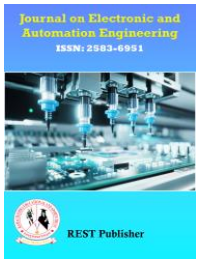
Journal on Electronic and Automation Engineering

Vol: 4(2), June 2025

REST Publisher; ISSN: 2583-6951 (Online)

Website: https://restpublisher.com/journals/jeae/

DOI: https://doi.org/10.46632/jeae/4/2/51



IoT Based Air Quality Monitoring System

***B. Satish Chandra, K. Rajakumar, K. Satish Nayak, K. Ajay**

Nalla Malla Reddy Engineering College, Hyderabad, Telangana, India.

*Corresponding Author Email: satishchandra.ece@nmrec.edu.in

Abstract: *Air pollution poses a serious threat to urban health and safety. This paper presents an IoT-based dynamic air quality monitoring system designed for real-time tracking of pollutants such as PM2.5, PM10, CO, NO2, SO2, and O3, along with temperature and humidity. The system uses a microcontroller, gas and particulate sensors, GPS for geo-tagging, and wireless communication (Wi-Fi, LoRa, or GSM) to transmit data to the cloud. Machine learning processes the data to predict trends and issue alerts via a mobile app, supporting public awareness and smart city management. This solution promotes sustainable urban living through efficient, scalable pollution monitoring.*

1. INTRODUCTION

Air pollution has become a critical concern, making continuous monitoring essential for public health and environmental sustainability. This project proposes an IoT-based air and sound pollution monitoring system capable of detecting harmful gases and noise levels in real time. The system uses gas and sound sensors connected to a microcontroller, which processes and transmits the data to an online server via the internet. This enables remote monitoring by authorities to take timely actions. Unlike existing commercial solutions limited to specific pollutants or communication modes, the proposed system offers an integrated, cost-effective solution using a wireless sensor network, cloud storage, and a user interface with alerts. The design supports indoor and outdoor applications, making it suitable for scalable smart city deployments.

2. EMBEDDED SYSTEMS

An embedded system is a specialized computer designed to perform specific tasks, often under real-time constraints, and is integrated into larger devices that may include mechanical and hardware components. Unlike general-purpose computers, embedded systems are tailored for dedicated functions and are typically powered by microcontrollers or digital signal processors (DSPs). These systems are found in a wide range of applications, from simple devices like digital watches and MP3 players to complex systems such as traffic control, industrial automation, and even air traffic management. The complexity of embedded systems can vary greatly, from basic setups with a single microcontroller to advanced architectures involving multiple processors and communication networks. Programming for embedded systems differs from traditional PC development, often requiring optimization for cost and efficiency, as the hardware is chosen to minimize expense even if it increases programming difficulty.

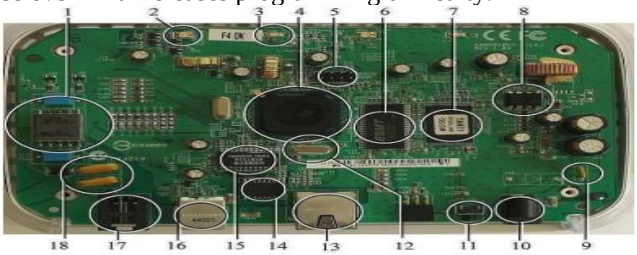


FIGURE 1. A modern example of embedded system

History: In the earliest years of computers in the 1930–40s, computers were sometimes dedicated to a single task, but were far too large and expensive for most kinds of tasks performed by embedded computers of today. Over time however, the concept of programmable controllers evolved from traditional electromechanical sequencers, via solid state devices, to the use of computer technology. It was built from transistor logic and had a hard disk for main memory. When the Minuteman II went into production in 1966, the D-17 was replaced with a new computer that was the first high-volume use of integrated circuits.

Tools: Embedded development makes up a small fraction of total programming. There's also a large number of embedded architectures, unlike the PC world where 1 instruction set rules, and the UNIX world where there's only 3 or 4 major ones. This means that the tools are more expensive. It also means that they're lowering featured, and less developed. On a major embedded project, at some point you will almost always find a compiler some sort. Debugging tools are another issue. Since you can't always run general programs on your embedded processor, you can't always run a debugger on it. This makes fixing your program difficult. Special hardware such as JTAG ports can overcome this issue in part. However, if you stop on a breakpoint when your system is controlling real world hardware (such as a motor), permanent equipment damage can occur. As a result, people doing embedded programming quickly become masters at using serial IO channels and error message style debugging.

Resources: To save costs, embedded systems frequently have the cheapest processors that can do the job. This means your programs need to be written as efficiently as possible. When dealing with large data sets, issues like memory cache misses that never matter in PC programming can hurt you. Luckily, this won't happen too often- use reasonably Embedded applications generally use deterministic memory techniques and avoid the default "new" and "malloc" functions, so that leaks can be found and eliminated more easily. Other resources programmers expect may not even exist. For example, most embedded processors do not have hardware FPUs (Floating-Point Processing Unit).

Real Time Issues: Embedded systems frequently control hardware, and must be able to respond to them in real time. Failure to do could cause inaccuracy in measurements, or even damage hardware such as mtors. 2.2 Need for Embedded Systems: The uses of embedded systems are virtually limitless, because every day new products are introduced to the market that utilizes embedded computers in novel ways. In recent years, hardware such as microprocessors, microcontrollers, and FPGA chips have become much cheaper. So when implementing a new form of control, it's wiser to just buy the generic chip and write your own custom software for it. Producing a custom made chip to handle a particular task or set of tasks costs far more time and money. Many embedded computers even come with extensive libraries, so that "writing your own software" becomes a very trivial task indeed. From an implementation viewpoint, there is a major difference between a computer and an embedded system. Embedded systems are often required to provide RealTime response. The main elements that make embedded systems unique are its reliability and ease in debugging.

Debugging: Embedded debugging may be performed at different levels, depending on the facilities available. From simplest to most sophisticate, they can be roughly grouped into the following areas:

• Interactive resident debugging, using the simple shell provided by the embedded operating system (e.g. Forth and Basic)
• External debugging using logging or serial port output to trace operation using either a monitor in flash or using a debug server like the Remedy Debugger which even works for heterogeneous multi core systems.
• An in-circuit debugger (ICD), a hardware device that connects to the microprocessor via a JTAG or Nexus interface. This allows the operation of the IoT based Air Quality Monitoring System 8 Dept. of ECE NMREC microprocessor to be controlled externally, but is typically restricted to specific debugging capabilities in the processor.
• An in-circuit emulator replaces the microprocessor with a simulated equivalent, providing full control over all aspects of the microprocessor.
• A complete emulator provides a simulation of all aspects of the hardware, allowing all of it to be controlled and modified and allowing debugging on a normal PC.
• Unless restricted to external debugging, the programmer can typically load and run software through the tools, view the code running in the processor, and start or stop its operation.

The view of the code may be as assembly code or source code. Because an embedded system is often composed of a wide variety of elements, the debugging strategy may vary. For instance, debugging a software (and microprocessor)

centric embedded system is different from debugging an embedded system where most of the processing is performed by peripherals (DSP, FPGA, coprocessor). An increasing number of embedded systems today use more than one single processor core. A common problem with multi-core development is the proper synchronization of software execution. In such a case, the embedded system design may wish to check the data traffic on the busses between the processor cores, which requires very low-level debugging, at signal/bus level, with a logic analyzer, for instance.

Reliability: Embedded systems often reside in machines that are expected to run continuously for years without errors and in some cases recover by them if an error occurs. Therefore, the software is usually developed and tested more carefully than that for personal computers, and unreliable mechanical moving parts such as disk drives, switches or buttons are avoided. Specific reliability issues may include: Embedded systems are used in critical applications where downtime is unsafe or costly, such as in space systems, aircraft navigation, chemical plants, and financial networks. These systems must run continuously and reliably. To handle errors, techniques like watchdog timers, redundant subsystems, and limp modes are used. Security and reliability are improved through Trusted Computing Base (TCB) architectures and embedded hypervisors, which isolate faults and allow safe subsystem restarts without affecting the whole system.

Explanation of Embedded Systems:
Embedded systems use various software architectures depending on application needs. A Simple Control Loop runs continuously, calling subroutines that manage specific hardware or software components. In Interrupt Controlled Systems, tasks are triggered by external events like timers or data input, allowing faster response times for critical tasks. Non-pre-emptive multitasking resembles a hidden control loop where tasks run in their own environments and yield control during idle periods. This makes adding new tasks easier. Microkernels, a step above real-time operating systems, handle basic functions like memory and CPU switching, while user-level processes manage advanced operations like networking or file systems.

Stand Alone Embedded System:
Embedded systems receive input from sensors or human commands (like button presses), process the data, and produce output — all in a standalone manner. These are known as standalone embedded systems, such as microwave ovens and air conditioners. Some embedded systems are designed to perform specific tasks within a fixed time and are called real-time embedded systems. These are classified into two types: **Hard real-time systems**, which must meet strict timing deadlines to avoid equipment failure, and **Soft real-time systems**, where slight delays are acceptable and do not cause harm — like in a TV remote control.

Network communication embedded systems:

Embedded systems enable wide-range network communication by connecting devices over the internet. For example, a web camera connected to a computer with internet access can transmit images or videos to another system anywhere in the world. In a smart home setup, a webcam installed at the door can detect a visitor, capture their image, and send it to the homeowner's computer. The user receives an alert with the image and can unlock the door remotely with a mouse click. This demonstrates how embedded systems support real-time, network-based communication and control.

Different types of processing units:
The central processing unit (CPU) in embedded systems can be a microcontroller, microprocessor, or digital signal processor (DSP). Microcontrollers are low-cost and efficient, with built-in components like memory, communication interfaces, and analog-to-digital converters, reducing the need for external parts. Microprocessors are more powerful and suited for complex tasks but require additional external components, leading to higher power consumption. Digital Signal Processors (DSPs) are used in applications focused on signal processing, such as audio, video, or communication systems.

Applications of embedded systems:

Consumer applications:

FIGURE 2. Automatic coffee makes equipment

FIGURE 3. Fax machine, Printing machine

3. HARDWARE DESCRIPTION

In this chapter the block diagram of the project and design aspect of independent modules are considered. block diagram is shown in fig: 3.1:

Block diagram:

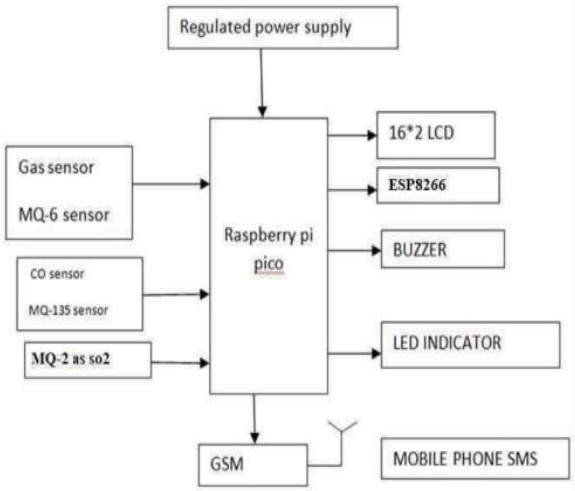


FIGURE 4. Block Diagram

Raspberry Pi Pico is a low-cost, high-performance microcontroller board with flexible digital interfaces. It incorporates Raspberry Pi's own RP2040 microcontroller chip, with a dualcore Arm Cortex M0+ processor running up to 133 MHz, embedded 264KB of SRAM, and 2MB of onboard Flash memory, as well as 26 x multi-function GPIO pins. For software development, either Raspberry Pi's C/C++ SDK or the Micro Python is available. There are also complete development resources and tutorials to help you get started easily, and integrate it into end products quickly.

FIGURE 5. Raspberry Pic Pico

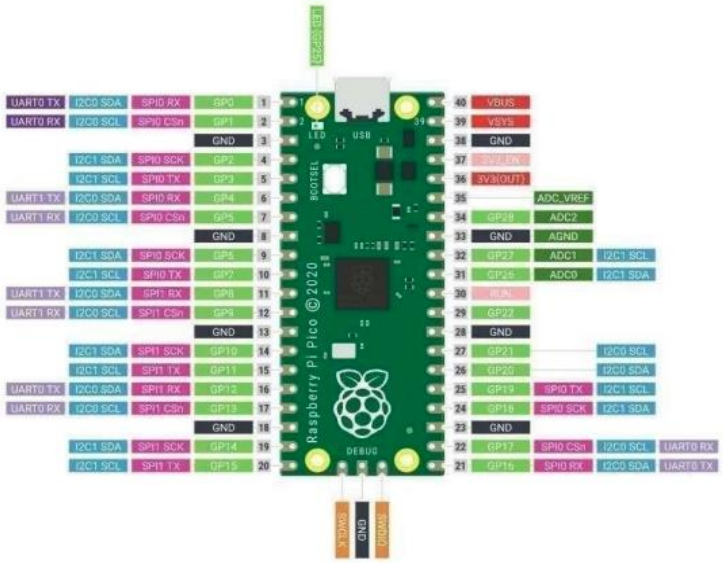


FIGURE 6. Pinout diagram of the Raspberry Pi Pico

4. SOFTWARE DESCRIPTION

This project is implemented using following software's:
•Express PCB – for designing circuit
• Arduino C compiler - for compilation part Proteus 7 (Embedded C) – for simulation part

Express PCB:
Breadboards are great for prototyping equipment as it allows great flexibility to modify a design when needed; however, the final product of a project, ideally should have a neat PCB, few cables, and survive a shake test. Not only is a proper PCB neater but it is also more durable as there are no cables which can yank loose. Express PCB is a software tool to design PCBs specifically for manufacture by the company Express PCB (no other PCB maker accepts Express PCB files). It is very easy to use, but it does have several limitations. Express PCB has been used to design many PCBs (some layered and with surface-mount parts. Print out PCB patterns and use the toner transfer method

with an Etch Resistant Pen to make boards. However, Express PCB does not have a nice print layout. Here is the procedure to design in Express PCB and clean up the patterns so they print nicely.

Preparing Express PCB for First Use:

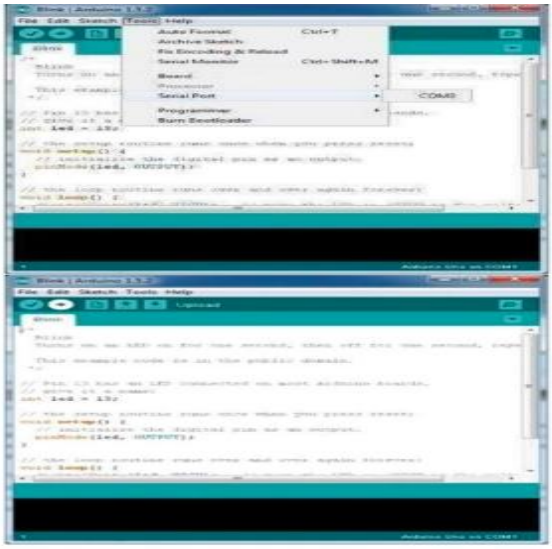
Express PCB comes with a less then exciting list of parts. So, before any project is started head over to Audio logical and grab the additional parts by morsel, ppl, and tangent, and extract them into your Express PCB directory. At this point start the program and get ready to setup the workspace to suit your style. Click View -> Options. In this menu, setup the units for "mm" or "in" depending on how you think, and click "see through the top copper layer" at the bottom. The standard color scheme of red and green is generally used but it is not as pleasing as red and blue.

The interface: When a project is first started you will be greeted with a yellow outline. This yellow outline is the dimension of the PCB. Typically, after positioning of parts and traces, move them to their final position and then crop the PCB to the Iot based Air Quality Monitoring correct size. However, in designing a board with a certain size constraint, crop the PCB to the correct size before starting.

FIGURE 7. Show The Toolbar in Which Each Button Has the Following Functions

Design Considerations:

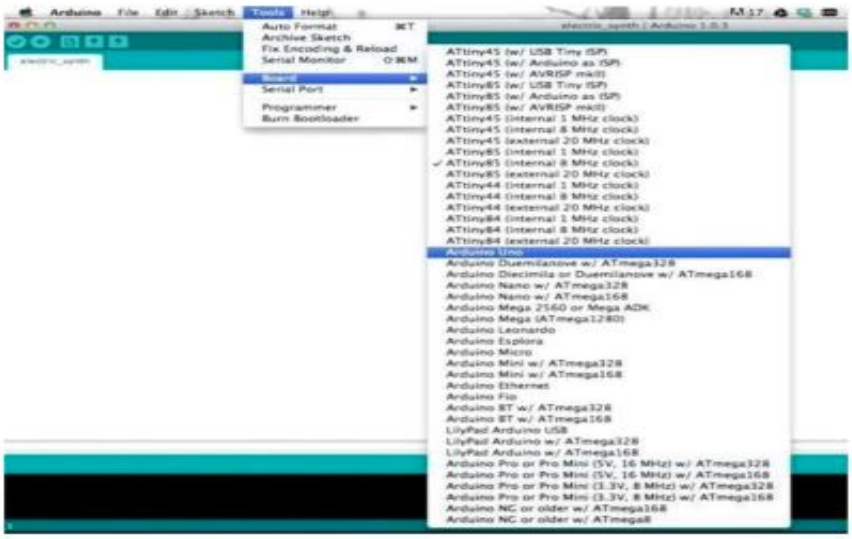
Before starting a project there are several ways to design a PCB and one must be chosen to suit the project's needs. When making a PCB you have the option of making a single sided board, or a double sided board. Single sided boards are cheaper to produce and easier to etch, but much harder to design for large projects. If a lot of parts are being used in a small space it may be difficult to make a single sided board without jumping over traces with a cable. While Iot based Air Quality Monitoring there's technically nothing wrong with this, it should be avoided if the signal travelling over the traces is sensitive (e.g. audio signals). A double sided board is more expensive to produce professionally, more difficult to etch on a DIY board, but makes the layout of components a lot smaller and easier. It should be noted that if a trace is running on the top layer, check with the components to make sure you can get to its pins with a soldering iron. Large capacitors, relays, and similar parts which don't have axial leads can NOT have traces on top unless boards are plated professionally. When using a double sided board you must consider which traces should be on what side of the board. Generally, put power traces on the top of the board, jumping only to the bottom if a part cannot be soldiered onto the top plane (like a relay), and viceversa. Some projects like power supplies or amps can benefit from having a solid plane to use for ground. In power supplies this can reduce noise, and in amps it minimizes the distance between parts and their ground connections, and keeps. Launch and Blink! After following the appropriate steps for your software install, we are now ready to test your first program with your Arduino board! Launch the Arduino application.

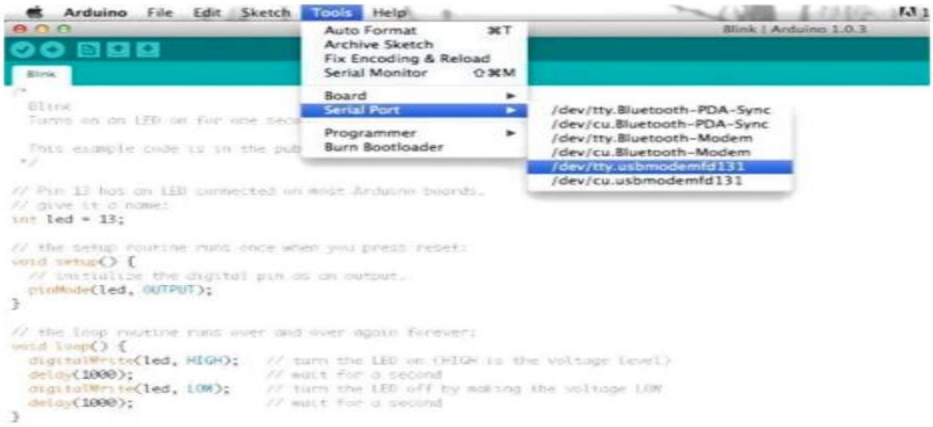


If you're not sure which serial device is your Arduino, take a look at the available ports, then unplug your Arduino and look again. The one that disappeared is your Arduino. With your Arduino board connected, and the Blink sketch open, press the 'Upload' button After a second, you should see some LEDs flashing on your Arduino, followed by the message 'Done Uploading' in the status bar of the Blink sketch

Launch and Blink!

Once the software installation is complete, you can test your first program with the Arduino board. Launch the Arduino application and prepare to write and upload the code. This process will help you verify that your setup is working correctly and give you hands-on experience with programming and interacting with your Arduino board.





5. ADVANTAGES AND DISADVANTAGES

Advantages:

1. Wi-Fi-based user-friendly interface
2. Low power consumption
3. Capable of controlling both high and low voltage devices
4. Long operational life
5. Wireless data transmission via Wi-Fi

6. Fast system response
7. Efficient and cost-effective design

Disadvantages:

1. No feedback or status updates from devices
2. Limited operational range

Applications:

1. Useful in environments where human presence is risky or impossible
2. Widely used in military applications, especially for explosive detection using robots
3. Can be employed to control various electronic devices remotely

6. RESULTS

The project "IOT Based Air Pollution Monitoring and Alerting System to Prevent environment" was System uses air sensors to sense presence of harmful gases/compounds in the air and constantly transmit this data to microcontroller. Also, system keeps measuring sound level and reports it to the online server over IOT.

7. CONCLUSION

Integrating features of all the hardware components used have been developed in it. Presence of every module has been reasoned out and placed carefully, thus contributing to the best working of the unit. Secondly, using highly advanced ICs with the help of growing technology, the project has been successfully implemented. Thus, the project has been successfully designed and tested.

Future Scope: Air pollution occurs when harmful substances including particulates and biological molecules are introduced into Earth's atmosphere. It may cause diseases, allergies or death in humans; it may also cause harm to other living organisms such as animals and food crops, and may damage the natural or built environment. Human activity and natural processes can both generate air pollution. We can use more sensors along with this system.

REFERENCES

[1]. Thing Speak Math Works [https://www.mathworks.com/help/thingspeak] (Used for cloud platform integration and data visualization setup.)

[2]. Senthilkumar Meyyappan, A. Bharath Naik, A. Uma Sai and Ch. Keerthi, "Improving Weather Forecasting Accuracy Using Machine Learning", *Journal on Electronic and Automation Engineering,* Vol. 2(4), December 2023, pp. 9-18.

[3]. Senthilkumar Meyyappan and N. Selvamuthukumaran, "Network Selection in Heterogeneous Wireless Systems using GRA Method", *Journal on Electronic and Automation Engineering*, Vol. 4(1), March 2025, pp. 127-132.

[4]. ESP8266 Node MCU – [https://docs.espressif.com/projects/esp8266] (Technical specifications and programming details of the microcontroller.)

[5]. M. Senthil Kumar and M. Gopinath, "An Efficient Polynomial Pool-Based Scheme for Distributed Heterogeneous WSNs", *International Journal of Modern Engineering Research (IJMER)*. (Vol.3, Issue 6, Nov-Dec.2013, PP 3328-3335, ISSN: 2249-6645).

[6]. Arduino IDE and Library References – [https://www.arduino.cc] (Programming environment and libraries used for sensor integration and ESP8266 coding.)

[7]. M. Senthil Kumar and L. Praveen, "An Assuring Approach for Tree-Based Routing Topology in WSNs", *International Journal of Emerging Trends in Engineering and Development (IJETED).* (Issue 3, Vol.6, November 2013, ISSN: 2249 – 6149).

[8]. Senthilkumar Meyyappan, G. Lava Kumar, G. Niharika and G. Chakradhar, "Cellular Network Signal Strength Analyser", *Journal on Electronic and Automation Engineering*, Vol. 4(1), March 2025, pp. 165-174.

[9]. Sharma, S., & Kumar, R. (2020). IoT Based Smart Water Quality Monitoring System. International Journal of Engineering Research & Technology (IJERT), Vol. 9 Issue 06. (Used as literature basis for IoT-based water monitoring architecture.)

[10]. Senthilkumar Meyyappan, K. Susmitha, K. Vaishnavi and M. Sai Rao, "Condition Based Monitoring and Maintenance System for Underground Metro Stations", *Journal on Electronic and Automation Engineering*, Vol. 4(1), March 2025, pp. 175-182.

[11]. M. Singh, A. Patel (2021). Design and Implementation of IoT Based Water Quality Monitoring System. IEEE International Conference on Smart Systems and Inventive Technology (ICSSIT).

[12]. C.I. Vimalarani and M. Senthil Kumar, "Energy Efficient PCP Protocol for k-Coverage in Sensor Networks", *IEEE International Conference on Computational Intelligence and Computing Research, IEEE Proceedings, 2010.*

[13]. M. Kavitha, T. Maheshwaran and M. Senthil Kumar, "Secure Routing in MANETs with Key Management", *International Journal on Engineering Technology and Sciences (IJETS).* (Vol.1, Issue 6, October 2014, ISSN (P): 2349 – 3968, ISSN (O): 2349 - 3976).

[14]. M. Senthil Kumar, "Energy Efficient Techniques for Transmission of Data in Wireless Sensor Networks", *Journal of Computing Technologies (JCT)*. (Vol.5, Issue 2, February 2016, ISSN: 2278 – 3814).

[15]. M. Senthil Kumar and Ashish Chaturvedi, "Energy-Efficient Coverage and Prolongs for Network Lifetime of WSN using MCP*", European Journal of Scientific Research (EJSR).* (Vol.95, No.2, January 2013, ISSN: 1450 – 216X / 1450 – 202X).

[16]. SIM800L GSM Module AT Commands Reference – [https://lastminuteengineers.com/sim800l-gsm-module-arduino-tutorial] (Used for integrating SMS alert functionality).

[17]. M. Senthil Kumar and C. Sridhathan, "Impact of Mobility on the Routine of Enhanced – DSDV Protocol in Mobile Ad-hoc Networks", *International Journal of Applied Engineering Research (IJAER)*. (Vol.13, No.14, 2018, PP 11674-11679, ISSN: 0973-4562).

[18]. M. Kavitha, T. Maheshwaran and M. Senthil Kumar, "Ensure Data Transmission in Mobile Ad-Hoc Networks", *International Journal on Engineering Technology and Sciences (IJETS)*. (Vol.2, Issue 4, April 2015, ISSN (P): 2349 – 3968, ISSN (O): 2349 - 3976).

[19]. M. Senthil Kumar and Ashish Chaturvedi, "A Novel Enhanced Coverage Optimization Algorithm for Effectively Solving Energy Optimization Problem in WSN", *Research Journal of Applied Sciences, Engineering and Technology (RJASET).* (Issue 4, Vol.7, January 2014, ISSN: 2040 – 7459 & e-ISSN: 2040 – 7467).

[20]. Senthilkumar Meyyappan, Kalyan Kasturi, G. Vijaya Lakshmi, J. Srinija Reddy and K. Grace Sampoorna, "Improvement of LEACH Protocol for Enhancing Features of WSN", *Journal on Electronic and Automation Engineering*, Vol. 2(4), December 2023, pp. 19-26.

