++](#) using special rules of code structuring. The Arduino IDE supplies a [software library](#) from the [Wiring](#) project, which provides many common input and output procedures. User-written code only requires two basic functions, for starting the sketch and the main program loop, that are compiled and linked with a program stub *main()* into an executable [cyclic executive](#) program with the [GNU tool chain](#), also included with the IDE distribution. The Arduino IDE employs the program *avrdude* to convert the executable code into a text file in hexadecimal encoding that is loaded into the Arduino board by a loader program in the board's firmware.

### 2.12.2 CONFIGURING SIMULATOR:

Simulator for Arduino is the most full featured Arduino Simulator available at the present time (watch the latest video below).

The benefits and features of the Arduino Simulator are:

- The ability to teach and demonstrate the inner workings of an Arduino sketch
- Test out a sketch without the hardware, or prior to purchasing hardware

- Debug a sketch
- Demonstrate a project to a potential customer
- Develop a complicated sketch faster than using the hardware

Download the free version below with a short delay timer on loading a sketch, and when ready upgrade to the Pro Version. *Simulator for Arduino* Pro Version is currently used in many countries over six continents. The download consists of a zip file containing a setup.exe file which installs an exe file, help files, images and examples. It is designed for the Arduino Uno, Mega and most other common Arduino boards and does the following:

- Steps through the program line by line. If a new line is selected, the program will continue from that point.
- Performs digitalWrite, digitalRead and pinMode for pins 0-53
- AnalogRead for pins 0-16 and analogWrite for digital pins 0-53
- Emulates Serial, LCD output, Ethernet, Servo, SD card, EEPROM, SoftSerial, SPI, Wire
- If, while, for, switch, do while loop functionality
- Subroutines (multi-level) with arguments
- View variables in real-time
- Step Into, Step Over, Step Out of or Run mode
- Ability to edit sketch or open in Arduino IDE
- Tabs for separate files in the sketch
- Context-sensitive help
- 2 and 4 line LCD support only with improvised CGRAM
- 2 dimensional arrays (without initialisation)
- Breakpoint now with a conditional option
- load custom libraries automatically after setting the Library Directory
- Change the font, size and style of the Simulator
- Advanced watch for easy variable viewing
- Minimize mode for demo/training

- Limited support for custom libraries
- Limited support for pointer and structures

### 2.12.3 COMPILATION:

A number of things have to happen for your Arduino code to get onto the Arduino board. First, the Arduino environment performs some minor pre-processing to turn your sketch into a C++ program. It then gets passed to a compiler (avr-gcc), which turns the human readable code into machine readable instructions (or object files). Then your code gets combined with (linked against), the standard Arduino libraries that provide basic functions like `digitalWrite()` or `Serial.print()`. The result is a single Intel hex file, which contains the specific bytes that need to be written to the program memory of the chip on the Arduino board. This file is then uploaded to the board: transmitted over the USB or serial connection via the bootloader already on the chip or with external programming hardware.

Sketches are compiled by `avr-gcc` and `avr-g++` according to the variables in the `boards.txt` file of the selected board's [platform](#).

The include path includes:

- The board's variant folder (as specified by the `build.variant` property in `boards.txt` for the selected board)
- The core folder (as specified by the `build.core` property in `boards.txt` for the selected board)
- The avr include directory (`hardware/tools/avr/avr/include/`)
- `libraries/{librarydir}` sub-folder of the Arduino IDE installation folder where library lives
- `libraries/{librarydir}` sub-folder of the hardware package of the currently selected board where library lives
- `libraries/{librarydir}` sub-folder of the sketchbook where library lives

The sketch is built in a temporary directory in the system-wide temporary directory



(e.g. /tmp on Linux).

The .c and .cpp files of the target are compiled and output with .o extensions to this directory, as is the main sketch file and any other .c or .cpp files in the sketch and any .c or .cpp files in any libraries which are #included in the sketch.

Before compiling each .c or .cpp file, an attempt is made to reuse the previously compiled .o file, which speeds up the build process. A special .d (dependency) file provides a list of all other files included by the source. The compile step is skipped if the .o and .d files exist and have timestamps newer than the source and all the dependent files. If the source or any dependent file has been modified, or any error occurs verifying the files, the compiler is run normally, writing a new .o & .d file. After a new board is selected from the Tools menu, all .c and .cpp files are rebuilt on the next compile.

These .o files are then linked together into a static library and the main sketch file is linked against this library. Only the parts of the library needed for your sketch are included in the final .hex file, reducing the size of most sketches.

The .hex file is the final output of the compilation which is then uploaded to the board.

If verbose output during compilation is checked in the Preferences dialog, the complete command line of each external command executed as part of the build process will be printed in the editor console.

#### **2.12.4 DEBUG:**

On Arduino, once you run your program you are unable to see what's happening inside your device and how your code is running. The only assistance you're given comes in the form of messages on your serial monitor or by LED display. Especially when other IDEs usually have debugging tools that utilize breakpoints, steps, call stack, watch, local/global variables and other components to double check through code.

However, debugging Arduino code isn't the same as debugging code on a PC. The two are very different in terms of how they are approached. The main reason for this is that Arduino code is usually used to control physical outputs or receiving physical inputs to/from the real world and the debugging process has to take those into account.

In practice, this means that Arduino developers often have to explore alternative methods and tools to debug their code. In this post, we'll break down everything from Arduino code debugging to using a simulator to debug Arduino. This will give you all the information you need to produce code that works.

### ➤ **Step 1: Check your hardware**

As mentioned above, if your circuito.io code doesn't work, chances are you have a problem with your hardware. This means you need to begin Hardware debugging.

Here are some basic troubleshooting steps for hardware issues:

- 1. Check wiring** - One of the first things you should do when you wire your circuit is to check that you have connected everything correctly. If you've put the jumper wires in the wrong pin of the Arduino, or the wrong placement on the breadboard, your program will not be able to interact with the component. Double-check all the components and wires to make sure everything is securely in place. If you have a complex circuit, sometimes it's just better to start from scratch, testing each component at a time to eliminate the problem.
- 2. Check Soldering** - If some of your parts required soldering during setup, make sure that you soldered them properly. Poorly soldered parts are one of the most common causes of technical failure. If you can't tell, resolder it.
- 3. Check Power Supply** - If there's still a problem with your circuit, you need to check the [power supply](#). Using a multimeter, you can check 2 things:

### ➤ **Step 2: Code debugging**

If your test code works, it's time to write your own code. When writing your own Arduino code, there are a number of basic rules and best practices you should follow. By

following the instructions below, you'll have a much easier time if you need to debug it later on:

## 2.13 BLYNK APP

Blynk is an Internet of Things Platform aimed to simplify building mobile and web applications for the Internet of Things. Easily connect 400+ hardware models like Arduino, ESP8266, ESP32, Raspberry Pi and similar MCUs and drag-n-drop IOT mobile apps for iOS and Android in 5 minutes



Blynk is a Platform with IOS and Android apps to control Arduino, Raspberry Pi and the likes over the Internet. It's a digital dashboard where you can build a graphic interface for your project by simply dragging and dropping widgets.

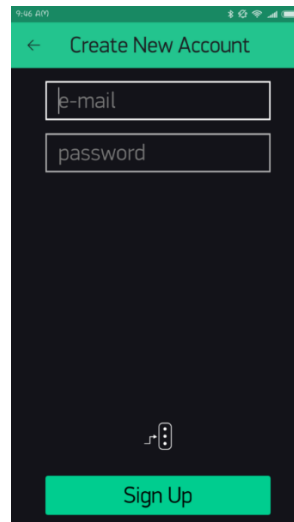
Blynk was designed for the Internet of Things. It can control hardware remotely, it can display sensor data, it can store data, visualize it and do many other cool things.

There are three major components in the platform:

- **Blynk App** - allows to you create amazing interfaces for your projects using various widgets we provide.
- **Blynk Server** - responsible for all the communications between the smartphone and hardware. You can use our Blynk Cloud or run your [private Blynk server](#)

locally. It's open-source, could easily handle thousands of devices and can even be launched on a Raspberry Pi.

- **Blynk Libraries** - for all the popular hardware platforms - enable communication with the server and process all the incoming and outgoing commands

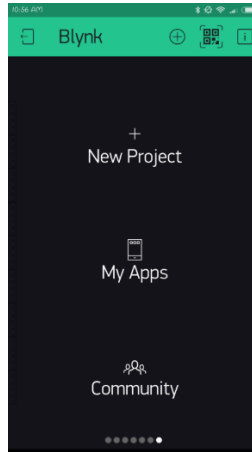


You'll also need to install the **Blynk Arduino Library**, which helps generate the firmware running on your ESP8266. Download the latest release from <https://github.com/blynkkk/blynk-library/releases>, and follow along with the directions there to install the required libraries.

### 2.13.1 CREATE A BLYNK PROJECT

Click the “Create New Project” in the app to create a new Blynk app. Give it any name. Blynk works with hundreds of hardware models and connection types. Select the Hardware type. After this, select connection type. In this project we have select WiFi connectivity

The **Auth Token** is very important – you'll need to stick it into your ESP8266's firmware. For now, copy it down or use the “E-mail” button to send it to yourself.



### 2.13.2 ADD WIDGETS TO THE PROJECT

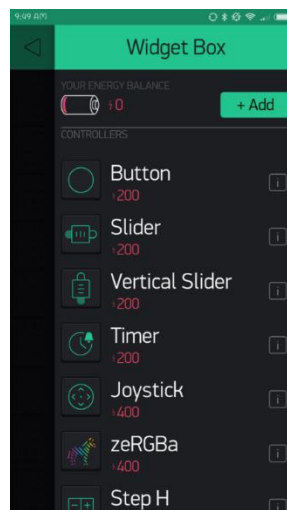
Then you'll be presented with a blank new project. To open the widget box, click in the project window to open.

We are selecting a button to control Led connected with NodeMCU.

1. Click on Button.
2. Give name to Button say led.
3. Under OUTPUT tab- Click pin and select the pin to which led is connected to NodeMCU, here it is digital pin 2, hence select digital and under pin D2. And Click continue.

Under MODE tab- Select whether you want this button as “push button” or “Switch”.

You have successfully created a GUI for Arduino.



## 3.CODE

### 3.1 ARDUINO CODE

```
#include <LiquidCrystal_I2C.h>

#include "MAX30100_PulseOximeter.h"

#include <Servo.h>

Servo servo;

LiquidCrystal_I2C lcd(0x27,16,2);

#define REPORTING_PERIOD_MS    1000

#define ONE_WIRE_BUS A0

OneWire oneWire(ONE_WIRE_BUS);

DallasTemperature sensors(&oneWire);

PulseOximeter pox;

uint32_t tsLastReport = 0;

float t=0;

int sp = 0;

int hb;

int ir,i=0;

const int IRR=12;

void onBeatDetected()

{

    Serial.println("Beat!");

}

void setup()

{
```

```

Serial.begin(9600);

servo.attach(7);

servo.write(90);

lcd.init();

lcd.backlight();

lcd.clear();

lcd.setCursor(0,0);

lcd.print("Covid Door Lock ");

lcd.setCursor(0,1);

lcd.print("  Welcome  ");

pinMode(IRR,INPUT);

pinMode(8,OUTPUT);

digitalWrite(8,HIGH);

pox.setOnBeatDetectedCallback(onBeatDetected);

pox.begin();

pox.setIRLedCurrent(MAX30100_LED_CURR_7_6MA);
}

void loop()
{
  lcd.clear();

  lcd.setCursor(0,0);

  lcd.print("Covid Door Lock ");

  lcd.setCursor(0,1);

  lcd.print("  Welcome  ");

```

```

delay(1000);

ir=digitalRead(IRR);

if(ir == 0)
{
    digitalWrite(8,LOW);

    lcd.clear();

    lcd.setCursor(0,0);

    lcd.print(" Please Place ");

    lcd.setCursor(0,1);

    lcd.print("Finger On Scanner");

    while(i <= 8)
    {
        pox.update();

        if (millis() - tsLastReport > REPORTING_PERIOD_MS)
        {
            hb = pox.getHeartRate();

            sp = pox.getSpO2();

            Serial.print("Heart rate:");

            Serial.print(hb);

            Serial.print("bpm / SpO2:");

            Serial.print(sp);

            Serial.println("%");

            tsLastReport = millis();

            if(sp >0 && hb > 0)

```



```

    {
        i = i+1;

        lcd.clear();

        lcd.setCursor(0,0);

        lcd.print("Scanning.....");

    }

}

}

while(i > 8 && i < 15)

{

    sensors.requestTemperatures();

    t = sensors.getTempCByIndex(0);

    t=(t* 9.0) / 5.0 + 32.0;

    Serial.println("Temperature is: ");

    Serial.println(t);

    i = i+1;

    delay(1000);

}

lcd.clear();

lcd.setCursor(0,0);

lcd.print("HB:    SP2:  ");

lcd.setCursor(3,0);

lcd.print(hb);

lcd.setCursor(13,0);

```

```

lcd.print(sp);

lcd.setCursor(0,1);

lcd.print("Temp:   'F");

lcd.setCursor(6,1);

lcd.print(t);

delay(5000);

if(t<100 && sp > 94)
{
    Serial.println("Granted");

    lcd.clear();

    lcd.setCursor(0,0);

    lcd.print(" Access Granted ");

    lcd.print("!!Door Opened!!!");

    servo.write(0);

    delay(2000);

    servo.write(90);
}
else
{
    Serial.println("Denied");

    lcd.clear();

    lcd.setCursor(0,0);

    lcd.print("!Access Denied!!");

    lcd.setCursor(0,1);

```

```

    lcd.print("!!Door Closed!!!");

    servo.write(90);

    delay(5000);

}

if(i==15)

{

    i=0;

    hb = 0;

    sp = 0;

}

}

else

{

    digitalWrite(8,HIGH);

}

}

```

### **3.2 NODEMCU CODE**

```

#define BLYNK_TEMPLATE_ID "TMPLnYieLJ3M"

#define BLYNK_DEVICE_NAME "covid_detect"

#define BLYNK_FIRMWARE_VERSION    "0.1.0"

#define BLYNK_PRINT Serial

#define APP_DEBUG

#include "BlynkEdgent.h"

```

```

float temp;

int a,count,sp,hb;

WidgetLED led(V1);

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

    BlynkEdgent.begin();

    pinMode(D0,INPUT);
}

void loop()
{
    if (Serial.available())
    {
        DeserializationError err = deserializeJson(doc, Serial);
        if (err == DeserializationError::Ok)
        {
            Serial.print("Heartbeat = ");

            hb = doc["count1"].as<int>();

            Serial.println(hb);

            Serial.print("SpO2 = ");

            sp = doc["count2"].as<int>();

            Serial.println(sp);

            Serial.print("Temp = ");

            temp = doc["count3"].as<float>();

```

```

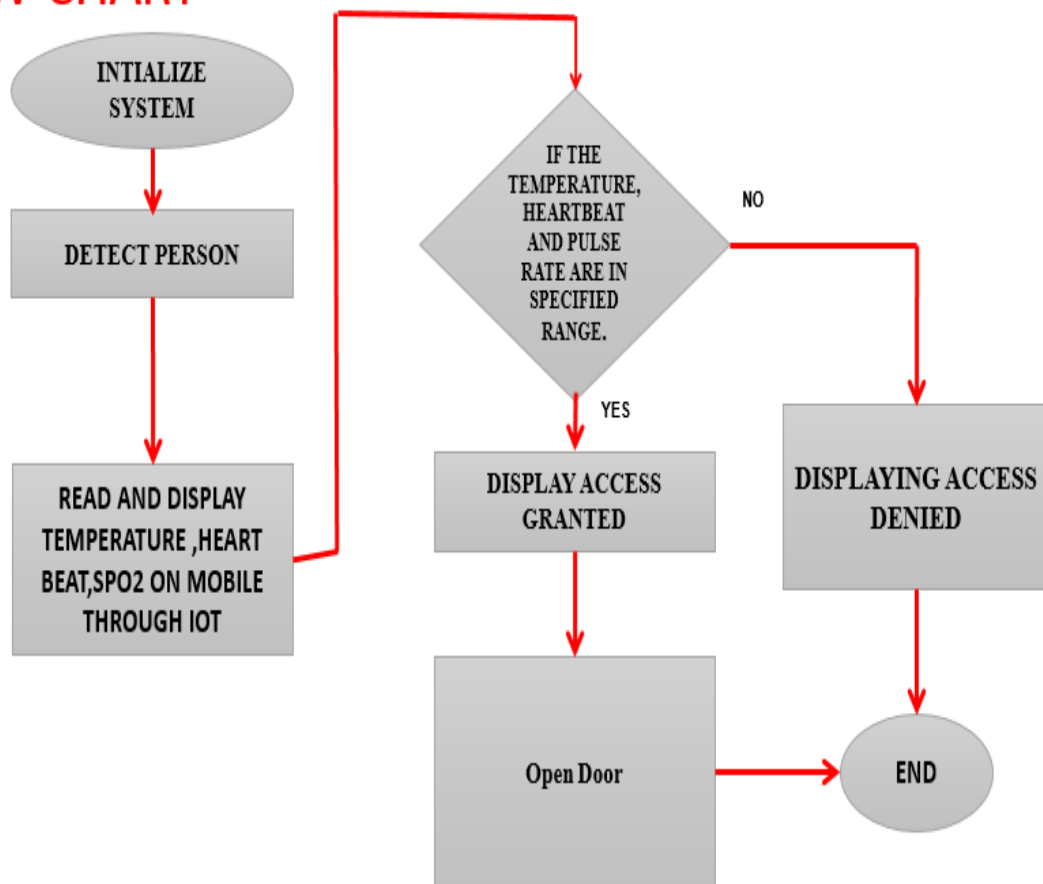
    Serial.println(temp);
}
else
{
    Serial.print("deserializeJson() returned ");
    Serial.println(err.c_str());
    while (Serial.available() > 0)
        Serial.read();
}
}
if(temp>0)
{
    Blynk.virtualWrite(V6, sp);
    Blynk.virtualWrite(V7, hb);
    Blynk.virtualWrite(V8, temp);
}
a=digitalRead(D0);
if(a == 0)
{
    led.on();
}
else
{
    led.off();
}

```

```
}  
  
BlynkEdgent.run();  
  
}
```

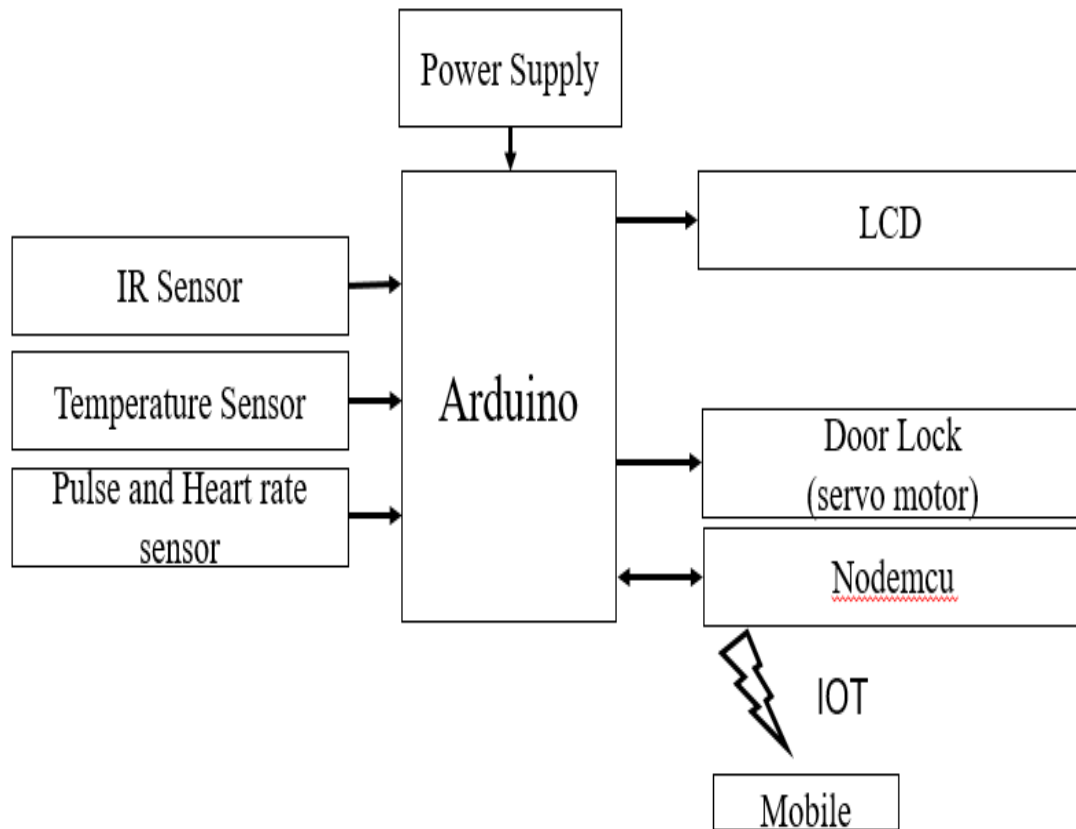
## 4.FLOW CHART

### FLOW CHART



## 5.BLOCK DIAGRAM

### BLOCK DIAGRAM:



---

## CONCLUSION

Intended objectives were successfully achieved in the prototype model developed. The developed product is easy to use, economical and does not require any special training. In conclusion we can say that by implementing this project we can identify the COVID suspected and stop entering those people into the campus (corporate companies) using smart monitoring systems. By using camera we can also check whether the person is wearing mask or not and we can intimate him using LCD display.

The COVID-19 pandemic has resulted in a global health crisis as thousands of people die from the disease every day. The fatality rate can be minimized if proper treatment is administered at the right time. Various steps, including regular monitoring of pulse rate, SpO2 level, and temperature, have been taken to ensure proper treatment. Considering the abovementioned facts, an IoT-based smart health monitoring system was developed

for COVID-19 patients. The system runs through an IoT-based mobile application, and both the doctor and the patient can receive alerts from this system during emergencies. Therefore, individuals can use this system effectively anywhere. Advanced features can be added in the future because the entire system is IoT-based.

## **FUTURE SCOPE**

The Future Scope of the Project we will introduce iot based covid-19 suspect smart entrance monitoring system will reduce the man power and human error . The system is cost-effective, noninvasive, and versatile in nature, which makes it easier to screen patients' well-being regardless of where they are. Additionally, it provides real-time alerts to concerned individuals and medical experts about any circumstance that requires prompt consideration.

## **REFERENCES**

- 1.L. Garg, E. Chukwu, N. Nasser, C. Chakraborty, and G. Garg “Anonymity Preserving Io -Based COVID-19 and Other Infectious Disease Contact racing Model,”, IEEE Access, vol. , pp. 2-159414,31August2020.DOI: 10.1109/ACCESS.2020.3020513
2. S. Pandya, A. Sur, K. Kotec a, “Smart epidemic tunnel: Io -based sensor- fusion assistive technology for COVID- 19 disinfection” International Journal of Pervasive Computing and Communications, DOI: 10.1108/IJPCC07-2020-0091.
3. M. N. Mohammed, H. Syamsudin, S. Al-Zubaidi, Sairah A.K, R. Ramli, E Yusuf, “Novel COVID-19 Detection and Diagnosis System Using IoT Based Smart Helmet, “International Journal of Psychological Rehabilitation,Vol.24, DOI:10.3720/IJPR/V2417/PR270221
4. M. Nasajpour, S. Pouriyeh, R M. Parizi, M. Dorodchi, M. Valero, H. R Arabnia, “Internet of Things for Current COVID-19 and Future Pandemics: An Exploratory Study.”, Journal of Health care Informatics , 325–364 (2020), Springer, vol. 4, DOI: <https://doi.org/10.1007/s41666-020-00080-6>



5. Pras ant Agarwal, Ravi Kumar G and Pooja Agarwal “Io based Framework for Smart Campus: COVIDReadiness”, World Conference on Smart Trends in Systems, Security and Sustainability (WorldS4), IEEE, London, UK; 27-28 July 2020, DOI : 10.1109/WorldS450073.2020.9210382

6. Imam Hossain, Dipankar Das, Md. Golam Ras ed, “Internet of Things based model for smart campus: challenges and limitations” 2 International Conference on Computer, Communication, Chemical, Materials, and Electronic Engineering (IC4ME2), IEEE, Rajshahi, Bangladesh, 11-12 July, 2019. DOI: 10.1109/IC4ME247184.2019.9036629.