A Project Report

On

IOT BASED URBAN UTILITIES MANAGEMENT

Submitted in partial fulfilment of the requirements for the award of degree of

BACHELOR OF TECHNOLOGY

In

ELECTRICAL AND ELECTRONICS ENGINEERING

Submitted By

Mr. KONAKALLA LAKSHMI SHANMUKH PAVAN SAI	**21ME1A0222**
Mr. SURISETTI VEERENDRA KUMAR	**22ME5A0216**
Mr. BANDUCHODE KUMAR	**21ME1A0205**
Mr. CHILAKABATTINA JAYA SAI KUMAR	**21ME1A0208**
Mr. BALANAGU SAI KRISHNA	**21ME1A0203**

Under the Guidance of

Mr. R. NAVEENKUMAR

Department of Electrical and Electronics Engineering

RAMACHANDRA COLLEGE OF ENGINEERINIG

(AUTONOMOUS)

(Approved by AICTE, New Delhi, Permanently Affiliated to JNTUK: Kakinada)

Accredited by NAAC (A+) & NBA, An ISO 9001:2015 Certified Institution

NH-16 Bypass Road, Vatluru (V), ELURU-534 007, Eluru Dist., A. P.

2021-2025

RAMACHANDRA COLLEGE OF ENGINEERING

(AUTONOMOUS)

DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING

BONAFIED CERTIFICATE

This is to certify that this project report entitled "**IOT BASED URBAN UTILITIES MANAGEMENT**" is bonafide work done by **KONAKALLA LAKSHMI SHANMUKH PAVAN SAI (21ME1A0222), SURISETTI VEERENDRA KUMAR (22ME5A0216), BANDUCHODE KUMAR (21MEIA0205), CHILAKABATTINA JAYA SAI KUMAR (21ME1A0208), Mr. BALANAGU SAI KRISHNA (21ME1A0203)** submitted in partial fulfilment of the requirements for the award of the degree of Bachelor of Technology in Electrical and Electronics Engineering in A.Y. 2024-25. The results of investigation enclosed in this report have been verified and found satisfactory. The results embodied in this project report have not been submitted to any other University or Institute for the award of any other degree or diploma.

Project guide	**Head of the Department**
Mr. R. NAVEEENKUMAR, M.tech	**Mr. JAJULA SURESH, M.tech**
ASSISTANT PROFESSOR	ASSOCIATIVE PROFESSOR

Internal Examiner **External Examiner**

DECLARATION

We are **KONAKALLA LAKSHMI SHANMUKH PAVAN SAI (21ME1A0222), SURISETTI VEERENDRA KUMAR (22ME5A0216), BANDUCHODE KUMAR (21MEIA0205), CHILAKABATTINA JAYA SAI KUMAR (21ME1A0208), Mr. BALANAGU SAI KRISHNA (21ME1A0203) here by declare the project report titled " IOT BASED SMART PRE-PAID ENERGY METER WITH PRIORITIZED LOAD SWITCHING** " under the supervision of **Mr. R. Naveenkumar**, **ASSISTANT PROFESSOR** of Electrical and Electronics Engineering is submitted in partial fulfilment of the requirements for the award of the degree of Bachelor of Technology in Electrical and Electronics Engineering. This is a record of work carried out by us and the results embodied in this project have not been reproduced or copied from any source. The results embodied in this project report have not been submitted to any other University or Institute for the award of any other degree or diploma.

Mr. KONAKALLA LAKSHMI SHANMUKH PAVAN SAI	**21ME1A0222**
Mr. SURISETTI VEERENDRA KUMAR	**22ME5A0216**
Mr. BANDUCHODE KUMAR	**21ME1A0205**
Mr. CHILAKABATTINA JAYA SAI KUMAR	**21ME1A0208**
Mr. BALANAGU SAI KRISHNA	**21ME1A0203**

ACKNOWLEDGEMENT

Many individuals have contributed to the fulfilment of this work in one or many ways. This work could not have seen light without the help of all these people, it will be our pleasure and responsibility to acknowledge their contributions.

It is a great pleasure to express our deep sense of gratitude and profound thanks to our Guide **Mr. R NAVEENKUMAR**, Assistant Professor of Electrical & Electronics Engineering, who kindly accepted to guide us and supported us with positive encouragement, patient guidance and fruitful discussions throughout the course of the Project work

We thank the Project Coordinator **Mr. P.VICTOR BABU**, Assistant Professor Department of EEE, for their valuable guidance and support throughout the development of this project.

We would like to express our sincere thanks to **Mr. JAJULA SURESH**, **Head of the Department**, Electrical & Electronics Engineering, for his help and cooperation in carrying out the project in a successful manner.

We wish to thank **Dr. S. S. SARMA**, Dean Academies, for having permitted us to do the Project work at Department of Electrical & Electronics Engineering and he has been a constant source of encouragement.

We express our deepest gratitude to **Dr. M MURALIDHARA RAO**, **Principal**, Ramachandra College of Engineering, Eluru for his valuable suggestions during preparation of draft in our document.

We take this space to extend our thankfulness to **Mr. K. SAI ROHITH**, Managing Director, Ramachandra College of Engineering, Eluru, for their constant support.

We thank to all the **Teaching and Non-Teaching** staff members, Department of Electrical and Electronics Engineering for their affection and immense support throughout duration our Project work.

We thank our parents, friends, family, and all our gurus who stood for us and taught us to stand against all odds in life.

INDEX

List of Figures

74

IOT BASED URBAN UTILITIES MANAGEMENT

ABSTRACT

A smart city solution called the IoT-Based Urban Utilities Management System was created to automate and track important public utility services like water distribution, manhole condition, and street lighting. The ESP32 microcontroller, which is at the center of the system, communicates with a number of input and output modules to guarantee effective and perceptive urban utility management. In order to ensure that the street lighting system is not only turned on but also operating, it is automated with a Real-Time Clock (RTC) for time-based control and an LDR (Light Dependent Resistor) for real-time defect detection. Relay2 is also used by the system to regulate the streetlights. ultrasonic sensor is used by the manhole monitoring device to identify spills, obstructions, and abnormal water levels. A buzzer is activated for nearby notice and alerts are sent via the GSM module if a critical level is reached. The system has a smart water pump that is turned on by Relay1 to control water distribution in public areas. The ESP32 is used to monitor and control this, and it can be activated by a linked mobile application (APP) or on a schedule. Local monitoring and IoT-based control are made possible by the system's RPS (Regulated Power Supply) and LCD display, which also gives real-time feedback. This project serves as an example of an urban utility management strategy that is scalable, real-time, and economical. It is a major step toward future-ready smart city infrastructure since it decreases manual intervention, conserves energy and water, and improves public safety.

CHAPTER 1

INTRODUCTION

Urbanization is expanding rapidly across the globe, and with it comes the need for smarter, more sustainable infrastructure to efficiently manage public utilities. Traditional utility management methods often require manual operation and monitoring, which can lead to delays, inefficiencies, wastage of resources, and high maintenance costs. To address these challenges, the integration of Internet of Things (IoT) technology in urban infrastructure presents a promising solution.

This project, titled "IoT-Based Urban Utilities Management System", is a comprehensive approach toward automating and monitoring essential utilities such as street lighting, manhole status detection, and water distribution systems. It combines real-time sensors, wireless communication modules, and a centralized microcontroller (ESP32) to enable smart decision-making, reduce manual effort, and improve service delivery to the public.

The core motivation behind this project is to enhance the reliability and efficiency of city services through automation and real-time monitoring. Urban areas, especially in developing regions, face problems like unnecessary power consumption due to continuously running street lights, delayed responses to manhole blockages that cause sanitation issues, and inefficient public water usage due to manual systems.

Smart solutions using IoT can eliminate these issues by enabling:

- Automated control of utilities based on time and sensor data
- Real-time monitoring and alerts for faults or abnormal conditions
- Remote access and control through wireless modules
- Fault detection and prevention, improving maintenance cycles

The IoT-Based Urban Utilities Management System is a compact yet powerful integration of multiple subsystems that together enhance the efficiency of essential urban services. At the core of the system is the ESP32 microcontroller, which acts as the central control unit, interfacing with various sensors, modules, and output devices. The system is powered by a Regulated Power Supply (RPS) to ensure stable operation of all components. To automate streetlight control, a Real-Time Clock (RTC) is used to turn lights on and off based on predefined time settings, reducing unnecessary power usage. In addition, a Light Dependent Resistor (LDR) is employed to detect whether the light is actually illuminating when it’s supposed to. If the light fails to turn on despite the signal, this fault is detected and can be reported, thereby improving maintenance accuracy. The manhole monitoring system uses an ultrasonic sensor to measure the water level inside the manhole. If the level exceeds a safe limit, the system activates a buzzer for local alerts and sends a message via the GSM module to the concerned authorities, enabling quick response to prevent accidents or overflow.

For public water distribution, the system includes a water pump controlled via a relay, which is activated automatically or through user input via a mobile application. The relay modules (Relay1 and Relay2) are used to switch the pump and the streetlight circuit, respectively. An LCD display is also included to provide real-time status updates directly on-site, offering transparency and immediate feedback. Wireless communication is enabled through both GSM and Wi-Fi modules, allowing users or municipal officials to monitor and control the system remotely through an IoT-based mobile application. This remote access ensures real-time updates, reduces manual intervention, and enhances the reliability of urban services.

1.1 PROBLEM STATEMENT:

Urban infrastructure in many cities still relies on manual monitoring and control methods for essential public utilities such as street lighting, drainage systems, and water distribution. These outdated systems often lead to a range of challenges, including excessive energy consumption due to streetlights being left on unnecessarily, delayed detection of manhole overflows or blockages, and inefficient or wasteful water usage in public areas.

Moreover, the lack of real-time monitoring and fault detection mechanisms results in delayed responses to system failures, posing risks to public safety and increasing maintenance costs. The absence of automation and smart control leads to unnecessary manpower usage, increased operational expenses, and reduced efficiency in service delivery.

Therefore, there is a critical need for a smart, real-time, and centralized utility management system that can monitor, control, and report the status of these public services automatically. A solution integrating IoT technology, sensors, and wireless communication is essential to address these urban challenges efficiently, ensuring sustainability, safety, and reliability.

In many urban areas, inefficient utility management leads to problems such as:

- Wastage of electricity due to improper streetlight control.
- Water wastage from uncontrolled public water access.
- Public safety risks caused by overflowing or open manholes.
- Lack of centralized real-time monitoring and reporting.

1.2 AIM AND OBJECTIVES :

To design and implement an IoT-based smart urban utilities management system that automates and monitors key public services such as street lighting, manhole water level detection, and public water distribution using ESP32, real-time sensors, and wireless communication technologies.

OBJECTIVES:

- To automate street lighting operations using a Real-Time Clock (RTC) for scheduled control and an LDR sensor for verifying the physical status of the lights.
- To develop a fault detection mechanism using the LDR sensor to identify whether the streetlight is functioning when it is expected to be ON.
- To monitor manhole water levels using an ultrasonic sensor to detect blockages or overflow conditions in real-time.
- To provide instant alerts to municipal authorities using a GSM module and a buzzer system whenever abnormal conditions are detected.
- To control public water distribution through an automated water pump managed by a relay and monitored via ESP32.
- To enable real-time system monitoring and control through a mobile application using IoT (Wi-Fi).

CHAPTER 2

LITERATURE REVIEW

Suseendran et al. represented the brightness controlling of the street light using sensors, based on IoT (Internet of Things), video vehicle detection and LDR (Light Decreasing Resistance) sensor. Each lamp unit encompassed two sensors- video vehicle detection sensor and LDR sensor. All the data were collected and processed on a regular basis which had required a huge data processing system.

Lavric et al. revealed a practical implementation of street light monitoring and controlling system employing WSN (Wireless Sensor Network System). The authors had focused on the software-based method and then performed a realtime implementation using limited number of lights. This archetype included Doppler sensor to allow vehicle detection. According to the appearance of vehicle, lights for that particular region increased intensity.

Archibong et al. worked with IoT based PV solar self-powered lighting system for street with anti-vandalism monitoring and tracking competency. LDR sensor switched the lights on-off along with an IR (InfraRed) sensor having the ability to save power. The anti-vandalism system was installed with a User Interface (UI), where the street lights communicated with the people and devices through wi-fi module at the control station.

Prasad et al. projected a case-study of Nagpur Street lighting system, where LED lights were utilized along with motion detection system. The arbitration showed that energy consumption was lessen by 55% per month.

Abdullah et al. worked to provide a suitable system to minimize the power dissipation. The prototype comprised LDR, IR, battery and LED along with Arduino uno, which was skilled in increasing the intensity of light based on the speed of the vehicle. This project concluded that, this system will retain about 40% energy per month.

Sukhothai et al. using Lora WAN in accompany with motion detector and illuminance sensor to mitigate the power consumption. In this model, every street light had been furnished with Lora WAN communication module, which controlled the LED lights by exchanging data with main server.

Mary et al. investigated on the control of street lights using lamp unit, sensor unit and access unit. The lamp and the sensor units accommodated sensors for identifying motion and brightness of controller, LED device and communication device. For the access unit, Zigbee was used. Whenever there was any motion, lights' brightness was proliferated otherwise it was dimmed or low. All the above divulged papers are prototype based. Only the work with WSN network system represented a field work with implementing the concept only for controlling four lighting units. This paper portrays a practically implemented method with control of 300 lights of 2.5km distance.

In the year 2016, Manish Kumar published a paper on streetlight regulation using a Zigbee wireless module. A transmission module, an LDR, and a microcontroller were among the components. Wireless connectivity with the lamp module is possible thanks to Zigbee. The computer analyses day-night variations and lamp safety using two LDR sensors. After the LDR data has been read, it is transferred to the microcontroller and then to the transmission module. The data is sent to the control center via wireless Zigbee, which monitors and manages each streetlight. The device connects to a Zigbee wireless network with a limited range. A GSM-based automatic streetlight control system was introduced by Prof. K.Y. Rajput and three other TSE Mumbai researchers to calculate different parameters, the device has a server microcontroller and sensors such as smoke sensors, noise sensors, light sensors, and so on. This device will sense ambient temperatures and noise levels and give an intervention signal to the machine. the problem is that the GSM modem must be mounted in each streetlight, which makes the unit very costly. There are also several compatibility issues. This model is more expensive because it depends heavily on hardware to control and monitor the computer.

M. Abhishek et al. developed a solar-powered traffic flow-based streetlight control system. They used an 8052 series microcontroller and replaced standard bulbs with LED lights, resulting in a threefold reduction in energy consumption. Sensors on either side of the road detect vehicle activity and instruct the microcontroller to turn the lights on and off as needed and when there are vehicles present or moving are the lights switched on. Otherwise, even though it's night time or there's bad weather, all the lights are switched off .A GSM-based smart street light monitoring and control system is an electronic system that uses timed operated switching of street lights to increase an industry's performance and accuracy.

Smart Street Lighting System once exploitation GSM, ancient street lighting schemes in locations with a restricted variety of passersby square measure left on for the rest of the night. As a result, a significant amount of energy is lost in vain. Quick reacting, dependable running, and power-saving street lighting systems are now a reality thanks to the widespread availability of flexible-lighting technologies such as light-emitting diode lamps and ubiquitous wireless internet access. This paper explains the Intelligent Street Lighting (ISL) scheme, which is a first step in satisfying the need for flexible public lighting systems

CHAPTER 3

EXISTING SYSTEM

Currently, municipal water supply systems rely on manual operation, where workers are responsible for turning the water induction motor pump on and off based on predefined schedules or demand. This manual process can lead to inefficiencies such as delayed water distribution, excessive energy consumption, and human errors that may result in water wastage or inadequate supply. Additionally, unexpected failures or maintenance requirements in the water pump system may not be detected promptly, leading to service disruptions.

Similarly, streetlights in many areas are still controlled manually by municipal workers. This method has several drawbacks, including delays in switching lights on or off, energy wastage due to lights remaining on during daylight hours, and potential security risks when lights fail to be activated on time. The reliance on human intervention increases the chances of operational inefficiencies and maintenance delays.

Manhole monitoring is another major concern in urban infrastructure. At present, manholes are not routinely monitored in real-time. The condition of a manhole is typically noticed only when it becomes clogged or overflows, causing traffic disruptions, health hazards, and potential accidents. Without an early warning system, municipal authorities can only take action after a problem has already occurred, leading to reactive rather than proactive maintenance.

To address these issues, this project introduces an IoT-based automated system that eliminates manual intervention in water supply management, streetlight operation, and manhole monitoring. By integrating IoT technology with an ESP32 microcontroller, the system automates water pump control, ensuring efficient and scheduled water distribution while reducing human dependency. An RTC module is used to automatically turn streetlights on and off at predefined times, optimizing energy consumption and enhancing public safety. Additionally, an ultrasonic sensor continuously monitors water levels in manholes, providing real-time alerts to prevent overflow and allowing authorities to take preventive action before a major issue arises.

This innovative approach improves municipal resource management, enhances operational efficiency, reduces maintenance costs, and ensures a safer and more reliable urban infrastructure.

CHAPTER 4

PROPOSED SYSTEM

This project utilizes an ESP32 microcontroller to automate and optimize municipal services, including scheduled water supply, smart streetlight control, and manhole monitoring. The system integrates a Real-Time Clock (RTC), ultrasonic sensors, an LDR, IoT connectivity, and a GSM module to ensure efficient and reliable operation.

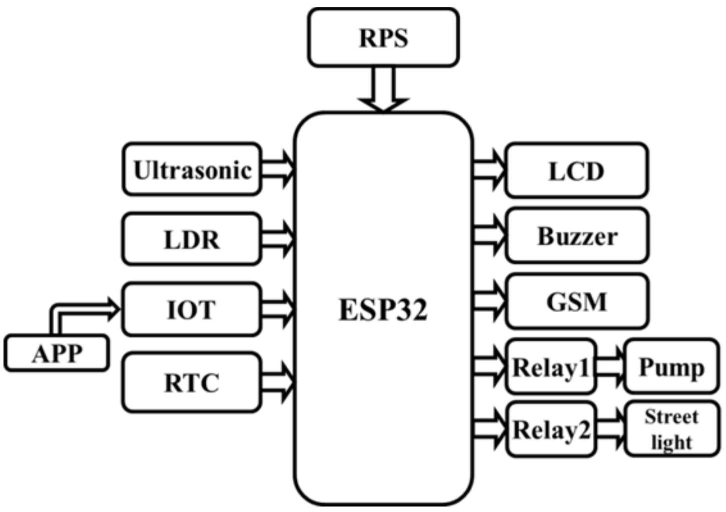


Fig 4.1: Flowchart of IoT based Urban Utility Management

The RTC module is programmed with a predefined schedule for municipal water distribution.At the scheduled time, the ESP32 microcontroller activates the water pump, allowing water to flow through the distribution network.If the system detects pump failure, an alert is sent via SMS to the registered authorities, and a buzzer alarm is triggered.The entire water distribution status is displayed on the LCD screen and updated to the IoT cloud for remote monitoring.The RTC ensures streetlights turn on and off according to a pre-set schedule, eliminating the need for manual operation.A Light Dependent Resistor (LDR) continuously monitors the brightness of the streetlights.If a streetlight fails to illuminate even after being switched on, the system triggers a buzzer alarm and sends an SMS alert to the authorities for maintenance action.The real-time status of streetlights is displayed on the LCD screen and updated to the IoT dashboard for centralized monitoring.An ultrasonic sensor is placed inside the manhole to continuously monitor the water level.If the water level exceeds a predefined threshold, indicating a potential clogging or overflow, the system immediately: Activates a buzzer alarm for local alerting.

Sends an SMS notification to the municipal authorities for quick intervention.Logs the incident data to the IoT cloud, allowing authorities to track and analyze clogging trends for preventive maintenance.

All operational data, including water supply status, streetlight conditions, and manhole water levels, is displayed on an LCD screen.The system continuously updates the IoT cloud, enabling municipal authorities to remotely monitor and manage city infrastructure in real time from a dashboard.In case of any fault detection, the system provides instant SMS alerts and buzzer notifications to ensure prompt maintenance action.

This fully automated IoT-based system significantly enhances municipal service efficiency by eliminating manual intervention, ensuring reliable water supply, optimizing streetlight operation, and preventing manhole overflows.

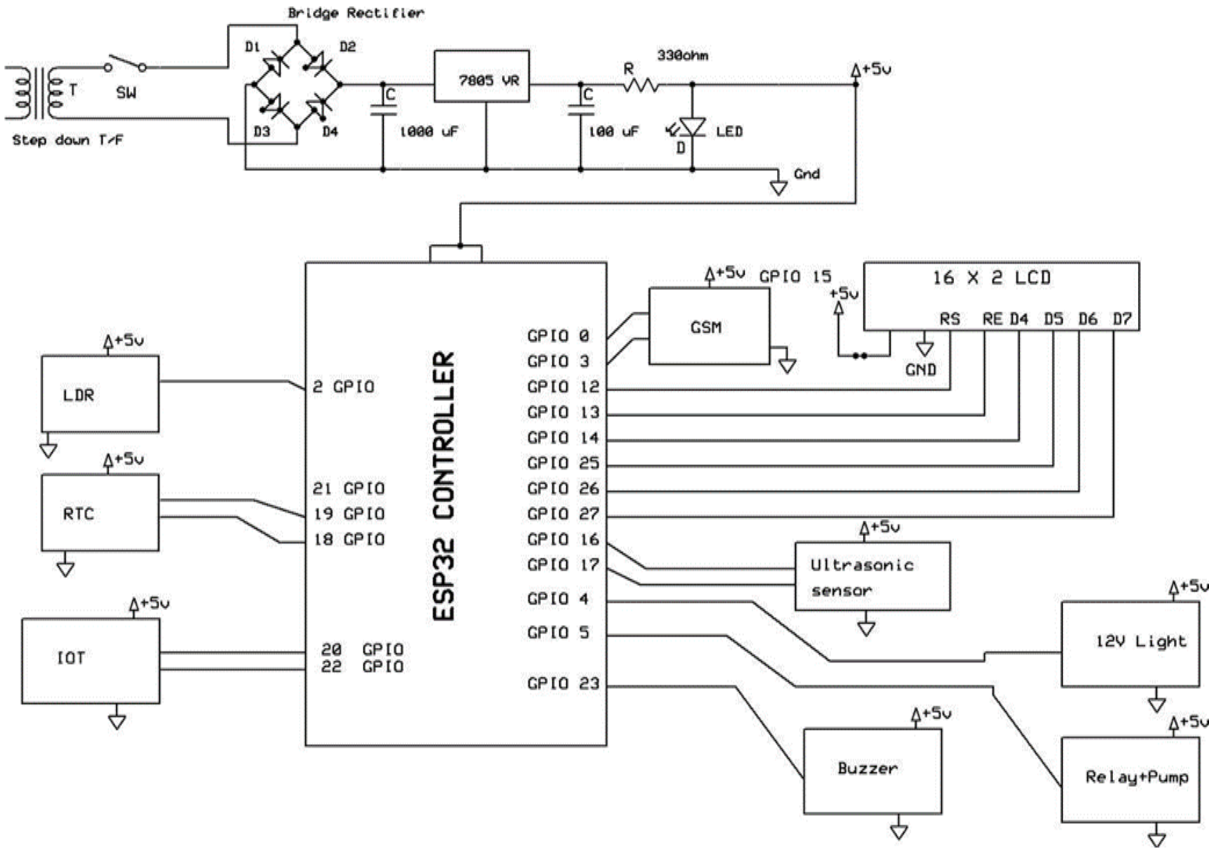


Fig 4.2: Schematic Diagram of IoT based Urban Utilities Management

In this project, we are using ESP32 Microcontroller. It has a total of 40 pins. Here the 20 and 22 are connected to the IOT, for transmitting and Receiving the data. 12, 13, 14, 25, 26, 27 pins are connected to 16*2 LCD, 0, and 3 pins are connected to a GSM modem which can send alert messages to the registered user, the 23 pin is connected to the buzzer, the 5 pin is connected to the pump. 2 pin is connected to the LDR sensor, 4 pin is connected to 12v LED. A4-A5 pins are connected to RTC.

CHAPTER 5

EMBEDDED SYSTEMS

5.1 Embedded Systems:

An embedded system is a computer system designed to perform one or a few dedicated functions often with real-time computing constraints. It is embedded as part of a complete device often including hardware and mechanical parts. By contrast, a general-purpose computer, such as a personal computer (PC), is designed to be flexible and to meet a wide range of end-user needs. Embedded systems control many devices in common use today.

Embedded systems are controlled by one or more main processing cores that are typically either microcontrollers or digital signal processors (DSP). The key characteristic, however, is being dedicated to handle a particular task, which may require very powerful processors. For example, air traffic control systems may usefully be viewed as embedded, even though they involve mainframe computers and dedicated regional and national networks between airports and radar sites. (Each radar probably includes one or more embedded systems of its own.)

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. Complexity varies from low, with a single microcontroller chip, to very high with multiple units, peripherals and networks mounted inside a large chassis or enclosure.

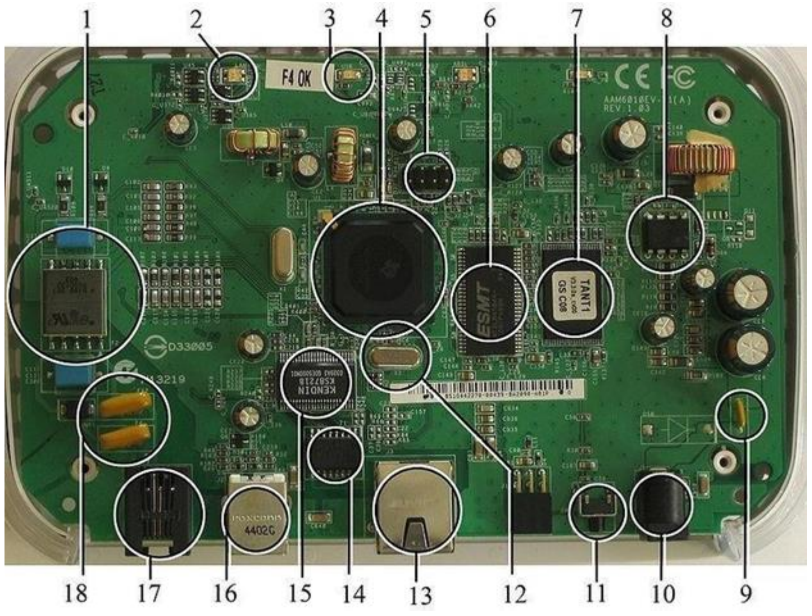


Fig 5.1: A modern example of embedded system

Labeled parts include microprocessor (4), RAM (6), flash memory (7).Embedded systems programming is not like normal PC programming. In many ways, programming for an embedded system is like programming PC 15 years ago. The hardware for the system is usually chosen to make the device as cheap as possible. Spending an extra dollar a unit in order to make things easier to program can cost millions. Hiring a programmer for an extra month is cheap in comparison. This means the programmer must make do with slow processors and low memory, while at the same time battling a need for efficiency not seen in most PC applications. Below is a list of issues specific to the embedded field.

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.

One of the first recognizably modern embedded systems was the Apollo Guidance Computer, developed by Charles Stark Draper at the MIT Instrumentation Laboratory. 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 bug of 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 efficient algorithms to start, and optimize only when necessary. Of course, normal profilers won't work well, due to the same reason debuggers don't work well.

Memory is also an issue. For the same cost savings reasons, embedded systems usually have the least memory they can get away with. That means their algorithms must be memory efficient (unlike in PC programs, you will frequently sacrifice processor time for memory, rather than the reverse). It also means you can't afford to leak memory. 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). These resources either need to be emulated in software, or avoided altogether.

Real Time Issues:

Embedded systems frequently control hardware, and must be able to respond to them in real time. Failure to do so could cause inaccuracy in measurements, or even damage hardware such as motors. This is made even more difficult by the lack of resources available. Almost all embedded systems need to be able to prioritize some tasks over others, and to be able to put off/skip low priority tasks such as UI in favor of high priority tasks like hardware control.

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 Real-Time response. The main elements that make embedded systems unique are its reliability and ease in debugging.

Debugging:

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, co-processor). 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 avoide.

5.2 Explanation of Embedded Systems:

Software Architecture:

There are several different types of software architecture in common use. Simple Control Loop: In this design, the software simply has a loop. The loop calls subroutines, each of which manages a part of the hardware or software.

Interrupt Controlled System: Some embedded systems are predominantly interrupting controlled. This means that tasks performed by the system are triggered by different kinds of events. An interrupt could be generated for example by a timer in a predefined frequency, or by a serial port controller receiving a byte. These kinds of systems are used if event handlers need low latency and the event handlers are short and simple.

Usually these kinds of systems run a simple task in a main loop also, but this task is not very sensitive to unexpected delays. Sometimes the interrupt handler will add longer tasks to a queue structure. Later, after the interrupt handler has finished, these tasks are executed by the main loop. This method brings the system close to a multitasking kernel with discrete processes.

Stand Alone Embedded System:

These systems takes the input in the form of electrical signals from transducers or commands from human beings such as pressing of a button etc.., process them and produces desired output. This entire process of taking input, processing it and giving output is done in standalone mode. Such embedded systems comes under stand alone embedded systems

Eg: microwave oven, air conditioner etc..

Network communication embedded systems:

A wide range network interfacing communication is provided by using embedded systems. Eg:Consider a web camera that is connected to the computer with internet can be used to spread communication like sending pictures, images, videos etc.., to another computer with internet connection throughout anywhere in the world. Consider a web camera that is connected at the door lock. Whenever a person comes near the door, it captures the image of a person and sends to the desktop of your computer which is connected to internet. This gives an alerting message with image on to the desktop of your computer, and then you can open the door lock just by clicking the mouse. Fig: 5.2 show the network communications in embedded systems.

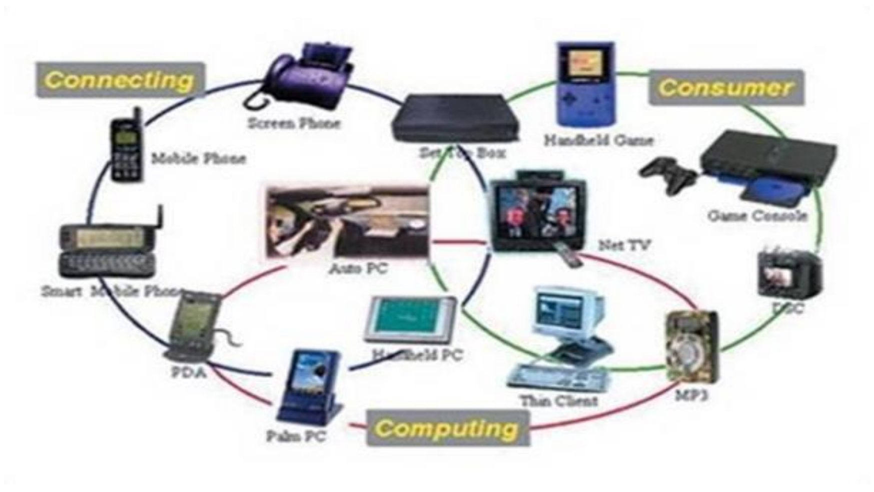


Fig 5.2 : A modern example of Network communication embedded systems

Different types of processing units:

The central processing unit (c.p.u) can be any one of the following microprocessor, microcontroller, digital signal processing.

- Among these Microcontroller is of low cost processor and one of the main advantage of microcontrollers is, the components such as memory, serial communication interfaces, analog to digital converters etc.., all these are built on a single chip. The numbers of external components that are connected to it are very less according to the application.
- Microprocessors are more powerful than microcontrollers. They are used in major applications with a number of tasking requirements. But the microprocessor requires many external components like memory, serial communication, hard disk, input output ports etc.., so the power consumption is also very high when compared to microcontrollers.
- Digital signal processing is used mainly for the applications that particularly involved with processing of signals

CHAPTER 6

HARDWARE DESCRIPTION

ESP32 CONTROLLER:

ESP32 is the SoC (System on Chip) microcontroller which has gained massive popularity recently. Whether the popularity of ESP32 grew because of the growth of IoT or whether IoT grew because of the introduction of ESP32 is debatable. If you know 10 people who have been part of the firmware development for any IoT device, chances are that 7−8 of them would have worked on ESP32 at some point. Before we delve into the actual reasons for the popularity of ESP32, let's take a look at some of its important specifications. The specs listed below belong to the ESP32 WROOM 32 variant.

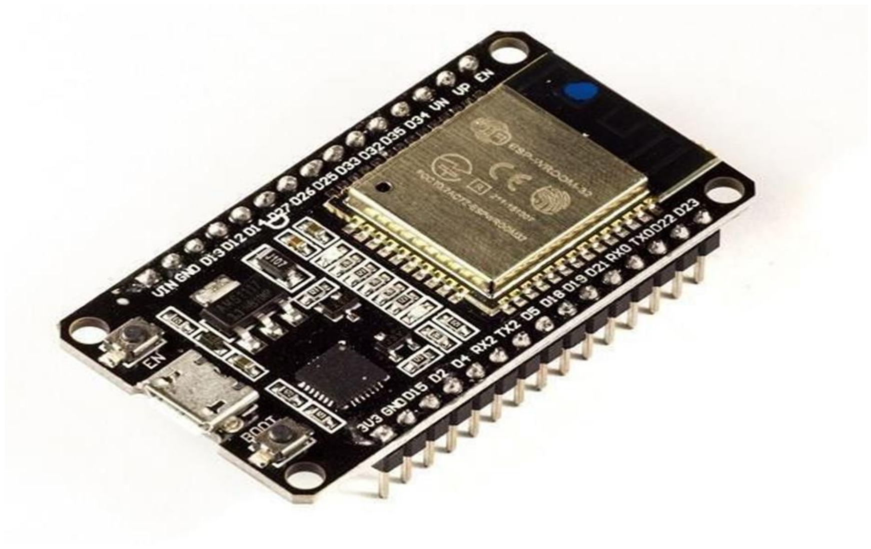


Fig 6.1 : ESP32 CONTROLLER

- Integrated Crystal− 40 MHz
- Module Interfaces− UART, SPI, I2C, PWM, ADC, DAC, GPIO, pulse counter, capacitive touch sensor
- Integrated SPI flash− 4 MB
- ROM− 448 KB (for booting and core functions)
- SRAM− 520 KB
- Integrated Connectivity Protocols− WiFi, Bluetooth, BLE
- On−chip sensor− Hall sensor
- Operating temperature range− −40 − 85 degrees Celsius
- Operating Voltage− 3.3V
- Operating Current− 80 mA (average)

With the above specifications in front of you, it is very easy to decipher the reasons for ESP32's popularity. Consider the requirements an IoT device would have from its microcontroller (µC). If you've gone through the previous chapter, you'd have realized that the major operational blocks of any IoT device are sensing, processing, storage, and transmitting. Therefore, to begin with, the µC should be able to interface with a variety of sensors. It should support all the common communication protocols required for sensor interface: UART, I2C, SPI. It should have ADC and pulse counting capabilities. ESP32 fulfills all of these requirements. On top of that, it also can interface with capacitive touch sensors. Therefore, most common sensors can interface seamlessly with ESP32.

Secondly, the µC should be able to perform basic processing of the incoming sensor data, sometimes at high speeds, and have sufficient memory to store the data. ESP32 has a max operating frequency of 40 MHz, which is sufficiently high. It has two cores, allowing parallel processing, which is a further add-on. Finally, its 520 KB SRAM is sufficiently large for processing a large array of data onboard. Many popular processes and transforms, like FFT, peak detection, RMS calculation, etc. can be performed onboard ESP32. On the storage front, ESP32 goes a step ahead of the conventional microcontrollers and provides a file system within the flash. Out of the 4 MB of onboard flash, by default, 1.5 MB is reserved as SPIFFS (SPI Flash File System). Think of it as a mini−SD Card that lies within the chip itself. You can not only store data, but also text files, images, HTML and CSS files, and a lot more within SPIFFS.

The newer models are available with combined Wi-Fi and Bluetooth connectivity, or just Wi-Fi connectivity. There are several different chip models available, including:

- ESP32-D0WDQ6 (and ESP32D0WD)
- ESP32-D2WD
- ESP32-S0WD
- System in package (SiP) – ESP32-PICO-D4
- ESP32 S series
- ESP32-C series
- ESP32-H series

The ESP32 is most commonly used for mobile devices, wearable tech, and IoT applications, such as Nabto Edge. Moreover, since Mongoose OS introduced an ESP32 IoT Starter Kit, the ESP32 has gained a reputation as the ultimate chip for hobbyists and IoT developers. It’s suitable for commercial IoT, and its capabilities

Regulated Power Supply:

A regulated power supply converts unregulated AC (Alternating Current) to a constant DC (Direct Current). A regulated power supply is used to ensure that the output remains constant even if the input changes. A regulated DC power supply is also known as a linear power supply; it is an embedded circuit and consists of various blocks. The regulated power supply will accept an AC input and give a constant DC output.

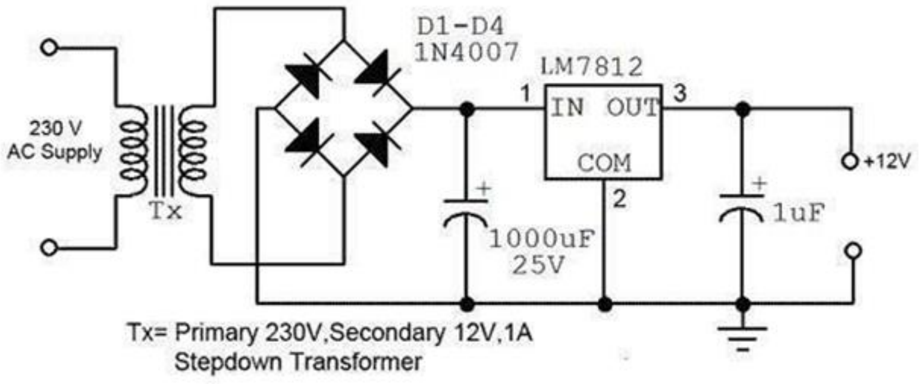


Fig 6.2: Block diagram of power supply

Voltage Regulator:

A voltage regulator (also called a ‘regulator’) with only three terminals appears to be a simple device, but it is in fact a very complex integrated circuit. It converts a varying input voltage into a constant ‘regulated’ output voltage. Voltage Regulators are available in a variety of outputs like 5V, 6V, 9V, 12V and 15V. The LM78XX series of voltage regulators are designed for positive input. For applications requiring negative input, the LM79XX series is used. Using a pair of ‘voltage-divider’ resistors can increase the output voltage of a regulator circuit.

It is not possible to obtain a voltage lower than the stated rating. You cannot use a 12V regulator to make a 5V power supply. Voltage regulators are very robust. The only way to destroy a regulator is to apply reverse voltage to its input. Reverse polarity destroys the regulator almost instantly. Fig: 3.3.11 shows voltage regulator.

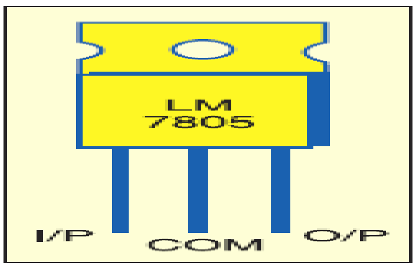


Fig 6.3: Voltage Regulator

LCD DISPLAY:

One of the most common devices attached to a micro controller is an LCD display. Some of the most common LCD's connected to the many microcontrollers are 16x2 and 20x2 displays. This means 16 characters per line by 2 lines and 20 characters per line by 2 lines, respectively.

Basic 16x 2 Characters LCD

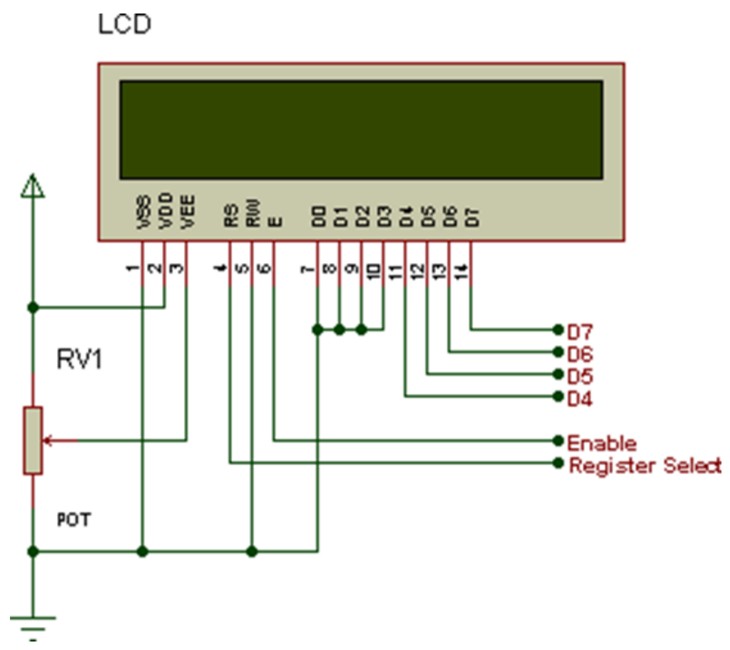


Fig 6.4: LCD Pin diagram

PIN DESCRIPTION:

Mode	Typ	Unit
Transmit 802.11b, CCK 1Mbps, POUT=+19.5dBm	215	mA
Transmit 802.11b, CCK 11Mbps, POUT=+18.5dBm	197	mA
Transmit 802.11g, OFDM 54Mbps, POUT =+16dBm	145	mA
Transmit 802.11n, MCS7, POUT=+14dBm	135	mA
Receive 802.11b, packet length=1024 byte, -80dBm	60	mA
Receive 802.11g, packet length=1024 byte, -70dBm	60	mA
Receive 802.11n, packet length=1024 byte, -65dBm	62	mA
Standby	0.9	mA
Deep sleep	10	uA
Power save mode DTIM 1	1.2	mA
Power save mode DTIM 3	0.86	mA
Total shutdown	0.5	uA

WIFI MODULE:

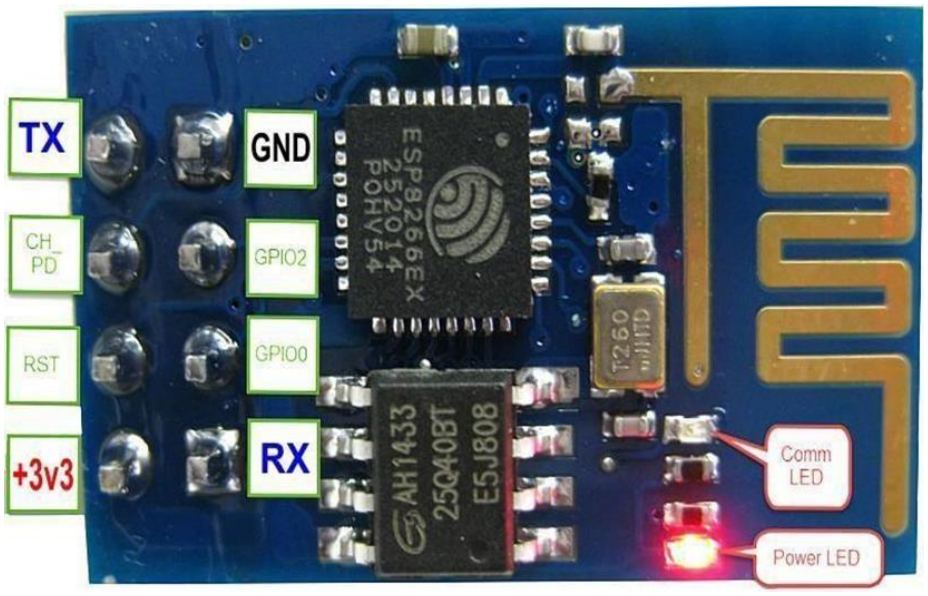


Fig 6.5: Wi-fi module

ESP8266 offers a complete and self-contained Wi-Fi networking solution, allowing it to either host the application or to offload all Wi-Fi networking functions from another application processor.

When ESP8266 hosts the application, and when it is the only application processor in the device, it is able to boot up directly from an external flash. It has integrated cache to improve the performance of the system in such applications, and to minimize the memory requirements.

Alternately, serving as a Wi-Fi adapter, wireless internet access can be added to any microcontroller-based design with simple connectivity through UART interface or the CPU AHB bridge interface.

ESP8266 on-board processing and storage capabilities allow it to be integrated with the sensors and other application specific devices through its GPIOs with minimal development up-front and minimal loading during runtime. With its high degree of on-chip integration, which includes the antenna switch balun, power management converters, it requires minimal external circuitry, and the entire solution, including front-end module, is designed to occupy minimal PCB area.

Sophisticated system-level features include fast sleep/wake context switching for energy-efficient VoIP, adaptive radio biasing for low-power operation, advance signal processing, and spur cancellation and radio co-existence features for common cellular, Bluetooth, DDR, LVDS, LCD interference mitigation.

Applications:

- Medical equipment
- Electronic test equipment
- Industrial machinery Interface
- Serial terminal
- Advertising system
- EPOS
- Restaurant ordering systems
- Gaming box
- Security systems
- R&D Test units
- Climatizing units
- PLC Interface
- Simulators
- Environmental monitoring
- Lab development
- Student projects
- Home automation
- PC external display
- HMI operator interface

LDR Sensor:

A photo resistor or light dependent resistor or cadmium sulfide (CdS) cell is a resistor whose resistance decreases with increasing incident light intensity. It can also be referenced as a photoconductor.

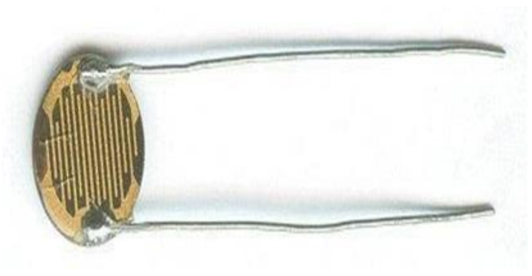


Fig 6.6: LDR (Photoresistor)

The internal components of a photoelectric control for a typical American streetlight:

The photo resistor is facing rightwards, and controls whether current flows through the heater which opens the main power contacts. At night, the heater cools, closing the power contacts, energizing the street light. The heater/bimetal mechanism provides a built-in time- delay.

A photo resistor or light dependent resistor or cadmium sulfide (CdS) cell is a resistor whose resistance decreases with increasing incident light intensity. It can also be referenced as a photoconductor.A photo resistor or light dependent resistor or cadmium sulfide (CdS) cell is a resistor whose resistance decreases with increasing incident light intensity. It can also be referenced as a photoconductor.

A photoelectric device can be either intrinsic or extrinsic. An intrinsic semiconductor has its own charge carriers and is not an efficient semiconductor, e.g. silicon. In intrinsic devices the only available electrons are in the valence band, and hence the photon must have enough energy to excite the electron across the entire band gap. Extrinsic devices have impurities, also called do pants, and added whose ground state energy is closer to the conduction band; since the electrons do not have as far to jump, lower energy photons (i.e., longer wavelengths and lower frequencies) are sufficient to trigger the device. If a sample of silicon has some of its atoms replaced by phosphorus atoms (impurities), there will be extra electrons available for conduction. This is an example of an extrinsic semiconductor.

Applications:

Photo resistors come in many different types. Inexpensive cadmium sulfide cells can be found in many consumer items such as camera light meters, street lights, clock radios, alarms, and outdoor clocks.

They are also used in some dynamic compressors together with a small incandescent lamp or light emitting diode to control gain reduction.

Lead sulfide (PbS) and indium antimonide (InSb) LDR’s (light dependent resistor) are used for the mid infrared spectral region. Ge:Cu photoconductors are among the best far-infrared detectors available, and are used for infrared astronomy and infrared spectroscopy.

ULTRA SONIC SENSOR:

Ultrasonic principle:

Ultrasonic sensors emit short, high-frequency sound pulses at regular intervals. These propagate in the air at the velocity of sound. If they strike an object, then they are reflected back as echo signals to the sensor, which itself computes the distance to the target based on the time-span between emitting the signal and receiving the echo.

As the distance to an object is determined by measuring the time of flight and not by the intensity of the sound, ultrasonic sensors are excellent at suppressing background interference.

Virtually all materials which reflect sound can be detected, regardless of their color. Even transparent materials or thin foils represent no problem for an ultrasonic sensor.

Micro sonic ultrasonic sensors are suitable for target distances from 30 mm to 10 m and as they measure the time of flight they can ascertain a measurement with pinpoint accuracy. Some of our sensors can even resolve the signal to an accuracy of less than 0.18 mm.

Ultrasonic sensors can see through dust-laden air and ink mists. Even thin deposits on the sensor membrane do not impair its function.

Sensors with a blind zone of just 30 mm and an extremely narrow beam spread are finding totally new applications these days: measuring levels in yoghurt pots and test tubes as well as scanning small bottles in the packaging sector - no trouble for our sensors. Even thin wires are reliably detected.

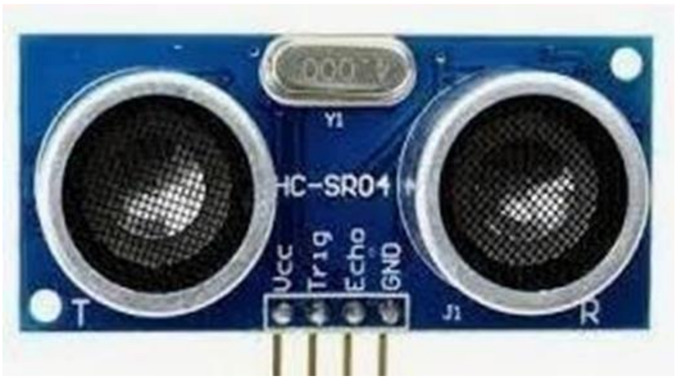


Fig 6.7 : Ultrasonic sensor module

Connection:

The SRF004 must be mounted above the buggy (e.g. by usinga small home-made aluminum bracket (not supplied)). The SRF004 has five solder connections which must be connectedvia wires to the solder joints on the bottom of the buggy PCB.

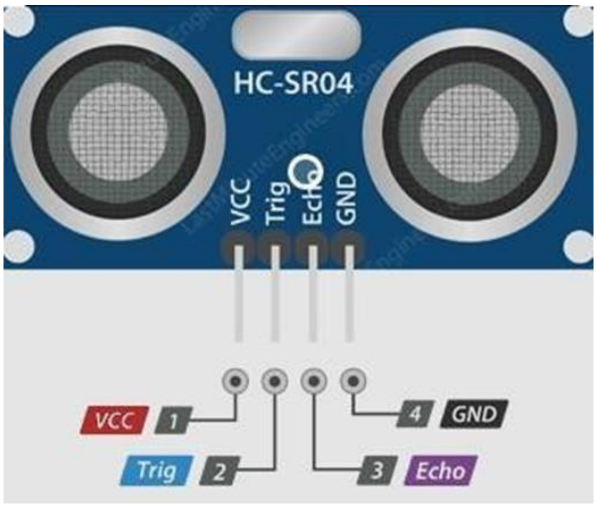


Fig 6.8: Connections Of Ultrasonic sensor module

1. Hole 1 – 5v Supply – to PIC chip leg 14 (V+ Supply)
2. Hole 2 – Echo Output – to PIC chip leg 15 (input 6)
3. Hole 3 – Trigger Input – to PIC chip leg 9 (output 3)
4. Hole 4 – Mode – to PIC chip leg 5 (0V Ground)
5. Hole 5 – 0V Ground – to PIC chip leg 5 (0V Ground)

Note that both holes 4 and 5 must both be connected to 0V.It is recommended that the wire links across the bottom of the Buggy are secured in place using a glue-gun or similar.

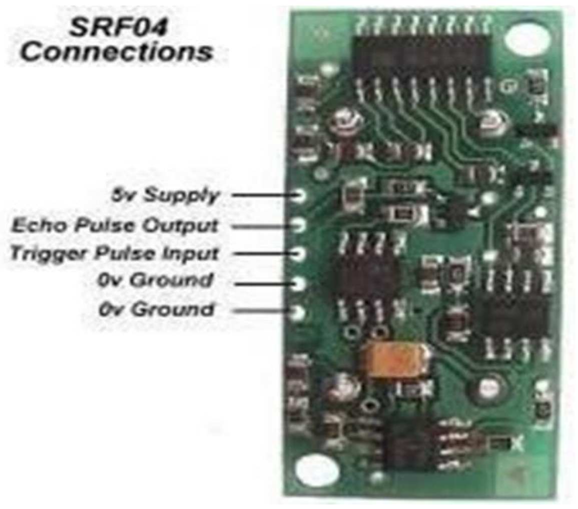


Fig 6.9: Ultrasonic sensors have set new standards in automation

Global System for Mobile Communication (GSM)

Definition:

GSM, which stands for Global System for Mobile communications, reigns (important) as the world's most widely used cell phone technology. Cell phones use a cell phone service carrier's GSM network by searching for cell phone towers in the nearby area. Global system for mobile communication (GSM) is a globally accepted standard for digital cellular communication.

GSM is the name of a standardization group established in 1982 to create a common European mobile telephone standard that would formulate specifications for a pan- European mobile cellular radio system operating at 900 MHz. It is estimated that many countries outside of Europe will join the GSM partnership.

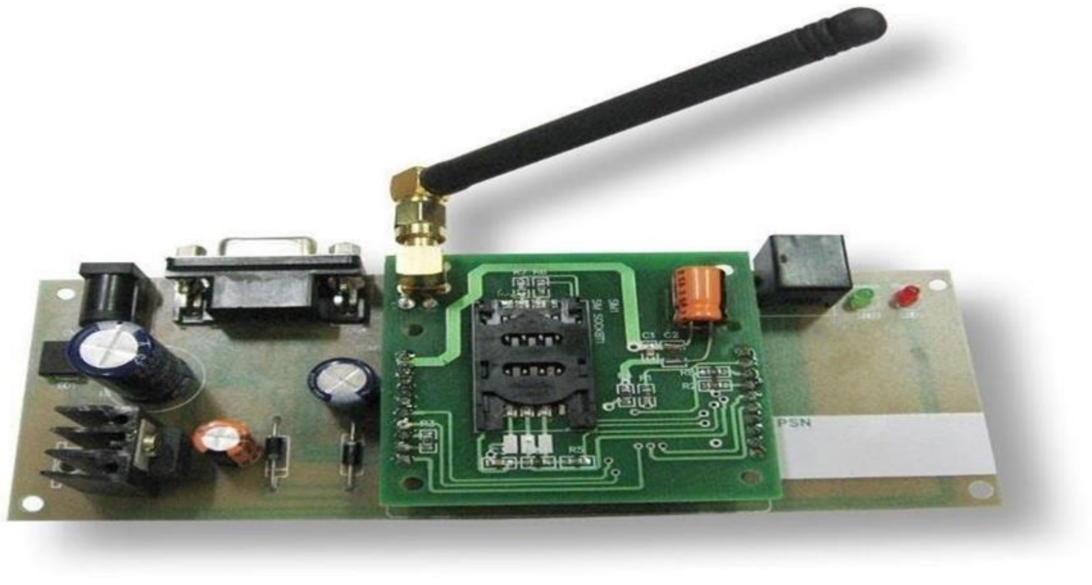


Fig 6.10: Ultrasonic sensors have set new standards in automation

MODEM SPECIFICATIONS:

The SIM300 is a complete Tri-band GSM solution in a compact plug-in module. Featuring an industry-standard interface, the SIM300 delivers GSM/GPRS900/1800/1900Mhz performance for voice, SMS, data and Fax in a small form factor and with low power consumption. The leading features of SIM300 make it deal fir virtually unlimited application, such as WLL applications (Fixed Cellular Terminal), M2M application, handheld devices and much more.

1. Tri-band GSM/GPRS module with a size of 40x33x2.85
2. Customized MMI and keypad/LCD support
3. An embedded powerful TCP/IP protocol stack

Based upon mature and field proven platform, backed up by our support service, from definition to design and production.

General Features:

- Tri-band GSM/GPRS900/1800/1900Mhz
- GPRS multi-slot class 10
- GPRS mobile station class –B
- Complaint to GSM phase 2/2+
- -class 4(2W @900MHz)
- -class 1(1W @/18001900MHz)
- Dimensions: 40x33x2.85 mm
- Weight: 8gm
- 7. Control via AT commands
- (GSM 07.07, 07.05 and SIMCOM enhanced AT commands)
- SIM application tool kit
- • supply voltage range 3.5 v
- Low power consumption
- Normal operation temperature: -20 ’C to +55 ‘C
- Restricted operation temperature : -20 ’C to -25 ‘C and +55 ’C to +70 ‘C
- storage temperature: -40 ‘C to +80 ‘C
- Specifications for Fax:
- Group 3 and class 1
- Specifications for Data:
- GPRS class 10: max 85.6 kbps (downlink)
- PBCCH support
- coding schemes Cs 1,2,3,4
- CSD upto 14.4 kbps
- USSD
- Non transperant mode
- PPP-stack
- Specifications for SMS via GSM/GPRS:
- Point to point MO and MT
- SMS cell broadcast
- Text and PDU mode

- Compatibility:
- At cellular command interface
- Specifications for voice: Tricodec
- -Half rate (HR)
- -Full rate (FR)
- -Enhanced full rate (EFR)
    - Hands free operation (Echo cancellation) Drivers:
- Microsoft windows mobile RIL driver MUX driver
- Interfaces:
- Interface to external SIM 3v 1.8v
- 60 pins board-to-board connector
- Two analog audio interfaces
- Keypad interfaces
- LCD interface
- RTC backup
- AT commands via serial interface
- Dual-Serial interfaces
- Antenna connector and antenna pad
- Approvals:
- FTA
- Local type approval
- CE
- Need of GSM:
- The GSM study group aimed to provide the followings through the GSM:
- Improved spectrum efficiency.
- International roaming.
- Low-cost mobile sets and base stations (BS)

Global Positioning System:

The Global Positioning System (GPS) is a burgeoning technology, which provides unequalled accuracy and flexibility of positioning for navigation, surveying and GIS data capture. The GPS NAVSTAR (Navigation Satellite timing and Ranging Global Positioning System) is a satellite- based navigation, timing and positioning system. The GPS provides continuous three-dimensional positioning 24 hrs a day throughout the world. The technology seems to be beneficiary to the GPS user community in terms of obtaining accurate data up to about100 meters for navigation, meter- level for mapping, and down to millimeter level for geodetic positioning. The GPS technology has tremendous amount of applications in GIS da ta collection, surveying, and mapping. The Global Positioning System (GPS) is a U.S. space-based radio navigation system that provides reliable positioning, navigation, and timing services to civilian users on a continuous worldwide basis -- freely available to all. For anyone with a GPS receiver, the system will provide location and time. GPS provides accurate location and time information for an unlimited number of people in all weather, day and night, anywhere in the world.

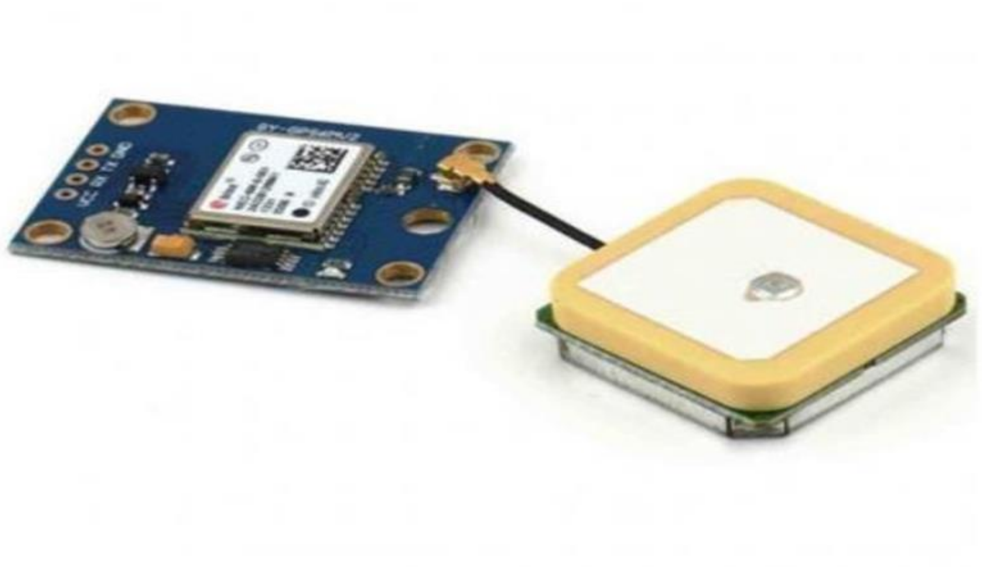


Fig 6.11: GSM Module

The Global Positioning System (GPS) is a satellite-based navigation system made up of a network of 24 satellites placed into orbit by the U.S. Department of Defense. GPS was originally intended for military applications, but in the 1980s, the government made the system available for civilian use. GPS works in any weather conditions, anywhere in the world, 24 hours a day. There are no subscription fees or setup charges to use GPS.

Features NEO-6M GPS Module:

- 5Hz position update rate
- Operating temperature range: -40 TO 85°CUART TTL socket
- EEPROM to save configuration settings
- Rechargeable battery for Backup
- The cold start time of 38 s and Hot start time of 1 s
- Supply voltage: 3.3 V
- Configurable from 4800 Baud to 115200 Baud rates. (default 9600)
- SuperSense ® Indoor GPS: -162 dBm tracking sensitivity
- Support SBAS (WAAS, EGNOS, MSAS, GAGAN

RTC (DS 1302):

Fig 6.12: RTC Module

The DS1302 trickle-charge timekeeping chip contains a real-time clock/calendar and 31 bytes of static RAM. It communicates with a microprocessor via a simple serial interface. The real-time clock/calendar provides seconds, minutes, hours, day, date, month, and year information. Only three wires are required to communicate with the clock/RAM: CE, I/O (data line), and SCLK (serial clock). Data can be transferred to and from the clock/RAM 1 byte at a time or in a burst of up to 31 bytes. The DS1302 is designed to operate on very low power and retain data and clock information on less than 1µW. The DS1302 has dual power pins, one for primary and another for backup.

FEATURES:

1.Real time clock counts seconds, minutes, hours, date of the month, month, day of the week, and year with leap year compensation valid up to 2100

2.31 x 8 RAM for scratchpad data storage

3.Serial I/O for minimum pin count

4.2.0–5.5 volt full operation

5.Uses less than 300 nA at 2.0 volts

6.Single–byte or multiple–byte (burst mode) data transfer for read or write of clock or RAM data

7.8–pin DIP or optional 8–pin SOICs for surface mount

8.Simple 3–wire interface

9.TTL–compatible (VCC = 5V)

10.Optional industrial temperature range –40°C to +85°C

11.DS1202 compatible

12.Real time clock counts seconds, minutes, hours, date of the month, month, day of the week, and year with leap year compensation valid up to 2100

13.31 x 8 RAM for scratchpad data storage

14.Serial I/O for minimum pin count

15.2.0–5.5 volt full operation

16.Uses less than 300 nA at 2.0 volts

17.Single–byte or multiple–byte (burst mode) data transfer for read or write of clock or RAM data

18.8–pin DIP or optional 8–pin SOICs for surface mount

19.Simple 3–wire interface

20.TTL–compatible (VCC = 5V)

21.Optional industrial temperature range –40°C to +85°C

22.DS1202 compatible

23.Added features over DS1202

24.Optional trickle charge capability to VCC1

25.Dual power supply pins for primary and backup power supplies

26.Backup power supply pin can be used for battery or super cap input

27.Additional scratchpad memory (7 bytes)

28.Added features over DS1202

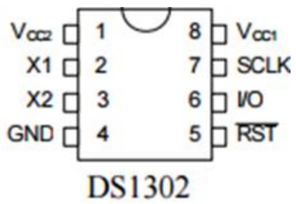
29.Optional trickle charge capability to VCC1

30.Dual power supply pins for primary and backup power supplies

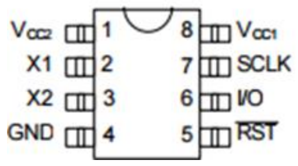
31.Backup power supply pin can be used for battery or super cap input

32.Additional scratchpad memory (7 bytes)

PIN ASSIGNMENT :



DS1302
8-Pin DIP (300-Mil)



DESCRIPTION:

The DS1302 Trickle Charge Timekeeping Chip contains a real time clock/calendar and 31 bytes of static RAM. It communicates with a microprocessor via a simple serial interface. The real time clock/calendar provides seconds, minutes, hours, day, date, month, and year information. The end of the month date is automatically adjusted for months with less than 31 days, including corrections for leap year. The clock operates in either the 24–hour or 12–hour format with an AM/PM indicator.

OPERATION:

The main elements of the Serial Timekeeper are shown in Figure 1: shift register, control logic, oscillator, real time clock, and RAM. To initiate any transfer of data, RST is taken high and 8 bits are loaded into the shift register providing both address and command information. Data is serially input on the rising edge of the SCLK. The first 8 bits specify which of 40 bytes will be accessed, whether a read or write cycle will take place, and whether a byte or burst mode transfer is to occur. After the first eight clock cycles have loaded the command word into the shift register, additional clocks will output data for a read or input data for a write. The number of clock pulses equals 8 plus 8 for byte mode or 8 plus up to 248 for burst mode.

AC 230V COOLER PUMP:

A pump is a device that moves fluids (liquids or gases), or sometimes slurries, by mechanical action. Pumps can be classified into three major groups according to the method they use to move the fluid: direct lift, displacement, and gravity pumps.

Pumps operate by some mechanism (typically reciprocating or rotary), and consume energy to perform mechanical work moving the fluid. Pumps operate via many energy sources, including manual operation, electricity, engines, or wind power, come in many sizes, from microscopic for use in medical applications to large industrial pumps.

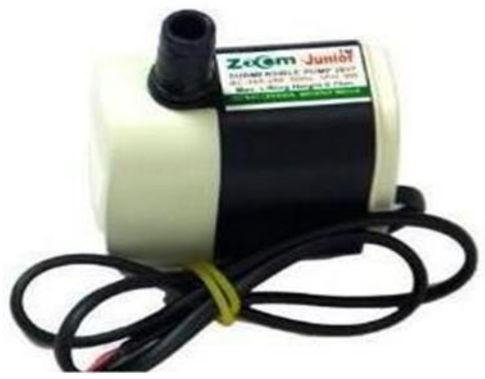


Fig 6.13: Water Dispenser

Mechanical pumps serve in a wide range of applications such as pumping water from wells, aquarium filtering, pond filtering and aeration, in the car industry for water- cooling and fuel injection, in the energy industry for pumping oil and natural gas or for operating cooling towers. In the medical industry, pumps are used for biochemical processes in developing and manufacturing medicine, and as artificial replacements for body parts, in particular the artificial heart and penile prosthesis.

When a casing contains only one revolving impeller, it is called a single-stage pump. When a casing contains two or more revolving impellers, it is called a double- or multi-stage pump.In biology ,many different types of chemical and biomechanical pumps have evolved; biomimicry is sometimes used in developing new types of mechanical pumps.

Rotary positive displacement pumps:

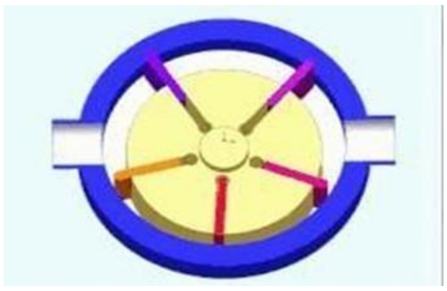


Fig 6.14: Rotary Positive Displacement Pumps

These pumps move fluid using a rotating mechanism that creates a vacuum that captures and draws in the liquid

Advantages: Rotary pumps are very efficient because they can handle highly viscous fluids with higher flow rates as viscosity increases.

Drawbacks: The nature of the pump requires very close clearances between the rotating pump and the outer edge, making it rotate at a slow, steady speed. If rotary pumps are operated at high speeds, the fluids cause erosion, which eventually causes enlarged clearances that liquid can pass through, which reduces efficiency.

Rotary positive displacement pumps fall into three main types:

- Gear pumps – a simple type of rotary pump where the liquid is pushed between two gears
- Screw pumps – the shape of the internals of this pump is usually two screws turning against each other to pump the liquid
- Rotary vane pumps – similar to scroll compressors, these have a cylindrical rotor encased in a similarly shaped housing. As the rotor orbits, the vanes trap fluid between the rotor and the casing, drawing the fluid through the pump.

Specifications:

- Single-phase Water Pump
- In-built Thermal Over Load Protector
- Power Rating: 0.018 kW
- Power Supply: 165V-230V/50HZ
- Motor Power: 0.05 hp

CHAPTER 7

SOFTWARE DESCRIPTION

The Arduino Software (IDE) makes it easy to write code and upload it to the board offline. We recommend it for users with poor or no internet connection. This software can be used with any Arduino board.here are currently two versions of the Arduino IDE, one is the IDE 2.0.0.

7.1 Arduino IDE – Compiler:

here are currently two versions of the Arduino IDE, one is the IDE 1.x.x and the other is IDE 2.x. The IDE 2.x is new major release that is faster and even more powerful to the IDE 1.x.x. In addition to a more modern editor and a more responsive interface it includes advanced features to help users with their coding and debugging.

The following steps can guide you with using the offline IDE (you can choose either IDE 1.x.x or IDE 2.x):

1. Download and install the Arduino Software IDE:
2. Arduino IDE 1.x.x (Windows, Mac OS, Linux, Portable IDE for Windows and Linux, ChromeOS), Arduino IDE 2.x

3.Connect your Arduino board to your device.

4. Open the Arduino Software (IDE).

The Arduino Integrated Development Environment - or Arduino Software (IDE) - connects to the Arduino boards to upload programs and communicate with them. Programs written using Arduino Software (IDE) are called sketches. These sketches are written in the text editor and are saved with the file extension .into Using the offline IDE 1.x.x

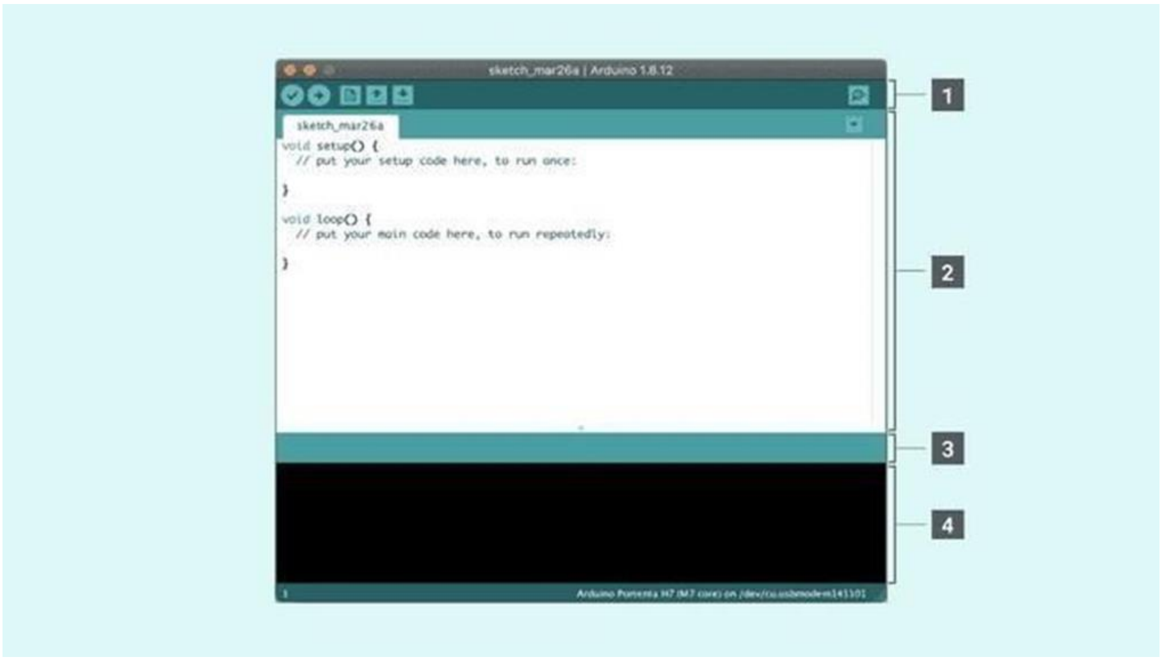
The editor contains the four main areas:

1. A Toolbar with buttons for common functions and a series of menus. The toolbar buttons allow you to verify and upload programs, create, open, and save sketches, and open the serial monitor.

2. The message area, gives feedback while saving and exporting and also displays errors.

3. The text editor for writing your code.

4. The text console displays text output by the Arduino Software (IDE), including complete error messages and other information. The bottom right-hand corner of the window displays the configured board and serial port.



```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

```
Arduino Portenta H7 (M7 core) on /dev/cu.usbmodem141101
```

5. Connect your Arduino or Genuino board to your computer.

6. Now, you need to select the right core & board. This is done by navigating to Tools > Board Arduino AVR Boards > Board. Make sure you select the board that you are using. If you cannot find your board, you can add it from Tools > Board > Boards Manager.

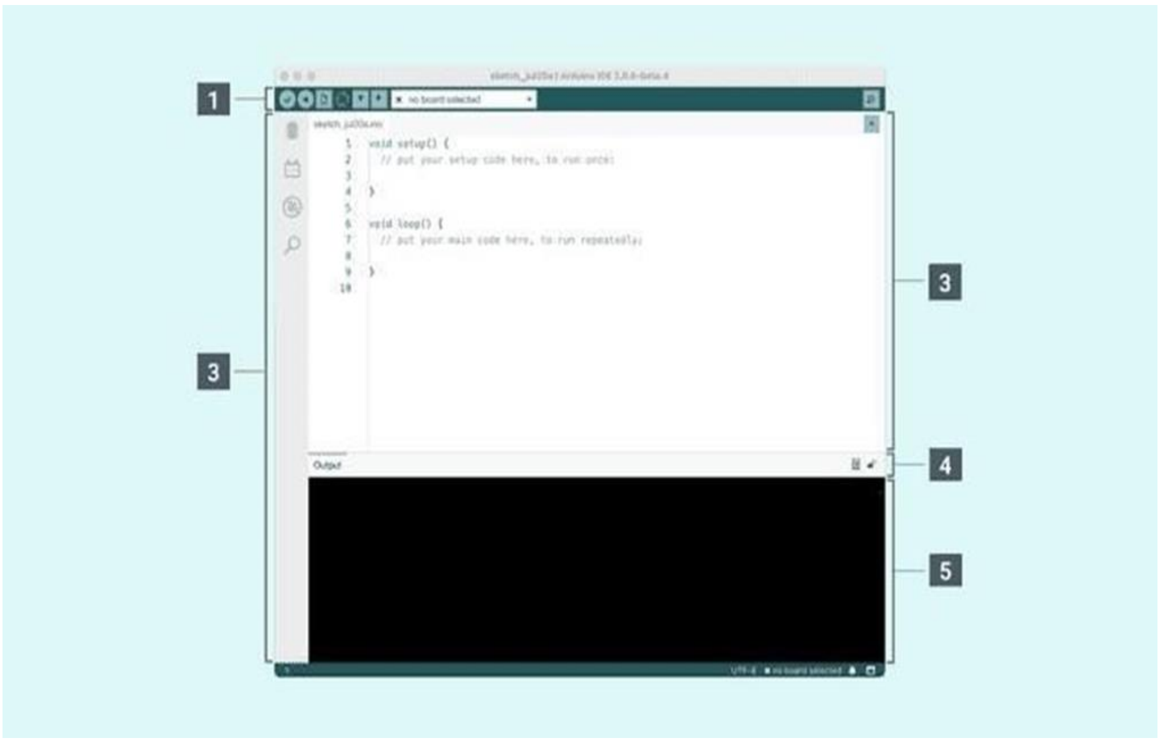
7. Now, let's make sure that your board is found by the computer, by selecting the port. This is simply done by navigating to Tools > Port, where you select your board from the list.

8.Selecting the port

9. Let’s try an example: navigate to File > Examples > 01.Basics > Blink

10. To upload it to your board, simply click on the arrow in the top left corner. This process takes a few seconds, and it is important to not disconnect the board during this process. If the upload is successful, the message "Done uploading" will appear in the bottom output area.

11. Once the upload is complete, you should then see on your board the yellow LED with an L next to it start blinking. You can adjust the speed of blinking by changing the delay number in the parenthesis to 100, and upload the Blink sketch again. Now the LED should blink much faster. The editor contains the four main areas:



```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

1. A toolbar with buttons for common functions and a series of menus. The toolbar buttons allow you to verify and upload programs, create, open, and save sketches, choose your board and port and open the serial monitor.

2. The Sidebar for regularly used tools. It gives you quick access to board managers, libraries, debugging your board as well as a search and replacement tool.

3. The text editor for writing your code.

4. Console controls gives control over the output on the console.

5. The text console displays text output by the Arduino Software (IDE), including complete error messages and other information.

The bottom right-hand corner of the window displays the configured board and serial port.

CHAPTER 8

SOURCE CODE

```
// DECLARATION

#include <ESP32Servo.h>

#include <LiquidCrystal.h> #include <stdio.h>

#include <Wire.h> #include "RTClib.h"

Servo myservo;

LiquidCrystal lcd(13, 12, 14, 27, 26, 25);

#define RXD2 19

#define TXD2 18

#define RXD2 20

#define TXD2 22

int ldr = 2; int pump = 5; int led = 4;

const int trigPin = 16; const int echoPin = 17;

int buzzer = 23;

// SETUP

void setup()

{

Serial.begin(9600);//serialEvent(); Serial2.begin(9600, SERIAL_8N1, RXD2, TXD2);

pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output pinMode(echoPin, INPUT);
// Sets the echoPin as an Input pinMode(buzzer, OUTPUT);

pinMode(led, OUTPUT);pinMode(pump, OUTPUT); pinMode(ldr, INPUT);

digitalWrite(led, LOW);digitalWrite(pump, LOW); digitalWrite(buzzer, HIGH);
```

```
lcd.begin(16, 2); lcd.print(" IOT Urban"); lcd.setCursor(0,1);

    lcd.print("Utilities Management"); delay(2500);

    if(!rtc.begin())

    {

    lcd.print("Couldn't find RTC"); while (1);

    }

    if(! rtc.isrunning())

    {

    lcd.print("RTC is NOT running!");

    }

    lcd.clear(); delay(1500);

    Serial.write("AT\r\n");     delay(3000);//okcheck(); Serial.write("ATE0\r\n");
        okcheck();                   Serial.write("AT+CWMODE=2\r\n");                  delay(3000);
    Serial.write("AT+CIPMUX=1\r\n");delay(3000);//                    okcheck();
    Serial.write("AT+CIPSERVER=1,23\r\n"); //     okcheck(); lcd.clear();

    lcd.print("Waiting For"); lcd.setCursor(0,1); lcd.print("Connection"); do{

    rcv = Serial.read();

    }while(rcv != 'C'); lcd.clear(); lcd.print("Connected"); delay(1500);

    lcd.clear(); gsminit();

    }

    // LOGIC

    void loop()

    {

    lcd.clear(); lcd.setCursor(0,0);lcd.print("U:");
    lcd.setCursor(8,0);lcd.print("L:");

    dist1 = ultra_dist(); lcd.setCursor(2,0);convertl(dist1);

    // delay(200);

    if(dist1 < 10)

    {

    beep();

    String msg_gsm="";

    msg_gsm = "Drainage_Level_Full:" + String(dist1); gsm_send(msg_gsm);

    }

    if(digitalRead(ldr) == LOW)

    {

    lcd.setCursor(10,0);lcd.print("Light");

    }

    if(digitalRead(ldr) == HIGH)

    {

    lcd.setCursor(10,0);lcd.print("Dark ");

    }

    delay(1000);

    lcd.clear();

    DateTime now = rtc.now();

    //yearv, monthv, dayv, hourv, minv, secv year_c = now.year();

     month_c = now.month(); day_c = now.day(); hour_c = now.hour(); min_c = now.minute();
    sec_c = now.second();
```

```
lcd.setCursor(0, 1); lcd.print(hour_c); lcd.print(':'); lcd.print(min_c); lcd.print(':');
lcd.print(sec_c); lcd.print(" ");

lcd.setCursor(0, 0); lcd.print(daysOfTheWeek[now.dayOfTheWeek()]); lcd.print(" ,");

lcd.print(day_c); lcd.print('/'); lcd.print(month_c); lcd.print('/'); lcd.print(year_c);

if(hour_c == hourv1 && min_c == minv1 && sec_c >=0 && sec_c <= 8)

{

digitalWrite(led, HIGH); led_status=1;

}

if(hour_c == hourv2 && min_c == minv2 && sec_c >=0 && sec_c <= 8)

{

digitalWrite(led, LOW); led_status=0;

}

if(hour_c == hourv3 && min_c == minv3 && sec_c >=0 && sec_c <= 8)

{

digitalWrite(pump, HIGH);

}

if(hour_c == hourv4 && min_c == minv4 && sec_c >=0 && sec_c <= 8)

{

digitalWrite(pump, LOW);

}

if(led_status == 1 && digitalRead(ldr) == HIGH)

{

beep();
```

```
String msg_gsm=""; msg_gsm = "Light_Fault"; gsm_send(msg_gsm); led_status=0;
}
while(Serial.available() < 0)
{
char inChar = (char)Serial.read();//delay(5); if(inChar == '@')
{sti=1;
}
if(sti == 1)
{
inputString += inChar;
}
if(inChar == '#')
{sti=0;
stringComplete = true;
}
}
if(stringComplete)
{
lcd.clear();
if(inputString[1] == 'A' || inputString[1] == 'a')
{
hourv1 = (((inputString[2]-48)*10) + (inputString[3]-48)); minv1 = (((inputString[5]-
48)*10) + (inputString[6]-48)); secv1 = (((inputString[8]-48)*10) + (inputString[9]-48));
```

```
lcd.clear();lcd.write("light ON Configured");

}

if(inputString[1] == 'B' || inputString[1] == 'b')

{

hourv2 = (((inputString[2]-48)*10) + (inputString[3]-48)); minv2 = (((inputString[5]-48)*10) + (inputString[6]-48)); secv2 = (((inputString[8]-48)*10) + (inputString[9]-48));

lcd.clear();lcd.write("light OFF Configured");

}

if(inputString[1] == 'C' || inputString[1] == 'c')

{

hourv3 = (((inputString[2]-48)*10) + (inputString[3]-48)); minv3 = (((inputString[5]-48)*10) + (inputString[6]-48)); secv3 = (((inputString[8]-48)*10) + (inputString[9]-48));

lcd.clear();lcd.write("Pump ON Config");

}

if(inputString[1] == 'D' || inputString[1] == 'd')

{

hourv4 = (((inputString[2]-48)*10) + (inputString[3]-48)); minv4 = (((inputString[5]-48)*10) + (inputString[6]-48)); secv4 = (((inputString[8]-48)*10) + (inputString[9]-48));

lcd.clear();lcd.write("Pump OFF Config");

}

if(inputString[1] == 'T' || inputString[1] == 't')

{

yearv = (((inputString[2]-48)*10) + (inputString[3]-48)); monthv = (((inputString[5]-48)*10) + (inputString[6]-48)); dayv = (((inputString[8]-48)*10) + (inputString[9]-48));
```

```
hourv = (((inputString[11]-48)*10) + (inputString[12]-48)); minv = (((inputString[14]-48)*10) + (inputString[15]-48)); secv = (((inputString[17]-48)*10) + (inputString[18]-48));

rtc.adjust(DateTime(yearv, monthv, dayv, hourv, minv, secv));

lcd.clear();lcd.write("D/T Configured");

}

sti=0; inputString = "";

stringComplete = false; delay(1000); lcd.clear();

}

delay(1000);

}

// FUNCTIONS

unsigned int ultra_dist()

{int ud=0; digitalWrite(trigPin, LOW); delayMicroseconds(2); digitalWrite(trigPin, HIGH); delayMicroseconds(10); digitalWrite(trigPin, LOW);

duration = pulseIn(echoPin, HIGH); distanceCm= duration*0.034/2;

ud = distanceCm; return ud;

}

void gsm_send(String gsm_string)

{

lcd.setCursor(15,1);lcd.write('G'); delay(5000);delay(4000);delay(4000); delay(4000); Serial2.write("AT+CMGS=\""); Serial2.write(pastnumber);

Serial2.write("\"\r\n"); delay(3000); Serial2.print(gsm_string); Serial2.write(0x1A);
```

```
delay(5000);delay(4000); delay(4000);delay(4000); lcd.setCursor(15,1);lcd.write(' ');

}

void beep()

{

digitalWrite(buzzer, LOW);delay(2000);digitalWrite(buzzer, HIGH);

}

void converts(unsigned int value)

{

unsigned int a,b,c,d,e,f,g,h;

a=value/10000; b=value%10000; c=b/1000; d=b%1000;

e=d/100; f=d%100;

g=f/10; h=f%10;

a=a|0x30; c=c|0x30; e=e|0x30; g=g|0x30; h=h|0x30;

Serial.write(a); Serial.write(c); Serial.write(e); Serial.write(g); Serial.write(h);

}

void convertl(unsigned int value)

{

unsigned int a,b,c,d,e,f,g,h;

a=value/10000; b=value%10000; c=b/1000; d=b%1000;

e=d/100; f=d%100;

g=f/10; h=f%10;

a=a|0x30; c=c|0x30; e=e|0x30; g=g|0x30; h=h|0x30;

// lcd.write(a); lcd.write(c);
```

```
lcd.write(e);

lcd.write(g);

lcd.write(h);

}

void gsminit()

{

Serial2.write("AT\r\n");    okcheck2();

Serial2.write("ATE0\r\n");          okcheck2(); Serial2.write("AT+CMGF=1\r\n");
    okcheck2();

Serial2.write("AT+CNMI=1,2,0,0\r\n");  okcheck2();
Serial2.write("AT+CSMP=17,167,0,0\r\n"); okcheck2();

lcd.clear();

lcd.print("SEND  MSG  STORE");  lcd.setCursor(0,1);  lcd.print("MOBILE  NUMBER");
do{

rcv = Serial2.read();

}while(rcv == '*'); readSerial(pastnumber);



lcd.clear(); lcd.print(pastnumber); pastnumber[10]='\0';

delay(4000);   delay(4000);   Serial2.write("AT+CMGS=\"");   Serial2.write(pastnumber);
Serial2.write("\"\r\n"); delay(3000); Serial2.write("Mobile no. registered\r\n");

Serial2.write(0x1A);        delay(4000); delay(4000);

}
```

CHAPTER 9

RESULTS

The IOT Smart Municipality appeared on the LCD when we switched on the regulated power source. After connecting to the IOT module, the output can be seen in the image below. The IOT Smart Municipality appeared on the LCD when we switched on the regulated power source. After connecting to the IOT module, the output can be seen in the image below.

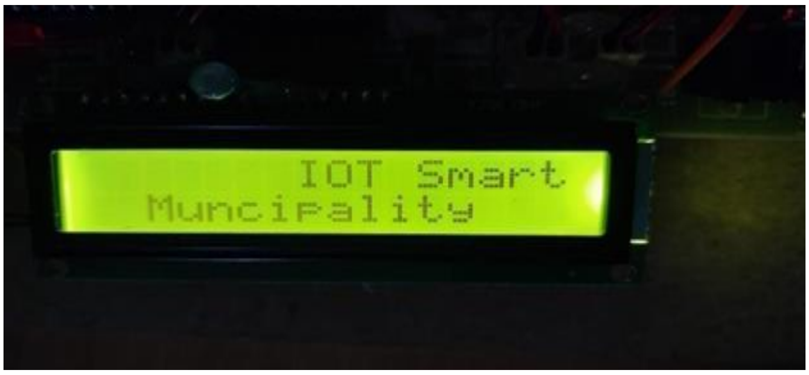


Fig 9.1: LCD Displays The Title Of The Project

When we activated the regulated power supply, the LCD showed the IOT Smart Municipality. The output will be observed in the below figure once we have accessed the IOT module.

The automated street light control system developed in this project delivered accurate and consistent performance during real-time testing. By utilizing the Real-Time Clock (RTC), the system successfully managed the operation of streetlights according to preset timings, ensuring that the lights switched ON during evening hours and OFF during the day. This automation greatly reduced unnecessary power usage and improved overall energy efficiency.

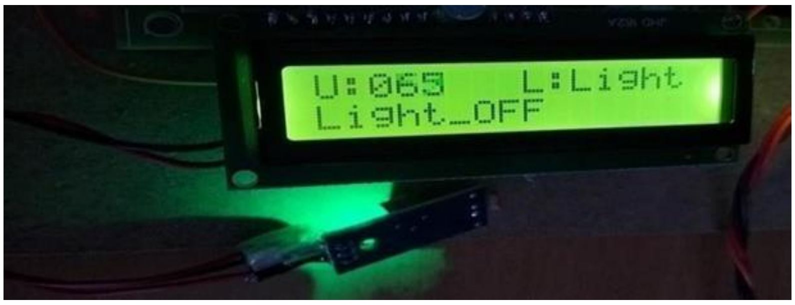
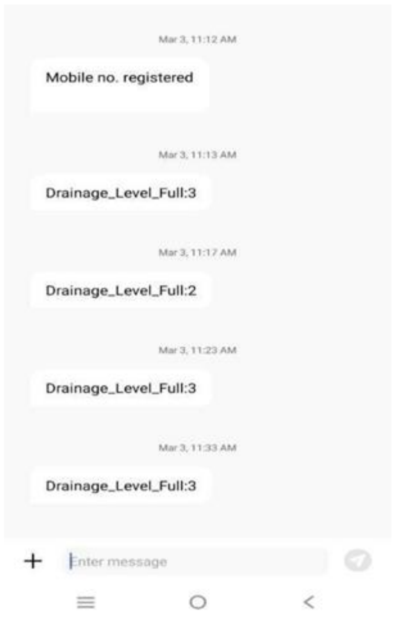


Fig 9.2: LCD Displays Street Light fault detection status

The integration of the LDR sensor played a crucial role in detecting any malfunction in the lighting system. It accurately identified whether the streetlight was functioning when activated, and any failure was promptly signaled through a buzzer and an alert via the GSM module. Additionally, the current status of the lighting system was shown on the LCD screen, helping with on-site monitoring and quick identification of issues. The module operated smoothly under various conditions, demonstrating its potential as a reliable solution for smart and energy-conscious urban infrastructure.

The manhole monitoring module efficiently utilized an ultrasonic sensor to measure water levels inside the drainage system. Integrated with the ESP32 microcontroller, the sensor consistently tracked variations in water height and accurately detected potential overflow or blockage scenarios. When the water level crossed the critical threshold, the system immediately activated a buzzer for local alert and transmitted real-time notifications through the GSM module, sending SMS alerts directly to registered mobile numbers of municipal personnel. This ensured that authorities were instantly informed, allowing them to take timely action and avoid hazardous conditions such as flooding or manhole overflow.



Mar 3, 11:12 AM

Mobile no. registered

Mar 3, 11:13 AM

Drainage_Level_Full:3

Mar 3, 11:17 AM

Drainage_Level_Full:2

Mar 3, 11:23 AM

Drainage_Level_Full:3

Mar 3, 11:33 AM

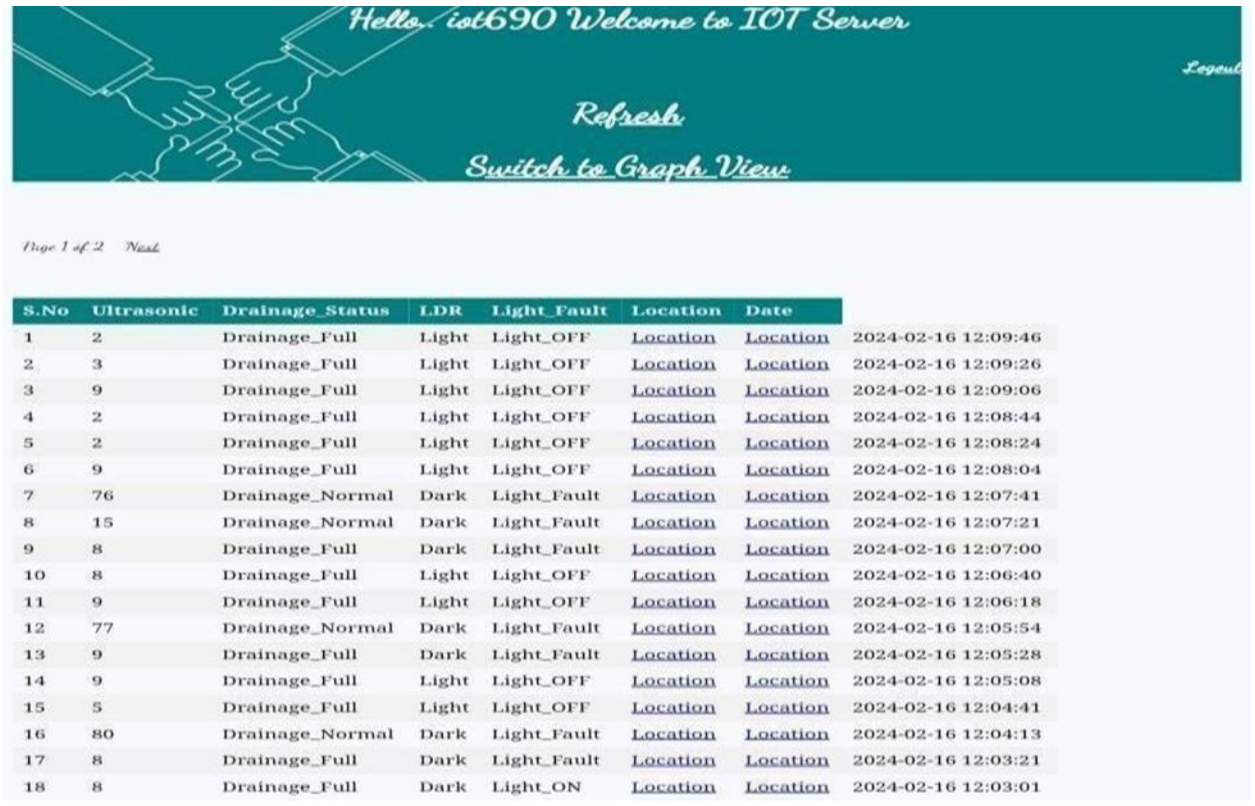
Drainage_Level_Full:3

Fig 9.3: Alerting Messages On Manhole Status Through Mobile Messages

The live data was also reflected on the LCD screen for easy onsite monitoring. The system operated smoothly throughout various testing scenarios and proved to be a dependable, real-time alert mechanism. Its ability to combine sensor-based detection with mobile messaging makes it highly practical for deployment in urban areas where early intervention can prevent public safety risks and infrastructure damage.

Alerting user through gsm module through messaging system in the mobile application which is easily detect the status easily respond to cons

Here, the picture depicts the output of the system that was uploaded to the internet via an ESP8266 IOT. The locations, including the date and time, are given to the municipalities along with the drainage condition and street fault data.



S.No	Ultrasonic	Drainage_Status	LDR	Light_Fault	Location	Date	
1	2	Drainage_Full	Light	Light_OFF	Location	Location	2024-02-16 12:09:46
2	3	Drainage_Full	Light	Light_OFF	Location	Location	2024-02-16 12:09:26
3	9	Drainage_Full	Light	Light_OFF	Location	Location	2024-02-16 12:09:06
4	2	Drainage_Full	Light	Light_OFF	Location	Location	2024-02-16 12:08:44
5	2	Drainage_Full	Light	Light_OFF	Location	Location	2024-02-16 12:08:24
6	9	Drainage_Full	Light	Light_OFF	Location	Location	2024-02-16 12:08:04
7	76	Drainage_Normal	Dark	Light_Fault	Location	Location	2024-02-16 12:07:41
8	15	Drainage_Normal	Dark	Light_Fault	Location	Location	2024-02-16 12:07:21
9	8	Drainage_Full	Dark	Light_Fault	Location	Location	2024-02-16 12:07:00
10	8	Drainage_Full	Light	Light_OFF	Location	Location	2024-02-16 12:06:40
11	9	Drainage_Full	Light	Light_OFF	Location	Location	2024-02-16 12:06:18
12	77	Drainage_Normal	Dark	Light_Fault	Location	Location	2024-02-16 12:05:54
13	9	Drainage_Full	Dark	Light_Fault	Location	Location	2024-02-16 12:05:28
14	9	Drainage_Full	Light	Light_OFF	Location	Location	2024-02-16 12:05:08
15	5	Drainage_Full	Light	Light_OFF	Location	Location	2024-02-16 12:04:41
16	80	Drainage_Normal	Dark	Light_Fault	Location	Location	2024-02-16 12:04:13
17	8	Drainage_Full	Dark	Light_Fault	Location	Location	2024-02-16 12:03:21
18	8	Drainage_Full	Dark	Light_ON	Location	Location	2024-02-16 12:03:01

Fig 9.4: IOT Server Shows Manhole Monitoring Status

The real-time IoT dashboard served as a centralized platform for monitoring the performance and status of the entire urban utility system, including manhole conditions and streetlight functionality. The data captured through various sensors, such as ultrasonic sensors and LDRs, was effectively displayed on the web server with time-stamped records. The ultrasonic sensor accurately reported the drainage status as either "Drainage_Full" or "Drainage_Normal," allowing authorities to assess the risk of overflow. The LDR sensor continuously tracked ambient light conditions and detected streetlight faults by comparing light levels with expected behavior, labeled clearly as "Light_ON," "Light_OFF," or "Light_Fault."

Each entry on the server included location links, making it easy to pinpoint affected areas, and all logs were updated with exact timestamps. This enhanced transparency, accountability, and ease of maintenance. For instance, records show that even when ambient conditions were dark, some lights registered a "Light_Fault," indicating bulb malfunction, which was flagged for attention. Similarly, abnormal ultrasonic readings triggered drainage alerts, which could then be verified and resolved proactively.

The implementation of the water dispenser module in this project has shown promising results in automating public water distribution. The system, driven by an ESP32 microcontroller and relay setup, successfully activated the water pump as per the scheduled timing configured using the Real-Time Clock (RTC).

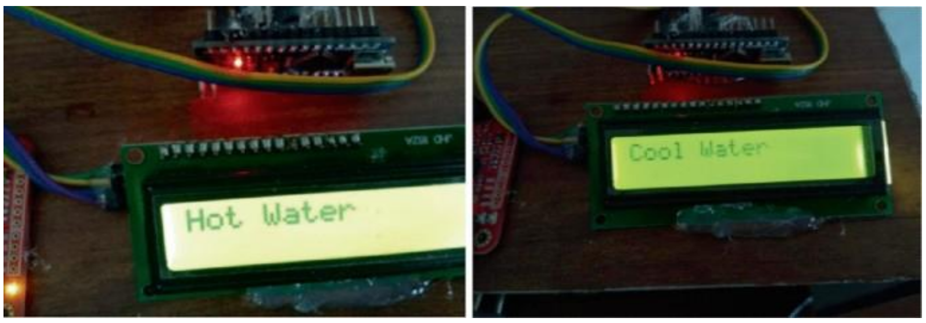


Fig 9.5: LCD Displays Water Dispenser Conditions

Real-time status updates of the pump were made available both through the LCD screen and the IoT cloud interface, offering clear visibility into the system's operation. During testing, the module contributed to substantial water savings by ensuring timely control of the dispenser

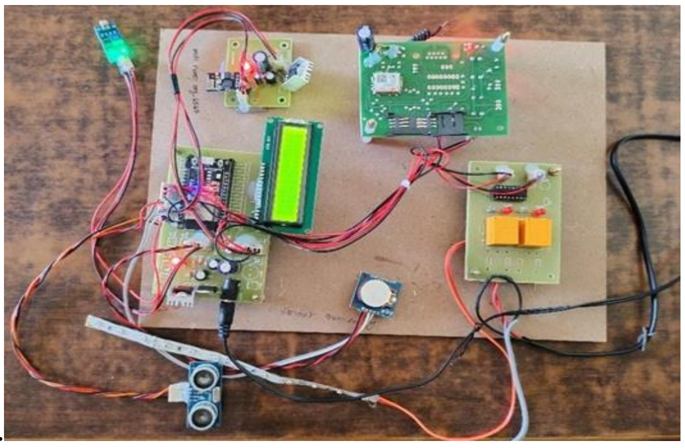

,.

Fig 9.6: IOT BASED URBAN UTILITIES MANAGEMENT KIT

CHAPTER 10

CONCLUSION AND FUTURE SCOPE

Conclusion:

This project successfully demonstrates an IoT-based municipal automation system that enhances water pump control, manhole monitoring, and streetlight management using the ESP32 microcontroller. By integrating IoT connectivity, ultrasonic sensors, an RTC module, and a GSM alert system, the solution ensures real-time monitoring, automated operations, and timely alerts, significantly reducing manual intervention. The system optimizes water distribution, prevents manhole overflows, and improves streetlight efficiency, contributing to better public safety, resource management, and urban sustainability.

Future scope:

Future enhancements will focus on AI-driven predictive maintenance and advanced data analytics to further enhance system reliability and efficiency, making it a scalable and smart solution for modern urban infrastructure.

1. AI-Powered Predictive Maintenance

- ➢ Use machine learning models to predict failures in water, electricity, or waste systems.
- ➢ Enable preemptive alerts for maintenance before breakdowns occur.

2. Digital Twin Integration

- ➢ Create a real-time digital replica of the city's utility infrastructure.
- ➢ Simulate changes, visualize faults, and test scenarios virtually.

3. Blockchain for Secure Data Transactions

- ➢ Ensure data integrity and transparency in utilities billing, usage records, and fault reports.

4. Smart Water and Energy Meters

- ➢ Enable remote tracking, auto-billing, and leak/fraud detection.
- ➢ Encourage dynamic pricing based on usage.

REFERENCES:

[1]Kumar, Manish, Ravinder Kumar, and Ritula Thakur. "Zigbee Based Smart Street Light Control System Using Lab VIEW." Int. Journal of Innovative Research in Science, Engineering and Technology 5.4 (2016).

[2]Abhishek, M., K. Chetan, and K. Arun Kumar. "Design and implementation of traffic flowbased street light control system with effective utilization of solar energy." Int. J. Sci. Eng. Adv. Technol 3.9 (2015): 195-499.

[3]Mohamed, Samir A. Elsagheer. "Smart street lighting control and monitoring system for electrical power saving by using VANET." Int'l J. of Communications, Network and System Sciences 6.08 (2013): 351.

[4]Amin, Chaitanya, Paridhi Holani AshutoshNerkar, and Rahul Kaul. "GSM based autonomous street illumination system for efficient power management." International Journal of Engineering Trends and Technology 4.1 (2013): 54-60.

[5]K.Y.Rajput, GargeyeeKhatav, Monica Pujari and Priyanka Yadav" Intelligent Street Lighting System Using GSM" International Journal of Engineering Science Invention Volume 2 Issue 3 , March, 2013.

[6]J.Arthi, W.Lydiapreethi, B. Gunasundari," IOT BASED SMART LED STREET LIGHTING SYSTEM" IJRTI | Volume 2, Issue 4 | ISSN: 2456-3315.

[7]Mohmmad Ashaq Sofi, Anima Nanda, Gulzar Ahmed Rather, Mohd Abass Sofi, "IN VITRO ANTIMICROBIAL ACTIVITY OF METHANOLIC LEAF EXTRACT OF CYNARA SCOLYMUS (ARTICHOKE) AGAINST SELECTED HUMAN PATHOGENS" , International Journal Of Advance Research In Science And Engineering http://www.ijarse.com IJARSE, Volume No. 10, Issue No. 04, April 2021 ISSN-2319-8354(E).

[8]Salvi, Ritisha, et al. "Smart Street light using Arduino uno microcontroller." Int. J. Innov. Res. Compute. Common. Eng 5 (2017): 5203-5206.

[9]Velaga, Nagendra R., and Amit Kumar. "Techno-economic evaluation of the feasibility of a smart street light system: a case study of rural India." Procedia-Social and Behavioral Sciences 62 (2012): 1220- 1224

[10]T. Baranidharan, A.Chinnadurai, R.M.Gowri, J. Karthikeyan. “Automated Water Distribution System Using PLC And SCADA” IJEEE, Volume 07, Issue 01, Jan- June 2015

[11]Mr. Prashantpalkar, Prof. (Dr.) ShrinivasPatil, Prof. Mrs. PoojaBelagali, Mr. AshishChougule. “Automation in Drinking Water Supply Distributed System and Testing of Water” IOSR Journal of Electronics & Communication Engineering (IOSR-JECE)

[12]Sujeet Rote, AdarshMourya, Tanmay Oak, Prof. Abhimanyu Yadav. “Automatic Water Distribution System using PLC” International Journal of Engineering Research & Technology (IJERT) (ISSN: 2278-0181) Vol. 6 Issue 04, April-2017