POWER SAVING SYSTEM FOR ELECTRICAL VEHICLES



**A Project Report**

*Submitted to the FACULTY of ENGINEERING of*

*JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY,KAKINADA*

*In partial fulfillment of the requirements for award of the Degree of*

***Bachelor of Technology***

in

***Electrical and Electronics Engineering***

By

S.RAKESH BABU RABIYA BASARIYA

(21481A0281) (21481A0275)

P.RUSHIKESH R.BHASKAR RAO

(21481A0263) (21481A0276)

Under the Guidance of

### CH.SUJATHA

Assistant Professor

**Department of Electrical and Electronics Engineering**

## SESHADRI RAO GUDLAVALLERU ENGINEERING COLLEGE

(An Autonomous Institute with Permanent Affiliation to JNTUK, Kakinada)

### SESHADRI RAO KNOWLEDGE VILLAGE

GUDLAVALLERU-521356 ANDHRA PRADESH **2025**

**SESHADRI RAO GUDLAVALLERU ENGINEERING COLLEGE**

(An Autonomous Institute with Permanent Affiliation to JNTUK, Kakinada)

### SESHADRI RAO KNOWLEDGE VILLEGE

GUDLAVALLERU-521356

**DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING**

****

CERTIFICATE

This is to certify that the project report entitled “**POWER SAVING SYSTEM FOR ELECTRICAL VEHICLES”**is a bonafide record of work carried out by **Mr. S.Rakesh Babu (21481A0281),Ms.Rabiya Basariya (21481A0275), Mr. P.Rushikesh (21481A0263)** and **Mr. R.Bhaskar Rao (21481A0276)** under my guidance and supervision in partial fulfillment of the requirements for award of the degree of Bachelor of Technology in ***Electrical And Electronics Engineering*** *of* ***Jawaharlal Nehru Technological University, Kakinada.***

##### Ch.Sujatha Dr.A. Amarendra(B. E ,M.Tech,PhD)

PROJECT GUIDE PROFESSOR & HOD

## ACKNOWLEDGEMENT

We are glad to express our deep sense of gratitude to **Ch.Sujatha, Assistant Professor**, Electrical and Electronics Engineering for her guidance and cooperation in completing this project. Through this we want to convey our sincere thanks to her for inspiring assistance during our project.

We express our heartfelt gratitude and deep indebtedness to our beloved Head of the Department

**Dr. A. Amarendra,** for his great help and encouragement in doing our project successfully.

We also express our gratitude to our principal **Dr. B. Karun Kumar**, for his encouragement and facilities provided during the course of project.

We express our heartfelt gratitude to our **Faculty members** and **Lab Technician**s for providing a great support for us in completing our project.

We thank one and all who have rendered help to us directly or indirectly in the completion of this work.

**Mr. S.Rakesh Babu (21481A0281),**

**Ms. Rabiya Basirya (21481A0275),**

**Mr.P.Rushikesh (21481A0263),**

**Mr. R.Bhaskar Rao (21481A0276).**

## CONTENTS

#### Page No.

**List of Figures i**

[List of Tables ii](#_bookmark0)

[Nomenclature iii](#_TOC_250008)

[Abstract iv](#_TOC_250007)

Chapter1:Introduction 6-8

* 1. Overview of Renewable Energy Systems 6
  2. Power Quality Challenges in DG Systems 6-7
  3. Objective of the Work 7
  4. Organization of the Thesis 8

[Chapter2: Non Conventional Energy Sources 9-2](#_bookmark1)5

* 1. [Introduction 9](#_TOC_250006)
  2. Schematic Diagram of the Proposed System 9
  3. [Solar Photovoltaics 10-19](#_TOC_250005)
  4. [Wind System 19-23](#_TOC_250004)
  5. Alternatives and Trends in Wind Systems 23-25

[Chapter3: Power Quality 26-35](#_TOC_250003)

* 1. [Classification of Power Quality Problems 26-33](#_TOC_250002)
  2. [Causes and Effects 33](#_TOC_250001)
  3. [Mitigation Techniques 33-35](#_TOC_250000)

|  |  |
| --- | --- |
| **Chapter4: Facts Devices** | **36-55** |
| 4.1 Overview of FACTS | 36-38 |
| 4.2 SVC and STATCOM | 38-48 |
| 4.3 TCSC and Series Devices | 49-54 |
| 4.4 Unified Power Flow Controller (UPFC) | 54-55 |
| **Chapter5: Unified Power Quality Conditioner (UPQC)** | **56-59** |
| 5.1 Introduction to UPQC | 56 |
| 5.2 Control Objectives of UPQC | 56-57 |
| 5.3 Operating Principle | 57-59 |
| **Chapter6: ANN based UPQC System** | **60-69** |
| 6.1 Artificial Neural Network Basics | 60-68 |
| 6.2 Implementation of ANN in UPQC | 68-69 |
| **Chapter7: Simulation results** | **70-71** |
| 7.1 Simulation Setup in MATLAB | 70 |
| 7.2 Output Waveforms | 71 |
| **Chapter 8: Conclusion** | **72** |
| **References** | **73-74** |
| **PROJECT PROFORMANCE:** | **75-80** |

## LIST OF FIGURES

|  |  |
| --- | --- |
| **Fig. No.** | **Figure Name** |
| Fig 2.1 | Schematic diagram of proposed system |
| Fig 2.3 | I-V Characteristics of solar cell |
| Fig 2.4 | Effect of irradiance on I-V and P-V curves |
| Fig 2.5 | Effect of Temperature on solar panel output |
| Fig 2.7 | Equivalent circuit of solar cell |
| Fig 2.9 | Power Coefficient curve |
| Fig 2.10 | Wind Turbine Power Curve |
| Fig 2.11 | Aerofoil Stall Effect |
| Fig 2.1**2** | Power density vs Wind Curves |
| Fig 4.1 to 4.15 | Facts devices Configuration |
| Fig 14.15 | UPQC configuration |
| Fig 14.16 | Equivalent circuit of UPQC |

## LIST OF TABLES

**Table Title Page No.**

**2.1: Key Specifications of MSX-60 PV Panel 12**

## NOMENCLATURE

ABBREVIATION DEFINITION

UPQC. Unified Power Quality Conditioner

ANN. Artificial Neural Network

DG. Distributed Generation

PV. Photovoltaic

MPPT. Maximum Power Point Tracking

VSC. Voltage Source Converter

THD. Total Harmonic Distortion

PI. Proportional Integral Controller

MPP. Maximum Power Point

SVC. Static VAR Compensator

STATCOM. Static Synchronous Compensator

TCSC. Thyristor Controlled Series Capacitor

DVR. Dynamic Voltage Restorer

PCC. Point of Common Coupling

## ABSTRACT

Electric vehicles (EVs) represent a sustainable alternative to fossil-fueled transportation, yet their widespread adoption is hindered by limitations in battery lifespan, charging infrastructure, and user concerns over range anxiety. Current EV systems lack intelligent power optimization, resulting in inefficient energy usage during critical low-battery conditions. This project proposes a Power Saving System that dynamically optimizes energy consumption in EVs using an ATmega328P microcontroller. The system automatically disables non-essential components—such as lights, fans, and infotainment systems—and regulates motor performance through an L293D motor driver to extend driving range. Additionally, it incorporates a manual activation feature to enable power-saving mode preemptively. By enhancing energy utilization and system reliability, this design contributes to more efficient EV operation and promotes broader adoption of electric mobility solutions.

### I. INTRODUCTION

Street Electric vehicles (EVs) are rapidly gaining traction as a sustainable alternative to internal combustion engine vehicles, offering zero emissions, reduced environmental impact, and lower operating costs. However, despite these benefits, EV adoption remains limited due to several practical concerns. Among the most prominent is the issue of battery limitations—both in terms of driving range and energy efficiency. As EV users often experience range anxiety, especially in areas with scarce charging infrastructure, there is an urgent need to address how power is managed within these vehicles.

Traditional EV systems focus heavily on providing warnings when the battery level becomes critically low but do not implement any measures to optimize power usage in response. This results in continued operation of non-essential systems like air conditioning, infotainment, and lighting, even when energy reserves are depleting. Consequently, energy that could otherwise extend the vehicle’s driving range is wasted, and the risk of the vehicle halting before reaching a charging station increases significantly.

To address this issue, the proposed project introduces a power-saving system that automatically manages energy usage based on battery status. By integrating a microcontroller-based control unit, the system intelligently identifies critical battery conditions and initiates energy conservation measures. These include the automatic shutdown of non-vital systems and adjustments in motor operation to reduce power draw. Furthermore, the system incorporates a manual override feature, allowing drivers to proactively engage power-saving mode even before the battery reaches low levels.

The core of the system revolves around the ATmega328P microcontroller, commonly used in Arduino platforms, due to its cost-effectiveness and flexibility. It controls various components such as relays, fans, LEDs, and a DC motor (representing vehicle propulsion) through interfaces like the L293D motor driver. This setup allows for real-time monitoring and dynamic control of energy-consuming components, ensuring that the vehicle operates within optimal power parameters under different conditions.

Additionally, the proposed design provides valuable visual and audible feedback to the driver using an LCD display and buzzer, respectively. These indicators convey battery status, system warnings, and active power-saving mode status. By enhancing user awareness and allowing both automatic and manual control of energy consumption, the system not only improves vehicle efficiency but also empowers users with greater control over their EV's performance.

In summary, this project aims to overcome one of the major barriers in EV adoption by introducing a practical, affordable, and adaptable solution for battery power management. By optimizing energy usage and extending vehicle range during critical situations, the system enhances the reliability of electric vehicles and contributes to the global effort toward sustainable and efficient transportation systems.

2.1 Background and Motivation

Electric vehicles (EVs) have emerged as a promising solution to the growing concerns over environmental pollution, climate change, and the depletion of fossil fuels. With the ability to reduce carbon emissions and promote cleaner transportation, EVs are positioned at the forefront of sustainable mobility. Governments and environmental agencies around the world are encouraging their adoption through subsidies, policy changes, and public awareness campaigns. Despite this global push, EVs are still not as widely adopted as conventional vehicles, primarily due to limitations related to battery life, charging infrastructure, and user trust in long-distance capabilities.

One of the key challenges faced by EV users is range anxiety—the fear of running out of battery before reaching a charging station. This fear is not entirely unfounded, as the current generation of EVs has a limited range per charge and requires considerable time to recharge. Moreover, most EVs lack intelligent systems that optimize battery usage in real-time. Instead, they rely on basic alerts that notify users of a low battery without actively managing or conserving power. As a result, energy is often consumed inefficiently, especially by non-essential components like air conditioning, infotainment, lighting systems, and other auxiliary features.

This inefficiency in energy use becomes even more problematic in regions with sparse charging facilities or during long-distance travel. In such cases, every unit of stored energy becomes valuable, and the need for effective power-saving mechanisms becomes apparent. Recognizing this gap, the proposed project is motivated by the necessity to design and implement a system that can actively monitor and manage energy consumption, especially during critical battery levels.

The idea is to introduce a microcontroller-based power-saving mechanism that automatically disables or limits the use of non-vital systems when the battery level is low. This intelligent control ensures that essential vehicle operations are prioritized, and available energy is utilized more judiciously. Furthermore, the inclusion of a manual override feature allows users to activate the power-saving mode even in non-critical conditions—demonstrating both flexibility and user autonomy.

Our motivation also stems from the goal of making electric vehicles more accessible and user-friendly. By offering a system that is affordable, easy to implement, and compatible with various EV models, we aim to bridge the gap between cutting-edge research in battery management systems and practical, real-world solutions. The simplicity of components like the ATmega328P microcontroller and L293D motor driver ensures that even low-cost EVs or educational prototypes can benefit from the system.

In essence, the background of growing EV usage and the motivation to enhance its reliability converge in this project. Through intelligent energy management, we hope to make electric vehicles not just greener, but also smarter and more dependable for everyday users.

2.2 Problem Statement

Electric vehicles (EVs) are an environmentally friendly alternative to traditional fuel-powered vehicles; however, their adoption is significantly hindered by limitations in battery performance and energy management. One of the most pressing concerns among EV users is the limited driving range, especially in areas lacking adequate charging infrastructure. Although most EVs are equipped with basic low battery alerts, they lack intelligent systems that can optimize energy consumption during critical power levels.

In current EV designs, non-essential components such as interior lighting, fans, infotainment systems, and other auxiliary loads continue to operate even when the battery is nearly depleted. This leads to inefficient energy usage and contributes to range anxiety—the fear of the vehicle running out of power before reaching a charging station. Existing Battery Management Systems (BMS) do not dynamically adapt to low-power conditions or prioritize energy-saving functions, thereby reducing overall vehicle efficiency and reliability.

There is a clear need for a smart, cost-effective, and easily implementable system that can manage and conserve battery power in real-time. Such a system should automatically disable non-essential components and regulate motor functions based on battery status while also offering manual control for user-initiated power-saving.

This project addresses this gap by proposing a microcontroller-based Power Saving System that improves energy utilization, extends EV driving range, and enhances user confidence, especially in critical battery conditions.

2.3 Objectives

The primary objective of this project is to design and implement a power-saving system for electric vehicles that optimizes battery usage and extends driving range, particularly during low-battery conditions. To achieve this, the project focuses on the following specific goals:

1. Utilize an ATmega328P microcontroller to monitor battery levels and automatically control the operation of non-essential electrical components in the vehicle.

2. Implement automatic disabling of auxiliary systems such as lighting, fans, and infotainment when the battery charge reaches a predefined threshold, conserving energy for essential functions like propulsion.

3. Incorporate a manual switch that allows users to activate the power-saving mode at any time, providing greater control over battery usage even before reaching critical levels.

4. Use an LCD display and buzzer to communicate system status, battery level, and power-saving mode activation to the user in real-time, enhancing driver awareness and interaction.

2.4 Methodology

The methodology for this project involves a systematic approach combining hardware integration, software development, and functional testing to build an intelligent battery power-saving system for electric vehicles (EVs). The system is centered around the ATmega328P microcontroller, which serves as the control unit for monitoring battery status and managing component behavior.

1. System Design and Planning

The project began with a review of existing battery management challenges in EVs and identifying components that consume energy unnecessarily during low-battery scenarios. A schematic diagram of the system architecture was developed, outlining connections between sensors, actuators, display modules, and the microcontroller. The system was designed to detect critical battery levels and respond by disabling non-essential systems such as fans, lights, and infotainment devices.

2. Hardware Selection and Assembly

Essential hardware components were selected based on functionality, cost, and compatibility. The ATmega328P microcontroller (Arduino Uno) acts as the central processor. The L293D motor driver was used to control a 12V DC motor, simulating the vehicle’s propulsion. Other components include a 16x2 LCD display for user feedback, a relay module to switch off high-power loads, a DC fan to simulate HVAC systems, a programmable LED matrix for adaptive lighting control, and a manual activation switch. All components were interconnected according to the designed circuit.

3. Power Management Logic Development

Using the Arduino IDE, embedded C code was written to implement the logic for battery monitoring and power-saving operations. A voltage divider circuit was employed to measure battery voltage through the analog input of the microcontroller. When the voltage drops below a critical threshold, the system initiates energy-saving procedures, including switching off auxiliary components and regulating motor speed using PWM signals.

4. Manual Activation Feature

A digital input switch was configured to allow users to manually activate power-saving mode, regardless of the battery level. This provides users with control over energy usage during extended journeys or when approaching areas with limited charging facilities.

5. User Interface and Alerts

An LCD module was programmed to display real-time battery voltage, system status (normal or power-saving mode), and warnings. A buzzer provides audible alerts when power-saving mode is activated or when the battery level becomes critically low, ensuring the user remains informed.

6. Testing and Validation

The complete system was tested in a simulated EV environment. Multiple scenarios were evaluated, such as normal operation, critical battery condition, and manual activation. The response of each component and the accuracy of the voltage sensing were verified. The system was validated for its ability to reduce power consumption and maintain essential functionality under low battery conditions.

2.5 Expected Benefits

• By automatically disabling non-essential systems when battery levels are critically low, the proposed system helps conserve power, allowing the vehicle to travel longer distances on a single charge. This is particularly useful in emergency situations or areas with limited access to charging infrastructure.

• Intelligent energy management and user feedback features reduce the stress and uncertainty associated with battery depletion. Drivers gain more confidence in managing their battery levels, knowing that the system will help preserve power when needed.

• The system ensures that energy is prioritized for essential operations like propulsion, while unnecessary consumption is minimized. This leads to more efficient use of stored energy and improved overall performance of the electric vehicle.

• With the inclusion of a manual activation switch, users are given the flexibility to initiate power-saving mode at their discretion. This empowers drivers to proactively manage energy consumption based on travel distance, terrain, or personal preferences.

• The use of low-cost, widely available components like the ATmega328P microcontroller and L293D motor driver makes this solution economically viable. It can be integrated into both new EVs and retrofitted into existing models with minimal cost and effort.

• By enhancing the reliability and practicality of electric vehicles, this system promotes greater adoption of clean transportation technologies. It supports environmental sustainability efforts by encouraging the shift from fuel-based to electric vehicles.

• Intelligent power-saving features reduce unnecessary battery drain, potentially increasing the lifespan of the battery and reducing the likelihood of vehicle breakdowns due to power loss.

2.6 Literature Survey

"A Comprehensive Scheme for Power Management of FC/SC/Battery Hybrid Systems" (2024)

Summary: This study proposes an innovative energy management system that integrates fuel cells (FC) and photovoltaic (PV) panels as primary power sources, aiming to enhance the efficiency and sustainability of EV power systems.

"Energy and Battery Management Systems for Electrical Vehicles: A Comprehensive Review" (2025)

Summary: This review article delves into the advancements in battery management systems (BMS) for EVs, analyzing previous technical developments and offering recommendations for future improvements to optimize battery performance and longevity.

"Some Critical Thinking on Electric Vehicle Battery Reliability: From Enhancement to Optimization" (2024)

Summary: This research offers comprehensive insights into the optimization of EV battery systems by integrating the reliability ecosystem with a dynamic system approach, presenting a novel lifecycle framework for enhancing battery reliability.

"A Novel Digital Control Scheme for Power Management in a Hybrid Electric Vehicle" (2023)

Summary: This study presents a novel digital control scheme specific to the integration of solar, battery, and fuel cell systems, addressing power management issues in hybrid electric vehicles to improve efficiency and performance.

These studies provide valuable insights into the latest advancements in EV energy management and battery optimization, contributing to the development of more efficient and reliable electric vehicles.

# CHAPTER-2

# EXISTING SYSTEM AND PROPOSED SYSTEM

### 2.1 EXISTING SYSTEM

Conventional electric vehicles (EVs) are equipped with basic battery monitoring systems that primarily focus on displaying the state of charge and issuing warnings when the battery level drops below a critical threshold. These systems are effective in alerting drivers about low battery conditions but offer little to no functionality in terms of actively managing power consumption. As a result, drivers are informed about battery depletion but are not assisted in conserving the remaining power, which limits the practical effectiveness of these alerts in critical situations.

A major shortcoming of the current EV architecture is the continuous operation of non-essential systems, even when battery charge is critically low. Devices such as interior and exterior lighting, air conditioning fans, infotainment systems, and other auxiliary components remain active unless manually turned off by the user. These systems, although not crucial for vehicle propulsion, consume a significant portion of the battery’s energy. This unnecessary drain on the battery not only reduces the potential driving range but also increases the risk of the vehicle stopping before reaching a charging station.

Existing battery management systems (BMS) lack the dynamic adaptability required to respond to real-time battery conditions. They are often designed with fixed thresholds and standard control logic, which do not account for the varying power needs of different components under different conditions. Without intelligent decision-making capabilities or automated control over energy distribution, these systems are unable to prioritize essential functions or cut off non-vital loads during emergencies. This static approach leads to inefficient energy usage and contributes to range anxiety among EV users.

### 2.2 PROPOSED SYSTEM

### The proposed Power Saving System for Electric Vehicles is designed to optimize battery usage and extend driving range when power levels are critically low. Using an ATmega328P microcontroller, the system intelligently manages energy consumption by automatically disabling non-essential components such as lights, fans, and infotainment systems while regulating motor speed through an L293D motor driver. Additionally, a manual activation switch allows users to enable power-saving mode even when the battery is not critically low, further optimizing efficiency. This system enhances energy utilization, reduces range anxiety, and improves the overall reliability of electric vehicles, making them more practical for longer journeys.

# CHAPTER 3

# BLOCK DIAGRAM AND WORKING

### BLOCK DIAGRAM

### 

### Fig 3.1 Block Diagram

### 

### System Workflow

### The workflow of the proposed Battery Power Saving System involves a step-by-step process where the microcontroller continuously monitors the battery status and dynamically manages the power distribution based on real-time data. The system operates through both automatic and manual control mechanisms to optimize energy usage in electric vehicles.

### Initialization and Monitoring

### When the system is powered on, the ATmega328P microcontroller initializes all components and starts monitoring the battery voltage using a voltage divider circuit connected to an analog input pin. It also checks the status of the manual activation switch.

### Battery Level Evaluation

### The microcontroller compares the real-time battery voltage against a predefined critical threshold value. If the voltage remains above this threshold, the system operates normally, allowing all components (motor, fan, lights, infotainment, etc.) to function without restrictions.

### Automatic Power-Saving Activation

### When the battery level falls below the critical limit, the microcontroller automatically activates power-saving mode. This triggers the relay module to disconnect non-essential components like the DC fan, lights, and infotainment system. Simultaneously, motor speed is reduced by adjusting the PWM signal sent to the L293D motor driver, thereby conserving additional energy.

### Manual Override Option

### At any point, the user can press the manual activation switch to engage power-saving mode, even if the battery is not critically low. This provides the driver with control over when to conserve energy, such as during long trips or when nearing a charging station.

### User Alerts and Feedback

### Throughout the operation, an LCD display shows real-time battery voltage, system status (normal or power-saving mode), and any warnings. A buzzer provides audible alerts during mode transitions or when the battery reaches critical levels, keeping the driver informed and engaged.

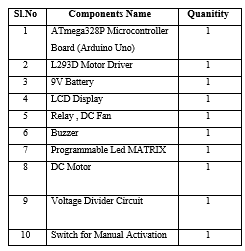
### Return to Normal Mode

### If the battery voltage returns to a safe level or the user deactivates manual power-saving mode, the system restores full operation. The relay reconnects non-essential components, and the motor returns to normal speed, ensuring a seamless transition between modes.

**CHAPTER-4**

**HARDWARE COMPONENTS AND DESCRIPTION**

**4.1 HARDWARE COMPONENTS**

****

**4.1 HARDWARE COMPONENTS**

**DESCRIPTION :**

**4.1 Arduino UNO**

The Arduino Uno is a microcontroller board based on the ATmega328 (datasheet). It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started. The Uno differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it features the Atmega8U2 programmed as a USB-to-serial converter. "Uno" means one in Italian and is named to mark the upcoming release of Arduino 1.0. The Uno and version 1.0 will be the reference versions of Arduno, moving forward. The Uno is the latest in a series of USB Arduino boards, and the reference model for the Arduino platform; for a comparison with previous versions, see the index of Arduino boards

The **ATmega328** is one kind of single-chip microcontroller formed with Atmel within the **megaAVR family**. The architecture of this Arduino Uno is a customized Harvard architecture with 8 bit [**RISC processor**](https://www.elprocus.com/difference-between-risc-and-cisc-architecture/) core. **Uno** include Arduino Pro Mini, Arduino Nano, Arduino Due, Arduino Mega, and Arduino Leonardo.

A blue circuit board with black and grey objects

AI-generated content may be incorrect.

***Fig 4.1 Ardunio uno***

**4.1.1 Features of Arduino Uno Board**

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

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

***4.1.2 Arduino Uno Pin Diagram***

The Arduino Uno board can be built with power pins, analog pins, ATmegs328, ICSP header, Reset button, [power LED](https://www.elprocus.com/solar-powered-led-street-light-control-circuit/), digital pins, test led 13, TX/RX pins, USB interface, an external [power supply](https://www.elprocus.com/difference-between-single-phase-and-three-phase-ac-power-supply/). The **Arduino UNO board description** is discussed below.

A blue circuit board with text

AI-generated content may be incorrect.

***Fig 4.1.2: Arduino Uno Board***

***4.1.3Power Supply***

The **Arduino Uno power supply** can be done with the help of a USB cable or an external power supply. The external power supplies mainly include AC to DC adapter otherwise a battery. The adapter can be connected to the Arduino Uno by plugging into the power jack of the Arduino board. Similarly, [**the battery**](https://www.elprocus.com/an-overview-of-bio-battery-working-principle-types-applications/) leads can be connected to the Vin pin and the GND pin of the POWER connector. The suggested voltage range will be 7 volts to 12 volts.

***6.1.4 Input & Output***

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

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

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

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

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

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

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

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

***6.1.5 Memory***

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

***6.1.6Communication***

The Arduino Uno ATmega328 offers UART TTL-[**serial communication**](https://www.elprocus.com/i2c-bus-protocol-tutorial-interface-applications/), and it is accessible on digital pins like TX (1) and RX (0). The software of an Arduino has a serial monitor that permits easy data. There are two LEDs on the board like RX & TX which will blink whenever data is being broadcasted through the USB.

A SoftwareSerial library permits for serial communication on Arduino Uno digital pins and the ATmega328P supports TWI (I2C) as well as [**SPI-communication**](https://www.elprocus.com/serial-peripheral-interface-spi-communication-protocol/). The Arduino software contains a wired library for simplifying the utilization of the I2C bus.

*4.1.5. PIN DESCRIPTION*

|  |  |  |
| --- | --- | --- |
| **Pin Category** | **Pin Name** | **Details** |
| Power | Vin, 3.3V, 5V, GND | Vin: Input voltage to Arduino when using an external power source.  5V: Regulated power supply used to power microcontroller and other components on the board.  3.3V: 3.3V supply generated by on-board voltage regulator. Maximum current draw is 50mA.  GND: ground pins. |
| Reset | Reset | Resets the microcontroller. |
| Analog Pins | A0 – A5 | Used to provide analog input in the range of 0-5V |
| Input/Output Pins | Digital Pins 0 - 13 | Can be used as input or output pins. |
| Serial | 0(Rx), 1(Tx) | Used to receive and transmit TTL serial data. |
| External Interrupts | 2, 3 | To trigger an interrupt. |
| PWM | 3, 5, 6, 9, 11 | Provides 8-bit PWM output. |
| SPI | 10 (SS), 11 (MOSI), 12 (MISO) and 13 (SCK) | Used for SPI communication. |
| Inbuilt LED | 13 | To turn on the inbuilt LED. |
| TWI | A4 (SDA), A5 (SCA) | Used for TWI communication. |
| AREF | AREF | To provide reference voltage for input voltage. |

*Tab 4..1*

***4.1.7 HOW TO USE ARDUINO BOARD***

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

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

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

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

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

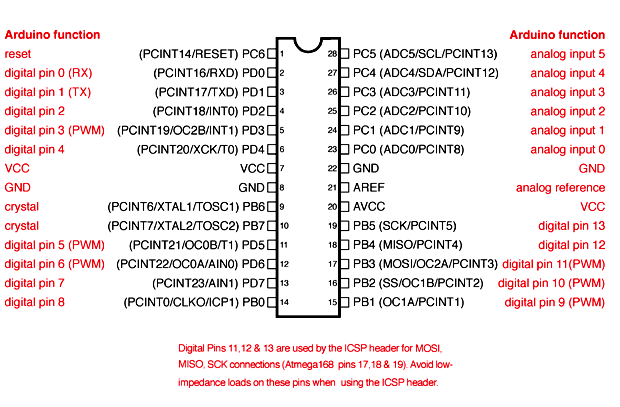
* **AREF:** Used to provide reference voltage for analog inputs with analogReference() function.
* **Reset Pin:** Making this pin LOW, resets the microcontroller.

*4.1.7 COMMUNICATION*

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

*4.1.8 Arduino Uno to ATmega328 Pin Mapping*

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



***Fig 4.1.3 : AT mega 328 pin***

*4.1.8.1 Applications*

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

Projects requiring Multiple I/O interfaces and communications

**4.2 L293D Motor Driver**

A motor driver is an essential component in controlling the movement of motors within electronic systems, especially in projects involving robotics, electric vehicles, and automation. It acts as an interface between the low-power control signals (from microcontrollers or other logic circuits) and the high-power requirements of motors. The motor driver amplifies the control signal, allowing it to manage larger currents needed to drive motors effectively. Without a motor driver, the microcontroller would not be able to supply the necessary power for motor operation, as microcontrollers typically output low current (milliampere range), while motors require much higher currents (amperes) to function.

One commonly used motor driver is the L293D, which is designed to drive bipolar stepper motors and DC motors. It contains built-in diodes to protect against voltage spikes caused by inductive loads when the motor is switched off, ensuring the longevity of the circuit. The L293D allows for bidirectional control of the motor, meaning it can rotate in both clockwise and counterclockwise directions. This flexibility is crucial in various applications, such as vehicle movement or robotic actuators, where precise motor control is necessary.

In an electric vehicle system, the motor driver, like the L293D, can be used to regulate the speed and direction of the motor, based on the signals received from the microcontroller. It allows for fine control over the vehicle's propulsion system, enabling energy-efficient operation. The driver can adjust motor speed by modulating the pulse width of the control signals (Pulse Width Modulation or PWM), optimizing energy usage based on the vehicle's speed and battery charge level. By using such a system, the vehicle can conserve power by efficiently controlling motor speed and ensuring that power is only used when necessary.

**A red and blue circuit board

AI-generated content may be incorrect.**

**4.3Motor**

An A motor is a device that converts electrical energy into mechanical motion. Motors are integral components in many applications, from simple household appliances to advanced robotic systems and electric vehicles. The most common types of motors used in these systems are DC (Direct Current) motors, AC (Alternating Current) motors, and stepper motors. Motors operate based on the principle of electromagnetism, where a current passing through a coil of wire generates a magnetic field that interacts with the permanent magnets or another magnetic field to create rotational motion.

DC motors are popular for their simplicity and ease of control. They have two main parts: the stator (which produces a magnetic field) and the rotor (which rotates). In a DC motor, a voltage is applied to the rotor windings, creating a magnetic field that interacts with the stator's field, causing the rotor to spin. The direction of the motor’s rotation can be easily controlled by reversing the polarity of the voltage. DC motors are widely used in applications where variable speed and control are essential, such as in electric vehicles, robotics, and conveyor belts.

In electric vehicles, the motor is the heart of the propulsion system, responsible for converting stored electrical energy from the battery into mechanical energy that moves the vehicle. The efficiency of the motor directly affects the vehicle's range, acceleration, and overall performance. Electric vehicle motors often use high-efficiency, low-loss technologies, such as brushless DC motors (BLDC), which offer better performance and longer lifespans compared to traditional brushed motors. By optimizing motor design and control strategies (such as using motor drivers and pulse-width modulation), the energy consumption can be better managed, improving the driving range and reducing energy waste.

A small electric motor with a metal rod

AI-generated content may be incorrect.

**Fig 4.3 Motor.**

* 1. **LED matrix**

The WS281B is a popular type of addressable RGB LED, often used in programmable LED matrices, strips, and displays. These LEDs are highly versatile because each individual LED can be independently controlled, allowing for a wide range of color displays and effects. The WS281B LED uses a 3-wire protocol (data, power, and ground), making it relatively easy to interface with microcontrollers like Arduino, Raspberry Pi, or ESP32. The data pin transmits control information to each LED, which then adjusts the color and brightness according to the input signal.

One of the main advantages of WS281B LEDs is their ability to form large, customizable arrays or matrices. By chaining multiple WS281B LEDs together, you can create large, full-color displays for various applications, including artistic displays, signage, or interactive lighting systems. Each WS281B LED in the chain has an integrated driver, which helps reduce wiring complexity and ensures the control signal reaches each LED in the chain. Since the LEDs are addressable, you can control them individually by sending data that specifies the color (RGB values) and brightness of each LED in the matrix.

To control a WS281B LED matrix, you typically use a microcontroller along with a dedicated library, such as the Adafruit NeoPixel or FastLED libraries, which simplify the programming process. These libraries handle the timing and data protocol required to communicate with the WS281B LEDs. For example, in an Arduino setup, you would connect the data pin of the first WS281B LED in the chain to a digital output pin on the microcontroller. Then, using software, you can set the colors of each individual LED in the matrix, allowing you to create dynamic lighting patterns, animations, or even display images and text.

The flexibility of the WS281B makes it a favorite choice for projects requiring complex, customizable lighting setups. Whether you're building a visual display, a holiday light show, or integrating interactive light effects into a product or environment, the WS281B LED matrix offers an efficient and powerful solution.

**A close-up of a circuit board

AI-generated content may be incorrect.**

**Fig 4.4 WS2812B 16-Bit LED Matrix**

* 1. **Relay Module**

A relay module is an electrically operated switch that allows a low-power circuit to control a high-power circuit. It is widely used in automation, IoT, and embedded systems to control electrical devices such as lights, fans, motors, and appliances.

**Key Components of a Relay Module:**

* Relay Coil – Energized by a low-voltage DC signal to trigger the switch.
* Common (COM) Terminal – Shared terminal in the switching circuit.
* Normally Open (NO) Terminal – The circuit is open when the relay is inactive; closes when activated.
* Normally Closed (NC) Terminal– The circuit is closed when the relay is inactive; opens when activated.
* Optocoupler (Optional) – Provides isolation between the control and power circuits.
* Transistor & Diode Protection Circuit – Protects against voltage spikes.
* Indicator LED – Shows the relay’s status (ON/OFF).
* Screw Terminals or Pin Headers – For easy connection to external devices.

**Types of Relay Modules:**

* Single-Channel Relay Module– Controls one device.
* Multi-Channel Relay Module – Available in 2, 4, 8, or more channels.
* Solid-State Relay (SSR) Module– Uses semiconductor components for silent operation.
* Electromechanical Relay (EMR) Module – Uses a physical contact mechanism.

**Specifications to Consider:**

* Operating Voltage (e.g., 5V, 12V, 24V DC)
* Switching Voltage & Current (e.g., AC 250V/10A, DC 30V/10A)
* Trigger Voltage & Logic (High-level or Low-level trigger)
* Response Time (Typically in milliseconds

**Applications of Relay Modules:**

* Home Automation (Smart lighting, fan control)
* Industrial Control Systems
* IoT and Embedded Projects
* Safety and Security Systems
* Automated Agriculture (Irrigation, Motor Control)

A close-up of a blue electronic device

AI-generated content may be incorrect.

**Fig 4.5 Relay module**

* 1. **Jumping wires**

Jumper wires are insulated conductors used to connect different components on a breadboard**,** circuit board**, or** microcontroller without the need for soldering. They are essential for prototyping and testing electronic circuits.

**Types of Jumper Wires:**

* Male-to-Male (M-M) – Pins on both ends, used for connecting two female headers (e.g., Arduino to a breadboard).
* Male-to-Female (M-F) – One pin and one socket, used for connecting modules or sensors with male pins.
* Female-to-Female (F-F) – Sockets on both ends, used to connect two male header pins.

**Common Features:**

* Insulated with PVC or silicone for flexibility and safety.
* Standard lengthsvary from 10cm to 30cm (or longer).
* Wire Gauge (typically 22-28 AWG).
* Color-coded insulation (Red for power, Black for ground, other colors for signals).
* **Reusable** for multiple projects.

**A close-up of a colorful cable

AI-generated content may be incorrect.**

**Fig 4.6 Jumper wires**

* 1. **LCD display**

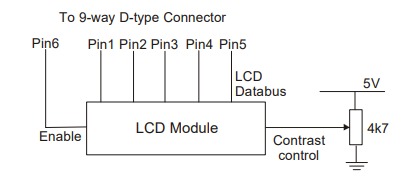
This is an LCD Display designed for E-blocks. It is a 16 character, 2-line alphanumeric LCD display connected to a single 9-way D-type connector. This allows the device to be connected to most E-Block I/O ports. The LCD display requires data in a serial format, which is detailed in the user guide below. The display also requires a 5V power supply. Please take care not to exceed 5V, as this will cause damage to the device. The 5V is best generated from the E-blocks Multi programmer or a 5V fixed regulated power supply.

**FEATURES ·**

* E-blocks compatible ·
* Low cost ·
* Compatible with most I/O ports in the E-Block range (requires 5 I/O lines via 9 way D-type connector) ·
* Ease to develop programming code using Flowcode icons.

****

**Fig 4.7.1 16x2 LCD**

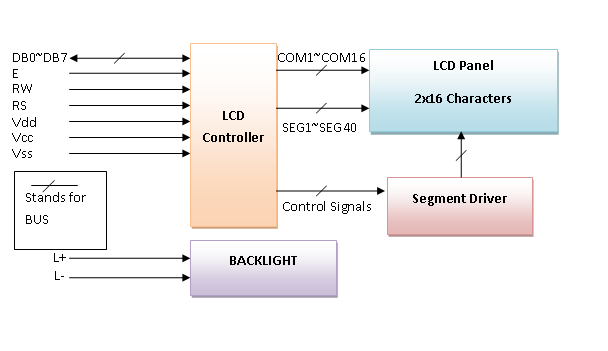
****

**Fig.4.7.2 LCD block diagram**

DATA/SIGNALS/EXECUTION OF LCD

Now that was all about the signals and the hardware. Let us come to data, signals and execution.LCD accepts two types of signals, one is data, and another is control. These signals are recognized by the LCD module from status of the RS pin. Now data can be read also from the LCD display, by pulling the R/W pin high. As soon as the E pin is pulsed, LCD display reads data at the falling edge of the pulse and executes it, same for the case of transmission. LCD display takes a time of 39-43µS to place a character or execute a command. Except for clearing display and to seek cursor to home position it takes 1.53ms to 1.64ms. Any attempt to send any data before this interval may lead to failure to read data or execution of the current data in some devices. Some devices compensate the speed by storing the incoming data to some temporary registers.

LCD displays have two RAMs, naming DDRAM and CGRAM. DDRAM registers in which position which character in the ASCII chart would be displayed. Each byte of DDRAM represents each unique position on the LCD display. The LCD controller reads the information from the DDRAM and displays it on the LCD screen. CGRAM allows user to define their custom characters. For that purpose, address space for first 16 ASCII characters are reserved for users. After CGRAM has been setup to display characters, user can easily display their custom characters on the LCD screen.



**Fig..4.7.3 LCD Block diagram**

The selection of RAM is set by the previous address set instruction. If the address set instruction of RAM is not performed before this instruction, the data that is read first is invalid, because the direction of AC is not determined. If the RAM data is read several times without RAM address set instruction before read operation, the correct RAM data from the second, but the first data would be incorrect, as there is no time to transfer RAM data. In case of DDRAM read operation, cursor shift instruction plays the same role as DDRAM address set instruction; it also transfers RAM data to the output data registers.

**4.8 Power supply (DC Adapter)**

A **DC adapter** (Direct Current adapter) is an external power supply that converts AC (Alternating Current) from a wall outlet into DC (Direct Current) required by various electronic devices. These adapters ensure stable voltage and current to prevent damage and enhance the efficiency of powered devices. DC adapters are widely used in consumer electronics, industrial equipment, medical devices, and IoT applications.

**Working Principle**

The primary function of a DC adapter is to transform and regulate power from an AC source into a usable DC output. This process involves several key components:

* AC Input Stage: The adapter receives AC voltage (typically 100V-240V, depending on the region).
* Rectification and Filtering: A rectifier converts AC into DC, and capacitors smooth out fluctuations.
* Voltage Regulation: A voltage regulator ensures a stable and precise output voltage.
* Output Stage: The final DC output is delivered to the connected device through a specific connector.

**Types of DC Adapters**

DC adapters come in various forms, depending on design and functionality. The major types include:

**Fixed Voltage DC Adapters**

* Provide a single, non-adjustable voltage (e.g., 5V, 9V, 12V, 19V, 24V, etc.).
* Commonly used for routers, gaming consoles, and laptops.

**Adjustable Voltage DC Adapters**

* Offer multiple voltage settings, adjustable via a switch or dial.
* Useful for powering multiple devices with different voltage requirements.

**Linear Power Adapters**

* Use a transformer to step down voltage before rectification.
* Provide clean, low-noise power but are bulkier and less efficient.

**Switching Power Adapters (SMPS)**

* Utilize high-frequency switching circuits for efficient voltage conversion.
* More compact, lightweight, and energy-efficient compared to linear adapters.

**Features and Specifications**

When selecting a DC adapter, the following specifications are crucial:

* Input Voltage**:** 100V-240V AC (universal input for global use).
* Output Voltage**:** Ranges from 3V to 48V DC, depending on device requirements.
* Output Current**:** Measured in Amperes (A), must match or exceed device needs.
* Power Rating**:** Expressed in Watts (W), calculated as Voltage × Current.
* Connector Type**:** Barrel jack, USB, or proprietary connectors.
* Polarity**:** Ensures correct power delivery to prevent device damage.

DC adapters are essential for various electronic and industrial applications, including:

* Consumer Electronics**:** Laptops, routers, gaming consoles, and LED lighting.
* Industrial Equipment**:** PLCs, motor controllers, and automation systems.
* Medical Devices**:** Portable monitors, diagnostic equipment, and infusion pumps.
* IoT and Embedded Systems**:** Microcontrollers, Raspberry Pi, and smart home devices.
* Battery Charging**:** Used in rechargeable battery packs and power banks.

**Safety and Protection Mechanisms**

To ensure safety, modern DC adapters include multiple protection features:

* Overvoltage Protection (OVP): Prevents excessive output voltage.
* Overcurrent Protection (OCP): Limits current to prevent device damage.
* Short-Circuit Protection (SCP): Cuts off power in case of a short circuit.
* Thermal Protection: Prevents overheating through automatic shutdown or cooling mechanisms.
* Surge Protection**:** Shields against voltage spikes and power surges.

**How to Select the Right DC Adapter**

* Choosing the appropriate DC adapter requires consideration of key factors:
* Voltage and Current Matching**:** Ensure the adapter’s output matches the device’s voltage and amperage.
* Connector Compatibility**:** Select the correct size and polarity to fit the device.
* Power Rating**:** Choose an adapter with a sufficient wattage margin.
* Efficiency and Certification**:** Opt for certified adapters (UL, CE, FCC) for safety and reliability.
* Portability and Design**:** Consider compact, lightweight options for mobility.



**Figure 4.8. Power supply DC Adapter.**

* 1. **Batttery**

A battery is a device that stores chemical energy and converts it into electrical energy to power electronic circuits and devices. It consists of one or more electrochemical cells, each containing a positive electrode (cathode), a negative electrode (anode), and an electrolyte that facilitates the movement of ions. When a circuit is connected, a chemical reaction occurs between the electrodes and the electrolyte, releasing electrons through the external circuit and powering the connected load. Batteries are widely used in applications ranging from small portable devices to large systems like electric vehicles and renewable energy storage.

In electric vehicles (EVs), the battery is the primary power source and plays a critical role in determining the vehicle’s range, performance, and charging time. Most modern EVs use lithium-ion batteries due to their high energy density, low weight, and rechargeability. These batteries are made up of many individual cells arranged in series and parallel to deliver the required voltage and capacity. The Battery Management System (BMS) monitors and protects the battery from overcharging, overheating, and deep discharging, which helps extend battery life and ensure safety.

Battery performance is affected by several factors, including temperature, discharge rate, and age. As batteries discharge and recharge over time, their capacity gradually decreases—a phenomenon known as battery degradation. To optimize energy usage, especially in systems like power-saving electric vehicles, it’s important to monitor the battery level and automatically adjust energy consumption based on its state. This approach not only conserves power but also prolongs the battery’s life and increases the reliability and efficiency of the entire system.

**Switch :**

An on/off switch is a basic yet essential electrical component used to control the flow of electricity in a circuit. It allows the user to manually open or close the circuit, thereby turning a device or system on or off. When the switch is in the "on" position, it completes the circuit, allowing current to flow. When in the "off" position, it breaks the circuit and stops the flow of electricity. On/off switches come in various forms, including toggle, rocker, push-button, and slide switches, and are chosen based on the application and user preference.

In electronic systems such as power-saving systems for electric vehicles, an on/off switch can be used to activate or deactivate specific modes, such as entering a low-power state. For example, a manual activation switch could be used by the driver to enable the power-saving mode even when the battery is not critically low. This allows for more efficient energy use, especially during long journeys or in stop-and-go traffic. The switch can be connected to a microcontroller (like the ATmega328P), which then adjusts system behavior based on the switch's state.

When integrating an on/off switch into a microcontroller-based system, it's important to implement debouncing—either through hardware (using a capacitor and resistor) or software—to ensure reliable readings. Mechanical switches can create brief fluctuations (bounces) in the signal as contacts settle, which may cause the system to misinterpret multiple presses. By adding a simple debounce routine in code, the microcontroller can detect clean, accurate switch transitions, making the system more reliable and responsive.

A red switch with a white number on it

AI-generated content may be incorrect.

**FIG 4.10 SWITCH**

* 1. **Buzzer**

A buzzer is an audio signaling device that produces sound when electrical voltage is applied. It is commonly used in electronic systems to provide audible alerts, feedback, or alarms. Buzzers come in two main types: active and passive. An active buzzer has a built-in oscillator and generates sound when powered, while a passive buzzer requires an external signal (like a PWM signal from a microcontroller) to produce sound. Depending on the application, either type can be used to create tones, beeps, or alarms.

In embedded systems like a power-saving system for electric vehicles, a buzzer can serve as a warning or notification device. For example, it can alert the user when the battery level drops below a critical threshold or when power-saving mode is activated. This provides immediate auditory feedback, enhancing user awareness without the need to constantly monitor visual indicators. The buzzer can also be programmed to produce different beep patterns to convey various messages, such as single beeps for mode changes or continuous beeping for critical battery warnings.

Buzzers are typically connected to a digital output pin of a microcontroller like the ATmega328P. For a passive buzzer, you can use Pulse Width Modulation (PWM) to generate specific frequencies and tones. For an active buzzer, simply setting the pin HIGH or LOW will turn the sound on or off. The simplicity, low power consumption, and effectiveness of buzzers make them a valuable addition to many interactive and safety-critical systems.

A black round object with red and black wires

AI-generated content may be incorrect.

**FIG 4.11 BUZZER**

* 1. **DC fan**

A DC fan is an electric fan powered by a direct current (DC) voltage source, typically used for cooling or ventilation in electronic devices, vehicles, and industrial systems. It operates by converting electrical energy into mechanical energy through a DC motor, which spins the fan blades to create airflow. The speed of a DC fan can be easily controlled by adjusting the voltage or using Pulse Width Modulation (PWM), making it highly efficient and adaptable to different cooling needs.

In the context of a power-saving system for electric vehicles, a DC fan is often used for cooling internal components such as the motor driver, battery pack, or microcontroller unit. While important for maintaining optimal operating temperatures, the fan itself consumes power. Therefore, in power-saving mode, the system can be programmed to turn off or reduce the fan speed when it's not critically needed, helping to conserve battery life without significantly compromising system performance.

DC fans are typically small, lightweight, and come in various sizes and voltage ratings (like 5V, 12V, or 24V). They are connected to the control system using transistors or MOSFETs, allowing microcontrollers like the ATmega328P to switch them on or off automatically based on temperature sensors or power-saving triggers. Integrating a fan smartly within an energy management system not only protects sensitive electronics but also contributes to overall system efficiency by running only when necessary.

A black fan with red wire

AI-generated content may be incorrect.

**FIG 4.12 DC FA**

# CHAPTER-5

**SOFTWARE COMPONENTS**

**5.1 ARDUINO IDE**

The Arduino Integrated Development Environment ([IDE](https://en.wikipedia.org/wiki/Integrated_development_environment)) is a [cross-platform](https://en.wikipedia.org/wiki/Cross-platform) application (for [Windows](https://en.wikipedia.org/wiki/Windows), [macOS](https://en.wikipedia.org/wiki/MacOS), [Linux](https://en.wikipedia.org/wiki/Linux)) that is written in functions from [C](https://en.wikipedia.org/wiki/C_(programming_language)) and [C++](https://en.wikipedia.org/wiki/C%2B%2B_(programming_language)). It is used to write and upload programs to [Arduino](https://en.wikipedia.org/wiki/Arduino) compatible boards, but also, with the help of third-party cores, other vendor development boards.

The source code for the IDE is released under the [GNU General Public License](https://en.wikipedia.org/wiki/GNU_General_Public_License), version 2. The Arduino IDE supports the languages [C](https://en.wikipedia.org/wiki/C_(programming_language)) and [C++](https://en.wikipedia.org/wiki/C%2B%2B) using special rules of code structuring. The Arduino IDE supplies a [software library](https://en.wikipedia.org/wiki/Software_library) from the [Wiring](https://en.wikipedia.org/wiki/Wiring_(development_platform)) project, which provides many common input and output procedures.

User-written code only requires two basic functions, for starting the sketch and the main program loop, that are compiled and linked with a program stub main() into an executable [cyclic executive](https://en.wikipedia.org/wiki/Cyclic_executive) program with the [GNU toolchain](https://en.wikipedia.org/wiki/GNU_toolchain), also included with the IDE distribution. The Arduino IDE employs the program avrdude to convert the executable code into a text file in hexadecimal encoding that is loaded into the Arduino board by a loader program in the board's firmware. By default, avrdude is used as the uploading tool to flash the user code onto official Arduino boards.

Arduino microcontrollers are pre-programmed with a boot loader that simplifies uploading of programs to the on-chip flash memory. The default bootloader of the Arduino Uno is the Optiboot bootloader. Boards are loaded with program code via a serial connection to another computer.

Some serial Arduino boards contain a level shifter circuit to convert between RS-232 logic levels and transistor–transistor logic (TTL) level signals. Current Arduino boards are programmed via Universal Serial Bus (USB), implemented using USB-to-serial adapter chips such as the FTDI FT232. Some boards, such as later-model Uno boards, substitute the FTDI chip with a separate AVR chip containing USB-to-serial firmware, which is reprogrammable via its own ICSP header. Other variants, such as the Arduino Mini and the unofficial Boarduino, use a detachable USB-to-serial adapter board or cable, Bluetooth or other methods. When used with traditional microcontroller tools, instead of the Arduino IDE, standard AVR in-system programming (ISP) programming is used.

The Arduino/Genuino Uno has a number of facilities for communicating with a computer, another Arduino/Genuino board, or other microcontrollers. The ATmega328 provides UART TTL (5V) serial communication, which is available on digital pins 0 (RX) and 1 (TX). An ATmega16U2 on the board channels this serial communication over USB and appears as a virtual com port to software on the computer. The 16U2 firmware uses the standard USB COM drivers, and no external driver is needed.

However, on Windows, a .inf file is required. Arduino Software (IDE) includes a serial monitor which allows simple textual data to be sent to and from the board. The RX and TX LEDs on the board will flash when data is being transmitted via the USB-to-serial chip and USB connection to the computer (but not for serial communication on pins 0 and 1). A SoftwareSerial library allows serial communication on any of the Uno's digital pins.

Arduino IDE is an open-source Integrated Development Environment (IDE) that is widely used by developers and hobbyists to program and control Arduino boards. It provides a simple and user-friendly interface that allows users to write, compile, and upload code to an Arduino board.

The Arduino IDE is designed to be a cross-platform software, which means that it can run on different operating systems, including Windows, Mac OS, and Linux. It is written in Java programming language and is based on the Processing development environment.

The IDE comes with a built-in text editor that supports syntax highlighting for different programming languages, such as C and C++. The editor also has features such as auto-completion and error highlighting to make coding easier and more efficient.

One of the key features of the Arduino IDE is its library manager. The library manager is an online repository of libraries that can be easily downloaded and added to your Arduino projects. These libraries contain pre-written code that can be used to interface with sensors, actuators, and other hardware components, saving time and effort for the developer.

The Arduino IDE also has a Serial Monitor, which allows users to communicate with the Arduino board via the serial port. This is useful for debugging and testing purposes, as it enables users to send and receive data between the board and the computer.

In addition, the IDE comes with a bootloader that enables users to upload sketches (code) to the board without the need for an external programmer. This makes it easy for beginners to get started with programming and experimenting with different projects.

Overall, the Arduino IDE is a powerful tool that makes it easy for developers and hobbyists to program and control Arduino boards. Its simplicity and user-friendliness make it a popular choice among beginners, while its advanced features make it a valuable tool for experienced developers.



Fig.5.1.Arduino IDE

Rather than requiring a physical press of the reset button before an upload, the Arduino/Genuino Uno board is designed in a way that allows it to be reset by software running on a connected computer. One of the hardware flow control lines (DTR) of the ATmega8U2/16U2 is connected to the reset line of the ATmega328 via a 100 nanofarad capacitor. When this line is asserted (taken low), the reset line drops long enough to reset the chip.

This setup has other implications. When the Uno is connected to a computer running Mac OS X or Linux, it resets each time a connection is made to it from software (via USB). For the following half-second or so, the bootloader is running on the Uno. While it is programmed to ignore malformed data (i.e. anything besides an upload of new code), it will intercept the first few bytes of data sent to the board after a connection is opened.

A screenshot of a computer program

AI-generated content may be incorrect.

Fig.52.Arduino IDE console

DESIGN APPROACH: SOFWARE

The Arduino C program consists of 2 functions, namely setup () and loop (). The setup () function is run every time the Arduino board is turned on. While the loop () function is run continuously as long as the Arduino board lives. The computer will send measurement commands to the microcontroller via serial communication. The flowchart for Arduino Uno programming. We used embedded “C” on Arduino IDE platform to code the controller. Voice control action is done by BTvoice control android application.

This application software uses android phone’s internal voice recognition to forward voice command to the microcontroller Pairs with Bluetooth Serial Modules and sends in the recognized voice as a string. while android device is paired with microcontroller via Bluetooth we can give audio input like forward ,left, right back as well as line follower input. In this application if one says Chair the phone will return a sting \*Chair# to your Bluetooth module. Here ‘\*’and‘#’ is for start and stop bits respectively. Fig. 7 is showing the flow chart for voice control.

The Arduino IDE (Integrated Development Environment) is a software application used to write, compile, and upload code to Arduino boards. Here's an overview of its features and functionality:

1. Code Editor: The Arduino IDE provides a text editor where you can write and edit your Arduino sketch (code). It supports syntax highlighting, auto-indentation, and code completion, making it easier to write and debug code.

2. Sketch Structure: Arduino sketches are written in the C or C++ programming language and follow a specific structure. Each sketch consists of two main functions: setup() and loop(). The setup() function is called once when the Arduino board is powered on or reset, while the loop() function is called repeatedly for as long as the board is powered on.

3. Library Manager: The Arduino IDE includes a Library Manager that allows you to easily install and manage libraries (collections of pre-written code) for various sensors, actuators, and other components. These libraries provide convenient functions and classes to interface with external hardware.

4. Serial Monitor: The Arduino IDE includes a Serial Monitor tool that allows you to communicate with the Arduino board over the serial port. You can use it to send and receive data, debug your code, and display sensor readings or other output messages.

5. Board Manager: The Arduino IDE supports a wide range of Arduino-compatible boards, including official Arduino boards as well as third-party boards from other manufacturers. The Board Manager allows you to select the appropriate board type and install board-specific drivers and firmware.

6. Upload and Compile: Once you've written your Arduino sketch, you can compile it into machine code using the Arduino IDE. The IDE then uploads the compiled code to the Arduino board via a USB connection, allowing you to run your program on the board.

7. Integrated Examples: The Arduino IDE includes a collection of example sketches that demonstrate various features and functionalities of Arduino boards and peripherals. These examples cover topics such as digital and analog input/output, communication protocols (e.g., UART, SPI, I2C), and sensor interfacing.

8. Open Source: The Arduino IDE is open-source software, meaning its source code is freely available for modification and redistribution. This allows developers to contribute improvements, fixes, and new features to the IDE, ensuring its ongoing development and compatibility with new hardware and software platforms.

Overall, the Arduino IDE provides a user-friendly and intuitive environment for developing projects with Arduino boards, making it accessible to beginners and experienced users alike. Its simplicity, versatility, and extensive community support make it a popular choice for prototyping and experimenting with electronics and embedded systems.

A screenshot of a sketching window

AI-generated content may be incorrect.

Fig.5.3.ArduinoIDE editor page.

A screenshot of a computer program

AI-generated content may be incorrect.

Fig.5.4.ArduinoIDE menu page.

A computer screen with text and a error

AI-generated content may be incorrect.

Fig.5.5.ArduinoIDE program page.

The Arduino Integrated Development Environment (IDE) is a software application used to write, compile, and upload code to Arduino boards. Here's an overview of its key features and functionality:

1. Code Editor: The Arduino IDE provides a simple and user-friendly code editor where you can write and edit your Arduino sketches (programs). It supports syntax highlighting and automatic indentation to make writing code easier.

2. Sketch Structure: Arduino sketches are organized into two main functions: `setup()` and `loop()`. The `setup()` function is called once when the Arduino board is powered on or reset, and it is used to initialize variables, pin modes, and other settings. The `loop()` function is then called repeatedly after `setup()` and is where you put the main code that you want the Arduino to execute continuously.

3. Built-in Libraries: The Arduino IDE comes with a collection of built-in libraries that provide pre-written code for common tasks such as controlling digital and analog pins, communicating with sensors and actuators, and interfacing with displays and other peripherals.

4. Serial Monitor: The Arduino IDE includes a Serial Monitor tool that allows you to communicate with your Arduino board over the serial port. You can use it to send and receive data between the Arduino and your computer, which is useful for debugging and monitoring the behavior of your sketches.

5. Board Manager: The Arduino IDE includes a Board Manager feature that allows you to easily add support for different types of Arduino boards and microcontrollers. You can install board definitions for specific Arduino models, as well as third-party boards from other manufacturers.

6. Library Manager: The Arduino IDE also includes a Library Manager tool that makes it easy to install and manage additional libraries for your projects. You can search for libraries by name or browse through categories, and then install them with just a few clicks.

7. Upload Tool: Once you've written and tested your Arduino sketch, you can use the Arduino IDE to upload it to your Arduino board. The IDE compiles your code into machine-readable binary files and sends them to the board via USB or another serial connection.

8. Cross-platform Support: The Arduino IDE is available for Windows, macOS, and Linux, making it accessible to users on a wide range of operating systems.

Overall, the Arduino IDE provides a convenient and beginner-friendly environment for writing and uploading code to Arduino boards, making it a popular choice for hobbyists, students, and professionals alike.

The Arduino Integrated Development Environment (IDE) is a software application used to write, compile, and upload code to Arduino boards. Here are some key features and functionalities of the Arduino IDE:

1. Code Editor: The IDE provides a simple yet powerful code editor where you can write and edit your Arduino sketches (programs). It supports syntax highlighting, auto-indentation, and code completion to assist you while writing code.

2. Sketch Structure: Arduino sketches consist of two main functions: `setup()` and `loop()`. The `setup()` function is executed once when the Arduino board is powered on or reset, while the `loop()` function is executed continuously in a loop after the `setup()` function completes.

3. Library Manager: The IDE includes a library manager that allows you to easily install and manage Arduino libraries. Libraries contain pre-written code that you can use to extend the functionality of your Arduino projects, such as interfacing with sensors, displays, or communication modules.

4. Serial Monitor: The IDE includes a serial monitor tool that allows you to communicate with your Arduino board via the serial port. You can use the serial monitor to send data to the Arduino board and receive data back from it for debugging and testing purposes.

5. Board Manager: The IDE includes a board manager that allows you to select the type of Arduino board you are using and install the necessary board definitions and drivers. This enables the IDE to compile and upload code specifically for your Arduino board.

6. Sketch Upload: Once you have written your Arduino sketch, you can upload it to your Arduino board directly from the IDE using a USB cable. The IDE compiles the sketch into machine code, uploads it to the Arduino board's microcontroller, and starts execution automatically.

7. Examples and Tutorials: The IDE includes a variety of built-in examples and tutorials to help you get started with Arduino programming. These examples cover a wide range of topics and demonstrate how to use different Arduino features and components.

8. Cross-Platform Support: The Arduino IDE is available for multiple operating systems, including Windows, macOS, and Linux, making it accessible to a wide range of users.

Overall, the Arduino IDE provides a user-friendly and accessible platform for programming Arduino boards, making it suitable for beginners and experienced users alike to create interactive electronics projects.

The Arduino IDE (Integrated Development Environment) is a software application used for writing, compiling, and uploading code to Arduino microcontroller-based boards. Here's an overview of its key features and functionalities:

1. Code Editor: The Arduino IDE provides a text editor where you can write your Arduino sketches (code). It offers syntax highlighting, auto-indentation, and code completion features to assist with writing and editing code.

2. Library Manager: The IDE includes a library manager that allows you to easily search for and install additional libraries to extend the functionality of your Arduino projects. These libraries contain pre-written code that you can use to interface with sensors, actuators, displays, and other peripheral devices.

3. Board Manager: The Board Manager allows you to select the type of Arduino board you are using from a list of supported boards. It also provides options for installing additional board definitions for third-party boards or custom hardware.

4. Serial Monitor: The Serial Monitor is a tool for debugging and monitoring serial communication between the Arduino board and your computer. It displays data sent by the Arduino over the serial port and allows you to send data back to the Arduino for testing and troubleshooting purposes.

5. Compile and Upload: The IDE provides buttons for compiling and uploading your Arduino sketches to the connected Arduino board. The compilation process checks your code for errors and generates a binary file (hex file) that is then uploaded to the board via a USB connection.

6. Examples: The IDE includes a collection of example sketches covering a wide range of topics and applications. These examples serve as starting points for your own projects and provide demonstrations of how to use various Arduino functions and libraries.

7. Preferences: The Preferences menu allows you to customize various settings of the IDE, such as the editor font size, compiler warnings, and external editor preferences.

8. Integrated Help: The IDE includes built-in documentation and help resources accessible via the Help menu. It provides information on Arduino functions, libraries, and programming concepts to assist with learning and troubleshooting.

Overall, the Arduino IDE provides a user-friendly and intuitive environment for writing and uploading code to Arduino boards, making it accessible to beginners and experienced developers alike. Its simplicity and versatility make it a popular choice for prototyping and developing Arduino-based projects.

# CHAPTER-6

**RESULTS**

# CHAPTER-7

### CONCLUSION

In conclusion, the proposed Power Saving System for Electric Vehicles effectively enhances energy efficiency and extends driving range by intelligently managing the vehicle's electrical components. Key elements such as the ATmega328P microcontroller, L293D motor driver, manual on/off switch, buzzer, DC fan, and battery monitoring work together to optimize power consumption, especially during critical battery conditions. By automatically disabling non-essential systems like fans, lights, and infotainment units, and by regulating motor speed, the system ensures that available energy is used wisely and efficiently.

The inclusion of a manual power-saving switch empowers users to engage the mode proactively, reducing unnecessary energy usage even before reaching low battery levels. Components like the buzzer provide real-time alerts, while smart fan control ensures essential cooling without excessive power draw. The use of a programmable LED matrix or indicator system could also be explored to provide visual feedback, further improving user interaction.

**CHAPTER 8**

**FUTURE WORK**

Future work for the proposed Power Saving System can focus on integrating IoT and mobile app support for real-time monitoring and remote control, enabling users to track battery usage and activate power-saving features conveniently. Incorporating artificial intelligence could allow the system to learn from driving patterns and environmental conditions to dynamically optimize energy consumption. Additionally, support for solar charging, an enhanced user interface using LED matrices or touchscreens, and a modular design adaptable to various types of electric vehicles can further increase the system’s efficiency, usability, and applicability across a wide range of EV platforms**.**

### REFERENCES

[1] Vinay Teja, P., Pradep, R., Prathiksh, S., & Karthika, V. (2019). Power Saving System for Electric Vehicles. International Journal of Innovative Research in Science, Engineering and Technology, 8(3), 2450-2455.

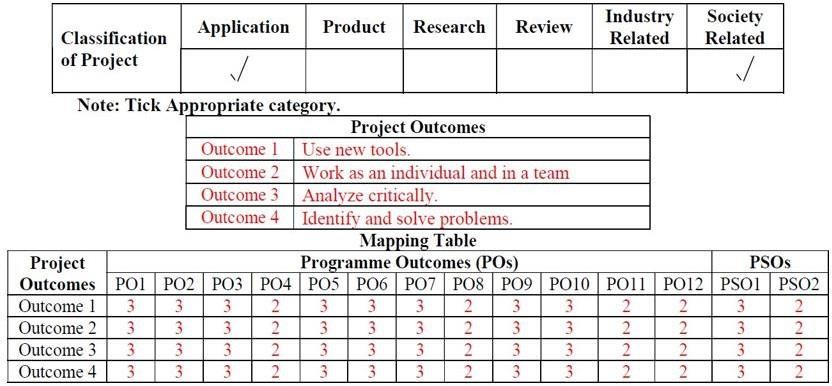
[2] Nagrale, S., Waghmare, A., Sahare, T., Nagwanshi, A., Sakhale, I., & Sathe, J. (2023). Arduino Based Battery Management System for Electric Vehicles. International Journal of Advanced Research in Innovative Ideas in Education, 9(2), 2248-2251.

[3] LinkDhavale, S. R., Mahajan, M., Mahajan, K., Chavan, A., & Sidwadkar, R. (2022). Smart Electrical Vehicle System. International Journal of Research Publication and Reviews, 3(5), 3465-3473.

[4] Arduino Forum. (2019). ATMEGA328P Power Saving.

[5] Instructables. (2013). How to Use the L293D Motor Driver - Arduino Tutorial.

### PROJECT PERFORMANCE

****

Note: Map each project outcomes with POs and PSOs with either 1 or 2 or 3 based on level of mapping as follows

1-Slightly (Low) mapped 2-Moderately (Medium) mapped 3-Substantially (High) mapped

**Program Outcomes:**

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering

fundamentals and an engineering specialization for the solution of complex engineering problems.

1. **Problem analysis:** Identify, formulate, research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
2. **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for public health and safety, and cultural, societal, and environmental considerations.
3. **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
4. **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and Modern engineering and IT tools, including prediction and modeling to complex engineering activities, with an understanding of the limitations.
5. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal, and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
6. **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
7. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
8. **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
9. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with the society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
10. **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one’s own work, as a member and leader in a team, to manage projects and in multidisciplinary environments
11. **Life-long learning:** Recognizes the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

##### Program Specific Outcomes (PSO):

1. Apply the knowledge of circuit design, analog & digital electronics to the field of electrical and electronics systems
2. Analyze, design and develop control systems, industrial drives and power systems using modern tool

#### Student self Evaluation for the project

Student Name:

S.Rakesh Babu

Registration No:

21481A0281

Period of PROJECT: From:15-09-2023 To: 02-12-2023

Date of Evaluation:

Name of the Person in-charge: Ch.Sujatha

Address with mobile number: Gudlavalleru(v), Gudlavalleru(m), krishna(d),521356.

**Please rate your performance in the following areas:**

**Rating Scale: 1 is lowest and 5 is highest rank**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **1) Oral communication** | **1** | **2** | **3** | **4** | **5** |
| **2) Written communication** | **1** | **2** | **3** | **4** | **5** |
| **3) Proactiveness** | **1** | **2** | **3** | **4** | **5** |
| **4) Interaction ability with community** | **1** | **2** | **3** | **4** | **5** |
| **5) Positive Attitude** | **1** | **2** | **3** | **4** | **5** |
| **6) Self-confidence** | **1** | **2** | **3** | **4** | **5** |
| **7) Ability to learn** | **1** | **2** | **3** | **4** | **5** |
| **8) Work Plan and organization** | **1** | **2** | **3** | **4** | **5** |
| **9) Professionalism** | **1** | **2** | **3** | **4** | **5** |
| **10) Creativity** | **1** | **2** | **3** | **4** | **5** |
| **11) Quality of work done** | **1** | **2** | **3** | **4** | **5** |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **12) Time Management** | **1** | **2** | **3** | **4** | **5** |
| **13) Understanding the Community** | **1** | **2** | **3** | **4** | **5** |
| **14) Achievement of Desired Outcomes** | **1** | **2** | **3** | **4** | **5** |
| **15) OVERALL PERFORMANCE** | **1** | **2** | **3** | **4** | **5** |

**Date: Signature of the Student**

#### Evaluation by the person in-charge of the Project

Student Name:

P. Nataraj Venkateshan

Registration No:

22485A0223

Period of the project:

Date of Evaluation:

Name of the Person in-charge: SK. Rizwana

Address

: Gudiwada, Gudiwada mandal, Krishna District 521301

**Please rate the student’s performance in the following areas: Please note that your evaluation shall be done independent of the**

**student’s self-evaluation Rating Scale: 1 is lowest and 5 is highest rank**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **1) Oral communication** | **1** | **2** | **3** | **4** | **5** |
| **2) Written communication** | **1** | **2** | **3** | **4** | **5** |
| **3) Proactiveness** | **1** | **2** | **3** | **4** | **5** |
| **4) Interaction ability with community** | **1** | **2** | **3** | **4** | **5** |
| **5) Positive Attitude** | **1** | **2** | **3** | **4** | **5** |
| **6) Self-confidence** | **1** | **2** | **3** | **4** | **5** |
| **7) Ability to learn** | **1** | **2** | **3** | **4** | **5** |
| **8) Work Plan and organization** | **1** | **2** | **3** | **4** | **5** |
| **9) Professionalism** | **1** | **2** | **3** | **4** | **5** |
| **10) Creativity** | **1** | **2** | **3** | **4** | **5** |
| **11) Quality of work done** | **1** | **2** | **3** | **4** | **5** |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **12) Time Management** | **1** | **2** | **3** | **4** | **5** |
| **13) Understanding the Community** | **1** | **2** | **3** | **4** | **5** |
| **14) Achievement of Desired Outcomes** | **1** | **2** | **3** | **4** | **5** |
| **15) OVERALL PERFORMANCE** | **1** | **2** | **3** | **4** | **5** |

**Date: Signature of the Supervisor**