**IOT MANHOLE MONITORING &STREET LIGHT CONTROL SYSTEM WITH FAULT DETECTION & REPORTING FOR MUNCIPALITIES**

**ABSTRACT**

This project involves the design and implementation of an automated street light control system and an underground drainage monitoring system using IOT technology. Besides this, it can also check the status of the manhole lid. The system's low maintenance and low cost are some of its main advantages. This system can monitor the status of the street lights and manholes in real time. It can also notify the officials through a text message.

The IoT (Internet of Things) is a blooming technology that mainly concentrates on the interconnection of devices to one another and the people. The world is getting smarter these days, and people are drawn to the word "Smart". Given that India is one of the world's fastest-growing tech markets, we are incorporating a smart framework into the switch. The project's goal is to provide automatic control and fault detection for street light. The lights are switched ON/OFF automatically based on the intensity of sunlight using the LDR sensor.

Smart underground infrastructure is an important feature to be considered while implementing Drai Drainage system monitoring plays a vital role in keeping the city clean and healthy. Since manual monitoring is incompetent, this leads to slow handling of problems in drainage and consumes more time to solve. To mitigate all these issues, the system using a wireless sensor network, consisting of sensor nodes is designed. The proposed system is low cost, low maintenance, IoT based real time which alerts the managing station through a message when any manhole crosses its threshold values. This system reduces the death risk of manual scavengers who clean the underground drainage and also benefits the public.

**CHAPTER 1**

**INTRODUCTION**

An embedded system is a particular type of computer system that is primarily made to carry out several activities, including accessing, processing, storing, and controlling the data in various electronics-based systems. Embedded systems are made up of both hardware and software, with the software—often referred to as firmware—being integrated directly into the hardware. These systems' ability to provide the o/p within the allotted time is one of their most crucial qualities. Embedded systems assist in improving the accuracy and convenience of the task. So, both basic and complex gadgets routinely use embedded systems. The principal real-world uses for embedded systems are in a variety of appliances, including microwaves, calculators, TV remote controls, home security systems, and local traffic management systems.

An integral part of any drainage system is the access points into it when it comes to cleaning, clearing, and inspection. Metropolitan cities have adopted underground drainage systems and the city’s municipal corporation must maintain its cleanliness. If the sewage maintenance is not proper, ground water gets contaminated causing infectious diseases. Blockages drains during monsoon season, causing problems in the routine of the public. Hence, there should be a facility in the city’s corporation, which alerts the officials about blockages in sewers, their exact location. It mainly acknowledges in the field of alerting the people about the gas explosion, increase in the water level and the temperature level. It uses IoT to make the drainage monitoring system in a highly automotive by using sensor for detecting and sending alerts through GSM and GPS module to the authorities. This project overcomes the demerits by detecting drainage water blockage by installing water flow rate sensors at the intersection of nodes. When there is a blockage in a particular node, there is variation in the flow of drainage water which when cross the set value will display the alert in the managing station. Also other demerits are solved by detecting.

Temperature variations inside the manhole and alerting the same to the managing station. Also, flow rate sensors are used to detect the overflow of the drainage water and alert the same to the managing station through automatic message. Maintenance of manholes manually is tedious and dangerous due to the poor environmental conditions inside, so the main focus of this project is to provide a system which monitors water level, atmospheric temperature, water flow and toxic gases. If drainage gets blocked and sewage water overflows, it is sensed by the sensors and a message is sent to the municipality. It is, therefore, dangerous to go inside the manholes for inspection of its current state. To solve all the problems related to underground sanitation, a remote alarm system is necessary for transmitting data collected by the sensors set inside the manhole to the managing station. This includes components such as controller, memory, transceiver and battery to supply power. Manhole detection and monitoring system using IOT it is a very useful system to all of us by this we detect manhole condition in this system.

We used the different components like water flow sensor ,gas sensor , temperature and humidity sensor . This project overcome the demerit of paper by detecting drainage water flow speed rate by installing water flow rate sensor at the intersection of nodes when there is a blockage in a particular road there is variation in the flow of drainage in water which when across the seat value will display the alerts in the managing station by the system. we protect the health of municipality working staff. In this system we use different components this components is very high output and input components and very efficiency component buy this components and this system we detect any problem occur in manhole without any man. For a clean and healthy environment, many Indian cities have an underground drainage system that is controlled by the Municipal Corporation. The water in the drainage system is occasionally mixed with pure water due to poo upkeep Infections and diseases can be spread through the drainage system. Because of climate change, drainage is affected throughout the year, and the environment is dynamic, people's daily lives are disrupted. To fix all drainage system concerns and to send Blynk notifications to the municipal corporation informing them of the state of the drainage system so that officials can take the necessary steps to restore the drainage system. A gas sensor was used to detect the gas produced within the biowaste drainage system, preventing it from escaping. The pressure inside the drainage system produced an explosion.

The Internet of Things (IoT) is a network of physical objects that link to each other. that enables them to connect with one another. The IoT allows for remote sensing and monitoring of machines. It is a sophisticated Artificial intelligence is used in an automation and analytics environment to deliver creative and automated products and services. There frameworks have more accountability, power, and efficiency. IoT has a variety of automation applications, such as smart homes, smart parking, smart highways, and smart lighting

In our country, the corporation street light (HID lamps) consumes more energy, most of the time street lights are switched ON when they are not in use (roads are bare) and there are chances we forget to switch them off and also, we have all seen street light turned ON during the day. However, with the increasing importance for energy conservation and well maintenance are leads to protection of the natural resources for the future.

In order to overcome this issue, A smart street light (LED Lights) system can be used to replace conventional street lamp like HID (High-intensity discharge) lamps. Power savings due to increased current luminous efficiency, lower operating costs, high color building index, accelerated start-up, and durability are all benefits of LED lights over conventional technologies. It also makes out the Fault in any of the street light. Since the resources we depend on, such as hydro, thermal, and coal, are finite. Introducing energy-saving elements such as LDR, Relays and LEDs can light up a wide area with high-intensity light whenever required is the primary goal of the device. The Relay is used as an automatic switch that eliminates almost all manual labour. Here in this project, we have also used IR (Infrared) sensors, that detect the motion of an objects, LDR (Light Dependent Resistor) is also known as photo resistor, this resistor works on the principle of photo conductivity. GSM (Global System for Mobile Communication) module is used for texting message and sending to required mobile number.

Here the street light intensity is controlled by the controller when the LDR senses the dark the lights turn ON automatically, but lights glow as dim, as soon as the IR senses the moving object the street light glow as bright, if it fails to glow, the GSM module send the message to respective authorized mobile number.

Traditional street lighting systems often face challenges such as excessive energy consumption, inefficient maintenance practices, and delayed response to faults. The Street Light Control System addresses these issues by introducing a dynamic and responsive approach to lighting management. By integrating sensors, communication devices, and data analytics, the system allows for the automatic adjustment of lighting levels based on factors such as traffic patterns, weather conditions, and time of day. This not only enhances energy efficiency but also contributes to cost savings and environmental sustainability.

One of the key features of this system is its advanced fault detection and reporting capabilities. Through the deployment of sensors and monitoring devices, the system can promptly identify issues such as malfunctioning lights, electrical faults, or physical damage. In the event of a fault, the system generates real-time alerts and reports, facilitating swift response from municipal maintenance teams. This proactive approach minimizes downtime, reduces the risk of accidents related to poorly lit areas, and streamlines the overall maintenance process.

In addition to its energy optimization capabilities, the system is equipped with advanced Fault Detection & Reporting mechanisms. Sensors and monitoring devices are integrated to promptly detect faults or malfunctions in individual street lights, such as bulb failures, electrical issues, or damaged components. The incorporation of automated reporting ensures that municipalities receive detailed insights into the status of each street light, including energy consumption metrics and identified faults.

Moreover, the Street Light Control System offers predictive maintenance capabilities, leveraging historical data and predictive analytics to anticipate potential failures. By scheduling preventive maintenance based on these predictions, the system reduces the likelihood of unexpected downtime and ensures a more reliable and resilient street lighting network.

To facilitate seamless management, the system provides a user-friendly interface, making it accessible to municipal administrators with varying levels of technical expertise. In conclusion, the Street Light Control System with Fault Detection & Reporting represents a comprehensive and intelligent solution for municipalities looking to modernize their street lighting infrastructure, promising a safer, more energy-efficient, and sustainable urban environment.

**PROBLEM STATEMENT**

Developing an integrated IoT solution that combines sensors for manhole monitoring, smart street light controllers, and a centralized platform for fault detection and reporting. The system should enable real-time monitoring of manhole conditions, adaptive control of street lights, and automated fault detection with timely reporting. This solution aims to improve municipal infrastructure management, reduce maintenance costs, enhance public safety, and contribute to energy efficiency in urban areas.

Municipalities face challenges in efficiently managing urban infrastructure, particularly in monitoring the condition of manholes and controlling street lights. The existing systems often lack real-time data, leading to delayed responses to faults, increased maintenance costs, and inefficient energy usage. To address these issues, there is a need for an integrated IoT Manhole Monitoring & Street Light Control System with Fault Detection & Reporting for Municipalities.

**1)Manhole Condition Monitoring:**

Lack of real-time data on the condition of manholes.Inability to detect issues such as blockages, overflows, or structural damage promptly.Delayed response to manhole-related problems, leading to environmental and safety concerns.

**2)Street Light Control:**

Inefficient energy usage due to fixed schedules or manual control.Difficulty in promptly identifying and addressing malfunctions or outages.Inability to adapt to changing environmental conditions, leading to unnecessary energy consumption.

**3)Fault Detection & Reporting:**

Absence of a systematic approach for detecting faults in manholes and street lights.Limited capability to generate real-time reports on faults and their locations.Lack of automated notification systems for timely response and preventive maintenance.

**4)Data Integration and Accessibility:**

Inconsistent data integration across various municipal systems.Limited accessibility to relevant data for informed decision-making.Absence of a centralized platform for monitoring and managing both manholes and street lights.

**5)Cost Inefficiencies:**

High maintenance costs due to reactive rather than proactive maintenance.Unoptimized energy costs for street lighting.Increased operational expenses resulting from delayed fault detection and resolution.

**AIM:**

The aim of anIoT Manhole Monitoring and Street Light Control System with Fault Detection & Reporting for Municipalities seeks to revolutionize urban infrastructure management. For manhole monitoring, the system focuses on proactive maintenance, utilizing real-time data todetect blockages, overflow, or structural issues, thereby minimizing downtime and enhancing public safety.

**OBJECTIVES:**

* **Preventive Maintenance:** Implement real-time monitoring of manholes to detect issues such as blockages, overflow, or structural damage before they become major problems.
* **Reduce Downtime:** Minimize the time a manhole is out of service by quickly identifying and addressing any faults or malfunctions.
* **Energy Efficiency:** Optimize street lighting schedules and brightness levels based on real-time conditions such as ambient light, weather, and pedestrian activity, leading to energy savings.
* **Remote Control:** Enable municipalities to remotely control street lights, allowing for immediate adjustments or turn-offs during low-traffic periods.
* **Early Detection:** Employ sensors and analytics to identify faults or malfunctions in both manholes and street lights as soon as they occur.
* **Optimization:** Continuously optimize system parameters based on historical data to enhance overall performance and efficiency.
* **Public Safety:** Ensure the safety of pedestrians and motorists by promptly addressing issues such as malfunctioning street lights or hazardous manhole conditions.
* **Security Monitoring:** Incorporate security features to detect unauthorized access to manholes or tampering with street light infrastructure.
* **Operational Efficiency:** Improve operational efficiency by reducing the need for routine manual inspections through automated monitoring systems.
* **Resource Optimization:** Allocate resources more effectively by prioritizing maintenance efforts based on real-time data and criticality.
* **Waste Reduction:** Minimize environmental impact by addressing issues proactively, reducing the likelihood of emergencies and resulting in less waste.
* **Secure Communication:** Implement robust security measures to protect the data transmitted between devices and the central monitoring system.
* **Privacy Compliance:** Ensure compliance with privacy regulations and standards to protect the personal information of residents while collecting and analyzing data.
* **Public Awareness:** Utilize the system to enhance public awareness regarding the importance of proper waste disposal and conservation of energy.

**CHAPTER2**

**LITERATURE REVIEW**

**S. Sulthana et.al (2020) [1]** Everyone has the right to live in a healthy environment. Flooding due to obstructed drains is a common phenomenon in Bangladesh and many other developing countries, leading to unhygienic surroundings. The air becomes poisoned due to sewage gas, resulting in health issues. Stagnant water on roads contributes to the growth of Aedes mosquitoes. Manual tracking of drain conditions is challenging, and issues often become apparent only after widespread flooding occurs. To address this, the paper proposes a warning system that utilizes GSM techniques and IoT. The system employs sensors such as MQ135 for sewage gas, an ultrasonic sensor for measuring sewage distance, and a water level sensor to monitor water flow. When the water level reaches a certain threshold, the system sends a text message to the authorities, pinpointing the affected areas using GPS coordinates. Additionally, real-time data can be accessed by both authorities and the public through an online website implemented using NodeMCU.

**A. Pendharkar et.al (2020) [2]**This project aims to develop an **IoT-based technology** for monitoring sewage systems. The system detects humidity levels, temperature, and gas mixtures in real-time. It utilizes various sensors to measure gas concentrations and dynamically track changes. Specifically, it focuses on ensuring the safety of workers who operate under severe conditions. When gas levels exceed predefined thresholds, the system sends alerts to authorized personnel via connected mobile devices. Additionally, live video streaming allows monitoring for blockages.

**U. Andrijašević et.al (2020) [3]** this paper, the machine learning algorithm for **lid opening detection in telecommunication manholes**is presented. Telecommunication manholes network is equipped with smart Internet of Things (IoT) devices that use multiple sensors to monitor the system and detect various states of a manhole, such as lid opening, light detection, and object or person presence. The machine learning algorithm, developed with a recurrent neural network (RNN), specifically **long short-term memory (LSTM)**, detects whether the lid is open or closed based on the data generated by the IoT manhole monitoring system.

**P. Bhosale et.al (2021) [4]** smart cities, the **Internet of Things (IoT)** plays a crucial role in providing seamless and unique access to public resources, aiming for better utilization and optimization of power, water, and transportation facilities. Urban India faces a water crisis, with approximately **80% of sewage entering lakes, rivers, and groundwater untreated** due to ineffective wastewater infrastructure management. This research focuses on monitoring sewage infrastructure to predict blockages and prevent them from escalating into **Sanitary Sewer Overflows(SSO)**.The proposed **IoT-based system** tackles this issue comprehensively. It monitors the working of access manholes in a network, collecting multiple data parameters generated in and around these manholes. When the development of a blockage is detected, the system sends alerts. This geospatial data, along with the alerts, is observed via a centralized visual dashboard.

**M. Aarthi et.al (2021) [5]**The issue of **drainage overflow** is a significant concern in major cities, often caused by blockages. If left unresolved, it can lead to unhygienic conditions and health problems for nearby residents. To address this, an innovative system has been proposed. Additionally, a manhole monitoring system is integrated. When water flow is hindered due to sewage blockage, foul odors emanate from the manhole. These odors are produced by dangerous gases associated with stagnant dirty water. This system contributes to efficient sewage management, preventing blockages and promoting public health.

**S.A.Shamikh et.al (2017) [6]** The underground drainage system is a crucial component of urban infrastructure, serving as the lifeline of cities. However, manual management of drainage systems is inefficient, and maintaining clean and functional underground systems poses challenges. Accurate assessment of manhole covers, detection of blockages, and monitoring of toxic gases are essential for public health and safety. This project focuses on monitoring and managing underground and road-sided drainage systems using **IoT**and **Raspberry Pi.** his innovative system contributes to efficient sewage management, early detection of issues, and healthier urban environments.

**Muragesh Sk et.al (2015) [7]** has revolutionized real-world interactions by connecting physical objects and devices to sensor networks. In the context of urban infrastructure, efficient drainage management is crucial for maintaining cleanliness and preventing contamination. This paper presents the implementation and design of an **Underground Drainage and Manhole Monitoring System (UDMS)**for IoT applications. The key considerations in this design include low cost, low maintenance, rapid deployment, a high number of sensors, long life, and high-quality service. Efficient monitoring of manholes using sensors contributes to healthier urban environments and proactive maintenance.

**Lazarescu, M.T., et.al (2013) [8]** designed the **Internet of Things (IoT)** provides a virtual view, via the Internet Protocol, to a wide variety of real-life objects, ranging from cars and buildings to trees in a forest. Wireless sensor networks (WSNs) are well-suited for long-term environmental data acquisition, making them ideal for representing IoT applications. This paper presents the functional design and implementation of a complete WSN platform that can be used for a range of long-term environmental monitoring IoT applications. This innovative WSN platform contributes to efficient and cost-effective environmental monitoring, supporting a healthier and more sustainable future.

**Mr.Amey et.al (2016) [9]**The street lighting system consumes a substantial amount of electric power. Given the expensive cost per unit and limited resources for power generation, it is essential to explore energy-saving solutions. The existing street lighting systems often lead to unnecessary power consumption due to continuous lighting throughout the night. To address this, an **Automatic Street Light Control System** is proposed, which dynamically switches ON and OFF the lights based on traffic and light intensity.This innovative system contributes to energy-efficient street lighting, promoting sustainability and cost savings.

**Chaitanya Amin et.al (2013) [10]** This paper efficiently defines the control of street lighting systems and thereby **saves electricity**, which is a major concern worldwide. It also describes the use of **wireless sensor networks (WSN)** using **GSM** for streetlight monitoring and control. In summary, this innovative system contributes to efficient streetlight management, energy savings, and improved urban environments.

**Abdul Latif Saleem et.al (2015) [11]** Street light is a raised source of light that is commonly used along walkways and streets when the surrounding turns dark. Nowadays, most of the existing street light systems are wired which are not only difficult to construct but also has poor flexibility. To overcome this, wireless system is required. In this paper, we are using GSM technology which uses power efficiently by remotely monitoring and controlling the system. This system will ease the fault detection and maintenance. System allows us to make the most efficient use of the energy received from the sun to power street lights. Solar energy is collected with the aid of solar panel and battery is charged during day time and this energy is used to power street lights during night. Developed intelligent system turns the light ON and OFF depending on the vehicle or pedestrian movement, Real Time Clock and light intensity at the same time. Microcontroller processes the information from the sensors and is transferred to nearby control terminal (Base station with Raspberry PI as a compute module) to monitor the status of the street lamp using GSM technology via Short Message Service (SMS). Designed system is visualized by creating Graphical User Interface (GUI). Thus, the implementation of such system will result in energy saving, lower cost of maintenance, increased lifespan and hence the enhanced performance of the system.

**Gargeyee Khatav et.al (2013) [12]** Conventional street lighting systems in areas with a low frequency of passersby are online most of the night without purpose. The consequence is that a large amount of power is wasted meaninglessly. With the broad availability of flexible-lighting technology like light-emitting diode lamps and everywhere available wireless internet connection, fast reacting, reliably operating, and power-conserving street lighting systems become reality. The purpose of this work is to describe the Intelligent Street Lighting (ISL) system, a first approach to accomplish the demand for flexible public lighting systems.

**V.Sumathi et.al (2013) [13]** this paper, a new innovative street light system with optimized street light management and efficiency is presented. The system utilizes various sensors to control and guarantee a better efficient system. By implementing this system, cities can enhance streetlight efficiency, reduce energy consumption, and improve overall urban infrastructure.

**Ravinder Kumaret.al (2016) [14]** Street lights play a crucial role in urban infrastructure, but conventional systems often result in unnecessary power consumption due to continuous lighting even when there is low activity. To address this, the proposed **ZigBee-based automatic street light system** is smart and provides a safe nighttime environment for all road users, including pedestrians. The system reduces energy consumption, maintenance costs, and contributes to crime reduction. This innovative system contributes to efficient streetlight management, promoting sustainability and safer urban environment.

**Abhishek et.al (2015) [15]** The proposed system aims to efficiently manage street lighting by dynamically adjusting it based on traffic flow. By implementing this system, cities can achieve energy savings, reduce maintenance costs, and promote sustainable urban lighting.

**Mohamedet.al (2013) [16]**The huge amount of electrical power consumed in lighting streets is a concern in many countries. However, vehicles pass through specific areas at low rates during certain periods, and parts of the streets remain unoccupied by vehicles over time. In this paper, a system is proposed that automatically switches off the lights for street segments without vehicles and turns them on once vehicles are expected to arrive. The system utilizes **Vehicular Ad-Hoc Networks (VANET)** for communication and control.

**Rahul Kaul et.al (2013) [17]**This paper efficiently defines the control of street lighting systems and thereby **saves electricity**, which is a major concern worldwide. It also describes the use of wireless sensor networks using GSM for streetlight monitoring and control. The system provides remote access for streetlight maintenance and control. It takes automatic decisions for luminous control (ON/OFF/DIMMING) based on surrounding light intensity and time of the day. The system also senses various parameters like surrounding temperature, fog, carbon emissions, and noise intensities, suggesting corrective measures. Power theft control is integrated into the same system. The efficiency of the system allows installation in existing road conditions with an extra cost of only a single controlling computer.

**Monica Pujari et.al (2013) [18]**The conventional road lighting systems in areas with low pedestrian frequency remain online most of the night without reason, resulting in significant power wastage. With the availability of flexible lighting technology like light-emitting diode (LED) lights and wireless internet connection, responsive, reliable, and energy-saving street lighting systems have become a reality. The main idea behind this work is to describe the **Intelligent Street Lighting (ISL) system**, an initial approach to meeting the demand for adaptable public lighting systems. By using the ISL system, wastage of power can be significantly reduced.

**J.Arthi et.al [19]** proposed **Smart LED Street Lighting System** aims to design and execute advanced developments in IoT for energy-saving street lights. The best solution for electrical power wastage is the automation of street lights, eliminating manual operation. The system modifies street light illumination using sensors to achieve minimum electrical energy consumption. When an object presence is detected, street lights glow at their brightest mode; otherwise, they stay in dim mode during nighttime. Internet of Things (IoT) is used to visualize real-time updates of street processing and notify changes. This system reduces heat emissions, power consumption, maintenance and replacement costs, and carbon dioxide emissions.

**Salvi et.al (2017) [20]**The conventional street lighting systems often result in unnecessary power consumption due to continuous lighting even when there is low activity. To address this, the proposed **Smart Street Light System** utilizes an Arduino Uno microcontroller for efficient control. This innovative system contributes to energy savings, reduced maintenance, and improved urban lighting.

**Velaga et.al (2012) [21]**The majority of the Indian population lives in rural areas, where road transport plays a vital role in economic growth and development. However, the existing street lighting systems in rural areas are inefficient and consume a lot of electrical power. To address this, the authors propose a **Smart Street Light (SSL) system** that can dynamically switch ON and OFF the lights based on traffic flow and light intensity. The system uses **wireless sensor networks(WSN)** and **GSM** for communication and control. The authors evaluate the feasibility of the SSL system by considering a case study of the remotely located Khurhaan village in the state of Bihar, India. The results show that the SSL system can save up to **91,506 INR** annually compared to the conventional street light system. The paper also discusses the benefits and challenges of implementing the SSL system in rural areas.

**A .Sarma et.al (2016) [22]**It is very common these days that we see street light powered with solar panels. As the fossil fuels are degrading and also polluting the environment, the use of efficient power systems needs to be implemented. This paper presents a remote sense based street light system. This system can put effort to vary intensity according to the density of the traffic and pedestrians. Also, the high-intensity discharge lamp is replaced by LED’s. The system can be easily implemented widely where there is need of timely control. The paper also discusses the benefits and challenges of the proposed system.

**M .Samad et.al (2016) [23]** the paper presents the development of a low-cost sensor node for street lighting based on the criteria needed by the industry. The function of the sensor node is to sense or detect the motion or movement of an object or a car. The sensor node will act or response when it detects an object moving past its position or location. The street light will turn on and transmit data to another pole. Once the object has passed the sensor node, the light will turn off. The paper focuses on the development of a cost-effective sensor node. This sensor node will save the power supplied to the light by preventing it from turning on all night. This will also help to reduce maintenance cost and save on power consumption.

**S. R. Parekar et.al (2015) [24]** street light is a raised source of light that is commonly used along walkways and streets when the surrounding turns dark. Nowadays, most of the existing street light systems are wired which are not only difficult to construct but also has poor flexibility. To overcome this, wireless system is required. In this paper, we are using GSM technology which uses power efficiently by remotely monitoring and controlling the system. This system will ease the fault detection and maintenance. System allows us to make the most efficient use of the energy received from the sun to power street lights. Solar energy is collected with the aid of solar panel and battery is charged during day time and this energy is used to power street lights during night. Developed intelligent system turns the light ON and OFF depending on the vehicle or pedestrian movement, Real Time Clock and light intensity at the same time. Microcontroller processes the information from the sensors and is transferred to nearby control terminal (Base station with Raspberry PI as a compute module) to monitor the status of the street lamp using GSM technology via Short Message Service (SMS). Designed system is visualized by creating Graphical User Interface (GUI). Thus, the implementation of such system will result in energy saving, lower cost of maintenance, increased lifespan and hence the enhanced performance of the system.

**M. Karthikeyan et.al (2014) [25]** proposed system for the cloud based automatic system involves the automatic updating of the data to the lighting system. It also reads the data from the base station in case of emergencies. Zigbee devices are used for wireless transmission of the data from the base station to the light system thus enabling an efficient street lamp control system. Infrared sensor and dimming control circuit is used to track the movement of human in a specific range and dims/bright the street lights accordingly hence saving a large amount of power. In case of emergencies data is sent from the particular light or light system and effective measures are taken accordingly.

**S. Deo et.al (2014) [26]**A novel scheme for a Zigbee-based street light control is proposed with an aim to reduce the human error in the operation of street lights, decrease the energy consumption of the system, and ease the maintenance of the street light network. These objectives are achieved by creating a wireless Zigbee network of street lights that can be monitored from a base station. A new scheme that provides for the operation of alternate lights during low traffic hours, dusk and dawn has been devised. Additionally, an automatic mode of operation that utilizes light sensors to automatically switch ON street lights when light intensity falls below a certain level, has also been incorporated. The scheme was implemented, and a small scale working model was developed.

**CHAPTER 3**

**EXISTING SYSTEM**

The IoT Manhole Monitoring & Street Light Control System with Fault Detection & Reporting for Municipalities is a comprehensive solution designed to enhance the efficiency and management of urban infrastructure. This innovative system integrates Internet of Things (IoT) technology to monitor and control manholes and street lights, ensuring optimal functionality and timely fault detection.

In the context of manhole monitoring, the system employs various sensors to collect real-time data on parameters such as water level, gas emissions, and structural integrity within the manholes. These sensors are strategically placed to provide a holistic view of the manhole conditions. The collected data is then transmitted to a central server through wireless communication protocols, enabling municipalities to remotely monitor the status of manholes across the city.

Furthermore, the Street Light Control aspect of the system allows for intelligent and adaptive street lighting. IoT-enabled controllers are installed on streetlights, enabling municipalities to remotely manage and customize the lighting conditions based on factors such as time of day, weather conditions, and traffic patterns. This not only contributes to energy savings but also enhances public safety by ensuring well-lit streets.

A key feature of the system is its Fault Detection & Reporting mechanism. The collected data undergoes continuous analysis to identify anomalies or potential issues in both manholes and street lights. In case of any detected faults, the system generates real-time alerts and notifications, allowing municipal authorities to take prompt corrective actions. This proactive approach minimizes downtime, reduces maintenance costs, and ensures the longevity of the infrastructure.

To enhance accessibility and decision-making, the system provides a user-friendly dashboard for municipal officials. The dashboard presents a consolidated view of the entire infrastructure, displaying real-time status updates, historical data, and actionable insights. This enables authorities to make data-driven decisions, allocate resources efficiently, and prioritize maintenance tasks based on criticality.

There were various systemsbeing developed for the monitoring of manholes and street lights in municipalities, withfault detection and reporting. Here is an overview of existing systems.

**1)Manhole Monitoring System:**

* Sensor Networks: IoT (Internet of Things) sensors are often deployed in manholes to monitor various parameters such as water level, temperature, gas levels, and other environmental conditions.
* Wireless Communication: Data from sensors are transmitted wirelessly to a central control system, allowing real-time monitoring and analysis.
* Fault Detection: Anomalies, such as rising water levels or gas leaks, can trigger automated alerts or notifications for prompt action.

**2)Street Light Control System:**

* Smart Lighting Systems: IoT-based smart street lighting systems are becoming common, allowing municipalities to control street lights remotely.
* Energy Efficiency: These systems often include features like dimming or turning off lights during periods of low activity to conserve energy.
* Fault Detection: Sensors and monitoring devices can detect issues like burnt-out bulbs or electrical faults, triggering automatic alerts for maintenance teams.

**3)Integration and Centralized Monitoring:**

* Centralized Platforms: Many municipalities are integrating manhole monitoring and street light control into a centralized platform for more efficient management.
* Data Analytics: Utilizing data analytics to gain insights into usage patterns, potential faults, and overall system performance.

**4)Fault Detection & Reporting:**

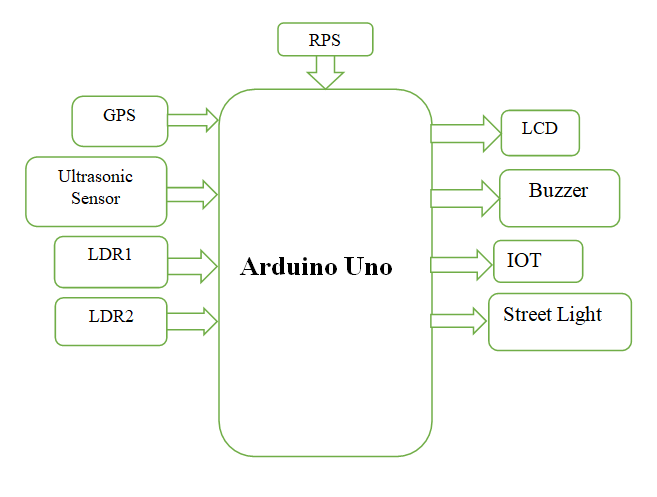
* Automated Alerts: Systems are designed to automatically detect faults or abnormalities and send alerts to maintenance teams.
* Remote Diagnostics: Some systems allow for remote diagnostics, enabling technicians to identify issues without physical inspection.
* Reporting Interfaces: Web-based interfaces or mobile applications for reporting and monitoring the status of manholes and street lights.

**CHAPTER 4**

**PROPOSED SYSTEM**

**In this project, we present a working model of IOT Manhole Monitoring & Street Light control system with fault detection & reporting for Muncipalitites. The main objective for implementing this project is** to enhance the municipalities in terms of Smart infrastructure management, public safety, and resource optimization**.** Implemented using IOT,Arduino Uno microcontroller, GPS, Ultrasonic sensor, LDR1, LDR2, Street light, RPS, LCD, buzzer, Arduino IDE tool, and embedded c language.

**BLOCK DIAGRAM:**

****

The proposed IoT-based system for manhole monitoring and street light control with fault detection and reporting capabilities represents a cutting-edge solution for municipalities to efficiently manage their infrastructure.

In the manhole monitoring aspect, the system would incorporate IoT sensors deployed within manholes across the municipality. These sensors would be equipped to monitor various parameters such as water levels, temperature, gas concentrations, and structural integrity. The sensors would continuously collect data and transmit it wirelessly to a central control hub. This control hub would host a cloud-based platform where the data would be analyzed in real-time using advanced algorithms.

For street light control, the system would involve retrofitting existing street lights with IoT controllers and sensors. These controllers would allow for remote monitoring and control of each street light, enabling municipalities to adjust lighting levels dynamically based on factors such as time of day, weather conditions, and traffic patterns. The sensors would also detect faults such as burnt-out bulbs, electrical issues, or vandalism.

The fault detection and reporting capabilities would be a key feature of the system. In both manhole monitoring and street light control, the system would be programmed to detect anomalies and faults automatically. For manholes, this could include detecting rising water levels, abnormal temperatures, or sudden changes in gas concentrations. For street lights, it could involve identifying burnt-out bulbs, fluctuations in power consumption, or physical damage.

Upon detecting a fault, the system would generate automated alerts and reports. These alerts would be sent to designated personnel responsible for maintenance and repairs. The reports would provide detailed information about the nature of the fault, its location, and any relevant environmental conditions. This would enable maintenance teams to respond promptly and efficiently, minimizing downtime and ensuring the continued operation of critical infrastructure.

Furthermore, the system would offer advanced analytics capabilities. It would store historical data collected from both manhole monitoring and street light control, allowing municipalities to identify trends, predict potential issues, and optimize resource allocation. This data-driven approach would enable municipalities to make informed decisions about infrastructure maintenance, upgrade, and expansion, ultimately enhancing the resilience and sustainability of urban infrastructure networks.

Overall, the proposed IoT-based system for manhole monitoring and street light control with fault detection and reporting represents a comprehensive solution for municipalities seeking to modernize their infrastructure management practices. By leveraging the power of IoT technology and advanced analytics, municipalities can improve the reliability, efficiency, and safety of their infrastructure while minimizing operational costs and environmental impact.

**Hardware Modules:**

* Sensors
* Microcontroller
* Communication Module
* Light Intensity Sensors
* Fault Detection Sensors
* Database
* User Interface

**Software Modules:**

* Sensor Integration
* Data Processing
* Remote Control
* APIs (Application Programming Interfaces)
* Alert Generation
* Access Control
* Authentication and Authorization

**Project Working:**

In this system there are totally five sections:

1. Regulated Power Supply
2. Input Section

* GPS
* Ultrasonic Sensor
* LDR1
* LDR2

1. Output Section

* LCD
* Buzzer
* IOT
* Street Light

1. Arduino Uno Microcontroller

There are five modules Controller, RPS, Input, software and output module. The power is supplied to the RPS module through an adapter. The adapter converts 230v AC to 12v DC and this 12v DC is give to the RPS module. The RPS module consists of voltage regulator 7805 which converts the 12v DC into 5v DC, capacitors are used to reduce noise and LED is used which indicates whether the power is supplied or not. This 5v DC power supply goes to each and every module.

The Input modules are GPS, Ultrasonic sensor, LDR1 and LDR2. The GPS is used for live tracking of the system, the Ultrasonic sensor is used for drainage level detection of fault, In LDR1 the street light in day time should be OFF and night time street light should be ON in the system, In LDR2 the fault detection happens.

The output modules are LCD, Buzzer, IOT and street light. The 16x2 LCD module shows the output in the kit. The Buzzer used in the system is piezoelectric buzzer it alerts the system.

The data is uploaded in the website by using ESP8266 IOT module. Street light used in the system is 12v LED.

An IoT-based Manhole Monitoring & Street Light Control System with Fault Detection & Reporting for Municipalities integrates various technologies to enhance the management and maintenance of urban infrastructure. The system begins with the installation of IoT sensors in manholes and streetlights across the city. These sensors are equipped with various capabilities such as measuring water levels,monitoring temperature, and detecting light intensity. They are wirelessly connected to a central control system.

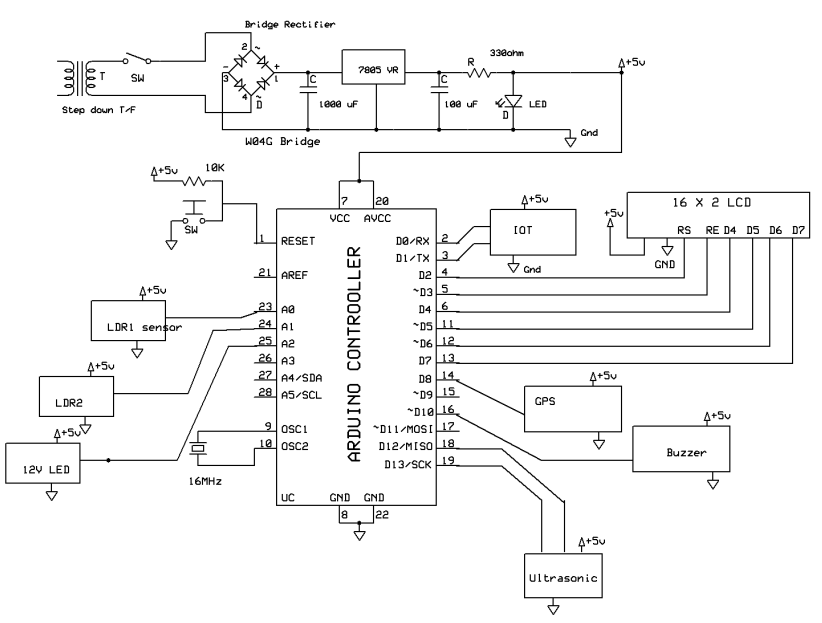
The sensors continuously collect data related to manhole conditions (such as water level, pressure, and presence of toxic gases) and streetlight status (light intensity, power consumption, etc.). This data is transmitted in real-time to the central control system using wireless communication protocols such as Wi-Fi,cellular networks. The central control system, typically located in the municipality's office or a centralized data center, collects and processes the data received from the IoT sensors.

Utilizing advanced algorithms and analytics, the central control system analyzes the data to identify any anomalies or faults in the manholes or streetlights. It continuously monitors the incoming data for deviations from normal operating conditions. If any anomalies or faults are detected, such as a rise in water level in a manhole beyond a certain threshold or a malfunctioning streetlight, the system triggers an alert/notification.

These notifications are sent out to relevant stakeholders, such as municipal authorities, maintenance crews, or utility companies, informing them about the detected fault or anomaly. Notifications can be sent via SMS, email, or through a dedicated dashboard or mobile application. Depending on the severity of the fault or anomaly detected, the system can trigger automated responses.

Furthermore, malfunctioning streetlights can be remotely turned off or dimmed until they are repaired. Municipal authorities can use the data collected by the system to schedule routine maintenance tasks, prioritize repairs, and optimize resource allocation. Overall, an IoT Manhole Monitoring & Street Light Control System with Fault Detection & Reporting enables municipalities to improve the efficiency, safety, and reliability of urban infrastructure while reducing maintenance costs and minimizing downtime.

**Schematic Diagram**

****

This is the pin diagram where all the hardware components are been connected components. this ARDUINO microcontroller having 28 pins. In which 14 GPIO pins as digital pins and 6 GPIO pins. 16MHz crystal oscillator connected internally. The step down transformer, Bridge rectifier capacitor with 1000f Resisters and led are connected in Regulated power supply which provide the 5v to the Arduino and all input/output modules.

* 16\*2 LCD Monitor has connected with the Digital pins 2,3,4,5,6,7.
* WIFI has connected to Digital Pins D0,D1 internal Transmitter and receiver pins.
* Ultrasonic connected to 12, 13 pins of the Arduino micro controller.
* IR sensor connected to digital pin A0
* LDR sensor connected to digital pin A1
* Buzzer alarm connected to digital pin 10
* LED sensor connected to digital pin A2
* GPS connected to digital pin 8

Design and implementation of smart city by using ARDUINO is done with 3 advanced applications smart drainage over flow, and Smart Street lighting system. Municipal drainage level will monitor and updated into over LCD and IOT module. Drainage system we use water detecting sensor which is automatically alert if it sense overflow of water. Street light applications we use LDR sensor along with LED indicator for street light.

**ADVANTAGES:**

The integration of IoT (Internet of Things) technology in Manhole Monitoring and Street Light Control Systems with Fault Detection & Reporting brings numerous advantages for municipalities.

Here are some key benefits of implementing such systems:

**Proactive Maintenance:**

The IoT-enabled Manhole Monitoring system allows municipalities to proactively address potential issues in sewer systems. By continuously monitoring parameters such as water levels and pressure, authorities can detect anomalies early on, preventing major failures and minimizing repair costs.

**Improved Public Safety:**

Street Light Control Systems with Fault Detection & Reporting contribute to enhanced public safety by ensuring well-lit urban environments. The proactive fault detection mechanism identifies malfunctions or outages promptly, reducing the risk of accidents and criminal activities in poorly lit areas.

**Energy Efficiency:**

The Street Light Control System utilizes IoT technology to optimize energy consumption. By dynamically adjusting lighting levels based on real-time conditions such as pedestrian activity and ambient light levels, municipalities can achieve significant energy savings and contribute to a more sustainable urban environment.

**Cost Savings:**

Both Manhole Monitoring and Street Light Control Systems lead to cost savings for municipalities. The early detection of faults in sewer systems minimizes the need for emergency repairs, while efficient street lighting management reduces energy costs and maintenance expenses.

**Remote Monitoring and Control:**

The IoT capabilities enable remote monitoring and control of both manholes and street lights. Municipal authorities can access real-time data, make adjustments, and receive automated reports from a centralized location, improving operational efficiency and reducing the need for physical inspections.

**Data-Driven Decision Making:**

The systems generate valuable data that can be analyzed to make informed decisions. Municipalities can use the data to optimize resource allocation, plan maintenance schedules, and improve overall infrastructure management.

**Enhanced Resilience and Reliability:**

The predictive maintenance features of these systems increase the resilience and reliability of critical infrastructure. By identifying potential issues before they escalate, municipalities can ensure the continuous functionality of sewer systems and street lights.

**Smart City Integration:**

These IoT-enabled systems can be seamlessly integrated into broader smart city initiatives. By connecting different components of urban infrastructure, municipalities can create a more interconnected and efficient urban environment.

**Environmental Impact:**

The energy optimization features of the Street Light Control System contribute to reducing carbon emissions, aligning with environmental sustainability goals. Additionally, the proactive maintenance of sewer systems helps prevent environmental contamination and mitigates the impact of infrastructure failures.

**User-Friendly Interfaces:**

The systems are designed with user-friendly interfaces, making them accessible to municipal administrators and operators with varying levels of technical expertise. This ensures effective utilization and management of the IoT-enabled solutions.

**DISADVANTAGES:**

While IoT (Internet of Things) solutions like Manhole Monitoring and Street Light Control Systems with Fault Detection & Reporting offer several benefits for municipalities, there are also potential disadvantages and challenges associated with their implementation.

Here are some considerations:

**Security Concerns:**

IoT devices are vulnerable to cyber-attacks, and if not properly secured, malicious actors could gain unauthorized access to the system. This could lead to unauthorized control of street lights, false reporting, or manipulation of data.

**Data Privacy Issues:**

The collection of data from IoT devices raises concerns about privacy. Monitoring systems may capture sensitive information, and municipalities need to implement robust data protection measures to ensure the privacy of citizens is maintained.

**High Initial Costs:**Implementing a comprehensive IoT system can be expensive, especially for municipalities with limited budgets. The cost of sensors, communication infrastructure, and software development can be significant.

**Maintenance Challenges:**

IoT devices and sensors require regular maintenance to ensure proper functionality. Manhole sensors and street light control systems may be exposed to harsh environmental conditions, leading to wear and tear. Frequent maintenance may be needed to keep the system operational.

**Compatibility Issues:**

Integrating IoT devices from different manufacturers may pose compatibility challenges. Ensuring seamless communication and interoperability between various components of the system can be complex.

**Limited Scalability:**

Some IoT solutions may have limitations when it comes to scalability. As municipalities grow or change, the system may struggle to adapt to increased demands or additional devices.

**Power Consumption:**

Many IoT devices rely on batteries or other power sources. Ensuring a stable and sustainable power supply for these devices, especially in remote or inaccessible locations, can be a challenge.

**False Alarms and Accuracy:**

IoT devices, especially those involved in fault detection, may generate false alarms or inaccurate reports. This could lead to unnecessary maintenance activities or overlook real issues, impacting the overall efficiency of the system.

**APPLICATIONS:**

The IoT (Internet of Things) Manhole Monitoring & Street Light Control System with Fault Detection & Reporting for municipalities offers a versatile set of applications that significantly improve urban infrastructure management.

Here are some key applications:

**Smart City Infrastructure Management:**

The system contributes to the development of smart cities by providing real-time monitoring and control of critical components like manholes and street lights. This helps municipalities in making data-driven decisions for better urban planning and resource allocation.

**Efficient Street Lighting:**

The Street Light Control System optimizes energy consumption by adjusting lighting levels based on real-time factors. This leads to significant energy savings for municipalities, contributing to sustainability goals and reducing operational costs.

**Public Safety Enhancement:**

The intelligent monitoring of manholes ensures the timely detection of issues such as overflow or blockage, preventing potential hazards and improving public safety. Properly functioning street lights also enhance visibility, contributing to a safer urban environment.

**Fault Detection and Reporting:**

The Fault Detection & Reporting feature helps municipalities identify issues in both manholes and street lights promptly. This proactive approach enables faster response times for maintenance and repairs, minimizing downtime and improving overall system reliability.

**Predictive Maintenance:**

The system's predictive maintenance capabilities anticipate potential failures in manholes and street lights based on historical data and analytics. This helps municipalities schedule maintenance activities in advance, reducing the likelihood of unexpected issues and extending the lifespan of infrastructure components.

**Environmental Impact Mitigation:**

By optimizing energy usage in street lighting and ensuring efficient operation of manholes, the system contributes to reducing the environmental footprint of municipalities. This aligns with sustainability goals and demonstrates a commitment to environmental responsibility.

**Traffic Management and Planning:**

The real-time data collected from the system can be utilized for intelligent traffic management. By adjusting street lighting based on traffic patterns, municipalities can enhance traffic flow and reduce congestion, leading to improved overall urban mobility.

**Remote Monitoring and Control:**

The ability to remotely monitor and control both manholes and street lights offers municipalities unprecedented flexibility. Administrators can access the system from anywhere, facilitating timely responses to emerging issues and enabling efficient management of city infrastructure.

**Data Analytics for Decision-Making:**

The system generates valuable data that can be analyzed to gain insights into usage patterns, performance metrics, and potential areas for improvement. This data-driven approach helps municipalities make informed decisions for better city planning and development.

**Integration with Existing Systems:**

The flexibility of the system allows seamless integration with other municipal systems, such as traffic management or emergency response systems. This createsa holistic approach to urban infrastructure management, improving overall efficiency and coordination.

**CHAPTER 5**

**EMBEDDED SYSTEMS**

**2.1 Embedded Systems:**

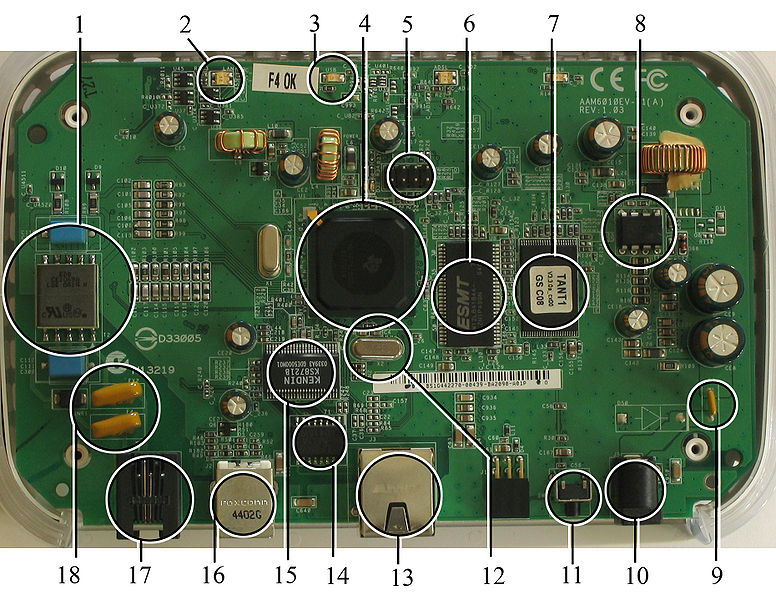
An embedded system is a computer system designed to perform one or a few dedicated functions often with real-time computing constraints. It is embedded as part of a complete device often including hardware and mechanical parts. By contrast, a general-purpose computer, such as a personal computer (PC), 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.)

Since the embedded system is dedicated to specific tasks, design engineers can optimize it to reduce the size and cost of the product and increase the reliability and performance. Some embedded systems are mass-produced, benefiting from economies of scale.

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.

In general, "embedded system" is not a strictly definable term, as most systems have some element of extensibility or programmability. For example, handheld computers share some elements with embedded systems such as the operating systems and microprocessors which power them, but they allow different applications to be loaded and peripherals to be connected. Moreover, even systems which don't expose programmability as a primary feature generally need to support software updates. On a continuum from "general purpose" to "embedded", large application systems will have subcomponents at most points even if the system as a whole is "designed to perform one or a few dedicated functions", and is thus appropriate to call "embedded". A modern example of embedded system is shown in fig: 2.1.



**Fig 2.1: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.

**2.1.1 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](http://en.wikipedia.org/wiki/Programmable_controllers) evolved from traditional [electromechanical](http://en.wikipedia.org/wiki/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](http://en.wikipedia.org/wiki/Apollo_Guidance_Computer), developed by [Charles Stark Draper](http://en.wikipedia.org/wiki/Charles_Stark_Draper) at the MIT Instrumentation Laboratory. At the project's inception, the Apollo guidance computer was considered the riskiest item in the Apollo project as it employed the then newly developed monolithic integrated circuits to reduce the size and weight. An early mass-produced embedded system was the Autonetics D-17 guidance computer for the [Minuteman missile](http://en.wikipedia.org/wiki/Minuteman_(missile)), released in 1961. It was built from [transistor](http://en.wikipedia.org/wiki/Transistor) [logic](http://en.wikipedia.org/wiki/Digital_circuit) and had a [hard disk](http://en.wikipedia.org/wiki/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.

**2.1.2 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.

**2.1.3 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.

**2.1.4 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.

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

**2.2.1 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 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, 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.

**2.2.2 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:

* The system cannot safely be shut down for repair, or it is too inaccessible to repair. Examples include space systems, undersea cables, navigational beacons, bore-hole systems, and automobiles.
* The system must be kept running for safety reasons. "Limp modes" are less tolerable. Often backup s are selected by an operator. Examples include aircraft navigation, reactor control systems, safety-critical chemical factory controls, train signals, engines on single-engine aircraft.
* The system will lose large amounts of money when shut down: Telephone switches, factory controls, bridge and elevator controls, funds transfer and market making, automated sales and service.

A variety of techniques are used, sometimes in combination, to recover from errors both software bugs such as memory leaks, and also soft errors in the hardware:

* Watchdog timer that resets the computer unless the software periodically notifies the watchdog
* Subsystems with redundant spares that can be switched over to
* software "limp modes" that provide partial function
* Designing with a Trusted Computing Base (TCB) architecture[6] ensures a highly secure & reliable system environment
* An Embedded Hypervisor is able to provide secure encapsulation for any subsystem component, so that a compromised software component cannot interfere with other subsystems, or privileged-level system software. This encapsulation keeps faults from propagating from one subsystem to another, improving reliability. This may also allow a subsystem to be automatically shut down and restarted on fault detection.
* Immunity Aware Programming

**2.3 Explanation of Embedded Systems:**

**2.3.1 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 interrupt 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.

* Cooperative Multitasking:

A non-preemptive multitasking system is very similar to the simple control loop scheme, except that the loop is hidden in an API. The programmer defines a series of tasks, and each task gets its own environment to “run” in. When a task is idle, it calls an idle routine, usually called “pause”, “wait”, “yield”, “nop” (stands for no operation), etc.The advantages and disadvantages are very similar to the control loop, except that adding new software is easier, by simply writing a new task, or adding to the queue-interpreter.

* Primitive Multitasking:

In this type of system, a low-level piece of code switches between tasks or threads based on a timer (connected to an interrupt). This is the level at which the system is generally considered to have an "operating system" kernel. Depending on how much functionality is required, it introduces more or less of the complexities of managing multiple tasks running conceptually in parallel.

As any code can potentially damage the data of another task (except in larger systems using an MMU) programs must be carefully designed and tested, and access to shared data must be controlled by some synchronization strategy, such as message queues, semaphores or a non-blocking synchronization scheme.

Because of these complexities, it is common for organizations to buy a real-time operating system, allowing the application programmers to concentrate on device functionality rather than operating system services, at least for large systems; smaller systems often cannot afford the overhead associated with a generic real time system, due to limitations regarding memory size, performance, and/or battery life.

* Microkernels And Exokernels:

A microkernel is a logical step up from a real-time OS. The usual arrangement is that the operating system kernel allocates memory and switches the CPU to different threads of execution. User mode processes implement major functions such as file systems, network interfaces, etc.

In general, microkernels succeed when the task switching and intertask communication is fast, and fail when they are slow. Exokernels communicate efficiently by normal subroutine calls. The hardware and all the software in the system are available to, and extensible by application programmers. Based on performance, functionality, requirement the embedded systems are divided into three categories:

**2.3.2 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..

**2.3.3 Real-time embedded systems:**

Embedded systems which are used to perform a specific task or operation in a specific time period those systems are called as real-time embedded systems. There are two types of real-time embedded systems.

* Hard Real-time embedded systems:

These embedded systems follow an absolute dead line time period i.e.., if the tasking is not done in a particular time period then there is a cause of damage to the entire equipment.

Eg: consider a system in which we have to open a valve within 30 milliseconds. If this valve is not opened in 30 ms this may cause damage to the entire equipment. So in such cases we use embedded systems for doing automatic operations.

* Soft Real Time embedded systems:

Eg: Consider a TV remote control system, if the remote control takes a few milliseconds delay it will not cause damage either to the TV or to the remote control. These systems which will not cause damage when they are not operated at considerable time period those systems comes under soft real-time embedded systems.

**2.3.4 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: 2.2 show the network communications in embedded systems.



**Fig 2.2: Network communication embedded systems**

**2.3.5 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

**2.4 APPLICATIONS OF EMBEDDED SYSTEMS:**

**2.4.1 Consumer applications:**

At home we use a number of embedded systems which include microwave oven, remote control, vcd players, dvd players, camera etc….



**Fig2.3: Automatic coffee makes equipment**

**2.4.2 Office automation:**

We use systems like fax machine, modem, printer etc…



**Fig2.4: Fax machine Fig2.5: Printing machine**

**2.4.3. Industrial automation:**

Today a lot of industries are using embedded systems for process control. In industries we design the embedded systems to perform a specific operation like monitoring temperature, pressure, humidity ,voltage, current etc.., and basing on these monitored levels we do control other devices, we can send information to a centralized monitoring station.



**Fig2.6: Robot**

In critical industries where human presence is avoided there we can use robot**s** which are programmed to do a specific operation.

**2.4.5 Computer networking:**

Embedded systems are used as bridges routers etc..



**Fig2.7: Computer networking**

**2.4.6 Tele communications:**

Cell phones, web cameras etc.



**Fig2.8: Cell PhoneFig2.9: Web camera**

**CHAPTER 6**

**HARDWARE DESCRIPTION**

**3.2 Micro controller:**



**Fig: 3.2 Microcontrollers**

**3.2.1 Introduction to Microcontrollers:**

Circumstances that we find ourselves in today in the field of microcontrollers had their beginnings in the development of technology of integrated circuits. This development has made it possible to store hundreds of thousands of transistors into one chip. That was a prerequisite for production of microprocessors, and the first computers were made by adding external peripherals such as memory, input-output lines, timers and other. Further increasing of the volume of the package resulted in creation of integrated circuits. These integrated circuits contained both processor and peripherals. That is how the first chip containing a microcomputer, or what would later be known as a microcontroller came about.

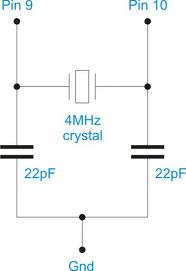
Microprocessors and microcontrollers are widely used in embedded systems products. Microcontroller is a programmable device. A microcontroller has a CPU in addition to a fixed amount of RAM, ROM, I/O ports and a timer embedded all on a single chip. The fixed amount of on-chip ROM, RAM and number of I/O ports in microcontrollers makes them ideal for many applications in which cost and space are critical.

The AVR is a modified Harvard architecture 8-bit RISC single chip microcontroller which was developed by Atmel in 1996. The AVR was one of the first microcontroller families to use on-chip flash memory for program storage, as opposed to One-Time Programmable ROM, EPROM, or EEPROM used by other microcontrollers at the time.

**Crystal Oscillator:**

XTAL1 and XTAL2 are input and output, respectively, of an inverting amplifier which can be configured for use as an On-chip Oscillator, Either a quartz

Crystal or a ceramic resonator may be used. The CKOPT Fuse selects between two different Oscillator amplifier modes. When CKOPT is programmed, the Oscillator output will oscillate a full rail-to-rail swing on the output. This mode is suitable when operatingin a very noisy environment or when the output from XTAL2 drives a second clockbuffer. This mode has a wide frequency range. When CKOPT is unprogrammed, theOscillator has a smaller output swing. This reduces power consumption considerably.

****

This mode has a limited frequency range and it cannot be used to drive other clockbuffers. For resonators, the maximum frequency is 8 MHz with CKOPT unprogrammed and16 MHz with CKOPT programmed. C1 and C2 should always be equal for both crystalsand resonators. The optimal value of the capacitors depends on the crystal or resonatorin use, the amount of stray capacitance, and the electromagnetic noise of the environment. For ceramic resonators, the capacitor values given by the manufacturer should be used.The Oscillator can operate in three different modes, each optimized for a specific frequency range. The operating mode is selected by the fuses CKSEL3..1

**3.2.2 Architecture:**

**Memory:** It has **8 Kb** of Flash program memory (10,000 Write/Erase cycles durability), **512 Bytes** of EEPROM (100,000 Write/Erase Cycles). **1Kbyte** Internal SRAM

**I/O Ports:** 23 I/ line can be obtained from three ports; namely Port B, Port C and Port D.

**Interrupts:**  Two External Interrupt source, located at port D. 19 different interrupt vectors supporting 19 events generated by internal peripherals.

**Timer/Counter:** Three Internal Timers are available, two 8 bit, one 16 bit, offering various operating modes and supporting internal or external clocking.

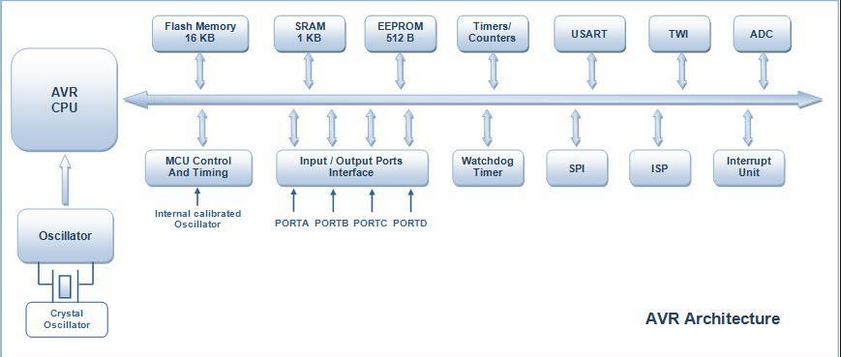
**SPI (Serial Peripheral interface):** ATmega8 holds three communication devices integrated. One of them is Serial Peripheral Interface. Four pins are assigned to Atmega8 to implement this scheme of communication.

**USART:**One of the most powerful communication solutions is [USART](http://www.circuitstoday.com/how-to-establish-a-pc-micro-controller-usart-communication) and ATmega8 supports both synchronous and asynchronous data transfer schemes. It has three pins assigned for that. In many projects, this module is extensively used for PC-Micro controller communication.

**TWI (Two Wire Interface):** Another communication device that is present in ATmega8 is Two Wire Interface. It allows designers to set up a commutation between two devices using just two wires along with a common ground connection, As the TWI output is made by means of open collector outputs, thus external pull up resistors are required to make the circuit.

**Analog Comparator:** A comparator module is integrated in the IC that provides comparison facility between two voltages connected to the two inputs of the Analog comparator via External pins attached to the micro controller.

**Analog to Digital Converter:**Inbuilt analog to digital converter can convert an analog input signal into digital data of **10bit** resolution. For most of the low end application, this much resolution is enough.



**Microcontroller:** Microcontroller can be termed as a single on chip computer which includes number of peripherals like RAM, EEPROM, Timers etc., required to perform some predefined task.

The computer on one hand is designed to perform all the general purpose tasks on a single machine like you can use a computer to run a software to perform calculations or you can use a computer to store some multimedia file or to access [internet](http://www.engineersgarage.com/articles/what-is-internet-history-working) through the browser, whereas the microcontrollers are meant to perform only the specific tasks, for e.g., switching the AC off automatically when room temperature drops to a certain defined limit and again turning it ON when temperature rises above the defined limit.

 There are number of popular families of microcontrollers which are used in different applications as per their capability and feasibility to perform the desired task, most common of these are [8051](http://www.engineersgarage.com/8051-microcontroller), **AVR** and [PIC](http://www.engineersgarage.com/articles/pic-microcontroller-tutorial) microcontrollers. In this article we will introduce you with **AVR** family of microcontrollers.

**AVR** was developed in the year 1996 by Atmel Corporation. The architecture of **AVR** was developed by Alf-Egil Bogen and Vegard Wollan. AVR derives its name from its developers and stands for **Alf-Egil Bogen Vegard Wollan RISC microcontroller**, also known as **A**dvanced **V**irtual **R**ISC. The AT90S8515 was the first microcontroller which was based on **AVR architecture** however the first microcontroller to hit the commercial market was AT90S1200 in the year 1997.

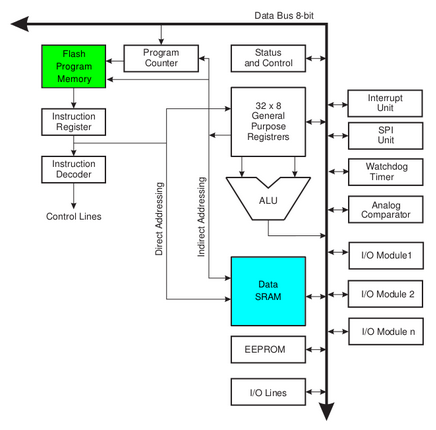
**AVR microcontrollers** are available in three categories:

1.      **TinyAVR** – Less memory, small size, suitable only for simpler applications

2.      **MegaAVR** – These are the most popular ones having good amount of memory (upto 256 KB), higher number of inbuilt peripherals and suitable for moderate to complex applications.

3.      **XmegaAVR** – Used commercially for complex applications, which require large program memory and high speed.

**ARCHITECTURE**



### Device architecture

Flash, EEPROM, and SRAM are all integrated onto a single chip, removing the need for external memory in most applications. Some devices have a parallel external bus option to allow adding additional data memory or memory-mapped devices. Almost all devices (except the smallest TinyAVR chips) have serial interfaces, which can be used to connect larger serial EEPROMs or flash chips.

#### Program memory

Program instructions are stored in non-volatile flash memory. Although the MCUs are 8-bit, each instruction takes one or two 16-bit words.

The size of the program memory is usually indicated in the naming of the device itself (e.g., the ATmega64x line has 64 kB of flash while the ATmega32x line has 32 kB).

There is no provision for off-chip program memory; all code executed by the AVR core must reside in the on-chip flash. However, this limitation does not apply to the AT94 FPSLIC AVR/FPGA chips.

#### Internal data memory

The data address space consists of the register file, I/O registers, and SRAM.

#### Internal registers

The AVRs have 32 single-byte registers and are classified as 8-bit RISC devices.

In most variants of the AVR architecture, the working registers are mapped in as the first 32 memory addresses (000016–001F16) followed by the 64 I/O registers (002016–005F16).

Actual SRAM starts after these register sections (address 006016). (Note that the I/O register space may be larger on some more extensive devices, in which case the memory mapped I/O registers will occupy a portion of the SRAM address space.)

Even though there are separate addressing schemes and optimized opcodes for register file and I/O register access, all can still be addressed and manipulated as if they were in SRAM.

In the XMEGA variant, the working register file is not mapped into the data address space; as such, it is not possible to treat any of the XMEGA's working registers as though they were SRAM. Instead, the I/O registers are mapped into the data address space starting at the very beginning of the address space. Additionally, the amount of data address space dedicated to I/O registers has grown substantially to 4096 bytes (000016–0FFF16). As with previous generations, however, the fast I/O manipulation instructions can only reach the first 64 I/O register locations (the first 32 locations for bitwise instructions). Following the I/O registers, the XMEGA series sets aside a 4096 byte range of the data address space which can be used optionally for mapping the internal EEPROM to the data address space (100016–1FFF16). The actual SRAM is located after these ranges, starting at 200016.

#### EEPROM

Almost all AVR microcontrollers have internal EEPROM for semi-permanent data storage. Like flash memory, EEPROM can maintain its contents when electrical power is removed.

In most variants of the AVR architecture, this internal EEPROM memory is not mapped into the MCU's addressable memory space. It can only be accessed the same way an external peripheral device is, using special pointer registers and read/write instructions which makes EEPROM access much slower than other internal RAM.

However, some devices in the SecureAVR (AT90SC) family use a special EEPROM mapping to the data or program memory depending on the configuration. The XMEGA family also allows the EEPROM to be mapped into the data address space.

Since the number of writes to EEPROM is not unlimited — Atmel specifies 100,000 write cycles in their datasheets — a well designed EEPROM write routine should compare the contents of an EEPROM address with desired contents and only perform an actual write if the contents need to be changed.

Note that erase and write can be performed separately in many cases, byte-by-byte, which may also help prolong life when bits only need to be set to all 1s (erase) or selectively cleared to 0s (write).

### Program execution

Atmel's AVRs have a two stage, single level pipeline design. This means the next machine instruction is fetched as the current one is executing. Most instructions take just one or two clock cycles, making AVRs relatively fast among eight-bit microcontrollers.

The AVR processors were designed with the efficient execution of compiled C code in mind and have several built-in pointers for the task.

**MCU speed**

The AVR line can normally support clock speeds from 0 to 20 MHz, with some devices reaching 32 MHz. Lower powered operation usually requires a reduced clock speed. All recent (Tiny, Mega, and Xmega, but not 90S) AVRs feature an on-chip oscillator, removing the need for external clocks or resonator circuitry. Some AVRs also have a system clock prescaler that can divide down the system clock by up to 1024. This prescaler can be reconfigured by software during run-time, allowing the clock speed to be optimized.

Since all operations (excluding literals) on registers R0 - R31 are single cycle, the AVR can achieve up to 1 MIPS per MHz, i.e. an 8 MHz processor can achieve up to 8 MIPS. Loads and stores to/from memory take two cycles, branching takes two cycles. Branches in the latest "3-byte PC" parts such as ATmega2560 are one cycle slower than on previous devices

**Features:**

• High-performance, Low-power Atmel®AVR® 8-bit Microcontroller

**• Advanced RISC Architecture**

– 130 Powerful Instructions – Most Single-clock Cycle Execution

– 32 × 8 General Purpose Working Registers

– Fully Static Operation

– Up to 16MIPS Throughput at 16MHz

– On-chip 2-cycle Multiplier

**• High Endurance Non-volatile Memory segments**

– 8Kbytes of In-System Self-programmable Flash program memory

– 512Bytes EEPROM

– 1Kbyte Internal SRAM

– Write/Erase Cycles: 10,000 Flash/100,000 EEPROM

– Data retention: 20 years at 85°C/100 years at 25°C(1)

– Optional Boot Code Section with Independent Lock Bits

In-System Programming by On-chip Boot Program

True Read-While-Write Operation

– Programming Lock for Software Security

**• Peripheral Features**

– Two 8-bit Timer/Counters with Separate Prescaler, one Compare Mode

– One 16-bit Timer/Counter with Separate Prescaler, Compare Mode, and Capture

Mode

– Real Time Counter with Separate Oscillator

– Three PWM Channels

– 8-channel ADC in TQFP and QFN/MLF package

Eight Channels 10-bit Accuracy

– 6-channel ADC in PDIP package

Six Channels 10-bit Accuracy

– Byte-oriented Two-wire Serial Interface

– Programmable Serial USART

– Master/Slave SPI Serial Interface

– Programmable Watchdog Timer with Separate On-chip Oscillator

– On-chip Analog Comparator

**• Special Microcontroller Features**

– Power-on Reset and Programmable Brown-out Detection

– Internal Calibrated RC Oscillator

– External and Internal Interrupt Sources

– Five Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, and

Standby

**• I/O and Packages**

– 23 Programmable I/O Lines

– 28-lead PDIP, 32-lead TQFP, and 32-pad QFN/MLF

**• Operating Voltages**

– 2.7V - 5.5V (ATmega8L)

– 4.5V - 5.5V (ATmega8)

**• Speed Grades**

– 0 - 8MHz (ATmega8L)

– 0 - 16MHz (ATmega8)

**• Power Consumption at 4Mhz, 3V, 25oC**

– Active: 3.6mA

– Idle Mode: 1.0mA

– Power-down Mode: 0.5µA

**Brown-out Detector:**

If the Brown-out Detector is not needed in the application, this module should be turned off. If the Brown-out Detector is enabled by the BODEN Fuse, it will be enabled in all sleep modes, and hence, always consume power. In the deeper sleep modes, this will contribute significantly to the total current consumption. Refer to “Brown-out Detection” on page 38 for details on how to configure the Brown-out Detector.

Internal Voltage Referencethe Internal Voltage Reference will be enabled when needed by the Brown-out Detector, the Analog Comparator or the ADC. If these modules are disabled as described in the sections above, the internal voltage reference will be disabled and it will not be consuming power. When turned on again, the user must allow the reference to start up before the output is used. If the reference is kept on in sleep mode, the output can be used immediately. Refer to “Internal Voltage Reference” on page 40 for details on the start-up time. Watchdog Timer If the Watchdog Timer is not needed in the application, this module should be turned off.

If the Watchdog Timer is enabled, it will be enabled in all sleep modes, and hence, always consume power. In the deeper sleep modes, this will contribute significantly to the total current consumption. Refer to “Watchdog Timer” on page 41 for details on how to configure the Watchdog Timer. Port Pins When entering a sleep mode, all port pins should be configured to use minimum power.

The most important thing is then to ensure that no pins drive resistive loads. In sleep modes where the both the I/O clock (clkI/O) and the ADC clock (clkADC) are stopped, the input buffers of the device will be disabled. This ensures that no power is consumed by the input logic when not needed. In some cases, the input logic is needed for detecting wake-up conditions, and it will then be enabled. Refer to the section “Digital Input Enable and Sleep Modes” on page 53 for details on which pins are enabled. If the input buffer is enabled and the input signal is left floating or have an analog signal level close to VCC/2, the input buffer will use excessive power.

**Power-on Reset:**

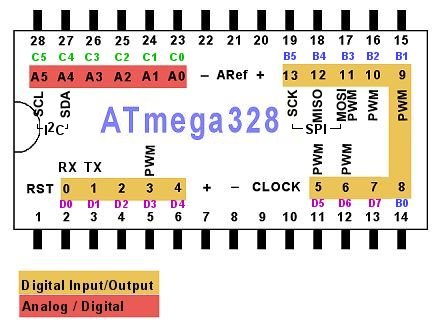
A Power-on Reset (POR) pulse is generated by an On-chip detection circuit. The detection level is defined in Table 15. The POR is activated whenever VCC is below thedetection level. The POR circuit can be used to trigger the Start-up Reset, as well as to detect a failure in supply voltage.

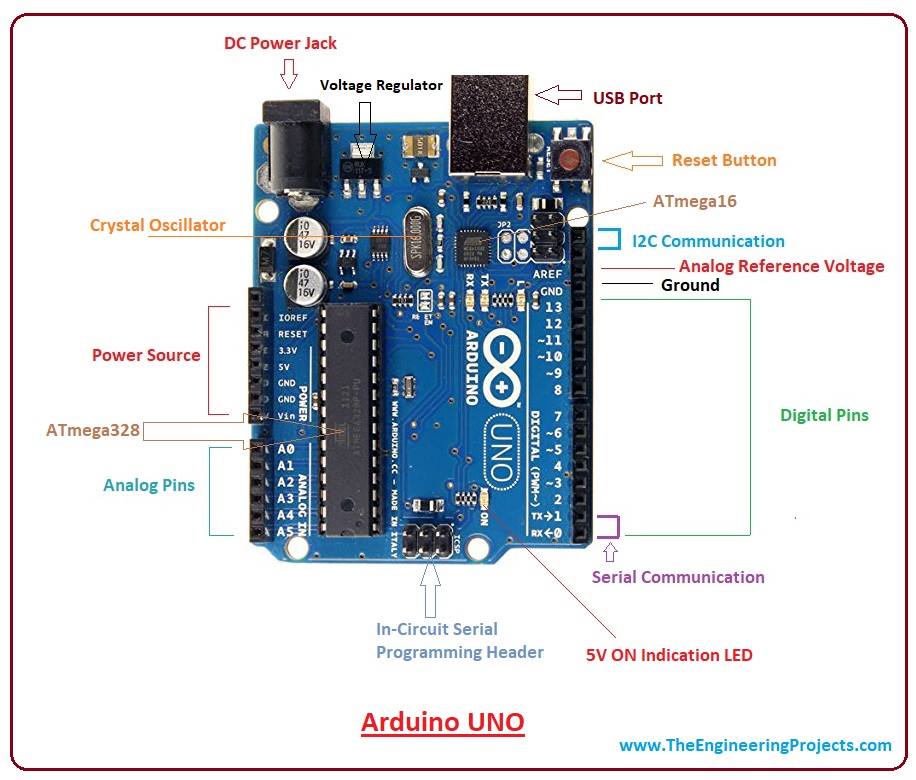
A Power-on Reset (POR) circuit ensures that the device is reset from Power-on. Reaching the Power-on Reset threshold voltage invokes the delay counter, which determineshow long the device is kept in RESET after VCC rise. The RESET signal is activated again, without any delay, when VCC decreases below the detection level.

**External Reset**:

An External Reset is generated by a low level on the RESET pin. Reset pulses longerthan the minimum pulse width (see Table 15) will generate a reset, even if the clock isnot running. Shorter pulses are not guaranteed to generate a reset. When the appliedsignal reaches the Reset Threshold Voltage – VRST on its positive edge, the delaycounter starts the MCU after the time-out period tTOUT has expired.

**3.2.3 Pin diagram:**





**Fig.3.2.4.PIN DIAGRAM OF ATMEGA328**

**VCC**  
Digital supply voltage magnitude of the voltage range between 4.5 to 5.5 V for the ATmega8 and 2.7 to 5.5 V for ATmega8L

**GND**  
Ground Zero reference digital voltage supply.

**PORTB (PB7.. PB0)**

PORTB is a port I / O two-way (bidirectional) 8-bit with internal pull-up resistor can be selected. This port output buffers have symmetrical characteristics when used as a source or sink. When used as an input, the pull-pin low externally will emit a current if the pull-up resistor is activated it. PORTB pins will be in the condition of the tri-state when RESET is active, although the clock is not running.

**PORTC (PC5.. PC0)**

PORTC is a port I / O two-way (bidirectional) 7-bit with internal pull-up resistor can be selected. This port output buffers have symmetrical characteristics when used as a source or sink. When used as an input, the pull-pin low externally will emit a current if the pull-up resistor is activated it. PORTC pins will be in the condition of the tri-state when RESET is active, although the clock is not running.

**PC6/RESET**  
If RSTDISBL Fuse programmed, PC6 then serves as a pin I / O but with different characteristics. PC0 to PC5 If Fuse RSTDISBL not programmed, then serves as input Reset PC6. LOW signal on this pin with a minimum width of 1.5 microseconds will bring the microcontroller into reset condition, although the clock is not running.

**PORTD (PD7.. PD0)**

PORTD is a port I / O two-way (bidirectional) 8-bit with internal pull-up resistor can be selected. This port output buffers have symmetrical characteristics when used as a source or sink. When used as an input, the pull-pin low externally will emit a current if the pull-up resistor is activated it. PORTD pins will be in the condition of the tri-state when RESET is active, although the clock is not running.

**RESET**Reset input pin. LOW signal on this pin with a minimum width of 1.5 microseconds will bring the microcontroller into reset condition, although the clock is not running. Signal with a width of less than 1.5 microseconds does not guarantee a Reset condition.

**AVCC**  
AVCC is the supply voltage pin for the ADC, PC3 .. PC0, and ADC7..ADC6. This pin should be connected to VCC, even if the ADC is not used. If the ADC is used, AVCC should be connected to VCC through a low-pass filter to reduce noise.

**Aref**  
Analog Reference pin for the ADC.

**ADC7 .. ADC6**

ADC analog input there is only on ATmega8 with TQFP and QFP packages / MLF.

**PORTS**

Term "port" refers to a group of pins on a microcontroller which can be accessed simultaneously, or on which we can set the desired combination of zeros and ones, or read from them an existing status. Physically, port is a register inside a microcontroller which is connected by wires to the pins of a microcontroller. Ports represent physical connection of Central Processing Unit with an outside world. Microcontroller uses them

The [Atmega8](http://www.protostack.com/product_by_model.php?model=IC-ATMEGA8-16PU) has 23 I/O ports which are organized into 3 groups:

* Port B (PB0 to PB7)
* Port C (PC0 to PC6)
* Port D (PD0 to PD7)

We will use mainly 3 registers known as **DDRX, PORTX**&**PINX**. We have total four PORTs on my ATmega16. They are **PORTA, PORTB, PORTC** and**PORTD**. They are multifunctional pins. Each of the pins in each port (total 32) can be treated as input or output pin.

**Applications**

AVR microcontroller perfectly fits many uses, from automotive industries and controlling home appliances to industrial instruments, remote sensors, electrical door locks and safety devices. It is also ideal for smart cards as well as for battery supplied devices because of its low consumption.

EEPROM memory makes it easier to apply microcontrollers to devices where permanent storage of various parameters is needed (codes for transmitters, motor speed, receiver frequencies, etc.). Low cost, low consumption, easy handling and flexibility make ATmega8 applicable even in areas where microcontrollers had not previously been considered (example: timer functions, interface replacement in larger systems, coprocessor applications, etc.).

In System Programmability of this chip (along with using only two pins in data transfer) makes possible the flexibility of a product, after assembling and testing have been completed. This capability can be used to create assembly-line production, to store calibration data available only after final testing, or it can be used to improve programs on finished products.

**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. The figure below shows the block diagram of a typical regulated DC power supply.

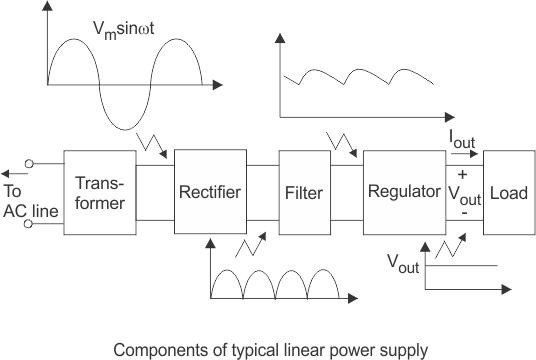


Figure 6.8 Block diagram of power supply

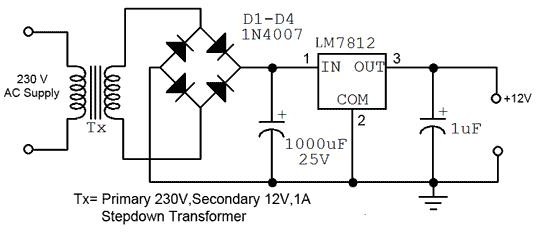
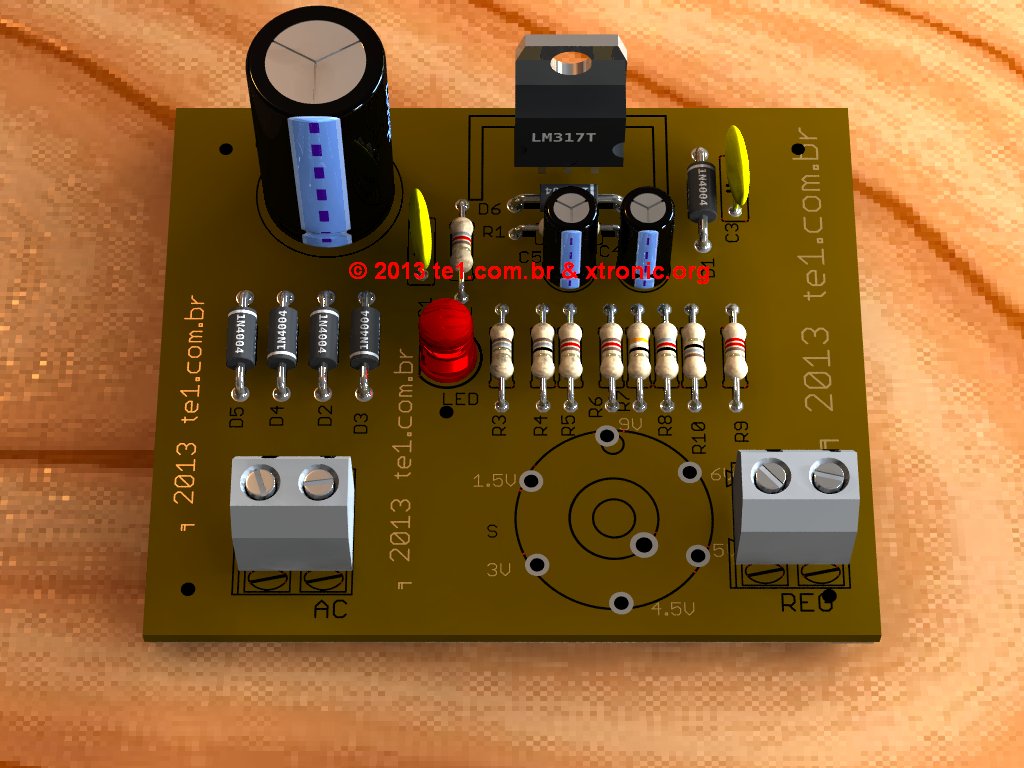
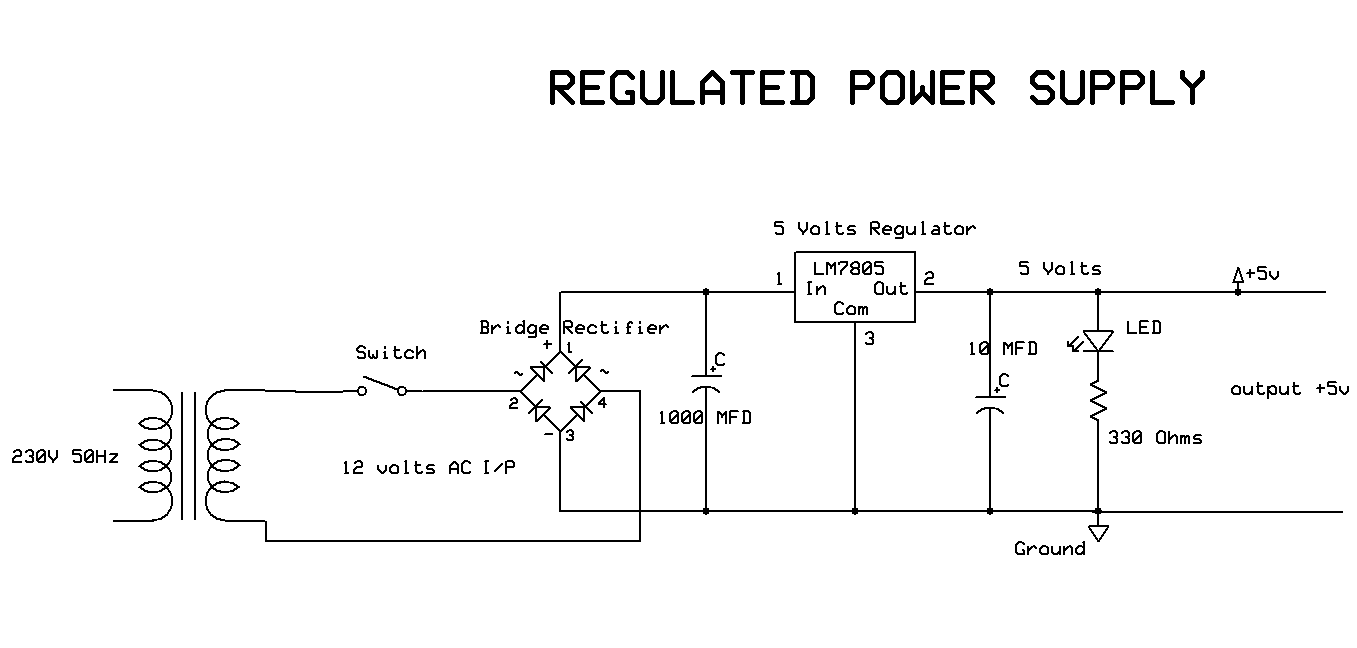


Figure 6.9 Power supply circuit diagram

****

**Fig6.10 Regulated Power Supply**

The basic circuit diagram of a regulated power supply (DC O/P) with led connected as load is shown in fig: 3.3.3.



**Fig 6.11 Circuit diagram of Regulated Power Supply with Led connection**

The components mainly used in above figure are

* 230V AC MAINS
* TRANSFORMER
* BRIDGE RECTIFIER(DIODES)
* CAPACITOR
* VOLTAGE REGULATOR(IC 7805)
* RESISTOR
* LED(LIGHT EMITTING DIODE)

The detailed explanation of each and every component mentioned above is as follows:

**Transformation:** The process of transforming energy from one device to another is called transformation. For transforming energy we use transformers.

**Step down transformer:**

Incase of step down transformer, Primary winding induces more flux than the secondary winding, and secondary winding is having less number of turns because of that it accepts less number of flux, and releases less amount of voltage.

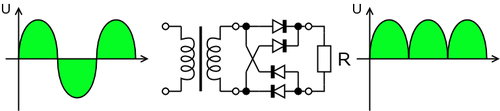
**Bridge full wave rectifier:**

The Bridge rectifier circuit is shown in fig:3.8, which converts an ac voltage to dc voltage using both half cycles of the input ac voltage. The Bridge rectifier circuit is shown in the figure. The circuit has four diodes connected to form a bridge. The ac input voltage is applied to the diagonally opposite ends of the bridge. The load resistance is connected between the other two ends of the bridge.

For the positive half cycle of the input ac voltage, diodes D1 and D3 conduct, whereas diodes D2 and D4 remain in the OFF state. The conducting diodes will be in series with the load resistance RL and hence the load current flows through RL.

For the negative half cycle of the input ac voltage, diodes D2 and D4 conduct whereas, D1 and D3 remain OFF. The conducting diodes D2 and D4 will be in series with the load resistance RL and hence the current flows through RL in the same direction as in the previous half cycle. Thus a bi-directional wave is converted into a unidirectional wave.

Input Output

****

**Fig 6.12 Bridge rectifier: a full-wave rectifier using 4 diodes**

**DB107:**

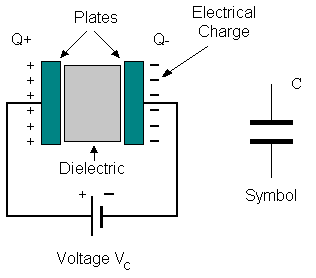
Now -a -days Bridge rectifier is available in IC with a number of DB107. In our project we are using an IC in place of bridge rectifier. The picture of DB 107 is shown in fig: 3.3.8.



**Fig 6.13. DB107**

**Capacitor Filtration:**

The process of converting a pulsating direct current to a pure direct current using filters is called as filtration.Electronic filters are electronic circuits, which perform signal-processing functions, specifically to remove unwanted frequency components from the signal, to enhance wanted ones.The **Capacitor** or sometimes referred to as a Condenser is a passive device, and one which stores energy in the form of an electrostatic field which produces a potential (static voltage) across its plates. In its basic form a capacitor consists of two parallel conductive plates that are not connected but are electrically separated either by air or by an insulating material called the Dielectric. When a voltage is applied to these plates, a current flows charging up the plates with electrons giving one plate a positive charge and the other plate an equal and opposite negative charge this flow of electrons to the plates is known as the Charging Current and continues to flow until the voltage across the plates (and hence the capacitor) is equal to the applied voltage Vcc. At this point the capacitor is said to be fully charged and this is illustrated below. The construction of capacitor and an electrolytic capacitor are shown in figures 3.3.9 and 3.3.10 respectively.

****

**Fig 6.14 Construction Of a Capacitor Fig 6.15 Electrolytic**

**Voltage Regulation:**

The process of converting a varying voltage to a constant regulated voltage is called as regulation. For the process of regulation we use voltage regulators.

**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. These can withstand over-current draw due to short circuits and also over-heating. In both cases, the regulator will cut off before any damage occurs. 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 6.16. Voltage Regulator**

**Resistor:**

A resistor is a two-terminal electronic component that produces a voltage across its terminals that is proportional to the electric current passing through it in accordance with Ohm's law:

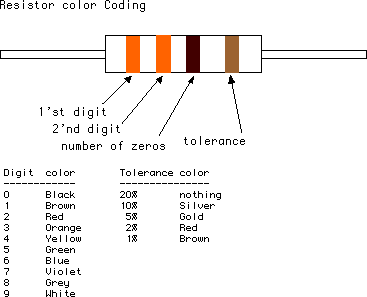
*V* = *IR*

Resistors are elements of electrical networks and electronic circuits and are ubiquitous in most electronic equipment. Practical resistors can be made of various compounds and films, as well as resistance wire (wire made of a high-resistivity alloy, such as nickel/chrome).

The primary characteristics of a resistor are the resistance, the tolerance, maximum working voltage and the power rating. Other characteristics include temperature coefficient, noise, and inductance. Less well-known is critical resistance, the value below which power dissipation limits the maximum permitted current flow, and above which the limit is applied voltage. Critical resistance is determined by the design, materials and dimensions of the resistor.

dissipated by a resistor (or the equivalent resistance of a resistor network) is calculated using the following:

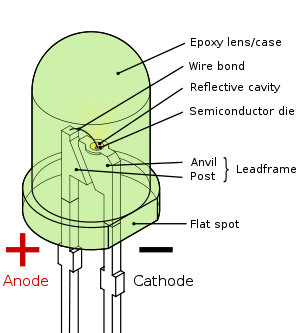
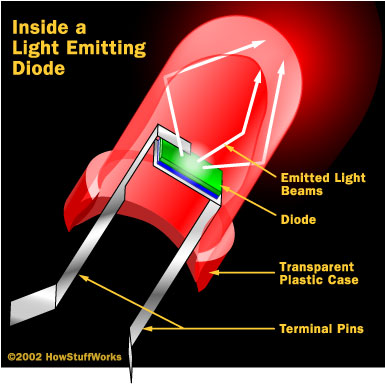
P = I^2 R = I V = \frac{V^2}{R}

****

**Fig 6.18 Resistor Fig 6.19 Color Bands In Resistor**

**LED:**

A light-emitting diode (LED) is a semiconductor light source. LED’s are used as indicator lamps in many devices, and are increasingly used for lighting. Introduced as a practical electronic component in 1962, early LED’s emitted low-intensity red light, but modern versions are available across the visible, ultraviolet and infrared wavelengths, with very high brightness. The internal structure and parts of a led are shown in figures 3.4.1 and 3.4.2 respectively.



**Fig.6.20 Inside a LED Fig 6.21. Parts of a LED**

**3.5 IOT ESP8266 MODULE**

**1 Introduction**

Espressif Systems’ Smart Connectivity Platform (ESCP) of high performance wireless SOCs, for mobile platform designers, provides unsurpassed ability to embed Wi-Fi capabilities within other systems, at the lowest cost with the greatest functionality

**2 Technology Overview**

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.

**3 Features**

* 802.11 b/g/n protocol
* Wi-Fi Direct (P2P), soft-AP
* Integrated TCP/IP protocol stack
* Integrated TR switch, balun, LNA, power amplifier and matching network
* Integrated PLL, regulators, and power management units
* +19.5dBm output power in 802.11b mode
* Integrated temperature sensor
* Supports antenna diversity
* Power down leakage current of < 10uA
* Integrated low power 32-bit CPU could be used as application processor
* SDIO 2.0, SPI, UART
* STBC, 1×1 MIMO, 2×1 MIMO
* A-MPDU & A-MSDU aggregation & 0.4s guard interval
* Wake up and transmit packets in < 2ms
* Standby power consumption of < 1.0mW (DTIM3)

**ESP8266 Applications**

* Smart power plugs
* Home automation
* Mesh network
* Industrial wireless control
* Baby monitors
* IP Cameras
* Sensor networks
* Wearable electronics
* Wi-Fi location-aware devices
* Security ID tags
* Wi-Fi position system beacons

**Specifications**

**7.1 Current Consumption**

The following current consumption is based on 3.3V supply, and 25C ambient, using internal regulators. Measurements are done at antenna port without SAW filter. All the transmitter’s measurements are based on 90% duty cycle, continuous transmit mode.

|  |  |  |  |
| --- | --- | --- | --- |
| **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 |

**CPU, Memory and Interfaces**

**CPU**

This chip embeds an ultra low power Micro 32-bit CPU, with 16-bit thumb mode. This CPU can be interfaced using:

* code RAM/ROM interface (iBus) that goes to the memory controller, that can also be used to access external flash memory,
* data RAM interface (dBus), that also goes to the memory controller
* AHB interface, for register access, and
* JTAG interface for debugging

**Memory Controller**

The memory controller contains ROM, and SRAM. It is accessed by the CPU using the iBus, dBus and AHB interface. Any of these interfaces can request access to the ROM or RAM modules, and the memory controller arbiters serve these 3 interfaces on a first-come-first-serve basis.

**AHB and AHB Blocks**

The AHB blocks performs the function of an arbiter, controls the AHB interfaces from the MAC, SDIO (host) and CPU. Depending on the address, the AHB data requests can go into one of the two slaves:

* APB block, or
* flash controller (usually for standalone applications).

Data requests to the memory controller are usually high speed requests, and requests to the APB block are usually register access.

The APB block acts as a decoder. It is meant only for access to programmable registers within ESP8266’s main blocks. Depending on the address, the APB request can go to the radio, SI/SPI, SDIO (host), GPIO, UART, real-time clock (RTC), MAC or digital baseband.

**Interfaces**

The ESP8266 contains several analog and digital interfaces described in the following sections.

**Master SI / SPI Control (Optional)**

The master serial interface (SI) can operate in two, three or four-wire bus configurations to control the EEPROM or other I2C/SPI devices. Multiple I2C devices with different device addresses are supported by sharing the 2-wire bus.

Multiple SPI devices are supported by sharing the clock and data signals, using separate software controlled GPIO pins as chip selects.

The SPI can be used for controlling external devices such as serial flash memories, audio CODECs, or other slave devices. It is set up as a standard master SPI device with 3 different enable pins:

* SPI\_EN0,
* SPI\_EN1,
* SPI\_EN2.

Both SPI master and SPI slave are supported with the latter being used as a host interface.

SPI\_EN0 is used as an enable signal to an external serial flash memory for downloading patch code and/or MIB-data to the baseband in an embedded application. In a host based application, patch code and MIB-data can alternatively be downloaded via the host interface. This pin is active low and should be left open if not used.

SPI\_EN1 is usually used for a user application, e.g. to control an external audio codec or sensor ADC, in an embedded application. This pin is active low and should be left open if not used.

SPI\_EN2 usually controls an EEPROM to store individual data, such as MIB information, MAC address, and calibration data, or for general use. This pin is active low and should be left open if not used.

**General Purpose IO**

There are up to 16 GPIO pins. They can be assigned to various functions by the firmware. Each GPIO can be configured with internal pull-up/down, input available for sampling by a software register, input triggering an edge or level CPU interrupt, input triggering a level wakeup interrupt, open-drain or push-pull output driver, or output source from a software register, or a sigma-delta PWM DAC.

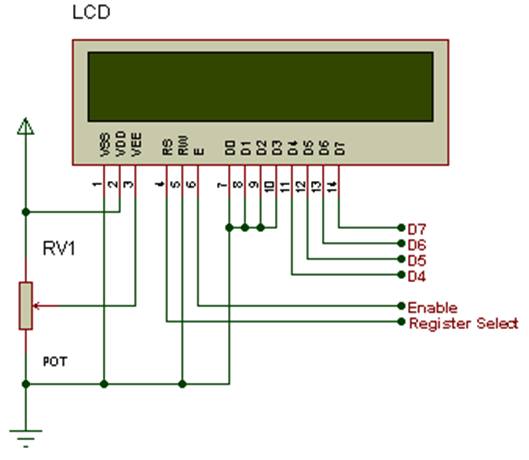
These pins are multiplexed with other functions such as host interface, UART, SI, Bluetooth coexistence, etc.

**3.7 LCD DISPLAY**

**LCD Background:**

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**

****

**Pin description:**

|  |  |  |
| --- | --- | --- |
| **Pin No.** | **Name** | **Description** |
| Pin no. 1 | **VSS** | Power supply (GND) |
| Pin no. 2 | **VCC** | Power supply (+5V) |
| Pin no. 3 | **VEE** | Contrast adjust |
| Pin no. 4 | **RS** | 0 = Instruction input 1 = Data input |
| Pin no. 5 | **R/W** | 0 = Write to LCD module 1 = Read from LCD module |
| Pin no. 6 | **EN** | Enable signal |
| Pin no. 7 | **D0** | Data bus line 0 (LSB) |
| Pin no. 8 | **D1** | Data bus line 1 |
| Pin no. 9 | **D2** | Data bus line 2 |
| Pin no. 10 | **D3** | Data bus line 3 |
| Pin no. 11 | **D4** | Data bus line 4 |
| Pin no. 12 | **D5** | Data bus line 5 |
| Pin no. 13 | **D6** | Data bus line 6 |
| Pin no. 14 | **D7** | Data bus line 7 (MSB) |

**Table 1: Character LCD pins with Microcontroller**

The LCD requires 3 control lines as well as either 4 or 8 I/O lines for the data bus. The user may select whether the LCD is to operate with a 4-bit data bus or an 8-bit data bus. If a 4-bit data bus is used the LCD will require a total of 7 data lines (3 control lines plus the 4 lines for the data bus). If an 8-bit data bus is used the LCD will require a total of 11 data lines (3 control lines plus the 8 lines for the data bus).

The three control lines are referred to as **EN**, **RS**, and **RW**.

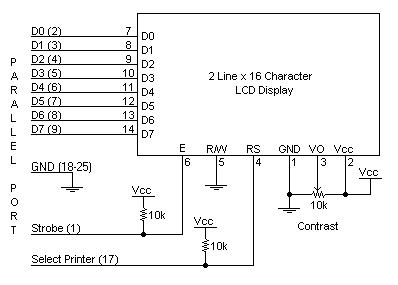
The **EN** line is called "Enable." This control line is used to tell the LCD that we are sending it data. To send data to the LCD, our program should make sure this line is low (0) and then set the other two control lines and/or put data on the data bus. When the other lines are completely ready, bring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

The **RS** line is the "Register Select" line. When RS is low (0), the data is to be treated as a command or special instruction (such as clear screen, position cursor, etc.). When RS is high (1), the data being sent is text data which should be displayed on the screen. For example, to display the letter "T" on the screen we would set RS high.

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

Finally, the data bus consists of 4 or 8 lines (depending on the mode of operation selected by the user). In the case of an 8-bit data bus, the lines are referred to as DB0, DB1, DB2, DB3, DB4, DB5, DB6, and DB7.

**Schematic:**

****

**Circuit Description:**

Above is the quite simple schematic. The LCD panel's Enable and RegisterSelect is connected to the ControlPort. The ControlPort is an open collector / open drain output. While most Parallel Ports have internal pull-up resistors, there is a few which don't. Therefore by incorporating the two 10K external pull up resistors, the circuit is more portable for a wider range of computers, some of which may have no internal pull up resistors.

We make no effort to place the Data bus into reverse direction. Therefore we hard wire the R/W line of the LCD panel, into write mode. This will cause no bus conflicts on the data lines. As a result we cannot read back the LCD's internal Busy Flag which tells us if the LCD has accepted and finished processing the last instruction. This problem is overcome by inserting known delays into our program.

The 10k Potentiometer controls the contrast of the LCD panel. Nothing fancy here. As with all the examples, I've left the power supply out. We can use a bench power supply set to 5v or use an onboard +5 regulator. Remember a few de-coupling capacitors, especially if we have trouble with the circuit working properly.

**SETB RW**

Handling the EN control line:

As we mentioned above, the EN line is used to tell the LCD that we are ready for it to execute an instruction that we've prepared on the data bus and on the other control lines. Note that the EN line must be raised/ lowered before/after each instruction sent to the LCD regardless of whether that instruction is read or write text or instruction. In short, we must always manipulate EN when communicating with the LCD. EN is the LCD's way of knowing that we are talking to it. If we don't raise/lower EN, the LCD doesn't know we're talking to it on the other lines.

Thus, before we interact in any way with the LCD we will always bring the **EN** line low with the following instruction:

**CLR EN**

And once we've finished setting up our instruction with the other control lines and data bus lines, we'll always bring this line high:

**SETB EN**

The line must be left high for the amount of time required by the LCD as specified in its datasheet. This is normally on the order of about 250 nanoseconds, but checks the datasheet. In the case of a typical microcontroller running at 12 MHz, an instruction requires 1.08 microseconds to execute so the EN line can be brought low the very next instruction. However, faster microcontrollers (such as the DS89C420 which executes an instruction in 90 nanoseconds given an 11.0592 MHz crystal) will require a number of NOPs to create a delay while EN is held high. The number of NOPs that must be inserted depends on the microcontroller we are using and the crystal we have selected.

The instruction is executed by the LCD at the moment the EN line is brought low with a final CLR EN instruction.

**Checking the busy status of the LCD:**

As previously mentioned, it takes a certain amount of time for each instruction to be executed by the LCD. The delay varies depending on the frequency of the crystal attached to the oscillator input of the LCD as well as the instruction which is being executed.

While it is possible to write code that waits for a specific amount of time to allow the LCD to execute instructions, this method of "waiting" is not very flexible. If the crystal frequency is changed, the software will need to be modified. A more robust method of programming is to use the "Get LCD Status" command to determine whether the LCD is still busy executing the last instruction received.

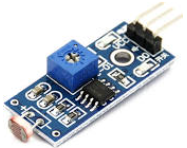
The "Get LCD Status" command will return to us two tidbits of information; the information that is useful to us right now is found in DB7. In summary, when we issue the "Get LCD Status" command the LCD will immediately raise DB7 if it's still busy executing a command or lower DB7 to indicate that the LCD is no longer occupied. Thus our program can query the LCD until DB7 goes low, indicating the LCD is no longer busy. At that point we are free to continue and send the next command.

**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:

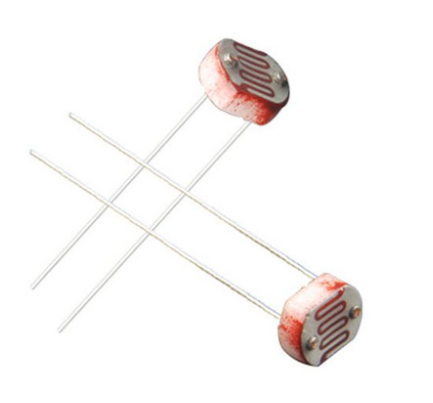
A **photo resistor** or **light dependent resistor** or **cadmium sulfide (CdS) cell** is a resistor whose resistance decreases with increasing incident light intensity. It can also be referenced as a **photoconductor**.



# Photo resistor



**The symbol for a photo resistor**

****

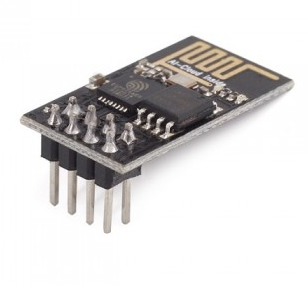
**Fig 3.5.1.LDR**

A light dependent resistor 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.mA photo resistor is made of a high resistance semiconductor. If light falling on the device is of high enough frequency, photons absorbed by the semiconductor give bound electrons enough energy to jump into the conduction band. The resulting free electron (and its whole partner) conducts electricity, thereby lowering resistance. 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.

**3.5 IOT ESP8266 MODULE**



**1 Introduction**

Espressif Systems’ Smart Connectivity Platform (ESCP) of high performance wireless SOCs, for mobile platform designers, provides unsurpassed ability to embed Wi-Fi capabilities within other systems, at the lowest cost with the greatest functionality

**2 Technology Overview**

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.

**3 Features**

* 802.11 b/g/n protocol
* Wi-Fi Direct (P2P), soft-AP
* Integrated TCP/IP protocol stack
* Integrated TR switch, balun, LNA, power amplifier and matching network
* Integrated PLL, regulators, and power management units
* +19.5dBm output power in 802.11b mode
* Integrated temperature sensor
* Supports antenna diversity
* Power down leakage current of < 10uA
* Integrated low power 32-bit CPU could be used as application processor
* SDIO 2.0, SPI, UART
* STBC, 1×1 MIMO, 2×1 MIMO
* A-MPDU & A-MSDU aggregation & 0.4s guard interval
* Wake up and transmit packets in < 2ms
* Standby power consumption of < 1.0mW (DTIM3)

**ESP8266 Applications**

* Smart power plugs
* Home automation
* Mesh network
* Industrial wireless control
* Baby monitors
* IP Cameras
* Sensor networks
* Wearable electronics
* Wi-Fi location-aware devices
* Security ID tags
* Wi-Fi position system beacons

**Specifications**

**7.1 Current Consumption**

The following current consumption is based on 3.3V supply, and 25C ambient, using internal regulators. Measurements are done at antenna port without SAW filter. All the transmitter’s measurements are based on 90% duty cycle, continuous transmit mode.

|  |  |  |  |
| --- | --- | --- | --- |
| **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 |

**CPU, Memory and Interfaces**

**CPU**

This chip embeds an ultra low power Micro 32-bit CPU, with 16-bit thumb mode. This CPU can be interfaced using:

* code RAM/ROM interface (iBus) that goes to the memory controller, that can also be used to access external flash memory,
* data RAM interface (dBus), that also goes to the memory controller
* AHB interface, for register access, and
* JTAG interface for debugging

**Memory Controller**

The memory controller contains ROM, and SRAM. It is accessed by the CPU using the iBus, dBus and AHB interface. Any of these interfaces can request access to the ROM or RAM modules, and the memory controller arbiters serve these 3 interfaces on a first-come-first-serve basis.

**AHB and AHB Blocks**

The AHB blocks performs the function of an arbiter, controls the AHB interfaces from the MAC, SDIO (host) and CPU. Depending on the address, the AHB data requests can go into one of the two slaves:

* APB block, or
* flash controller (usually for standalone applications).

Data requests to the memory controller are usually high speed requests, and requests to the APB block are usually register access.

The APB block acts as a decoder. It is meant only for access to programmable registers within ESP8266’s main blocks. Depending on the address, the APB request can go to the radio, SI/SPI, SDIO (host), GPIO, UART, real-time clock (RTC), MAC or digital baseband.

**Interfaces**

The ESP8266 contains several analog and digital interfaces described in the following sections.

**Master SI / SPI Control (Optional)**

The master serial interface (SI) can operate in two, three or four-wire bus configurations to control the EEPROM or other I2C/SPI devices. Multiple I2C devices with different device addresses are supported by sharing the 2-wire bus.

Multiple SPI devices are supported by sharing the clock and data signals, using separate software controlled GPIO pins as chip selects.

The SPI can be used for controlling external devices such as serial flash memories, audio CODECs, or other slave devices. It is set up as a standard master SPI device with 3 different enable pins:

* SPI\_EN0,
* SPI\_EN1,
* SPI\_EN2.

Both SPI master and SPI slave are supported with the latter being used as a host interface.

SPI\_EN0 is used as an enable signal to an external serial flash memory for downloading patch code and/or MIB-data to the baseband in an embedded application. In a host based application, patch code and MIB-data can alternatively be downloaded via the host interface. This pin is active low and should be left open if not used.

SPI\_EN1 is usually used for a user application, e.g. to control an external audio codec or sensor ADC, in an embedded application. This pin is active low and should be left open if not used.

SPI\_EN2 usually controls an EEPROM to store individual data, such as MIB information, MAC address, and calibration data, or for general use. This pin is active low and should be left open if not used.

**General Purpose IO**

There are up to 16 GPIO pins. They can be assigned to various functions by the firmware. Each GPIO can be configured with internal pull-up/down, input available for sampling by a software register, input triggering an edge or level CPU interrupt, input triggering a level wakeup interrupt, open-drain or push-pull output driver, or output source from a software register, or a sigma-delta PWM DAC.

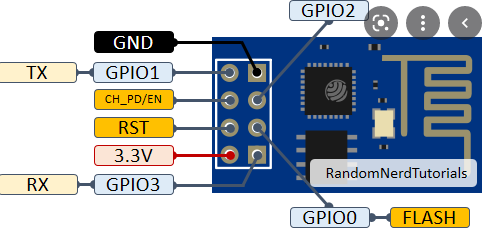
These pins are multiplexed with other functions such as host interface, UART, SI, Bluetooth coexistence, etc.

**Digital IO Pads**

The digital IO pads are bidirectional, non-inverting and tri-state. It includes input and an output buffer with tristate control inputs. Besides this, for low power operations, the IO can also be set to hold. For instance, when we power down the chip, all output enable signals can be set to hold low. Optional hold functionality can be built into the IO if requested. When the IO is not driven by the internal or external circuitry, the hold functionality can be used to hold the state to the last used state.

The hold functionality introduces some positive feedback into the pad. Hence, the external driver that drives the pad must be stronger than the positive feedback. The required drive strength is however small – in the range of 5uA.

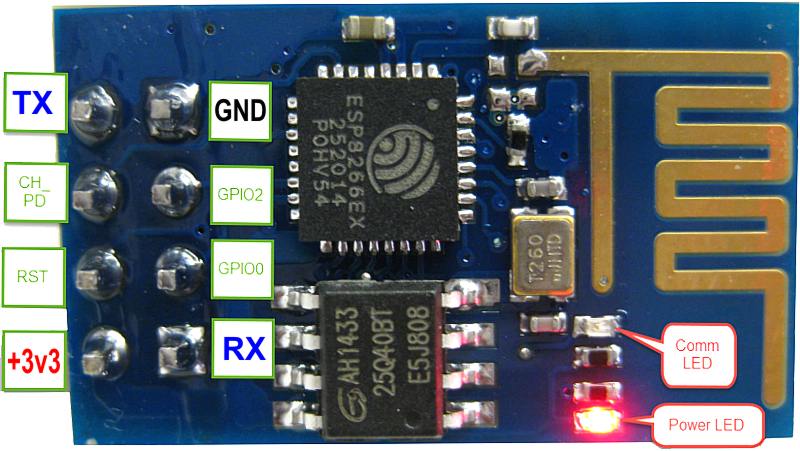
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Parameter** | **Symbol** | **Min** | **Max** | **Unit** |
| **Input low voltage** | **VIL** | **-0.3** | **0.25VIO** | **V** |
| **Input high voltage** | **VIH** | **0.75VIO** | **3.6** | **V** |
| **Input leakage current** |  | **IIL** | **50** | **nA** |
| **Output low voltage** | | **VOL** | **0.1VIO** | **V** |
| **Output high voltage** | | **VOH** | **0.8VIO** | **V** |
| **Input pin capacitance** | | **Cpad** | **2** | **pF** |
| **VDDIO** | **VIO** | **1.7** | **3.6** | **V** |
| **Maximum drive capability** | | **IMAX** | **12** | **mA** |
| **Temperature** | **Tamb** | **-20** | **100** | C |



**Applications**

* Wi-Fi Smart Hardware Converted from UART Serial Ports
* Sensor
* Smart Light
* Smart Plug

Wi-Fi (Short for Wireless Fidelity) is a wireless technology that uses radio frequency to transmit data through the air. Wi-Fi has initial speeds of 1mbps to 2mbps. Wi-Fi transmits data in the frequency band of 2.4 GHz. It implements the concept of frequency division multiplexing technology. Range of Wi-Fi technology is 40-300 feet.



2.2 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.

**2.2.1FEATURES**

* 802.11 b/g/n protocol
* Wi-Fi Direct (P2P), soft-AP
* Integrated TCP/IP protocol stack
* Integrated TR switch, balun, LNA, power amplifier and matching network
* Integrated PLL, regulators, and power management units
* +19.5dBm output power in 802.11b mode
* Integrated temperature sensor
* Supports antenna diversity
* Power down leakage current of < 10uA
* Integrated low power 32-bit CPU could be used as application processor
* SDIO 2.0, SPI, UART
* STBC, 1×1 MIMO, 2×1 MIMO
* A-MPDU & A-MSDU aggregation & 0.4s guard interval
* Wake up and transmit packets in < 2ms
* Standby power consumption of < 1.0mW (DTIM3)

**2.2.2 ESP8266 APPLICATIONS**

* Smart power plugs
* Home automation
* Mesh network
* Industrial wireless control
* Baby monitors
* IP Cameras
* Sensor networks
* Wearable electronics
* Wi-Fi location-aware devices
* Security ID tags
* Wi-Fi position system beacons

**3.6 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.3.5.1 Ultrasonic sensor module**

**Specification:**

The ultrasonic range sensor detects objects in it’s path and can be used to calculate the range to the object. It is sensitive enough to detect a 3cm diameterbroom handle at a distance of over 2m.

Voltage - 5v

Current - 0mA Typ. 50mA Max.

Frequency - 40 KHz

MaxRange - 3 m

MinRange - 3 cm

Sensitivity - Detect 3cm diameter broom handle at > 2 m

Input Trigger - 10uS Min. TTL level pulse

Echo Pulse - Positive TTL level signal, width proportional to range.

Small Size - 43mm x 20mm x 17mm height

****

**Fig.3.5.2Ultrasonic sensor timing diagram**

**Electrical connection:**

The SRF004 ultrasonic range finder has 5 connections pins. The power supply is connected to the 5V and 0V groundconnections on the SRF004. (Note that BOTHthe ‘Mode’(hole 4) and ‘0V Ground’ (hole 5) connections **MUST** beconnected to 0V for correct operation with the PICAXEsystem).

Take care not to overheat, and therefore damage, the solderconnection pads whilst makingconnections.

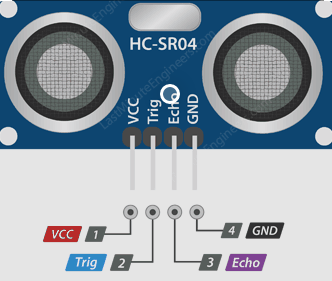
The SRF004 **Trigger Input** is connected to a PICAXE **output** pin.

The SRF004 **Echo Output** is connected to a PICAXE **input** pin.

**Ultrasonic sensors have set new standards in automation**

****

**Fig.3.5.4Pin configuration of 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.

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.

**3.5: GPS MODULE**

**Global Positioning System**

**3.5.1. Introduction:**

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 data 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.

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.

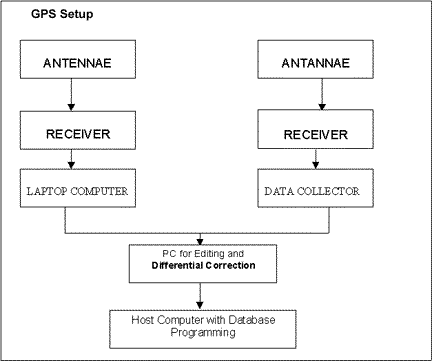
The GPS is made up of three parts: satellites orbiting the Earth; control and monitoring stations on Earth; and the GPS receivers owned by users. GPS satellites broadcast signals from space that are picked up and identified by GPS receivers. Each GPS receiver then provides three-dimensional location (latitude, longitude, and altitude) plus the time.

Individuals may purchase GPS handsets that are readily available through commercial retailers. Equipped with these GPS receivers, users can accurately locate where they are and easily navigate to where they want to go, whether walking, driving, flying, or boating. GPS has become a mainstay of transportation systems worldwide, providing navigation for aviation, ground, and maritime operations. Disaster relief and emergency services depend upon GPS for location and timing capabilities in their life-saving missions. Everyday activities such as banking, mobile phone operations, and even the control of power grids, are facilitated by the accurate timing provided by GPS. Farmers, surveyors, geologists and countless others perform their work more efficiently, safely, economically, and accurately using the free and open GPS signals.

**Geo positioning -- Basic Concepts:**   
By positioning we understand the determination of stationary or moving objects. These can be determined as follows:

1. In relation to a well-defined coordinate system, usually by three coordinate values and
2. In relation to other point, taking one point as the origin of a local coordinate system.

The first mode of positioning is known as point positioning, the second as relative positioning. If the object to be positioned is stationary, we term it as static positioning. When the object is moving, we call it kinematics positioning. Usually, the static positioning is used in surveying and the kinematics position in navigation.



This is a complete GPS module that is based on the NEO-6M. This unit uses the latest technology to give the best possible positioning information and includes a larger built-in 25 x 25mm active GPS antenna with a UART TTL socket. A battery is also included so that you can obtain a GPS lock faster. This is an updated GPS module that can be used with ardupilot mega v2. This GPS module gives the best possible position information, allowing for better performance with your Ardupilot or other Multirotor control platform.

The NEO-6M GPS engine on this board is a quite good one, with the high precision binary output. It has also high sensitivity for indoor applications. NEO-6M GPS Module has a battery for power backup and EEPROM for storing configuration settings. The antenna is connected to the module through a ufl cable which allows for flexibility in mounting the GPS such that the antenna will always see the sky for best performance. This makes it powerful to use with cars and other mobile applications.

The GPS module has serial TTL output, it has four pins: TX, RX, VCC, and GND. You can download the u-center software for configuring the GPS and changing the settings and much more.

**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)
* Separated 18X18mm GPS antenna

**CHAPTER 7**

**SOFTWARE DESCRIPTION**

he 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:

* **Arduino IDE 1.x.x** ([Windows](https://www.arduino.cc/en/Guide/Windows), [Mac OS](https://www.arduino.cc/en/Guide/macOS), [Linux](https://www.arduino.cc/en/Guide/Linux), [Portable IDE](https://www.arduino.cc/en/Guide/PortableIDE) for Windows and Linux, [ChromeOS](https://chrome.google.com/webstore/detail/arduino-create/dcgicpihgkmccjigalccipmjlnjopdfe)).
* [**Arduino IDE 2.x**](https://www.arduino.cc/en/Tutorial/getting-started-with-ide-v2/ide-v2-downloading-and-installing)

**2.** Connect your Arduino board to your device.

**3.** 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 .ino.

Using the offline IDE 1.x.x

The editor contains the four main areas:

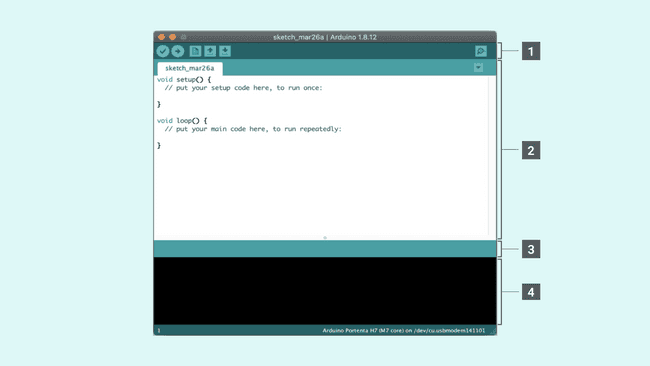
**1.** A **Toolbar with buttons** for common functions and a series of menus. The toolbar buttons allow you to verify and upload programs, create, open, and save sketches, and open the serial monitor.

**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.

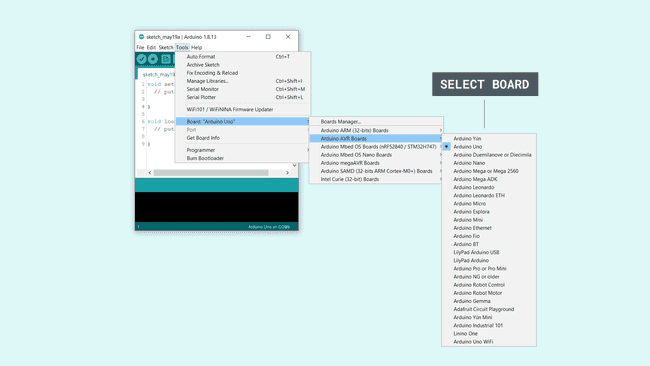
The bottom right-hand corner of the window displays the configured board and serial port.

[](https://docs.arduino.cc/static/ebf961f81c305e826d52e3401863b48e/29114/arduino-ide-interface.png)The Arduino Software IDE

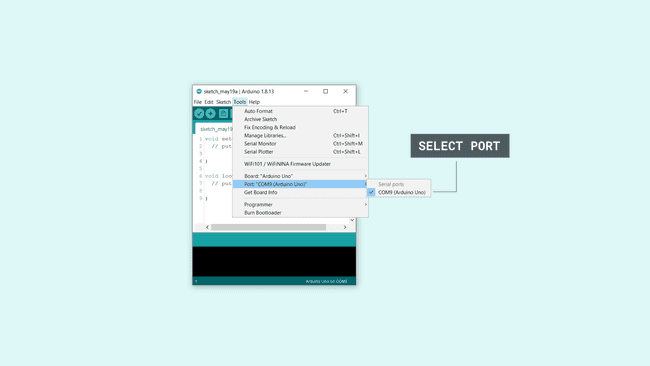
Now that you are all set up, **let’s try to make your board blink!**

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

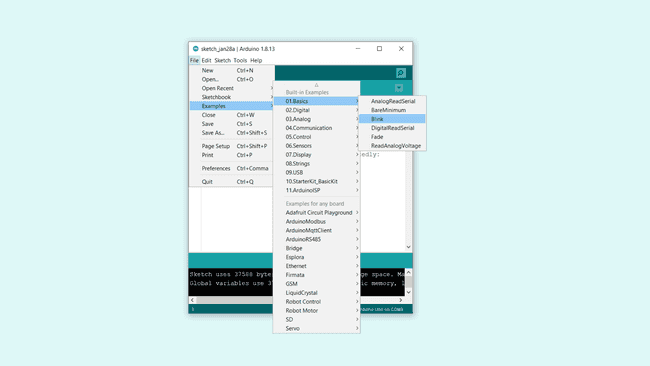
**6.** Now, you need to **select the right core & board**. This is done by navigating to **Tools > Board > Arduino AVR Boards > Board**. Make sure you select the board that you are using. If you cannot find your board, you can add it from **Tools > Board > Boards Manager**.

[](https://docs.arduino.cc/static/72ed26100bd60c7c0bb07779d3c6a96d/29114/install_avr_02.png)Selecting a board

**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.

[](https://docs.arduino.cc/static/6f0ce64a330509eeeebaaa899b81405a/29114/install_avr_03.png)Selecting the port

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

[](https://docs.arduino.cc/static/398ae037081cba436566abaf95d4e145/29114/install_avr_04.png)Opening an example

**9.** 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.

**10.** 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:

**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.

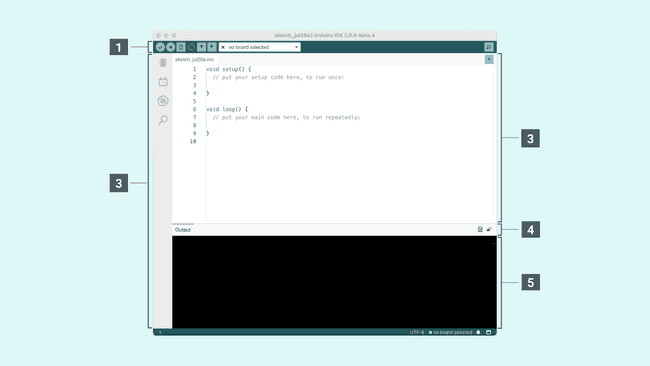
**2.** The **Sidebar** for regularly used tools. It gives you quick access to board managers, libraries, debugging your board as well as a search and replacement tool.

**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.

[](https://docs.arduino.cc/static/57a5ed239166aaa26f0dc02e450fe43a/29114/ide-20.png)The Arduino Software IDE

Now that you are all set up, **let’s try to make your board blink!**

**1.** **Connect your Arduino** or Genuino board to your computer.

**2.** Now, you need to **select the right board & port**. This is done from the toolbar. Make sure you select the board that you are using. If you cannot find your board, you can add it from the board manager in the sidebar.

[](https://docs.arduino.cc/static/984e20b7a0a79b5fc75d7bdf96f963d1/29114/select-board-port.png)Selecting a board & port

**3.** 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.

**CHAPTER 8**

**SOURCE CODE**

// DECLARATION

#include <LiquidCrystal.h>

#include <stdio.h>

LiquidCrystal lcd(6, 7, 5, 4, 3, 2);

#include <SoftwareSerial.h>

SoftwareSerial mySerial(8,9);

int led = A2;

int ldr2 = A1;

int ldr1 = A0;

const int trigPin = 12;

const int echoPin = 13;

int buzzer = 10;

// SETUP

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;

distancemm = (duration\*0.17);

ud = distancemm;

return ud;

}

//==== FUNCTION SETUP========

void setup()

{

char ret;

pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output

pinMode(echoPin, INPUT); // Sets the echoPin as an Input

pinMode(ldr1, INPUT); pinMode(ldr2, INPUT);

pinMode(relay, OUTPUT);pinMode(buzzer, OUTPUT);

digitalWrite(relay, LOW);digitalWrite(buzzer, HIGH);

Serial.begin(9600);

mySerial.begin(9600);

//Vehicle tracking using IOT

lcd.begin(16,2);

lcd.clear();

lcd.setCursor(0, 0);lcd.print(" IOT Smart ");

lcd.setCursor(0, 1);lcd.print(" Muncipality ");

delay(2500);

lcd.clear();lcd.print("Getting GPS...");

delay(15000);

lcd.clear();

lcd.setCursor(0,0);

lcd.write("Lat:");

for(ii=0;ii<=6;ii++) lcd.write(finallat[ii]);

lcd.setCursor(0,1);

lcd.write("Long:");

for(ii=0;ii<=7;ii++) lcd.write(finallong[ii]);

delay(1500);

wifiinit();

delay(2500);

lcd.clear();

lcd.print("U:"); //2,0

lcd.setCursor(8,0);

lcd.print("L:"); //10,0

}

========= LOGIC ============

void loop()

{

dista=0;

dist1=0;

dist2=0;

for(rtr1=0;rtr1<5;rtr1++)

{

dista = ultra\_dist();

dist1 = (dist1 + dista);

delay(10);

}

dist1 = (dist1/5);

lcd.setCursor(2,0);convertl(dist1);delay(10);

memset(drainage\_string,'\0',strlen(drainage\_string));

if(dist1 < 10)

{

beep();

strcpy(drainage\_string,"Drainage\_Full");

upload(finallat,finallong,dist1,drainage\_string,ldr\_string,fault\_string);

}

else

{

strcpy(drainage\_string,"Drainage\_Normal");

}

memset(ldr\_string,'\0',strlen(ldr\_string));

if(digitalRead(ldr1) == LOW)

{

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

strcpy(ldr\_string,"Light");

digitalWrite(relay, LOW);

lights='0';

}

if(digitalRead(ldr1) == HIGH)

{

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

strcpy(ldr\_string,"Dark");

digitalWrite(relay, HIGH);

lights='1';

}

memset(fault\_string,'\0',strlen(fault\_string));

if(digitalRead(ldr2) == HIGH && lights == '1')

{

lcd.setCursor(0,1);lcd.print("Light\_Fault");

strcpy(fault\_string,"Light\_Fault");

beep();

sts1++;

if(sts1 >= 2){sts1=2;}

if(sts1 == 1)

{

upload(finallat,finallong,dist1,drainage\_string,ldr\_string,fault\_string);

}

}

if(digitalRead(ldr2) == LOW && lights == '1')

{

lcd.setCursor(0,1);lcd.print("Light\_ON ");

strcpy(fault\_string,"Light\_ON");

sts1=0;

}

if(digitalRead(ldr2) == HIGH && lights == '0')

{

lcd.setCursor(0,1);lcd.print("Light\_OFF ");

strcpy(fault\_string,"Light\_OFF");

sts1=0;

}

if(digitalRead(ldr2) == LOW && lights == '0')

{

lcd.setCursor(0,1);lcd.print("Light\_OFF ");

strcpy(fault\_string,"Light\_OFF");

sts1=0;

}

}

//========== SERVER FUNCTIONS============

void upload(const char \*s1,const char \*s2,const char \*s3,const char \*s4)

{

delay(2000);

lcd.setCursor(15, 1);lcd.print("U");

myserialFlush();

mySerial.println("AT+CIPSTART=4,\"TCP\",\"projectsfactoryserver.in\",80");

sprintf(buff,"GET http://projectsfactoryserver.in/storedata.php?name=iot496&s1=%u&s2=%s&s3=%s\r\n\r\n",s1,s2,s3);

delay(8000);

}

char readserver(void)

{

char t;

delay(2000);

lcd.setCursor(15, 1);lcd.print("R");

myserialFlush();

mySerial.println("AT+CIPSTART=4,\"TCP\",\"projectsfactoryserver.in\",80");

//http://projectsfactoryserver.in/last.php?name=amvi001L

delay(8000);

memset(buff,0,strlen(buff));

sprintf(buff,"GET http://projectsfactoryserver.in/last.php?name=iot4L\r\n\r\n");

myserialFlush();

sprintf(bf2,"AT+CIPSEND=4,%u",strlen(buff));

mySerial.println(bf2);

delay(5000);

}

void clearserver(void)

{

delay(2000);

lcd.setCursor(15, 1);lcd.print("C");

myserialFlush();

mySerial.println("AT+CIPSTART=4,\"TCP\",\"projectsfactoryserver.in\",80");

//sprintf(buff,"GET http://projectsfactoryserver.in/storedata.php?name=iot1&s10=0\r\n\r\n");

delay(8000);

memset(buff,0,strlen(buff));

sprintf(buff,"GET http://projectsfactoryserver.in/storedata.php?name=iot4&s10=0\r\n\r\n");

}

//========== WIFI INITIALIZE FUNCTION==========

void wifiinit()

{

const char\* ssid = "iotserver";

const char\* password = "iotserver123";

char ret;

st:

mySerial.println("ATE0");

ret = check((char\*)"OK",50);

mySerial.println("AT");

ret = check((char\*)"OK",50);

if(ret != 0)

{

delay(1000);

goto st;

}

lcd.clear();lcd.setCursor(0, 0);lcd.print("CONNECTING");

mySerial.println("AT+CWMODE=1");

ret = check((char\*)"OK",50);

cagain:

myserialFlush();

mySerial.print("AT+CWJAP=\"");

mySerial.print(ssid);

mySerial.print("\",\"");

mySerial.print(password);

mySerial.println("\"");

if(check((char\*)"OK",300))goto cagain;

mySerial.println("AT+CIPMUX=1");

delay(1000);

lcd.clear();lcd.setCursor(0, 0);lcd.print("WIFI READY");

}

//========== GPS FUNCTION===============

void get\_gps()

{

lcd.clear();

lcd.print("Getting GPS Data");

lcd.setCursor(0,1);

lcd.print("Please Wait.....");

gps\_status=0;

int x=0;

while(gps\_status==0)

{

int str\_lenth=i;

coordinate2dec();

i=0;x=0;

str\_lenth=0;

}

}

void gps\_convert()

{

if(gps\_status)

{

// Serial.println(gpsString);

if(gpsString[0] == '$' && gpsString[1] == 'G' && gpsString[2] == 'P' && gpsString[3] == 'R' && gpsString[4] == 'M' && gpsString[5] == 'C')

{

// Serial.println("Don11111111111111111111111111111111111111111111111111111\r\n");

// Serial.write(gpsString[18]);Serial.write(gpsString[19]);Serial.write(gpsString[20]);Serial.write(gpsString[21]);Serial.write(gpsString[22]);

//lcd.setCursor(0,0);

for(ii=0;ii<9;ii++)

{

//lcd.write(gpsString[19+ii]);

msg1[ii] = gpsString[19+ii];

//Serial.write(msg1[ii]);

}

//Serial.println("\r\n");

//lcd.setCursor(0,1);

for(ii=0;ii<10;ii++)

{

//lcd.write(gpsString[32+ii]);

msg2[ii] = gpsString[32+ii];

// Serial.write(msg2[ii]);

}

convlat(lati); convlong(longi);

finallat[0] = msg1[0];

finallat[1] = msg1[1];

finallat[2] = '.';

finallat[3] = flat[0]; finallat[4] = flat[1];finallat[5] = flat[2];finallat[6] = flat[3];finallat[7] = '\0';

finallong[0] = msg2[0];

finallong[1] = msg2[1];

finallong[2] = msg2[2];

finallong[3] = '.';

finallong[4] = flong[0];finallong[5] = flong[1];finallong[6] = flong[2];finallong[7] = flong[3];finallong[8] = '\0';

}

}

}

======= Ultrasonic ==========

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;

distancemm = (duration\*0.17);

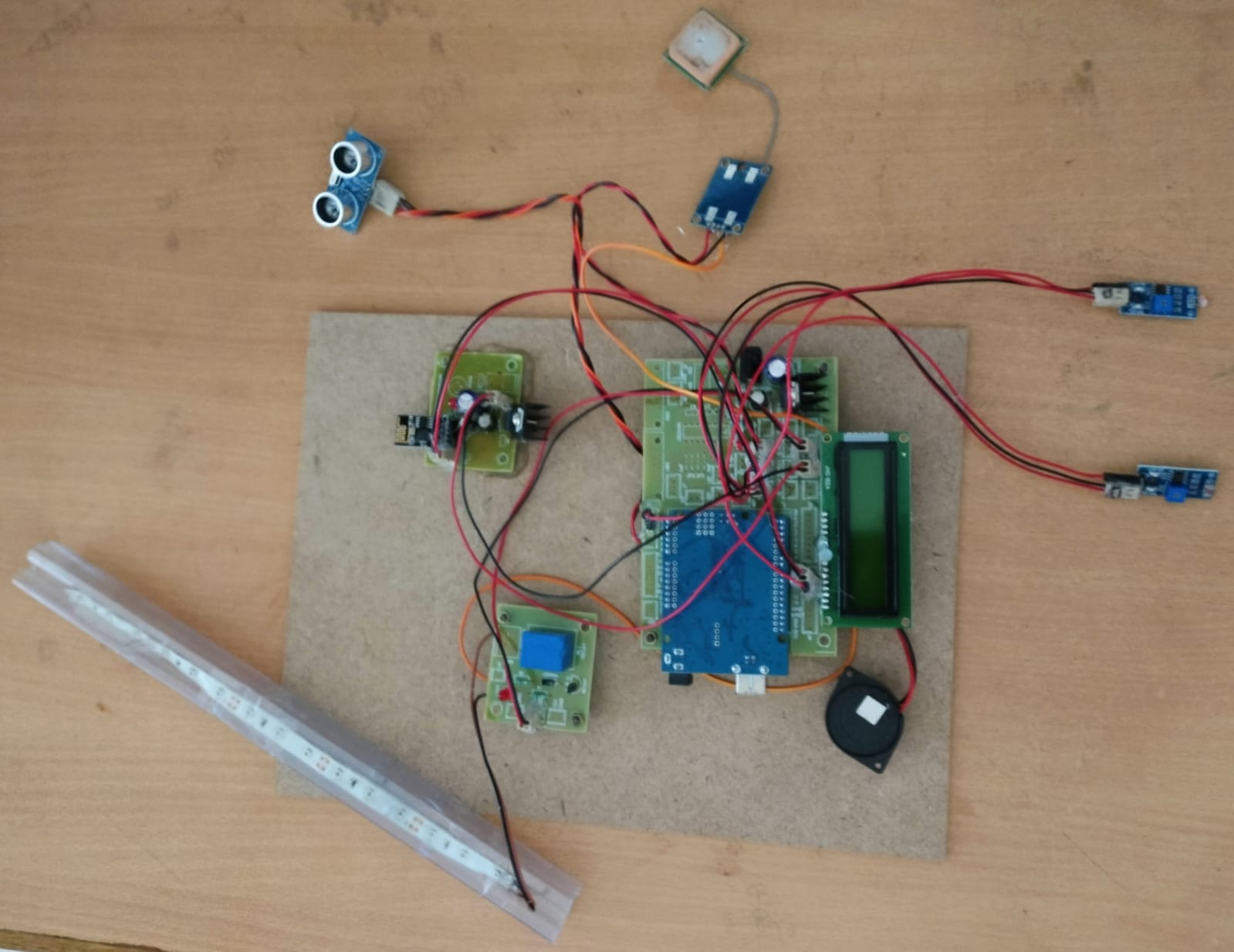
ud = distancemm;

return ud;

}

**CHAPTER 9**

**RESULTS**

****

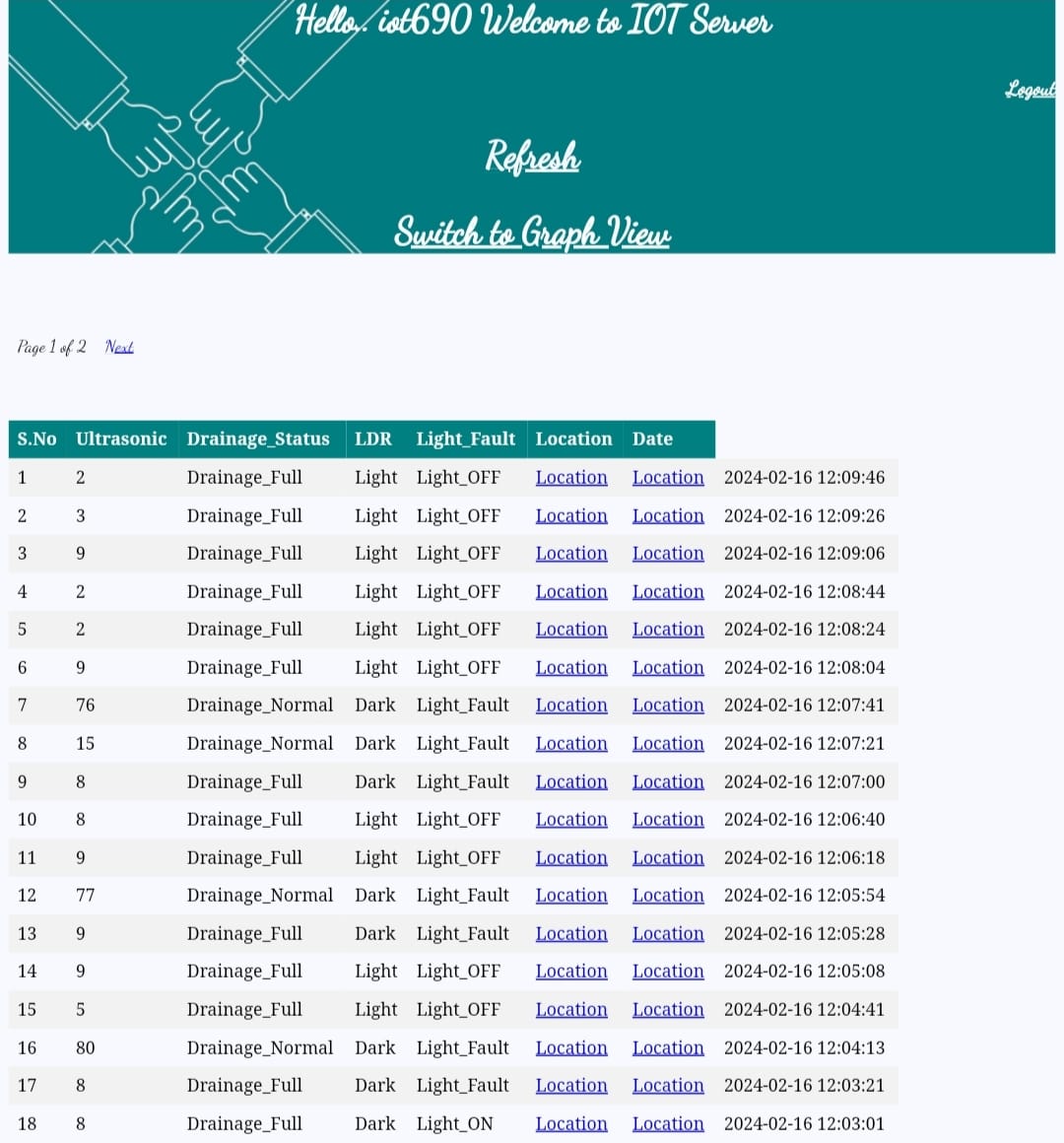
The above image shows the hardware equipment of the project. The kit is turned ON by giving the regulated power supply of 12v which is then converted to 5v dc current. The generated 5v dc current passes to every hardware component in the circuit.



When we turned on the regulated power supply, the LCD displayed the IOT Smart Muncipality. The output may be seen in the following image after we have connected to the IOT module.



Here LCD displays the light is in OFF as shown in above picture and Ultrasonic sensor level we can see in the above placed picture how the value is represented.



Here the image displays the output of the system is uploaded in the website by using ESP8266 IOT. The data of drainage status and street light fault is detected and the locations with date and time are reported to the municipalities.

**CHAPTER 10**

**CONCLUSION**

**CONCLUSION**

We Design and Implement the “IOT-MONHAOLE MONITORING & STREET LIGHT CONTROL SYSTEM WITH FAULT DETECTION AND REPORTING FOR MUNCIPALITIES” The main aim of the project is to benefitmunicipalities in terms of Smart infrastructure management, public safety, and resource optimization. In this project we are using the RPS, GPS, Ultrasonic sensor, LDR1, LDR2, LCD, Buzzer, Street light , IOT Module transmittin the data .And the data can be controlled by the Arduino Uno Micro Controller .By using the WIFI module connect the IOT Server. The data can display on the LCD display and at the same time IOT server. This system is simple tothe IoT Manhole Monitoring & Street Light Control System represents a transformative solution for municipalities, offering a scalable and cost-effective approach to managing critical infrastructure assets, enhancing public service delivery, and promoting the well-being of urban communities.

**REFERENCES**

[1] S Sultana, A Rahaman, AM Jhara, AC Paul “An IOT Based Smart Drain Monitoring System with Alert Messages”-Conference on Intelligent, 2020 Springer.

[2]A Pendharkar, J Chillapalli, K Dhakate, “IoT Based Sewage Monitoring System”, Available at SSRN, 2020, researchgate.net.

[3] U Andrijašević, J Kocić, V Nešić, “Lid Opening Detection in Manholes” 2020 - ieeexplore.ieee.org.

[4] P Bhosale, “Iot Based S[ystem for Detection of Sewage Blockages”, 2021, itin-industry.org.

[5] M Aarthi, A Bhuvaneshwaran, “Iot Based Drainage and Waste Management Monitoring and Alert System for Smart City”- Annals of the Romanian Society, 2021 - annalsofrscb.ro.

[6] Prof S. A. Shamikh1, Suvarna A. Sonawane2, ”Monitoring Smart City Application Using Raspberry PI based on IoT '' International Journal of Innovative Science, Engineering & Technology, Vol 5 Issue VIL, July 2017.

[7] Prof Muragesh SK1, Santhosha Rao2, “Automated Internet of Things For Underground Drainage and Manhole Monitoring Systems For Metropolitan Cities.” International Journal of Innovative Science, Engineering & Technology, Vol. 2 Issue 4, June 2015.

[8] Lazarescu, M.T., "Design of a WSN Platform for Long-Term Environmental Monitoring for IoT Applications," Emerging and Selected Topics in Circuits And Systems, IEEE Journal on, vol.3, no.1, pp.45, 54, March 2013.

[9] Design and Implementation of Automatic Street Light Controller for Energy Optimization Using FPGA, Mr. Amey J.Manekar and Dr. R.V. Kshirsagar, International Journal of Advanced Research in Computer and Communication Engineering, Vol. 5, Issue 6, June 2016.

[10] "GSM Based Autonomous Street Illumination System for Efficient Power Management," International Journal of Engineering Trends and TechnologyVolume4Issue1- 2013, Chaitanya Amin, AshutoshNerkar, Paridhi Holani, and Rahul Kaul.

[11] "Street Light Monitoring and Control System," International Journal of Engineering and Techniques, Volume 1 Issue 2, Mar-Apr 2015. Abdul Latif Saleem, Raja Sagar R, Sachin Datta N S, Sachin H S, and Usha M S.

[12] "Intelligent Street Lighting System Using GSM" by K.Y. Rajput, Gargeyee Khatav, Monica Pujari, and Priyanka Yadav March 2013 issue of the International Journal of Engineering Science Invention.

[13] V. Sumathi, A. Krishna Sandeep, and B. Tarun Kumar [5] International Journal of Engineering and Technology, Vol. 5, No. 5, October–November 2013, "Arm Based Street Lighting System with Fault Detection."

[14] 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).

[15] 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.

[16] 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.

[17] 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.

[18] 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.

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

[20] Salvi, Ritisha, et al. "Smart street light using arduino uno microcontroller." Int. J. Innov. Res. Comput. Commun. Eng 5 (2017): 5203-5206.

[21] 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.

[22] A. Sarma, G. Verma, S. Banarwal, H. Verma, "Street light power reduction system using microcontroller and solar panel", 2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom), pp. 2008-2010, 2016.

[23] Y. M. Yussoff, M. Samad, "Sensor node development for street lighting monitoring system", 2016 IEEE Symposium on Computer Applications & Industrial Electronics (ISCAIE), pp. 26-29, 2016.

[24] S. R. Parekar, M. M. Dongre, "An intelligent system for monitoring and controlling of street light using GSM technology", 2015 International Conference on Information Processing (ICIP), pp. 604-609, 2015.

[25] M. Karthikeyan, V. Saravanan, S. Vijayakumar, "Cloud based automatic street light monitoring system", 2014 International Conference on Green Computing Communication and Electrical Engineering (ICGCCEE), pp. 1-6, 2014.

[26] S. Deo, S. Prakash, A. Patil, "Zigbee-based intelligent street lighting system", 2014 2nd International Conference on Devices Circuits and Systems (ICDCS), pp. 1-4, 2014.