AN AI BASED VENTILATOR KPI USING EMBEDDED IOT DEVICE.

Roopa.S, Sneha.S, Priyanka.B, Soundarya.A, Aishwarya.A

[1] Roopa.S Asst prof, Electronics and Communication, SJM Institute of Technology, Karnataka, India
[2] Sneha.S Student, Electronics and Communication, SJM Institute of Technology, Karnataka, India
[3] Priyanka.B Student, Electronics and Communication, SJM Institute of Technology, Karnataka, India
[4] Soundarya.A Student, Electronics and Communication, SJM Institute of Technology, Karnataka, India
Aishwarya.A Student, Electronics and Communication, SJM Institute of Technology, Karnataka, India

ABSTRACT

The IoT-Based Smart Ventilator & Patient Monitoring System is a cost-effective and efficient alternative to conventional ventilators, designed to assist patients with reduced lung capacity, such as those affected during the COVID-19 pandemic. The system is built using Arduino for real-time ventilator control and vital signs monitoring, along with NodeMCU for IoT-based Telegram notifications. Sensors including the MAX30100 pulse oximeter, heartbeat sensor, and temperature sensor are used to measure SpO_2, heart rate, and body temperature, while an H-bridge-driven DC motor automates the breathing cycle by compressing an Ambu bag. The ventilator's breathing rate and inhalation-to-exhalation ratio are adjustable, ensuring adaptable respiratory support. Critical alerts are sent instantly to caregivers via Telegram in case of abnormal vitals or system faults. This portable, low-cost solution is suitable for emergency use, resource-limited healthcare facilities, and as a backup system during high demand situations.

Keyword : - *IoT-based ventilator, patient monitoring, Arduino Uno, NodeMCU ESP8266, MAX30100 sensor, Spo2 measurement, heart rate monitoring, temperature sensing, DC motor control, H-bridge driver, Ambu bag automation, real-time alerts.*

1. INTRODUCTION

A ventilator is a life-support device that assists or replaces spontaneous breathing by mechanically moving breathable air into and out of the lungs. Traditionally, ventilators are considered a last-resort medical tool used in critical care. However, during the COVID-19 pandemic, the demand for ventilators surged worldwide, creating shortages, especially in low-resource regions. This highlighted the urgent need for affordable, portable, and easy-to-build ventilator solutions that could be rapidly deployed during health crises.

The IoT-Based Smart Ventilator & Patient Monitoring System addresses this need by combining mechanical ventilation with continuous vital signs monitoring and IoT-enabled alerts. Using Arduino as the control unit, the system operates a DC motor through an H-bridge to automate the breathing cycle. Sensors such as the MAX30100 pulse oximeter, heartbeat sensor, and temperature sensor measure essential patient parameters, while a 16×2 LCD provides real-time local display. The NodeMCU module connects the device to the internet, sending instant notifications via Telegram to healthcare providers or caregivers when abnormal conditions or system faults are detected.

This design ensures not only respiratory assistance but also integrated patient health monitoring in a single, compact unit. The low-cost, open-source nature of the system makes it suitable for use in rural clinics, emergency medical response, and as a backup solution during times of high demand.

Respiratory diseases and injury-induced respiratory failure constitute a major public health problem in both developed and less developed countries. Asthma, chronic obstructive pulmonary disease and other chronic respiratory conditions are widespread. These conditions are exacerbated by air pollution, smoking, and burning of biomass for fuel. Underdeveloped and even the developed countries, considering the current global healthcare situation, do not have enough medical respiration systems. People nowadays face numerous respiratory issues, which may arise from various sources but certainly, they pose a great challenge for public health. Patients with underlying lung disease may develop respiratory failure under a variety of challenges and can be supported mechanical ventilation. These are machines which mechanically assist patients inspire and exhale, allowing the exchange of oxygen and carbon dioxide to occur in the lungs. a low-cost ventilator has been prototyped using the AMBU (Artificial Manual Breathing Unit) bag compression, which has the potential to be a solution for this problem of lack of ventilators. In order to make the system viable for patients of all ages and types, the designed system comes with the option of adjusting BPM (Breaths per Minute) of patients. The professionals operating this ventilator will be able to regulate the air volume so that only an adequate amount of air should be delivered to the patients without damaging their lug's tissues. Moreover, the inspiration to expiration ratio will also become a controllable factor. Similarly, Peep rate could be maintained at any desired level. The system is made so to provide two main modes of ventilation one being mandatory ventilation and the other is assisted ventilation to suit every kind of situation. Moreover, pressure and time-based triggers have been used in this prototype. The prototyping price for this device is lesser than its contemporaries.

Hence, this ventilator system is innovative, affordable and fully functional. This makes it a product which has all the potential to greatly reduce the pressure on the healthcare systems all over the world.
While there are enough ventilators for regular use, there is a lack of preparedness for cases of mass casualty such as influenza pandemics, natural disasters and massive toxic chemical releases. A part from that, if the person operating the mechanism lacks proper training, he could harm the lungs of the patient by improper compression. Hence, we came up with a device, which could trigger the AMBU bag. Therefore, a very affordable machine has been prototyped which could provide optimal functionality for mechanical ventilators.

1.1 Problem Statement:

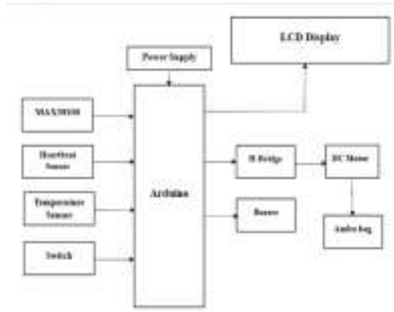
During medical emergencies such as the COVID-19 pandemic, the global healthcare system faced a severe shortage of ventilators. Conventional ventilators are expensive, complex, and require specialized maintenance, making them inaccessible in many rural or resource-limited healthcare facilities. Additionally, most ventilators focus solely on respiratory assistance without integrating continuous patient vital monitoring or remote alert mechanisms. This creates a gap in providing timely intervention when a patient's condition deteriorates, especially in situations where medical staff cannot be physically present at all times.
There is a need for a low-cost, portable, and easy-to-operate ventilator system that not only automates the breathing process but also continuously monitors key vital parameters such as oxygen saturation, heart rate, and body temperature. Furthermore, the system must be capable of sending real-time alerts to caregivers or healthcare professionals through IoT-based communication, ensuring immediate response in critical situations.

1.2 Objectives:

Create a low-cost and portable mechanical ventilator system that can be rapidly deployed in emergency situations. Use a DC motor with H-bridge control to regulate inhalation and exhalation cycles, with adjustable breathing rate (BPM) and I:E ratio to suit patient needs . Continuously measure vital parameters such as SpO_2, heart rate, and body temperature using sensors like the MAX30100, heartbeat sensor, and temperature sensor . Employ NodeMCU for internet connectivity to send immediate Telegram notifications to healthcare professionals or caregivers when abnormal vitals or faults are detected . Display real-time ventilator status and patient vitals on an LCD screen for on-site monitoring . Incorporate fault detection, emergency alerts, and safe motor control to prevent harm to the patient in case of malfunction.

2. METHODOLOGY

System Initialization: power is supplied to the Arduino-based ventilator system. Arduino initializes all sensors (MAX30100, heartbeat sensor, temperature sensor), LCD, DC motor, and H-bridge driver. NodeMCU connects to Wi-Fi for IoT communication.



Block Diagram

Vital Sign Monitoring: MAX30100 measures SpO_2 and heart rate. Heartbeat sensor provides backup pulse detection. Temperature sensor continuously measures patient body temperature. Arduino processes values and displays them on the LCD in real time.
Ventilation Control: Arduino controls the DC motor via L293D H-bridge. Motor compresses the Ambu bag during inhalation and releases during exhalation. Breathing parameters such as BPM (breaths per minute) and I:E ratio are pre-set and adjustable. Ventilation cycle repeats continuously.
Condition Analysis: Measured vitals are compared with preset safety limits: SpO_2, Abnormal heart rate, High temperature, Motor/sensor malfunction.
Alert Generation: For any abnormal or critical condition: Arduino sends alert data to NodeMCU via serial communication. NodeMCU sends real-time alerts to caregivers through Telegram. A local buzzer automatically activates.
Continuous Loop Operation: System continuously performs ventilation + monitoring + alerting. Runs until manually switched off or power is disconnected.

2.1 Working:

The IoT-Based Smart Ventilator & Patient Monitoring System operates by combining mechanical ventilation control with continuous vital signs monitoring and IoT-enabled alert notifications.
Power-Up and Initialization: When the system is powered on, the Arduino initializes the LCD, sensors (MAX30100 pulse oximeter, heartbeat sensor, temperature sensor), and motor control circuit via the H-bridge. The NodeMCU connects to a predefined Wi-Fi network to enable internet-based alerts through Telegram.
Patient Vital Monitoring: The MAX30100measures SpO2 (oxygen saturation) and heart rate. The Heartbeat sensor acts as an additional or backup pulse detection method. The Temperature sensor continuously measures the patient's body temperature. These values are displayed in real-time on the LCD screen.
Ventilation Control: The Arduino controls the DC motor through the H-bridge to automate inhalation and exhalation cycles. The breathing rate (BPM) and inhalation-to-exhalation (I:E) ratio are set in the code and can be adjusted for patient needs. During inhalation, the motor compresses an Ambu bag or blower system to deliver air into the patient's lungs. During exhalation, the motor either reverses or stops to allow the lungs to deflate naturally.
Abnormal Condition Detection: The Arduino continuously compares the sensor readings with preset safety thresholds: SpO_2 less than the safe limit (e.g., 92%), Heart rate outside the normal range, High body temperature indicating possible fever, Motor or sensor malfunction detection.

2.2 Scope:

The IoT-Based Smart Ventilator & Patient Monitoring System is designed to provide a low-cost, portable, and efficient respiratory support solution integrated with real-time health monitoring. The scope of the project covers the following aspects:

- Healthcare Applications – Suitable for emergency use during pandemics, disaster response, rural healthcare centers, and as a backup ventilator in hospitals.

- Integrated Monitoring – Continuous tracking of SpO_2, heart rate, and body temperature to enable early detection of patient deterioration.
- Remote Accessibility – IoT-based connectivity through NodeMCU allows caregivers and healthcare professionals to receive instant Telegram alerts from anywhere.
- Customizable Operation – Adjustable breathing parameters such as BPM and I:E ratio to match patient-specific needs.
- Educational & Research Use – Can serve as a training and prototyping platform for biomedical engineering students and researchers.
- Scalability – Potential for further development into a fully automated, closed-loop ventilator system with pressure/volume feedback control and AI-based health prediction.

3. HARDWARE AND SOFTWARE REQUIREMENTS

Functional Requirements:
These describe what the system must do:

1. Ventilation Control
   - Automate inhalation and exhalation cycles using a DC motor controlled via an H-bridge.
   - Allow adjustment of breathing rate (BPM) and inhalation-to-exhalation ratio.
2. Vital Signs Monitoring
   - Measure oxygen saturation (SpO_2) and heart rate using the MAX30100 sensor.
   - Monitor body temperature using a temperature sensor.
   - Provide backup heart rate monitoring using a separate heartbeat sensor.
3. Data Display
   - Display real-time patient vitals and ventilator status on a 16×2 LCD screen.
4. Alert System
   - Trigger a buzzer alarm when vital parameters go beyond safe limits.
   - Send instant IoT-based alerts to caregivers via Telegram using NodeMCU.
5. Data Communication
   - Transfer vital sign readings from Arduino to NodeMCU through serial communication.
   - Format and send alerts in a readable form to a pre-configured Telegram account.
6. Fault Detection
   - Detect motor or sensor malfunctions and trigger both local and remote alerts.

Non-Functional Requirements
These describe how the system should perform:

1. Reliability
   - Ensure continuous operation without frequent resets or hardware failures.
   - Maintain accuracy in vital sign measurements over extended periods.
2. Performance
   - The system should refresh and update patient vitals on the LCD at least twice per second.
   - IoT notifications should be sent within 3–5 seconds of detecting an abnormal condition.
3. Usability
   - Provide an intuitive LCD display for easy reading by medical staff.
   - Operate with minimal configuration for quick deployment in emergencies.
4. Scalability
   - Allow future addition of more sensors or a web dashboard without major redesign.
5. Portability
   - The system should be compact and lightweight for easy transportation.
6. Safety
   - Include mechanical and software safeguards to prevent over-compression of the Ambu bag.
   - Fail-safe mode in case of power or sensor failure, with an immediate alert.
7. Cost-Effectiveness
   - Use low-cost, easily available components to make the solution affordable for rural healthcare setups.

Hardware Requirements:

- Arduino

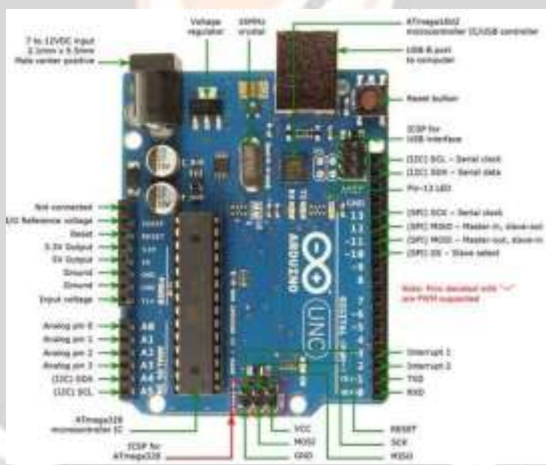
- LCD Display
- MAX30100 Sensor
- Temperature Sensor
- Heartbeat Sensor
- H-Bridge
- DC Motor
- Emergency Switch
- Power Supply

Software Requirements:

- Arduino IDE
- Embedded C

Arduino Uno:

Arduino Uno is a microcontroller board based on the ATmega328P. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz quartz crystal, a USB connection, a power jack, an ICSP header and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with an AC-to-DC adapter or battery to get started. You can tinker with your UNO without worrying too much about doing something wrong, worst case scenario you can replace the chip for a few dollars and start over again. "Uno" means one in Italian and was chosen to mark the release of Arduino Software (IDE) 1.0. The Uno board and version 1.0 of Arduino Software (IDE) were the reference versions of Arduino , now evolved to newer releases. The Uno board is the first in a series of USB Arduino boards, and the reference model for the Arduino platform; for an extensive list of current, past or outdated boards see the Arduino index of boards.[1]



Pin Specification

3.1 Software Requirements:

Arduino IDE

A program for Arduino may be written in any programming language for a compiler that produces binary machine code for the target processor. Atmel provides a development environment for their microcontrollers, AVR Studio and the newer Atmel Studio.

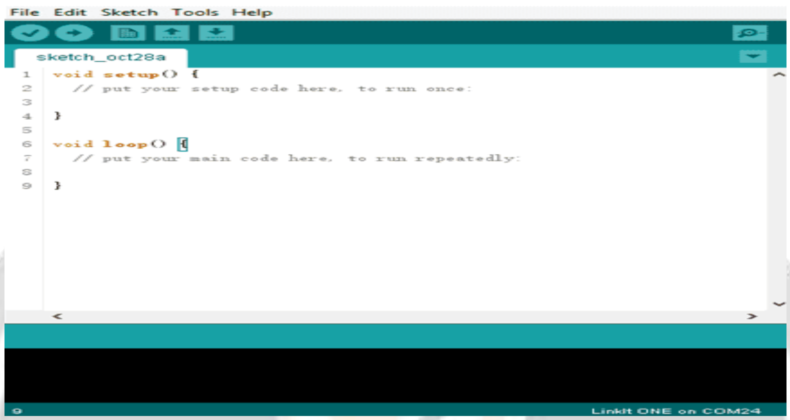
The Arduino project provides the Arduino integrated development environment (IDE), which is a cross-platform application written in the programming language Java. It originated from the IDE for the languages Processing and Wiring. It includes a code editor with features such as text cutting and pasting, 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.

A program written with the IDE for Arduino is called a sketch. Sketches are saved on the development computer as text files with the file extension .ino. Arduino Software (IDE) pre-1.0 saved sketches with the extension .pde.

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 toolchain, also included with the IDE distribution.

A minimal Arduino C/C++ sketch, as seen by the Arduino IDE programmer, consist of only two functions:

- setup(): This function is called once when a sketch starts after power-up or reset. It is used to initialize variables, input and output pin modes, and other libraries needed in the sketch.
- loop(): After setup() has been called, function loop() is executed repeatedly in the main program. It controls the board until the board is powered off or is reset.



An Arduino Sketch

Embedded C:

- When designing software for a smaller embedded system with the 8051, it is very common place to develop the entire product using assembly code. With many projects, this is a feasible approach since the amount of code that must be generated is typically less than 8 kilobytes and is relatively simple in nature. If a hardware engineer is tasked with designing both the hardware and the software, he or she will frequently be tempted to write the software in assembly language.
- The trouble with projects done with assembly code can is that they can be difficult to read and maintain, especially if they are not well commented. Additionally, the amount of code reusable from a typical assembly language project is usually very low. Use of a higher-level language like C can directly address these issues. A program written in C is easier to read than an assembly program.
- Since a C program possesses greater structure, it is easier to understand and maintain. Because of its modularity, a C program can better lend itself to reuse of code from project to project. The division of code into functions will force better structure of the software and lead to functions that can be taken from one project and used in another, thus reducing overall development time. A high order language such as C allows a developer to write code, which resembles a human's thought process more closely than does the equivalent assembly code. [25]The developer can focus more time on designing the algorithms of the system rather than having to concentrate on their individual implementation. This will greatly reduce development time and lower debugging time since the code is more understandable.
- By using a language like C, the programmer does not have to be intimately familiar with the architecture of the processor. This means that someone new to a given processor can get a project up and running quicker, since the internals and organization of the target processor do not have to be learned. Additionally, code developed in C will be more portable to other systems than code developed in assembly. Many target processors have C compilers available, which support ANSI C.
- All of this is not to say that assembly language does not have its place. In fact, many embedded systems (particularly real time systems) have a combination of C and assembly code. For time critical operations, assembly code is frequently the only way to go. One of the great things about the C language is that it allows

you to perform low-level manipulations of the hardware if need be, yet provides you the functionality and abstraction of a higher order language.

4. CONCLUSIONS

The Smart Ambulance Detection System using ESP32 successfully integrates real-time communication between the ambulance, traffic signal module, and web application to ensure faster and safer passage for emergency vehicles. By identifying the road through a 4×1 switch and transmitting the data via Zigbee, the system efficiently clears the respective traffic signal and updates live ambulance location and status on the web platform.
This intelligent approach reduces delay at intersections, minimizes manual intervention, and enhances the overall emergency response process. The system's modular design allows for easy expansion to more intersections and integration with existing smart city infrastructure, making it a cost-effective and scalable solution for improving urban healthcare emergency services.

5. REFERENCES

[1]. Reference 1: S.K. Powel son, Abdul Mohsen Al Husseini, HeonJu Lee, Justin Negrete, and Amelia Servi, "Design and prototyping of a low-cost portable mechanical ventilator,", Proceedings of the Design of Medical Devices Conference, Minneapolis, MN. (2010)
[2]. Reference 2: mCourses.lumenlearning.com.mBreathing,| BoundlessmBiology.v[online]nAvailablemat: https://courses.lumenlearning.com/boundlessbiology/chapter/breathing/ [Accessed,18th, Nov. 2018].
[3]. Reference 3: Kerechanin CW, Cytcgusm PN, Vincent JA, Smith DG, Wenstrand DS Development of Field Portable Ventilator Systems for Domestic and Military Emergency Medical Response, John Hopkins Apl. Tech. Digest Vol 25, Number 3. (2004)
[4]. Reference 4: Chan-Yeung M, Aït-Khaled N, White N, Ip MS, Tan WC "The burden and impact of COPD in Asia & Africa." Int J Tuberc Lung Dis. (2004)
[5]. Reference 5: ArchitaAgnihotri,"Human Body Respiration Measurement Using Digital Temperature Sensor with I2c Interface" International Journal of Scientific and Research Publications, Volume 3, Issue 3,
(2013)
[6]. Reference 6: nJ.S.A.Mary,kR.Divya,nMonitoring the Human Respiratory using Temperature Sensor, International Journal of Science and Engineering Application's", Volume 2, Issue 5, (2013)
[7]. Reference 7: Aït-Khaled N, Enarson D, Bousquet J (2001) Chronic respiratory diseases in developing countries: the burden and strategies for prevention and management, Bulletin of the World Health Organization, 79 (10)
[8]. Reference 8: Chan-Yeung M, Aït-Khaled N, White N, Ip MS, Tan WC (2004) The burden and impact of COPD in Asia & Africa. Int J Tuberc Lung Dis.
[9]. Reference 9: Webster's New World™ Medical Dictionary First, Second and Third Editions (May, 2008)
John Wiley & Sons, Inc.