A project report

on

DC MICROGRID INTEGRATION WITH BATTERY STORAGE SYSTEM

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

BACHELOR OF TECHNOLOGY

In

ELECTRICAL AND ELECTRONICS ENGINEERING

Submitted by

ROWLAJANNI JYOTSHNA	**21B91A02C8**
KANCHARAPU BHARGAVI	**21B91A0272**
PALLA ANAND KUMAR	**21B91A02B4**
NAGIDI BALAJI	**21B91A02A8**

Under the esteemed guidance of

Dr. P.JAGADEESH Ph.D.
Assistant Professor, EEE Department

DEPARTMENT OF ELECTRICAL & ELECTRONICS ENGINEERING

S.R.K.R. ENGINEERING COLLEGE (A)
(Affiliated to JNTU KAKINADA) (Recognized by A.I.C.T.E, New Delhi)
(Accredited by N.B.A., NAAC with ‘A+’ grade, New Delhi)
CHINNA AMIRAM, BHIMAVARAM-534204

(2025)

S.R.K.R. ENGINEERING COLLEGE (A)

(Affiliated to JNTU KAKINADA) (Recognized by A.I.C.T.E, New Delhi)
(Accredited by N.B.A., NAAC with 'A+' grade, New Delhi)

CHINNA AMIRAM, BHIMAVARAM-534204

DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING

Certificate

This is to certify that Mr./Ms. RJ.JYOTSHNA, K.BHARGAVI, P.ANAND KUMAR, N.BALAJI, of final year B. Tech., EEE, have carried out the project work on "DC MICROGRID INTEGRATION WITH BATTERY STORAGE SYSTEM" in partial fulfilment of the requirements for the award of the Degree of 'Bachelor of Technology' with specialization of Electrical and Electronics Engineering in S.R.K.R. Engineering College (A), Bhimavaram. This is a bonafide record of the work done by us during the academic year 2024 – 2025. The results of this project work have not been submitted to any other university or Institute for the award of any degree.

Guide	Head of the Department
Dr. P.JAGADEESH Ph.D.	**Dr. B.R.K. VARMA** M.E., Ph.D.
Assistant Professor	Professor & HOD

CERTIFICATE OF EXAMINATION

This is to certify that we had examined the project report and here by accord our approval of it as a study carried out and presented in a manner required for its acceptance in partial fulfilment of the award of the degree of **BACHELOR OF TECHNOLOGY** in **ELECTRICAL AND ELECTRONICS ENGINEERING** for which it has been submitted.

This approval does not necessarily endorse or accept every statement made, opinion expressed or conclusions drawn as recorded in the report. It only signifies the acceptance of the report for the purpose of which it is submitted.

INTERNAL EXAMINER **EXTERNAL EXAMINER**

DECLARATION

This thesis entitled "DC MICROGRID INTEGRATION WITH BATTERY STORAGE SYSTEM" has been carried out by us in the partial fulfilment of the requirements for the award of the degree of Bachelor of Technology in the Department of Electrical and Electronics Engineering, Sagi Rama Krishnam Raju Engineering College (A), Bhimavaram. We hereby declare that this thesis has not been submitted to any other university/institute for the award of any other degree/diploma.

Roll. No.	**Name**	**Signature**
21B91A02C8	RJ. Jyotshna	________________
21B91A0272	K. Bhargavi	________________
21B91A02B4	P. Anand Kumar	________________
21B91A02A8	N. Balaji	________________

ACKNOWLEDGEMENT

Words are inadequate to express the overwhelming sense of gratitude and humble regards to my supervisor Dr. P.JAGADEESH, Assistant Professor in Department of Electrical and Electronics Engineering for his constant motivation, support, expert guidance, constant supervision and constructive suggestion for the submission of my progress report of thesis work "DC MICROGRID INTEGRATION WITH BATTERY STORAGE SYSTEM".

We would like to express our deep sense of gratitude and sincere regards to Dr. B.R.K.VARMA, Head of Electrical and Electronics Engineering Department for his intense support throughout the project. We are over whelmed to thank our principal Dr. K. V. MURALI KRISHNAM RAJU and management of S.R.K.R Engineering completion College who has given all the facilities and support for the successful of our project.

We take this opportunity to express our acknowledgement to all the faculty members of Electrical and Electronics Engineering Department for giving valuable advices and support which paved the way for successful completion of our project.

We take this opportunity to convey our sincere thanks to all those who have directly and indirectly contributed for the successful completion of our project work

Project Associates

21B91A02C8

21B91A0272

21B91A02B4

21B91A02A8

ABSTRACT

A microgrid is a self-sufficient energy system that serves a discrete geographic footprint, such as a college campus, hospital, or business centre. With the development of computer networks and automation technology, precise monitoring of microgrid parameters has become essential. Various methods are employed to monitor the performance of a microgrid. This paper presents an IoT-based automated DC microgrid system. The primary aim of the developed system is to monitor the performance of a DC microgrid and automatically isolate the load upon the detection of over-voltage and under-voltage conditions. It primarily focuses on generating power using renewable energy resources, such as solar and wind energy, while monitoring the power supply and load voltages through the Blynk Application.

***Key Words*: DC Microgrid, Automation, Internet of Things (IoT), Blynk Application.**

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER -1

INTRODUCTION

The global energy demand is escalating, with India's electricity generation reaching 1,383.5 TWh in FY 2019-20, driven by urbanization and industrialization. Renewable energy sources, such as solar and wind, are pivotal for sustainable development, contributing significantly to India's energy mix (approximately 25% as of 2023). DC microgrids, which integrate distributed generators and loads, offer efficient power distribution by minimizing AC-DC conversion losses. These systems are ideal for localized applications, such as college campuses, hospitals, and urban EV charging stations. However, challenges like fault detection, load management, and real-time monitoring necessitate advanced automation. IoT technology enables remote monitoring and control, enhancing microgrid reliability and efficiency.

1.1 Motivation of the Project

The motivation for this project arises from the need to address inefficiencies in microgrid operation, such as manual monitoring and fault-related disruptions, particularly in urban settings like EV charging stations. Our academic experience in the Smart Systems Lab for IoT, where we explored sensor integration and cloud-based monitoring with platforms like Blynk, and the Solar System Lab, where we worked on solar PV systems and energy storage, inspired us to develop an IoT-based DC microgrid automation system. These labs highlighted the challenges of managing variable renewable energy outputs and the potential of IoT for real-time control, motivating us to create a scalable, reliable solution that reduces conversion losses and enhances sustainable power distribution in urban environments.

1.2 Aim of the Project

The primary aim of this project is to develop an IoT-enabled DC microgrid automation system capable of continuously monitoring electrical parameters and automatically responding to voltage anomalies. The system is designed to enhance the efficiency and reliability of renewable energy usage by automating the isolation of loads during over-voltage or under-voltage conditions. Additionally, the system is equipped to switch the load between available power sources, such as solar panels, wind-powered generators, and battery storage, based on real-time conditions. The integration of the Blynk IoT application enables users to remotely monitor and control the system, set voltage thresholds, and receive instant alerts in case of abnormal operating conditions.

1.3 Scope of the Project

The project focuses on designing and implementing a prototype DC microgrid with IoT-based automation. It includes hybrid power generation (solar and wind), real-time monitoring via Blynk, and automated fault response. The system is tested in a controlled environment simulating an urban microgrid for EV charging. Future scope includes integration with advanced energy storage and cloud-based analytics.

CHAPTER – 2

PROPOSED SYSTEM

2.1 Project Overview

The IoT-based DC microgrid automation system integrates renewable energy sources (solar panels and a DC geared motor simulating wind energy) with a battery energy storage system (BESS). The system monitors power parameters using voltage and current sensors, processes data via an ESP32 microcontroller, and controls loads with a 4-channel relay module. The Blynk application provides a user-friendly interface for real-time monitoring, threshold settings, and fault alerts. The system operates in two phases:

1. Power Generation and Monitoring: Generates power and measures voltages.
2. Fault Detection and Control: Isolates loads during faults based on user-defined thresholds.

2.2 System Architecture

- Power Sources: 12V, 15W solar panel; 150 RPM DC geared motor; 12V battery.
- Sensors: Voltage sensors (using voltage divider) and current sensor.
- Controller: ESP32 with Wi-Fi for IoT connectivity.
- Actuators: 4-channel 5V relay module for load switching.
- Interface: Blynk app for monitoring and control**.**

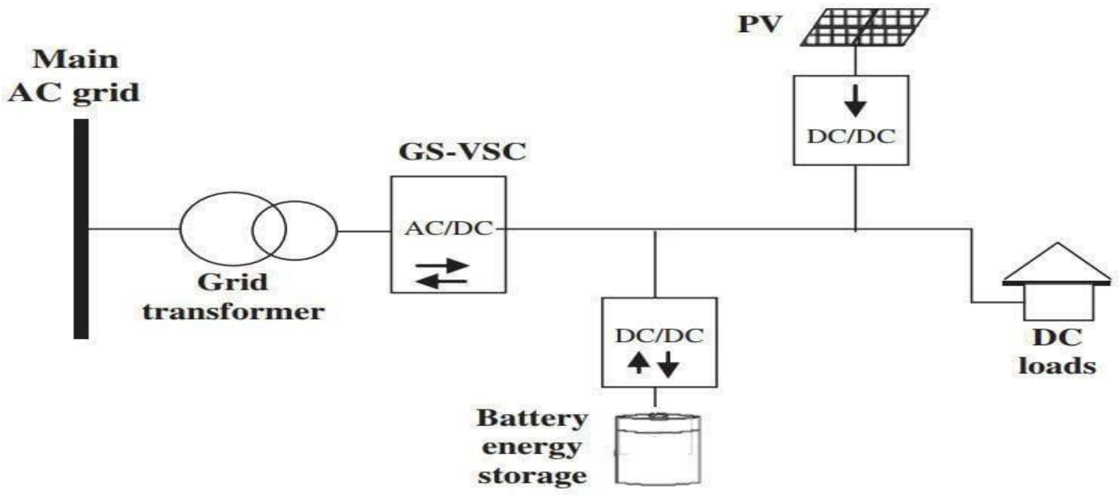


Fig.2.1- Architecture of DC Microgrid

2.3 Fundamental Concepts of Grid and Microgrid

To understand the context of the proposed IoT-based DC microgrid automation system, it is essential to define the key terms grid and microgrid, which form the foundation of modern power distribution systems.

2.3.1 Grid:

The electrical grid is an interconnected network designed to deliver electricity from power generation sources to consumers. It comprises generation stations (using conventional sources like coal and nuclear or renewable sources like solar and wind), transmission lines (high-voltage lines for long-distance power transfer), distribution networks (medium- and low-voltage lines for local delivery), and loads (residential, commercial, and industrial consumers). The grid operates as a centralized system, ensuring reliable power supply across large regions, but it faces challenges such as transmission losses and vulnerability to faults. In the context of this project, the grid serves as the main power system with which the microgrid can interface in parallel mode or operate independently.

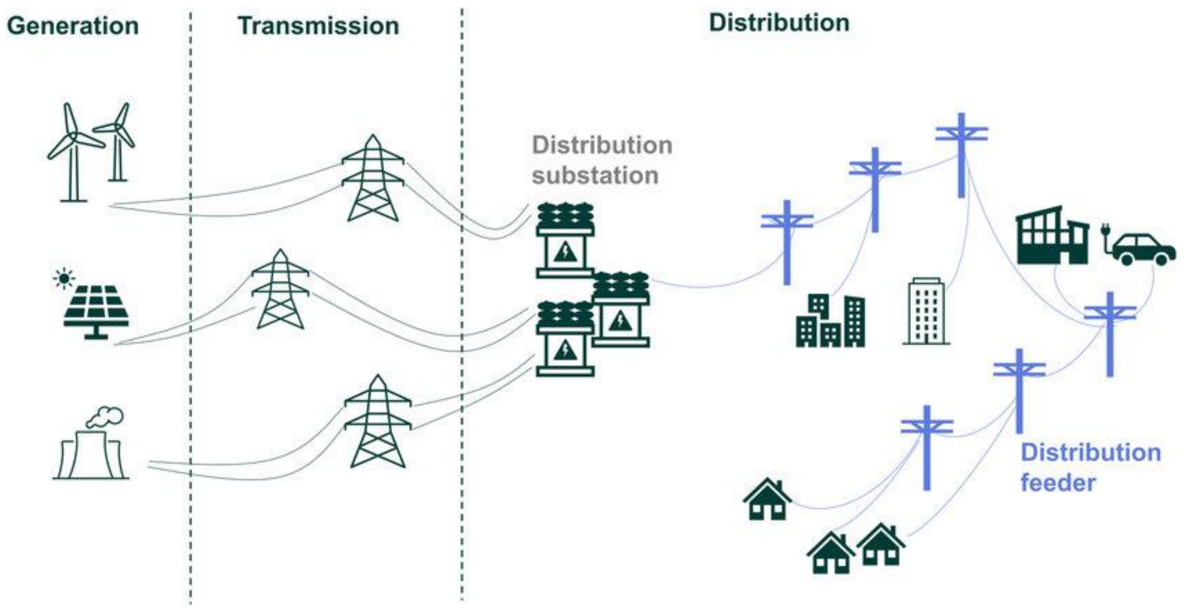


Fig.2.2- Layout of Grid

2.3.2 Microgrid:

A microgrid is a localized energy system that integrates distributed generators (DGs), energy storage systems, and loads within a defined geographic area, such as a campus, hospital, or urban building. Unlike the main grid, microgrids can operate in grid-connected mode (parallel to the grid) or islanded mode (isolated from the grid), providing flexibility and resilience. DC microgrids, as used in this project, minimize AC-DC conversion losses, making them ideal for integrating renewable sources like solar photovoltaic (PV) systems and wind turbines, which produce DC power. Microgrids enhance energy efficiency, support sustainable power distribution, and enable advanced automation, such as real-time monitoring and fault management, through technologies like IoT.

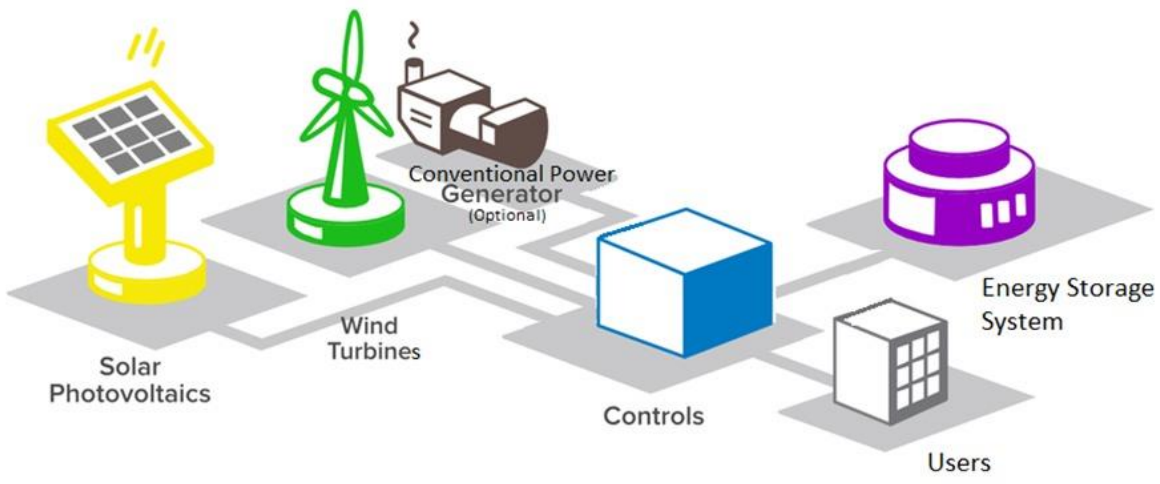


Fig.2.3 – Layout of Microgrid

CHAPTER – 3
LITERATURE REVIEW

3.1 Overview of Microgrid Automation

Microgrid automation is a critical field in modern power systems, focusing on real-time monitoring, fault detection, and load management to enhance efficiency and reliability. Microgrids, which integrate distributed generators (DGs), energy storage systems, and loads, can operate in grid-connected or islanded modes, offering flexibility for localized applications such as campuses, hospitals, and urban EV charging stations. DC microgrids, in particular, are highly efficient due to the absence of reactive power issues and skin effect, making them ideal for integrating renewable energy sources like solar photovoltaic (PV) systems and wind turbines. Automation techniques, including Supervisory Control and Data Acquisition (SCADA) systems, IoT-based monitoring, and advanced control strategies, have gained traction for their ability to optimize energy flow and ensure system stability. IoT technologies, with their remote access and scalability, are increasingly adopted to address challenges like variable power output and fault management, enabling seamless integration with cloud systems and user-friendly interfaces. This section reviews existing literature on microgrid automation, highlighting key systems, their contributions, and limitations, to contextualize the proposed IoT-based DC microgrid automation system.

3.2 Existing Systems

Several studies have explored automation techniques for DC microgrids, focusing on monitoring, control, and optimization. Below, we summarize key works, their methodologies, and their relevance to the proposed system.

Levi et al. (2018) developed a SCADA-based microgrid monitoring system designed for industrial applications [1]. The system utilized wireless smart sensors to interconnect microgrid components with a control room, enabling real-time data acquisition and power setting updates. The SCADA framework provided a common communication platform, ensuring efficient coordination among distributed energy resources. However, the system lacked remote access capabilities, limiting its applicability for distributed or urban settings where real-time monitoring from mobile devices is essential.

Kumar et al. (2020) proposed a fuzzy logic-based droop control strategy for battery energy storage systems in DC microgrids [2]. This approach improved voltage stability by dynamically adjusting power output based on load demand and source variability. The study demonstrated enhanced grid energy optimization for residential buildings with distributed energy resources. However, the reliance on complex hardware for fuzzy logic implementation increased system cost and complexity, making it less feasible for small-scale applications like urban EV charging stations.

Zhang et al. (2021) introduced an IoT-based battery monitoring system for DC microgrids, incorporating a communication channel to an Intelligent Electronic Device (IED), data acquisition, cloud storage, and a Human Machine Interface (HMI) [3]. The system embedded battery parameters (e.g., voltage, current, state of charge) into an IoT-enabled microcontroller, facilitating data storage and access via a cloud system. While innovative, the system relied on LCDs for data display, which restricted user interaction and lacked real-time fault alerts, reducing its effectiveness for dynamic microgrid environments.

Smith et al. (2022) presented a novel event-triggered distributed secondary control strategy for single-bus DC microgrids [4]. This approach used an event-triggering mechanism, allowing each converter to locally decide when to transmit signals to neighbouring converters, significantly reducing communication burden. The strategy improved system efficiency by minimizing unnecessary data exchanges. However, the study did not address user interfaces, leaving a gap in real-time monitoring and control for end-users, which is critical for urban microgrid applications.

Lee et al. (2023) developed a microgrid monitoring system using PIC microcontrollers interfaced with UART and an LCD display, programmed via MP Lab IDE [5]. The system included an IoT module to detect load status, energy consumption, and sensor values, with source code portability for device drivers. Despite its versatility, the system’s reliance on operation.

3.3 Research Gaps

The reviewed studies highlight significant advancements in microgrid automation, particularly in SCADA-based monitoring, IoT integration, and control strategies. However, several limitations persist across these systems, which the proposed IoT-based DC microgrid automation system aims to address:

1. **Lack of User-Friendly Interfaces:** Most systems, such as those by Zhang et al. [3] and Lee et al. [5], rely on LCDs for data display, which are cumbersome and limit user interaction. These interfaces are inadequate for real-time monitoring in dynamic environments, especially for urban applications requiring mobile access.
2. **Absence of Real-Time Fault Alerts:** Systems like those by Levi et al. [1] and Lee et al. [5] do not provide automated fault alerts, reducing their ability to respond promptly to overvoltage or undervoltage conditions. This is a critical drawback for ensuring microgrid reliability.
3. **Limited Remote Access:** The SCADA-based system by Levi et al. [1] and the control strategy by Smith et al. [4] lack remote monitoring capabilities, restricting their use in distributed or urban settings where users need access from remote locations.
4. **Complexity and Cost:** The fuzzy logic approach by Kumar et al. [2] requires complex hardware, increasing implementation costs and limiting scalability for small-scale microgrids.

The proposed system overcomes these gaps by integrating the Blynk application, an IoT platform that enables real-time monitoring, user-defined threshold settings, and fault alerts via a mobile interface. Unlike LCD-based systems, Blynk offers a scalable, user-friendly solution for remote access. The system automates load isolation during overvoltage (>15V) or undervoltage (<10V) conditions, addressing the lack of real-time fault response in existing works. By using cost-effective components like the ESP32 microcontroller and a 4-channel relay module, the system ensures affordability and scalability, making it suitable for urban applications such as EV charging stations. Furthermore, MATLAB Simulink simulations validate the system's performance, providing a robust framework for analyzing power flow and fault management, which is often absent in prior studies.

CHAPTER – 4

DESIGN ASPECTS

4.1 System Block Diagram

The block diagram illustrates the integration of power sources, sensors, ESP32, relays, and the Blynk server.

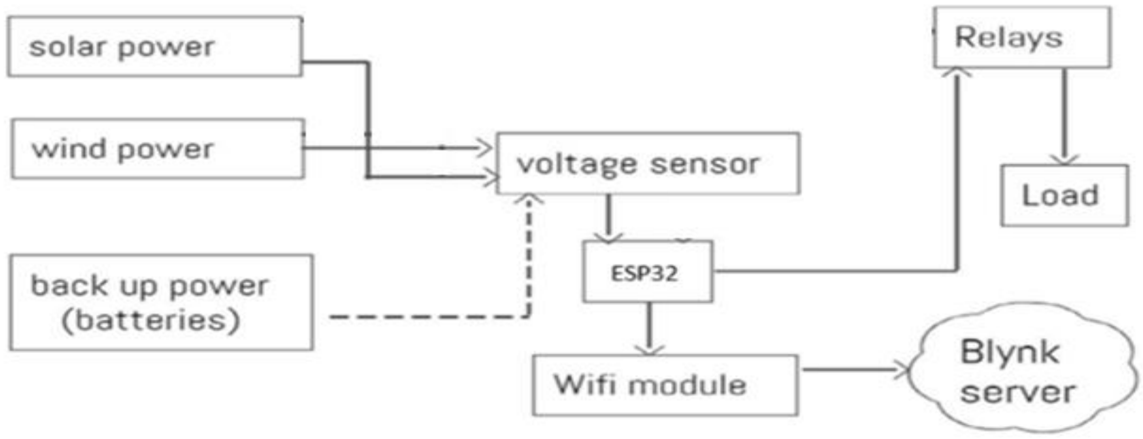


Fig.4.1 – Block Diagram of DC Microgrid

4.2 System Design

A hybrid power generation system consists of two renewable energy sources such as solar and wind (DC geared Motor). This increases the efficiency and power reliability of the system. Solar panel of rating 12V is used to capture solar energy and Hand driven 150RPM Dynamo is used to represent wind power. The battery backup is used to meet the load demand when power from solar and DC geared motor (representing wind) is not sufficient. Each source is provided with voltage sensor for purpose of collecting values of voltage at source end. The data is fed to ESP32 microcontroller. Current sensor is added at load end to measure current entering the load. Microcontroller is connected to Blynk server. The circuit diagram of developed system is shown in figure.

4.2.1 Design of Power Circuit

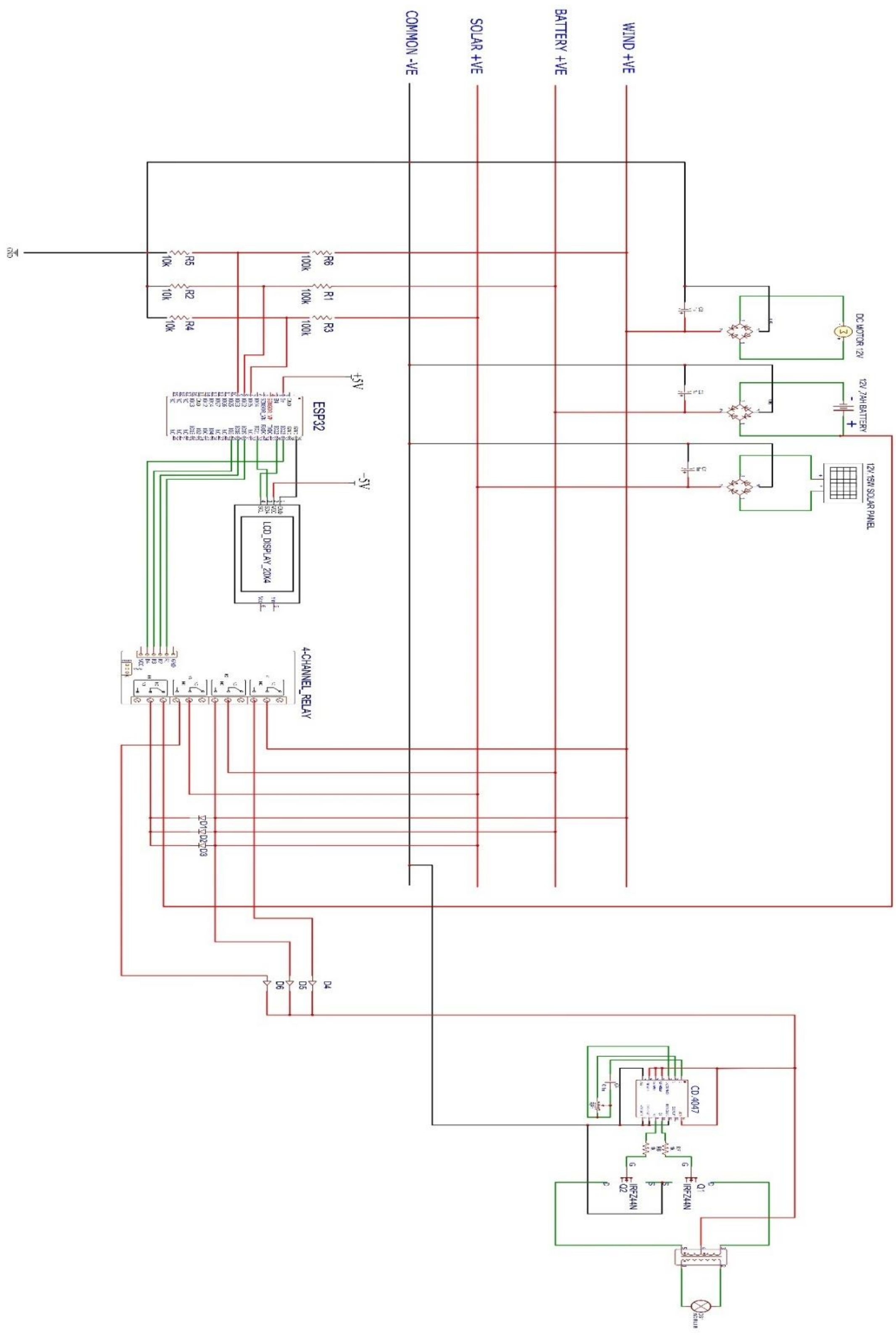


Fig.4.2 – Design of power Circuit of DC Microgrid

4.3 Embedded System

An embedded system is a dedicated computing platform for specific tasks. The ESP32 serves as the core controller.

4.3.1 Hardware Components

- **ESP32**: Dual-core SoC with 520 KB SRAM, Wi-Fi, and Bluetooth.
- **Sensors**: Voltage (25V max) and current (ACS712) sensors.
- **Relays**: 4-channel 5V module for load control.

4.3.2 Software Components

- **Arduino IDE**: Programs the ESP32.
- **Blynk Library**: Facilitates IoT communication.

4.3.3 Debugging Techniques

- **Serial Monitor**: Displays sensor data and system status.
- **JTAG Debugging**: For advanced troubleshooting.

4.3.4 Characteristics

- Application-specific: Monitors microgrid parameters.
- Real-time: Responds to faults within milliseconds.
- Low power: ESP32 consumes <5 μA in deep sleep.

4.4 ESP32 Microcontroller

4.4.1 Overview

The ESP32 is a low-cost SoC with 34 GPIO pins, supporting Wi-Fi, Bluetooth, and multiple communication protocols.

4.4.2 Powering the ESP32

- USB: 5V supply.
- VIN: 6-12V unregulated.
- 3.3V Pin: Direct 3.3V supply.

4.4.3 Input/Output Pins

- 34 digital pins (PWM, SPI, I2C, UART).
- 18 analog pins (12-bit ADC).

4.4.4 Special Functions

- Capacitive touch pins.
- RTC GPIOs for low-power modes.

ESP32 is created by Espressif Systems with a series of SoC (System on a Chip) and modules which are low cost with low power consumption.

This new ESP32 is the successor to the well-known ESP8266(became very popular with its inbuilt Wi-Fi). ESP32 not only has Built in Wi-Fi but also has Bluetooth and Bluetooth Low Energy. In other words, we can define ESP32 as “ESP8266 on Steroids”.

ESP32 chip ESP32-D0WDQ6 is based on a Tensilica Xtensa LX6 dual core microprocessor with an operating frequency of up to 240 MHz

The small ESP32 package has a high level of integrations such as:

- Antenna switches
- Balun to control RF
- Power amplifier
- Low noise reception amplifier
- Filters and power management modules

On top of all that, it achieves very low power consumption through power saving features including clock synchronization and multiple modes of operation. The ESP32 chip’s quiescent current is less than 5 μA which makes it the ideal tool for your battery powered projects or IoT applications.

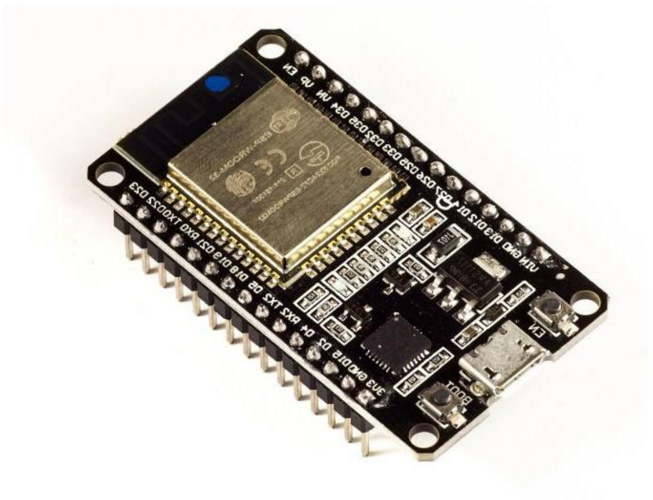


Fig.4.3 - ESP32 Microcontroller

4.4.5 Difference between ESP32 and ESP8266

As already mentioned above that **ESP32** is the successor of **ESP8266,** Lets learn what are the differences between ESP32 and ESP8266 with their features and specifications below.

- ESP32 has an additional core compared to ESP8266
- Faster **Wi-Fi**
- Increased number of GPIO (input/output) pins
- Compatibility with **Bluetooth 4.2** and **Bluetooth low energy** (*low energy*).

Additionally, the **ESP32** comes with touch-sensitive pins that can be used to "wake up" the **ESP32** from ***deep sleep*** **mode** and a built-in Hall effect sensor.

Although both boards are extremely cheap, the **ESP32** is slightly more expensive than ESP8266. ESP32 deserves it as it has more features than ESP8266.

We have differentiated the main technical specifications between the **ESP8266 and ESP32** in the below table.

4.4.6 Memory

In most of the microcontrollers based on Arduino, there are three types of memories:

- **Program memory:** to store the sketch.
- **SRAM memory**: to store the variables that are used in the code.
- **EEPROM memory:** to store variables that do not lose their value even when the device is turned off.

In ESP32 this does not happen, in fact there are more types of memories that are usually classified into internal and external.

The internal memories are those that are already included in the SoC, and the external are those that can be added to expand the capacity of the system.

Many ESP32- based development boards add external memory for a better performing system.

4.4.7 ESP32 Internal memories and their functions:

ROM memory (448 KiB): this memory is write-only, that is, you cannot reprogram it. This is where the codes that handle the Bluetooth stack, the Wi-Fi physical layer control, some general-purpose routines, and the bootloader to start the code from external memory are stored.

Internal SRAM memory (520 KiB): this memory is used by the processor to store both data and instructions. Its advantage is that it is much easier for the processor to access than the external SRAM.

RTC SRAM (16 KiB): this memory is used by the co-processor when the device operates in deep sleep mode.

Efuse (1 Kilobit): 256 bits of this memory are used by the system itself and the remaining 768 bits are reserved for other applications.

Flash embedded (Embedded flash): This memory is where our application code is stored. The amount of memory varies depending on the chip used:

- 0 MB (chips ESP32-D0WDQ6, ESP32-D0WD, ESP32-S0WD)
- 2 MB (chip ESP32-D2WD)
- 4 MB (Chip ESP32-PICO-D4)

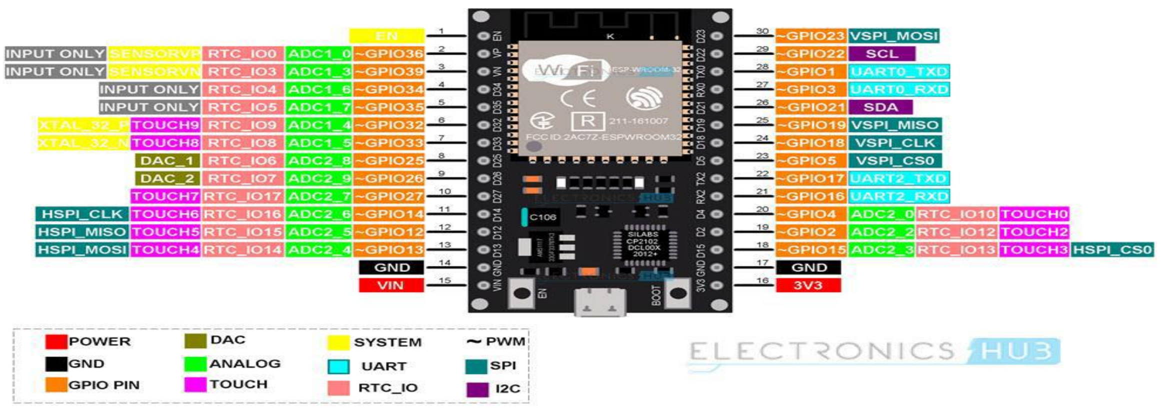
For ESP32s that do not have embedded memory or simply when memory is insufficient for your application, it is possible to add more memory externally:

- ➢ Up to 16 MB of external flash memory can be added. This way you can develop more complex applications.
- ➢ It also supports up to 8 MB of external SRAM memory.

Therefore, it is difficult for you to find yourself limited in memory when implementing an application using this platform.

4.5 Peripheral Features

The ESP32 has a great set of peripherals you can see from the above Block diagram.



EN 1
INPUT ONLY SENSORVP RTC_IO0 ADC1_0 ~GPIO36 2
INPUT ONLY SENSORVN RTC_IO3 ADC1_3 ~GPIO39 3
INPUT ONLY RTC_IO4 ADC1_6 ~GPIO34 4
INPUT ONLY RTC_IO5 ADC1_7 ~GPIO35 5
XTAL_32_P TOUCH9 RTC_IO9 ADC1_4 ~GPIO32 6
XTAL_32_N TOUCH8 RTC_IO8 ADC1_5 ~GPIO33 7
DAC_1 RTC_IO6 ADC2_8 ~GPIO25 8
DAC_2 RTC_IO7 ADC2_9 ~GPIO26 9
TOUCH7 RTC_IO17 ADC2_7 ~GPIO27 10
HSPI_CLK TOUCH6 RTC_IO16 ADC2_6 ~GPIO14 11
HSPI_MISO TOUCH5 RTC_IO15 ADC2_5 ~GPIO12 12
HSPI_MOSI TOUCH4 RTC_IO14 ADC2_4 ~GPIO13 13
GND 14
VIN 15
30 ~GPIO23 VSPI_MOSI
29 ~GPIO22 SCL
28 ~GPIO1 UART0_TXD
27 ~GPIO3 UART0_RXD
26 ~GPIO21 SDA
25 ~GPIO19 VSPI_MISO
24 ~GPIO18 VSPI_CLK
23 ~GPIO5 VSPI_CS0
22 ~GPIO17 UART2_TXD
21 ~GPIO16 UART2_RXD
20 ~GPIO4 ADC2_0 RTC_IO10 TOUCH0
19 ~GPIO2 ADC2_2 RTC_IO12 TOUCH2
18 ~GPIO15 ADC2_3 RTC_IO13 TOUCH3 HSPI_CS0
17 GND
16 3V3
EN
BOOT
POWER
GND
GPIO PIN
DAC
ANALOG
TOUCH
SYSTEM
UART
RTC_IO
~ PWM
SPI
I2C
ELECTRONICS HUB

Fig.4.4- ESP32 Pinout diagram and Pins

Pin Category	**Pin Name**	**Details**
Power	3.3V, 5V, GND	3.3V/5V supply, ground pins
Analog	ADC1_0 to ADC2_5	12-bit ADC, 0-3.3V range
Digital	GPIO0-GPIO39	Input/output, PWM, interrupts
Communication	UART, SPI, I2C	Serial, SPI, and I2C interfaces

Table 4-1 ESP32 Pin Configuration

As you can see from the above image of **ESP32 WROOM module pinout diagram**, all the different types of pins are mentioned in different colours which we are going to explain in detail below.

4.5.1 Digital pins

The ESP32 has a total of 34 digital pins. These pins are similar to Arduino digital pins which allows you to add LED display, OLED display, sensors, buttons, buzzers, etc. to our projects. Most of these pins support the use of internal pull-up, pull-down, and high impedance status as well. This makes them ideal for connecting buttons and matrix keyboards, as well as for applying LED control techniques such as the well-known Charlieplexing.
ESP32 WROOM module has 25 GPIO pins out of which there are only input pins, pins with input pull up and pins without internal pullup.
Maximum current drawn per a single GPIO is 40mA according to the “Recommended Operating Conditions” section in the ESP32 datasheet.

Input only pins:

- GPIO 34
- GPIO 35
- GPIO 36
- GPIO 39

Pins with pull up INPUT_PULLUP

- GPIO14
- GPIO16
- GPIO17
- GPIO18
- GPIO19
- GPIO21
- GPIO22
- GPIO23

Pins without internal pull up

- GPIO13
- GPIO25
- GPIO26
- GPIO27
- GPIO32
- GPIO33

4.5.2 ADC (Analog to digital converters)

Some of the pins listed in the pinout diagram can also be used to interact with analog sensors, same as analog pins of an Arduino board.

For this, the ESP32 has a 12-bits (0-4096 resolution which means when voltage observed is 0 the value is 0 and when max voltage like 3.3v is observed the value goes to 4096), 18-channel analog to digital converter, which means you can take readings from up to 18 analog sensors.

This allows you to develop very compact connected applications, even when using multiple analog sensors.

Analog input pins:

- ADC1_CH0 (GPIO 36)
- ADC1_CH1 (GPIO 37)
- ADC1_CH2 (GPIO 38)
- ADC1_CH3 (GPIO 39)

- ADC1_CH4 (GPIO 32)
- ADC1_CH5 (GPIO 33)
- ADC1_CH6 (GPIO 34)
- ADC1_CH7 (GPIO 35)
- ADC2_CH0 (GPIO 4)
- ADC2_CH1 (GPIO 0)
- ADC2_CH2 (GPIO 2)
- ADC2_CH3 (GPIO 15)
- ADC2_CH4 (GPIO 13)
- ADC2_CH5 (GPIO 12)
- ADC2_CH6 (GPIO 14)
- ADC2_CH7 (GPIO 27)
- ADC2_CH8 (GPIO 25)
- ADC2_CH9 (GPIO 26)

4.5.3 DAC (Digital to Analog Converters)

PWM signals are used on most Arduino boards to generate analog voltages. The ESP32 has two 8 bits digital to analog converters.

This allows two pure analog voltage signals to be generated. These converters can be used to:

- Control an analog circuit
- Manipulate the intensity of an LED
- Can even add a small amp and speaker to your project to play a song.

4.5.4 DAC Pins:

- DAC1 (GPIO25)
- DAC2 (GPIO26)

4.5.5 Capacitive Touch GPIOs

In case if you want to develop applications with **no mechanical buttons**, you can use the touch sensitive pins on ESP32s to achieve it.

These pins are capable of detecting the small variations produced when approaching a finger to the pin. In this way, it is possible to create all kinds of controls such as buttons or slide bars without the need for mechanical components.

Capacitive Touch pins:

- T0 (GPIO 4)
- T1 (GPIO 0)
- T2 (GPIO 2)
- T3 (GPIO 15)
- T4 (GPIO 13)
- T5 (GPIO 12)
- T6 (GPIO 14)
- T7 (GPIO 27)
- T8 (GPIO 33)
- T9 (GPIO 32)

4.5.6 RTC

As we already learnt about the RTC GPIO support in the core section. The GPIOs which are routed to the **RTC low-power management subsystem** can be used when the ESP32 is in deep sleep. These RTC GPIOs can be used to wake up the ESP32 from deep sleep when the **Ultra Low Power (ULP) co-processor** is running. The following GPIOs can be used as an external wake up source.

- RTC_GPIO0 (GPIO36)
- RTC_GPIO3 (GPIO39)
- RTC_GPIO4 (GPIO34)
- RTC_GPIO5 (GPIO35)
- RTC_GPIO6 (GPIO25)
- RTC_GPIO7 (GPIO26)
- RTC_GPIO8 (GPIO33)
- RTC_GPIO9 (GPIO32)
- RTC_GPIO10 (GPIO4)
- RTC_GPIO11 (GPIO0)
- RTC_GPIO12 (GPIO2)

- RTC_GPIO13 (GPIO15)
- RTC_GPIO14 (GPIO13)
- RTC_GPIO15 (GPIO12)
- RTC_GPIO16 (GPIO14)
- RTC_GPIO17 (GPIO27)

SD / SDIO / MMC driver

This peripheral allows the ESP32 to interact with SD and MMC cards directly. In fact, by combining this controller with the analog digital converter it is possible to improve our little audio player.

4.5.7 Communication Protocols

UART

Many microcontrollers have UART modules, which on Arduino are known as Serial ports. These allow asynchronous communications between two devices using only two pins.

The ESP32 has three UART ports:

- UART0
- UART1
- UART2

All of these are compatible with RS-232, RS-485 and IrDA protocols.

I2C

The ESP32 have two interfaces I2C or TWI that support the operating modes master and slave. Its features include:

- Standard mode (100 Kbit/s)
- Fast mode (400 Kbit/s)
- 7 and 10 bits addressing

I2C Pins

- GPIO 21 (SDA)
- GPIO 22 (SCL)

SPI

The ESP32 also has SPI communication. It has three fully functional buses:

Four transfer modes: this means that it is compatible with all or almost all SPI and QSPI devices available on the market.

- All SPI ports are capable of high speeds (theoretically up to 80 MHz).
- 64-byte buffer for transmission and reception.

By default the pin mapping SPI is:

SPI	**MOSI**	**MISO**	**CLK**
VSPI	GPIO 23	GPIO 19	GPIO 18
HSPI	GPIO 13	GPIO 12	GPIO 14

4.6 Blynk Application

4.6.1 Features

- Real-time data visualization.
- User-defined thresholds.
- Push notifications for faults.

Blynk is a low-code IoT platform that allows users to build and customize mobile apps to remotely control and monitor connected devices, without writing any code. It provides a drag-and-drop interface for creating apps, along with a web console for managing devices, users, and data.

Key Features:

- **Low-Code IoT Platform:**

Blynk simplifies IoT development by enabling users to create apps without extensive coding knowledge.

- **Mobile App Builder:**

The Blynk app allows users to design and customize mobile interfaces for controlling and monitoring connected devices.

- **Remote Control and Monitoring:**

Users can remotely control and monitor devices from anywhere in the world.

- **Data Visualization:**

Blynk allows users to visualize data from connected devices in real-time.

- **Scalability:**

Blynk is designed to support IoT projects of all sizes, from personal prototypes to commercial deployments**.**

- **Security:**

Blynk offers enterprise-grade security features, including encrypted communication and user authentication.

Over-the-Air (OTA) Updates:

The platform allows for easy firmware updates of connected devices remotely.

- **Automations:**

Blynk enables users to create automated scenarios based on device state or user actions.

4.6.2 Integration with ESP32

The ESP32 connects to the Blynk server via Wi-Fi, sending sensor data and receiving control commands

1. **Connect Hardware:**

Connect your devices to the Blynk platform using the Blynk library for your specific hardware platform (e.g., Arduino, ESP32).

2. **Design Your App:**

Use the Blynk app builder to create a mobile interface with widgets for controlling and monitoring your devices.

3. **Connect to the Cloud:**

The Blynk app communicates with the Blynk server, which in turn communicates with your connected devices.

4. **Remote Control and Monitoring:**

Control and monitor your devices remotely from your mobile app or the Blynk web console.
In summary, Blynk provides a comprehensive solution for building and managing IoT projects, offering a user-friendly platform for creating connected products and applications.

Fig.4.5 – Blynk application in Mobile

4.7 Key Components

4.7.1 Solar Panel

- **Specification**: 12V, 15W, p-type/n-type silicon cells.
- **Function**: Primary power source.

Solar panels are used for the harnessing of solar energy. Solar panels are made up of numerous solar cells that are linked in a series of parallel lines. P-type and n-type silicon are the two types of semiconductors used in solar cells. Electrons in the silicon are ejected when sunlight strikes a solar cell. If this happens in the electric field, then the electrons will travel from the n-type layer to the p-type layer through the external wire creating a flow of electricity. The solar power then generated can be stored for future use. 15W, 12V rated solar panel is used in our prototype to supply power to a 12V DC load.

Fig.4.6 – Solar Panel

4.7.2 DC Geared Motor

- **Specification**: 150 RPM, 12V, high-torque gearbox.
- **Function**: Simulates wind energy.

DC gear motor is a type of electric motor that combines a standard DC motor with a gearbox. This integration allows for increased torque output and reduced rational speed, which makes it suitable for applications that require high torque and precise output.

DC gear motor consists of a permanent magnet or electromagnet stator and a mature wound with the coil. When an electric current flows through the armature coils, it creates a magnetic field that interacts with the stator and generates torque that causes the armature to rotate.

The gearbox attached to the DC motor contains a series of gears that convert the motor's high speed and low torque output to low speed and high torque output.

These gears can be of various types such as spur gears, planetary gears, and helical gears, each offering different benefits in terms of efficiency, noise reduction, and load capacity.

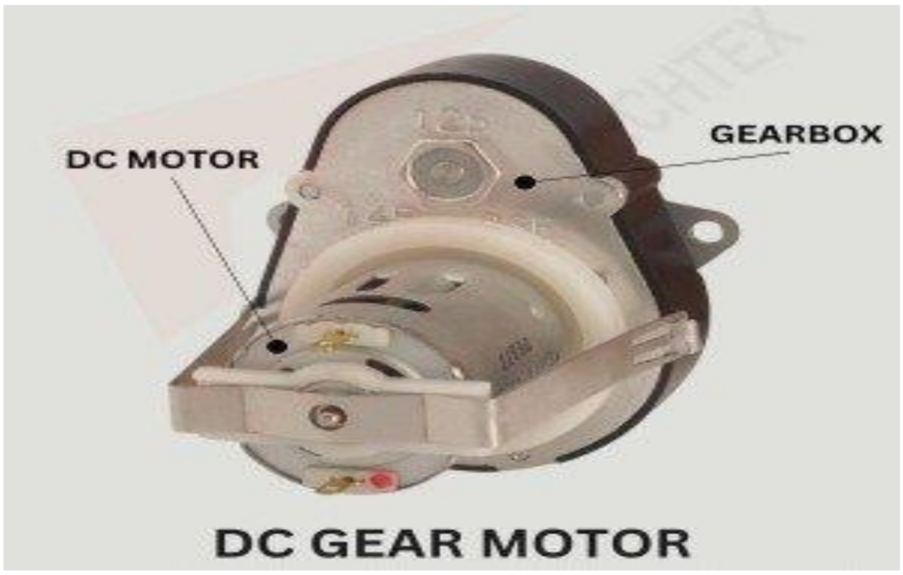


Fig.4.7 – DC Gear Motor

Working Of DC Gear Motor

A DC gear motor is a combination of a DC motor and a gearbox. It works on the principle of electromagnetic induction, where a current-carrying conductor placed in magnetic fields, experiences a force and causes the rotation of the motor. The gearbox is used to increase or decrease the speed of the motor depending on application requirements.

The DC motor consists of two main components: stator and rotor. The stator consists of permanent magnets or electromagnets that create a stationary magnetic field. At the same time, the rotor consists of an armature coil that carries electricity. The gearbox is connected to the shaft of the DC motor. It consists of a series of gears that transmit the rotational motion from the motor to its output shaft.

When the power is supplied to the DC motor, the armature coil energises and creates a magnetic field. The stationary magnetic of the stator interacts with the magnetic field of the armature coil and causes the rotor to rotate. Then the rotational motion of the motor is transmitted to the gearbox.

The gears in the gearbox either increase or decrease the speed and torque of the DC motor. Then the output shaft of the gearbox rotates at the desired speed and the torque as per the application's requirement.

Applications of DC Gear Motor

DC gear motors are used in a wide range of applications across various industries due to their ability to provide precise control over the speed and torque.

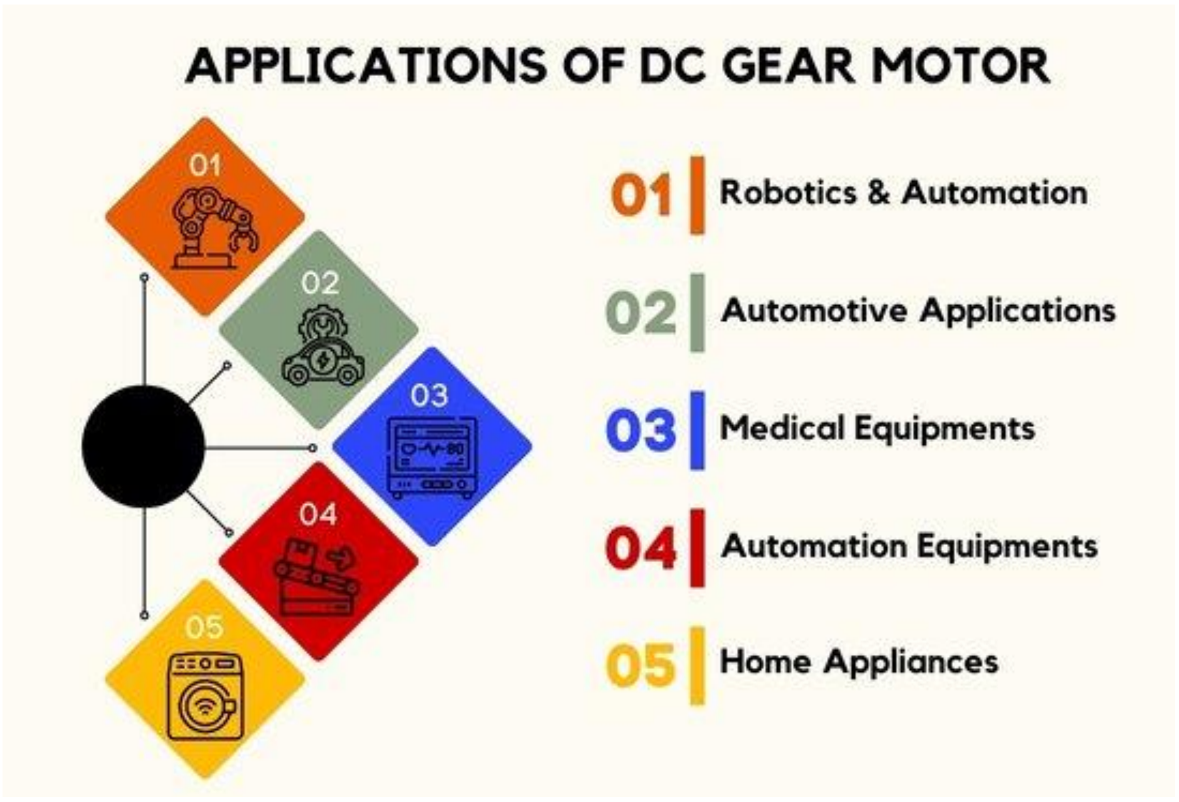


Fig.4.8 – Applications of DC Gear Motor

Here are some common applications of DC gear motors:

- **Robotics and Automation**

DC gear motors are used in robotics systems for driving robotic arms, grippers, and other equipment that require precise speed control. They are also used in automated assembly lines, pick and place machines, and material handling equipment.

- **Automotive Applications**

DC gear motors are used in power windows, sunproof, and other automotive accessories for controlling their movement. They are also used in various automotive systems such as windshield wipers and power seat adjustments.

- **Medical Equipment**

DC gear motors are used in medical equipment such as hospital beds, wheelchairs, and rehabilitation equipment for positioning and adjustment of patients.

- **Automation Equipment**

DC gear motors are used in automated doors, gates, and window openers for controlling the opening and closing mechanism. They are also used in various industrial automation equipment such as conveyor belts, packaging machines, and sorting systems.

- **Home Appliances**

DC gear motors are found in household appliances such as washing machines, dishwashers, and vacuum cleaners to drive their mechanism.

4.7.3 Voltage and Current Sensors

- **Voltage Sensor**: Measures up to 25V using voltage divider.

A voltage sensor is an electronic device used to measure the electrical potential difference between two points in a circuit. It converts this voltage into a readable signal, typically analog or digital, that can be processed by microcontrollers or data acquisition systems. One of the most common methods for sensing voltage is through a voltage divider circuit, which uses two resistors to scale down a higher voltage to a safer, measurable level. For more accurate or isolated measurements, especially in AC circuits or high-voltage environments, specialized sensors like the ZMPT101B or optocoupler-based sensors are used. These provide features such as signal amplification, noise filtering, and electrical isolation. Voltage sensors play a crucial role in applications like power supply monitoring, battery management systems, energy meters, and automation systems, helping ensure stability, efficiency, and safety in electrical and electronic setups.

- **Current Sensor**: ACS712, ±5A range.

A current sensor is a device used to measure the flow of electric current in a circuit, either AC or DC, and typically outputs a voltage signal that can be read by microcontrollers or monitoring equipment. Common types include shunt resistor-based sensors, which measure voltage drop across a known resistor using Ohm's law, and Hall effect sensors, which detect the magnetic field generated by current and are capable of measuring high currents without direct contact. Another type, the Rogowski coil, is used for high-speed AC current measurements in high-voltage applications. On the other hand, a voltage sensor measures the electrical potential difference between two points and converts it into a form readable by electronic systems. Basic voltage sensing is often done using a resistor voltage divider circuit, while more advanced sensors like the ZMPT101B for AC voltage or optocoupler-based sensors offer isolation and safety for high-voltage applications. Both current and voltage sensors are widely used in systems such as battery management, motor control, energy monitoring, and protection circuits to ensure efficient and safe operation.

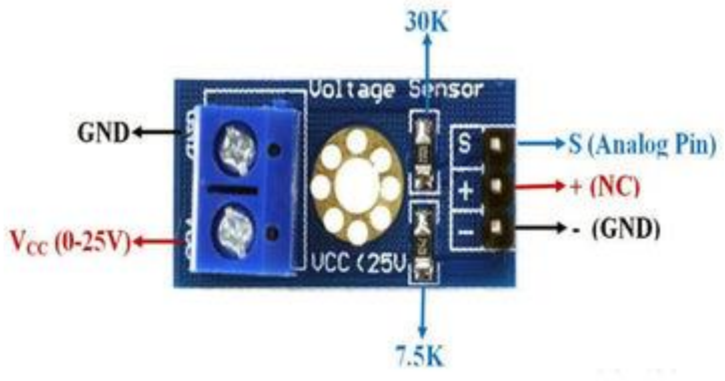


Fig.4.9 – Current Sensor

4.7.4 4-Channel Relay Module

- **Specification**: 5V, 10A at 250V AC/30V DC.
- **Function**: Load switching/isolation.

5V Relay Module 4 Channel is an indispensable tool for any electronics enthusiast or automation expert. This module provides seamless control over multiple devices, offering a reliable and efficient switching mechanism.

Whether you're building home automation systems, robotics projects, IoT applications, or industrial setups, this 5V relay module delivers outstanding performance and versatility.
Operating at 5V, this relay module is compatible with a wide range of microcontrollers, Arduino boards, Raspberry Pi, and other development platforms. Its compact design and straightforward interface make it easy to integrate into various projects without complex wiring or programming. Each channel on the relay module can handle a separate device, providing independent control for increased flexibility. With its low power consumption and high switching capacity, this module ensures efficient energy usage and reliable performance.
Furthermore, the 5V Relay Module 4 Channel incorporates safety features to protect connected devices from voltage fluctuations and surges, ensuring the longevity of your valuable equipment. Take advantage of this cost-effective and efficient 5V Relay Module 4 Channel to enhance your projects and achieve seamless device control like never before.

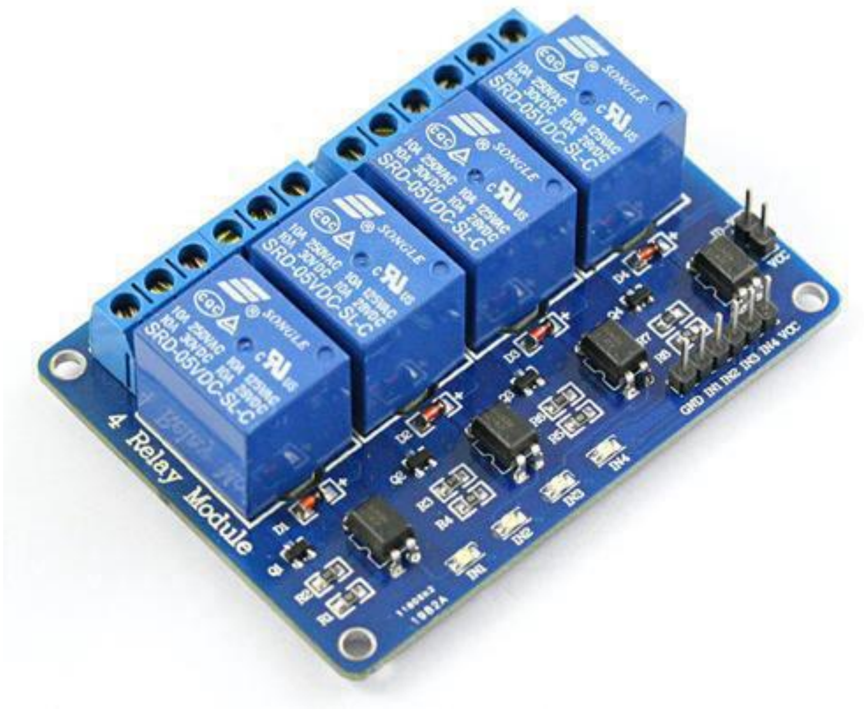


Fig.4.10- 4 Channel Relay Module

This is ADIY 4 Channel Isolated 5V 10A Relay Module, A wide range of microcontrollers such as Arduino, AVR, PIC, ARM and so on can control it.It is also able to control various appliances and other types of equipment with large current.

Relay output maximum contact is AC250V 10A and DC5V 10A. One can connect a microcontroller with standard interface directly to it.Red working status indicator lights are conducive to the safe use.

It has a wide range of applications such as all MCU control, industrial sector, PLC control, smart home control. Ready to get switching on your Raspberry Pi?!

This neat relay module features 4 x 5V relays rated at 10A/250V each. It is designed to switch up to 4 high current (10A) or high voltage (250V) loads with the help of microcontroller! Each relay can individually switch on/off by an opto-isolated digital input, which that can connect directly to a microcontroller output pin. It only requires a voltage of approx. 1.0V to switch the inputs on but can handle input voltages up to 5V. This makes it ideal for 1.0V to 5V devices.

Pin Description

Pin	**Details**
VCC	5V supply
GND	Ground
IN1-IN4	Control inputs
Output	Load connection

Table 4-2 Relay Module Pin Configuration

Product Features: 5V Relay module 4 channel

- 5V input for broad compatibility
- Four channels for independent device control
- Low power consumption for energy efficiency
- High switching capacity for various applications
- Compact and easy-to-use design
- Built-in protection against voltage spikes
- Ideal for Arduino, Raspberry Pi, and other microcontrollers

Specifications: 5V Relay module 4 channel

- Input Voltage: 5V DC
- One normally closed contact and one normally open contact

- Channel: 4 channel
- Relay Operating Voltage: 3.3V to 5V
- Triode drive, increasing relay coil
- High impedance controller pin
- Pull-down circuit for avoidance of malfunction
- Power supply indicator and Control indicator lamp
- Power supply and relay instructions, lit, disconnect is off;
- Input signal, signal, common Terminal and start conducting;
- Useful for appliance control;
- DC or AC signal, control, you can control the 220V AC load;
- There is a normally open and one normally close contact;
- The module is compliant with international safety standards, control and load areas isolation trench

Applications: 5V Relay module 4 channel

- Home Automation Systems
- Robotics and DIY Projects
- Internet of Things (IoT) Applications
- Industrial Automation
- Smart Home Devices
- Security Systems
- Lighting Control Systems

4.7.5 CD4047 IC

- **Function**: Timing and control in monostable/astable mode.
- **Specification**: 3-18V supply, low power.

The CD4047 is a **high voltage multivibrator IC**, that can operate in both Monostable and Astable mode. The IC requires an external resistor and capacitor to set the width of output pulse in monostable mode and the frequency of output pulse in Astable Mode.
The **monostable mode** has positive and negative edge trigger with programmable pulse width. The **Astable mode** provides good frequency stability at 50% duty cycle. The CD4047 IC can operate on 5V, 10V, 15V or even at 20V.

The output of the IC always comes in TTL which makes it easy to work with other TTL devices and **microcontrollers**.

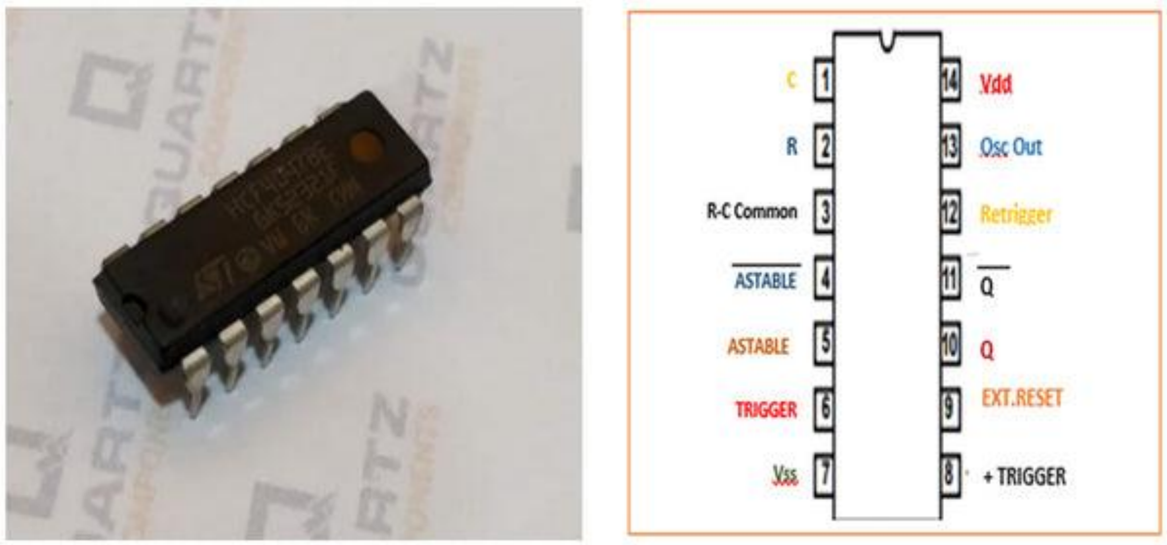
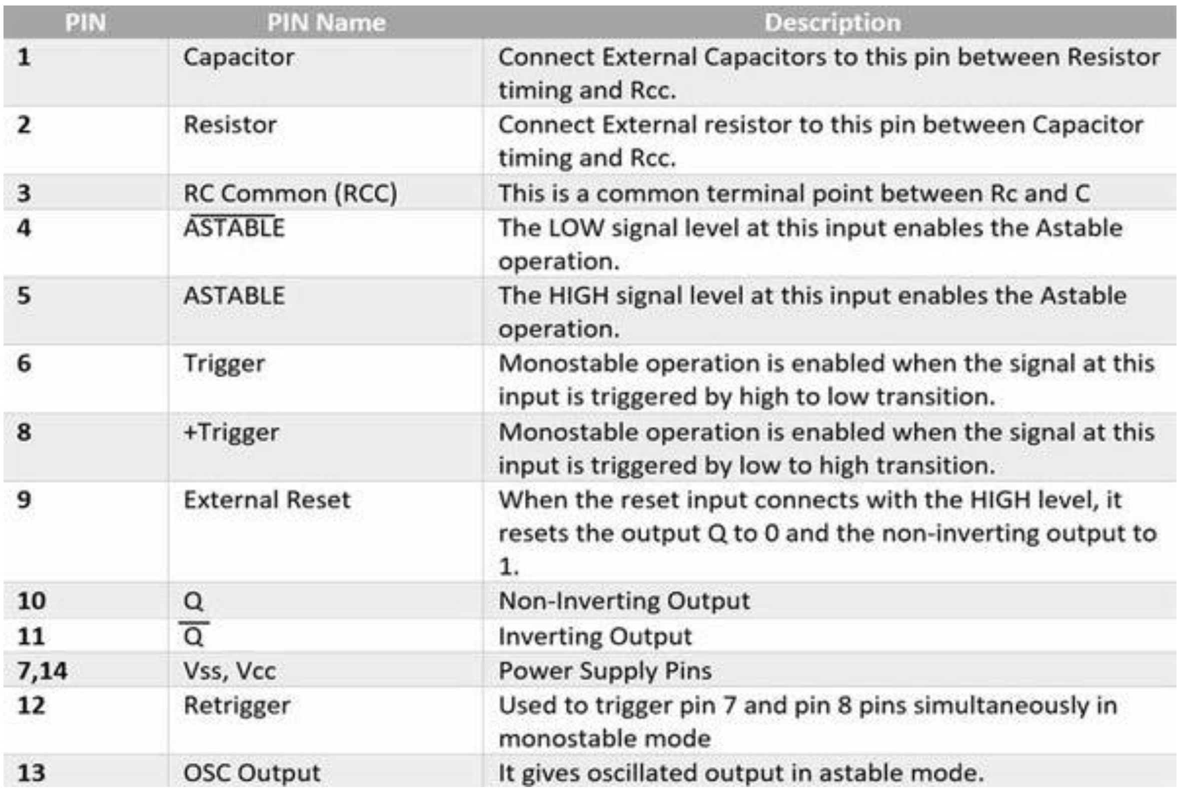


Fig.14- CD4047 Pin Diagram

PIN DESCRIPTIONS

Alternative to the CD4047 ICs, are the 555 Timer and the LM556/ dual timer IC.



PIN	PIN Name	Description
1	Capacitor	Connect External Capacitors to this pin between Resistor timing and Rcc.
2	Resistor	Connect External resistor to this pin between Capacitor timing and Rcc.
3	RC Common (RCC)	This is a common terminal point between Rc and C
4	$\overline{\text{ASTABLE}}$	The LOW signal level at this input enables the Astable operation.
5	ASTABLE	The HIGH signal level at this input enables the Astable operation.
6	Trigger	Monostable operation is enabled when the signal at this input is triggered by high to low transition.
8	+Trigger	Monostable operation is enabled when the signal at this input is triggered by low to high transition.
9	External Reset	When the reset input connects with the HIGH level, it resets the output Q to 0 and the non-inverting output to 1.
10	Q	Non-Inverting Output
11	$\overline{Q}$	Inverting Output
7,14	Vss, Vcc	Power Supply Pins
12	Retrigger	Used to trigger pin 7 and pin 8 pins simultaneously in monostable mode
13	OSC Output	It gives oscillated output in astable mode.

Features of CD4047 - Monostable/Astable Multivibrator IC

- Wide Supply Voltage: 3.0V to 15V
- High Noise Immunity
- Buffered inputs
- Requires only one external Resistor and Capacitor
- Offers both monostable and astable mode operations
- Low power consumption
- Monostable Features: Duty-cycle approaches 100%, Retrigger option for pulse width expansion, positive/ negative edge trigger

Astable Features: 50% duty-cycle, Oscillator output available, good astable frequency stability

Component	**Specification**
Solar Panel	12V, 15W
DC Geared Motor	150 RPM, 12V
Voltage Sensor	0-25V
Current Sensor	±5A
ESP32	520 KB SRAM, Wi-Fi
Relay Module	5V, 10A

Table 4-3 Component Specifications

4.7.6 Resistors:

Resistors are passive elements that **oppose/restrict the flow of current.**

A voltage is developed across its terminal, proportional to the current through the resistor.

$$V = IR$$

Units: Ohms (Ω)

behave like a tiny rechargeable battery. (**store energy and release it later.)** are **made** of two parallel conductors separated by a dielectric are used for **filtering, tuning, separating signals, etc.**

The ability of a capacitor to store charge is called “**Capacitance**”

C = Q/V (amount of charge stored/applied voltage)

The unit of capacitance is the **Farad.** Commonly used capacitances are much smaller than 1 Farad, micro-Farads (10^{-6} Farad, **µF**), nano-Farads (10^{-9} Farad, **nF)**, pico-Farads (10^{-12} Farad, **pF**).

4.7.7 Wires and Jumper Cables:

A jumper wire is an electric wire that connects remote electric circuits used for printed circuit boards. By attaching a jumper wire on the circuit, it can be short-circuited and short-cut (jump) to the electric circuit.

By placing the jumper wire on the circuit, it becomes possible to control the electricity, stop the operation of the circuit, and operate a circuit that does not operate with ordinary wiring. Also, when specification change or design change is necessary on the printed circuit board, reinforcement of the defective part, partial stop of the unnecessary function, and change of the circuit configuration of the unnecessary output part by attaching or detaching the jumper wire can do.

SHOW A jumper wire (NSL: New Showa Lead) is a lead-free tin-plated annealed copper wire. Tin plating is tin: 99.2%, copper: 0.8%.

In general, it is said that hot plating is difficult to control the plating thickness compared with electroplating, but we control the plating thickness by the original processing method. It also supports various environmental surveys such as RoHS Directive and REACH. These will facilitate connections between components on the breadboard or PCB.

4.7.8 LCD 16×2 Pin Configuration and Its Working

Nowadays, we always use the devices which are made up of LCDs such as CD players, DVD players, digital watches, computers, etc. These are commonly used in the screen industries to replace the utilization of CRTs. Cathode Ray Tubes use huge power when compared with LCDs, and CRTs heavier as well as bigger. These devices are thinner as well power consumption is extremely less. The LCD 16×2 working principle is, it blocks the light rather than dissipate. This article discusses an overview of LCD 16X2, pin configuration and its working.

What is the LCD 16×2?

The term LCD stands for liquid crystal display. It is one kind of electronic display module used in an extensive range of applications like various circuits & devices like mobile phones, calculators, computers, TV sets, etc.

These displays are mainly preferred for multi-segment light-emitting diodes and seven segments. The main benefits of using this module are inexpensive; simply programmable, animations, and there are no limitations for displaying custom characters, special and even animations, etc.

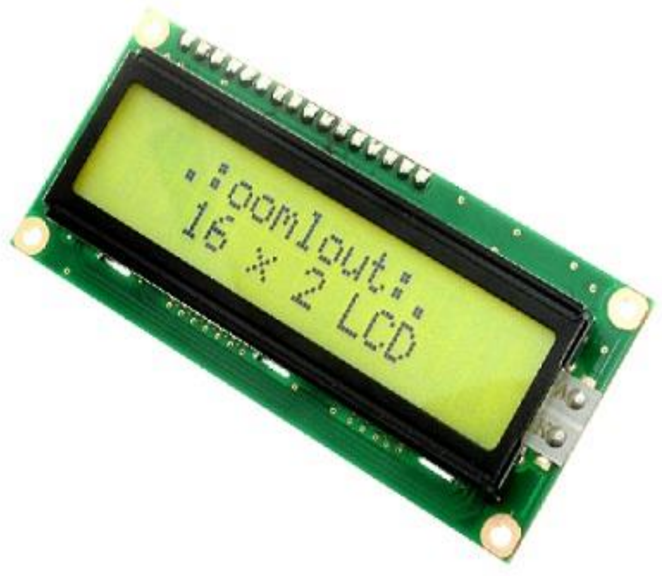


Fig.15- 16X2 LCD

LCD 16×2 Pin Diagram

The 16×2 LCD pinout is shown below.

- Pin1 (Ground/Source Pin): This is a GND pin of display, used to connect the GND terminal of the microcontroller unit or power source.
- Pin2 (VCC/Source Pin): This is the voltage supply pin of the display, used to connect the supply pin of the power source.
- Pin3 (V0/VEE/Control Pin): This pin regulates the difference of the display, used to connect a changeable POT that can supply 0 to 5V.

- Pin4 (Register Select/Control Pin): This pin toggles among command or data register, used to connect a microcontroller unit pin and obtains either 0 or 1(0 = data mode, and 1 = command mode).
- Pin5 (Read/Write/Control Pin): This pin toggles the display among the read or writes operation, and it is connected to a microcontroller unit pin to get either 0 or 1 (0 = Write Operation, and 1 = Read Operation).
- Pin 6 (Enable/Control Pin): This pin should be held high to execute Read/Write process, and it is connected to the microcontroller unit & constantly held high.
- Pins 7-14 (Data Pins): These pins are used to send data to the display. These pins are connected in two-wire modes like 4-wire mode and 8-wire mode. In 4-wire mode, only four pins are connected to the microcontroller unit like 0 to 3, whereas in 8-wire mode, 8-pins are connected to microcontroller unit like 0 to 7.
- Pin15 (+ve pin of the LED): This pin is connected to +5V
- Pin 16 (-ve pin of the LED): This pin is connected to GND.

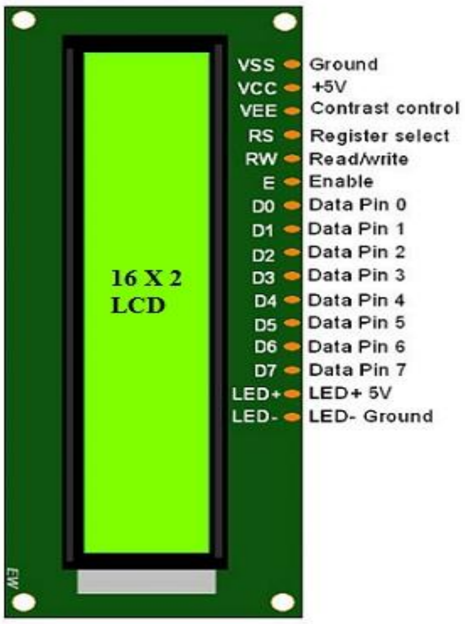
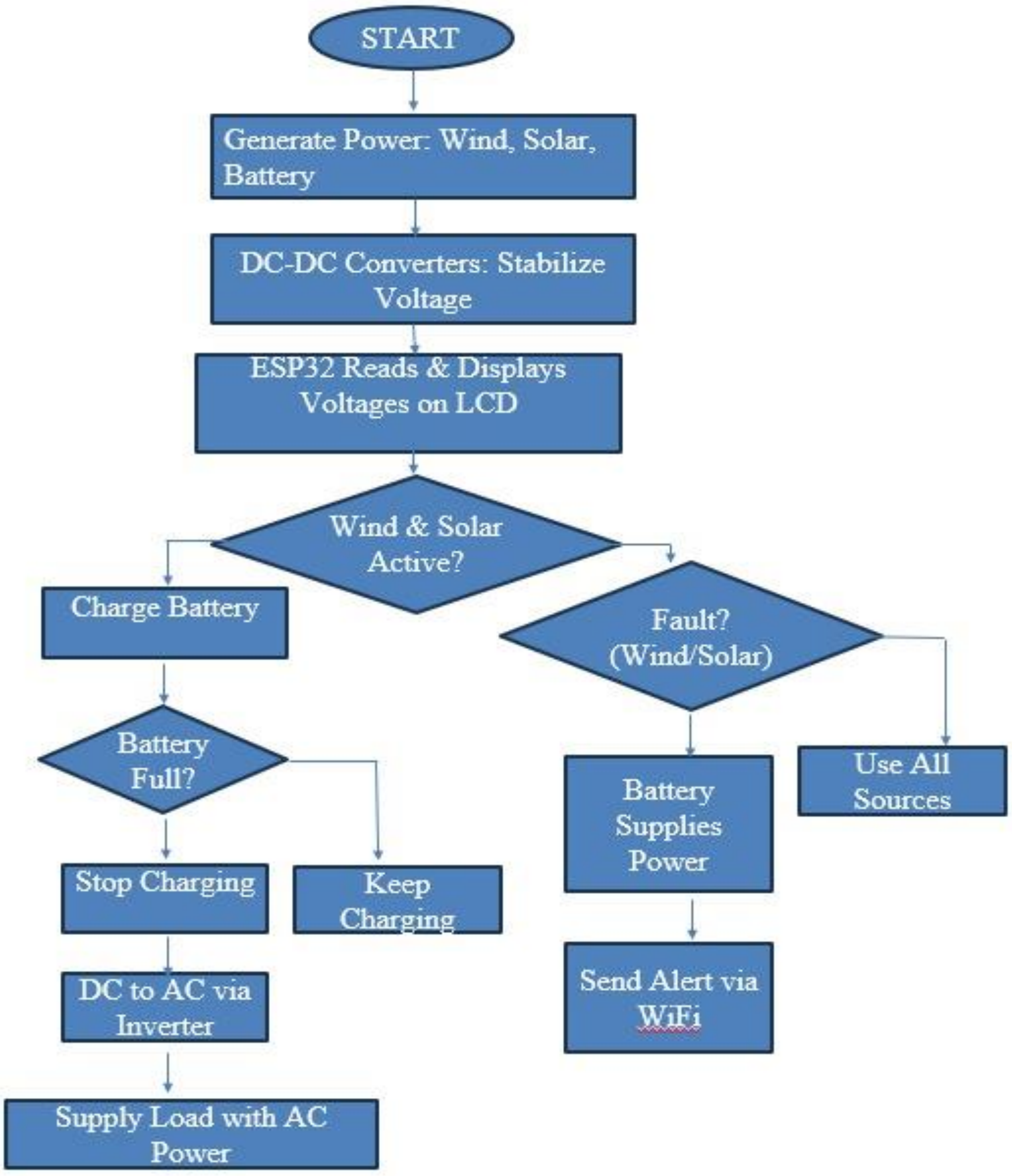


Fig16- LCD-16×2-pin-diagram

CHAPTER 5
IMPLEMENTATION ASPECTS

5.1 Hardware Implementation



5.1 Flow Chart

5.1.1 Component Assembly

Components were assembled on a prototyping board, with the ESP32 as the central controller. The solar panel and DC geared motor were connected to the DC bus via converters.

5.1.2 Circuit Connections

- **Sensors:** The solar, battery, and wind voltage sensors are connected to the ESP32’s ADC pins GPIO 33, GPIO 35, and GPIO 32 respectively, using voltage divider circuits to scale down the voltage levels.

- **Relays:** The relays for controlling power sources are connected to the ESP32's GPIO pins — solar to GPIO 19, battery to GPIO 18, wind to GPIO 5, and discharge relay to GPIO 23. These pins are configured as digital outputs.

- **LCD Display:** The 20x4 I2C LCD is connected to the ESP32 via I2C communication with SDA on GPIO 21 and SCL on GPIO 22.

- **Power Supply:** The ESP32 is powered by a regulated 5V supply, while the solar, wind, and battery sources operate at 12V. All components share a common ground for stable operation.

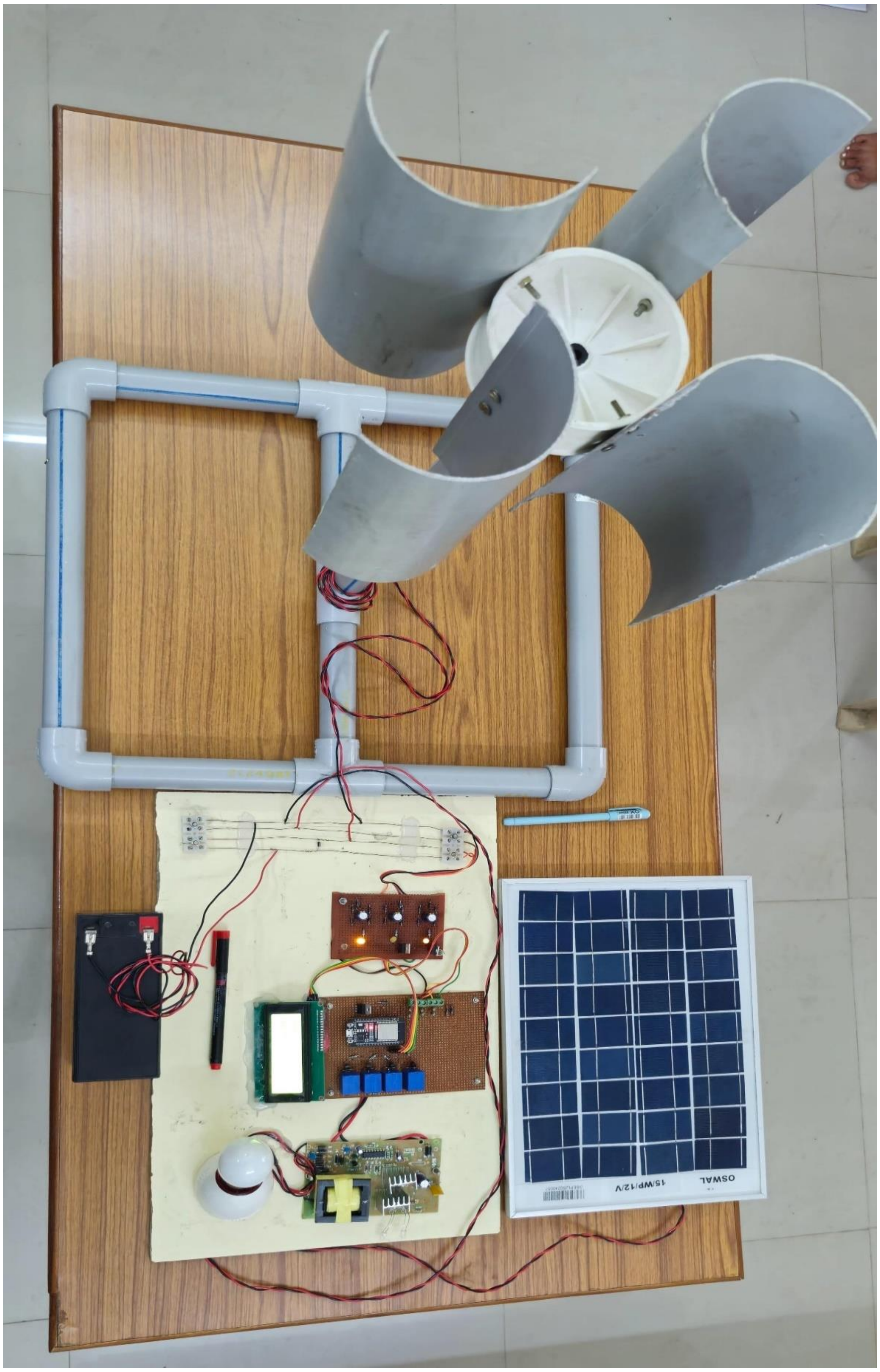


Fig.5.2 DC Microgrid Integration with Battery Storage System

5.2 Software Implementation

5.2.1 Arduino IDE Installation

1. Download Arduino IDE from the official website.
2. Install ESP32 board support.
3. Add Blynk library via Library Manager.

5.2.2 Programming Structure

5.2 Arduino Programming:

Arduino programs can be divided in three main parts: **Structure**, **Values (variables and constants)**, and **Functions**.

5.2.1 Structure

Let us start with the Structure. Software structure consists of two main functions: divided into three parts. They are structure, values.
The basic structure of Arduino programming language is simple and runs in two parts.

These two parts, or functions, enclose blocks of statements.

cpp

Copy Edit

```
Void setup ()

{

  Statements;

}



Void loop ()

{

}
```

Where setup() is the preparation, loop() is the execution. Both functions are required for the program to work. The setup function should follow the declaration of any variables at the beginning of the program. It is the first function to run the program, is run only once and is used.

The loop function follows next and includes the code to be executed continuously by reading inputs, triggering outputs. This function is the core of Arduino programs and does a lot of work.

5.2.2 SETUP

The setup function is called once when the program starts. We use it to initialize pin modes or to begin serial communication. It must be included in the program even if there are no statements to run.

Syntax:

```
cpp

CopyEdit

Void setup ()

{

  pinMode (pin, OUTPUT); // sets the 'pin' as output

}
```

5.2.3 LOOP

The loop() function does precisely what its name suggests and loops consecutively allowing the program to change, respond, and control the Arduino board.

Syntax:

```
cpp

CopyEdit

Void loop ()

{
  digitalWrite (pin, HIGH); // Turns pin on

  delay (1000); // pauses for one second

}
```

5.2.4 OTHER STRUCTURES – CONTROL STRUCTURES

There are many control structures which are used in the programming language like if, if else, while, do while, for, switch case, break, continue, return, goto. They are mainly used for controlling after a certain condition is reached. Some control structures used in this project are listed below.

IF

If statements test whether a certain condition has been reached, such as a value being above a certain value, and executes the statements inside the brackets if it is true. If false, the program skips over the statement.

Syntax:

cpp

CopyEdit

```
If (some variable ?? value)

{

  Statements;

}
```

While

While loops will loop continuously, and infinitely, until the expression inside the parenthesis becomes false. Something must change the tested variable, or the while loop will never exit. This could be in the code such as an incremented variable, or an external condition, such as testing a sensor.

Syntax:

```
cpp

CopyEdit

While (variable < 200)

{

  Controlstatement;

  Variable++;

}
```

5.2.5 Functions:

pinMode() Function

The pinMode() function is used to configure a specific pin to behave either as an input or an output. It is possible to enable the internal pull-up resistors with the mode

INPUT_PULLUP. Additionally, the INPUT mode explicitly disables the internal pull-ups.

pinMode() Function Syntax

```
cpp

CopyEdit

Void setup ()

{

  pinMode (pin, mode);

}
```

- pin: the number of the pin whose mode you wish to set
- mode: INPUT, OUTPUT, or INPUT_PULLUP.

digitalWrite() Function

The digitalWrite() function is used to write a HIGH or a LOW value to a digital pin. If the Pin has been configured as an OUTPUT with pinMode(), its voltage will be set to the corresponding value:
5V (or 3.3V on 3.3V boards) for HIGH, 0V (ground) for LOW. If the pin is configured as an INPUT, digitalWrite() will enable (HIGH) or disable (LOW) the internal pullup on the input pin.

It is recommended to set the pinMode() to INPUT_PULLUP to enable the internal pull-up resistor. If you do not set the pinMode() to OUTPUT, and connect an LED to a pin, when calling digitalWrite(HIGH), the LED may appear dim. Without explicitly setting pinMode(), digitalWrite() will have enabled the internal pull-up resistor, which acts like a large current limiting resistor.

digitalWrite() Function Syntax

cpp

CopyEdit

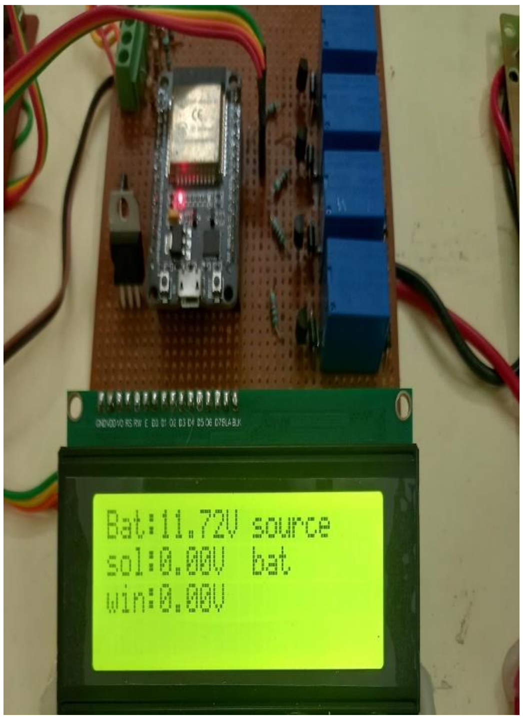
```
Void loop ()

{

  digitalWrite (pin, value);

}
```

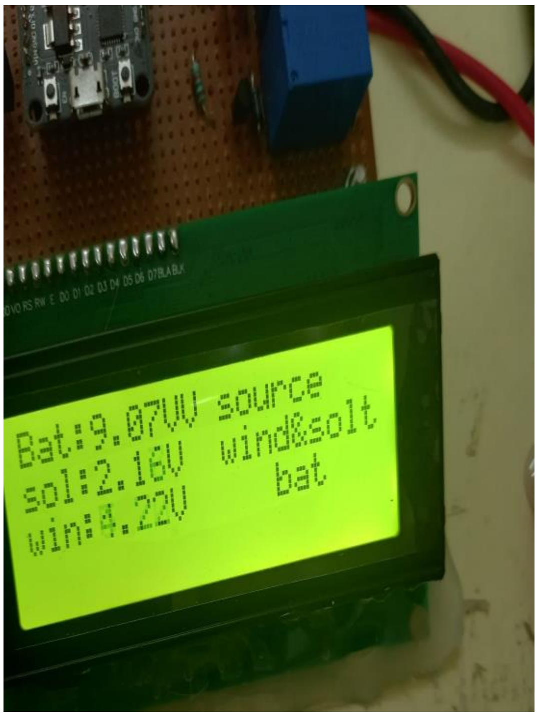
- pin: the number of the pin whose mode you wish to set
- value: HIGH, or LOW

CHAPTER 6

RESULT

6.1Hardware Output

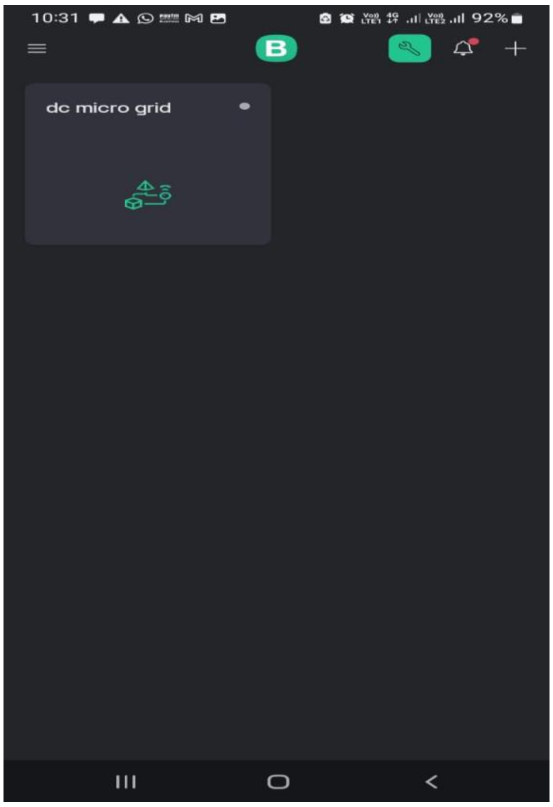


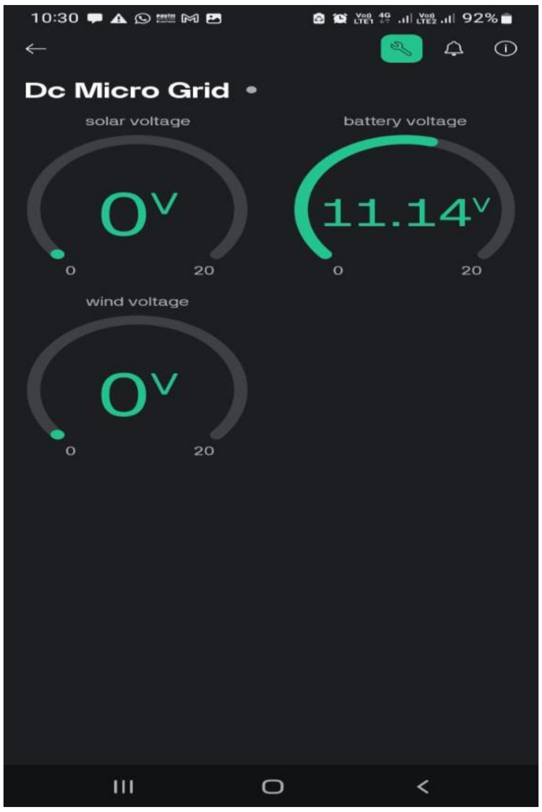


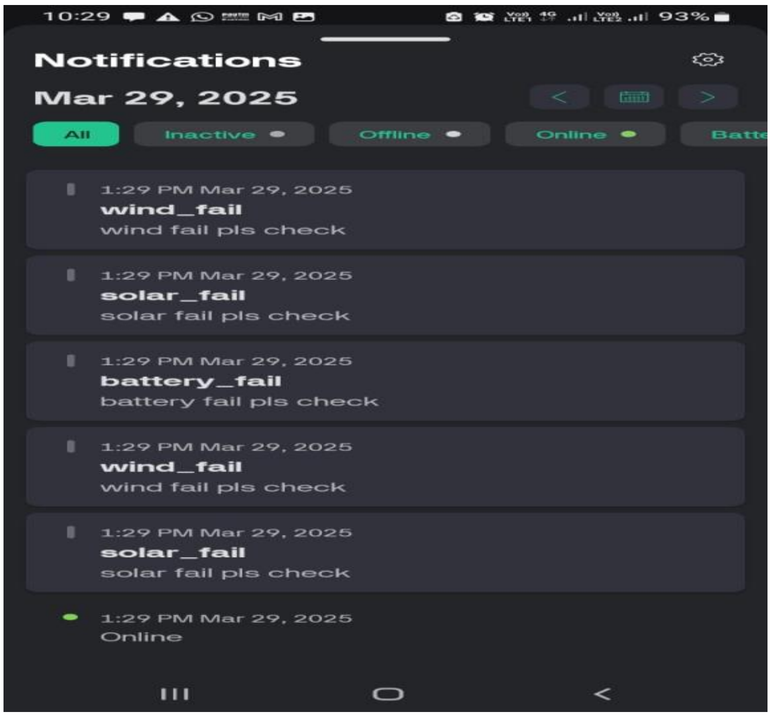
The hardware output displayed on the LCD screen shows the voltage readings for three different power sources: Battery, Solar, and Wind. Here's the explanation based on the current readings:

- **Bat: 11.72V source**: The battery is providing a voltage of 11.72V, which suggests it is operational and supplying power to the system.
- **Sol: 0.00V bat**: The solar panel is not generating any voltage (0.00V), indicating that it is either not receiving sunlight, is disconnected, or is malfunctioning.
- **Win: 4.22V bat**: The wind energy source is producing 4.22V, and the label "bat" indicates that this energy is being used to charge the battery.

6.2 Software Blynk Application Output







Notifications
Mar 29, 2025

1:29 PM Mar 29, 2025
wind_fail
wind fail pls check

1:29 PM Mar 29, 2025
solar_fail
solar fail pls check

1:29 PM Mar 29, 2025
battery_fail
battery fail pls check

1:29 PM Mar 29, 2025
wind_fail
wind fail pls check

1:29 PM Mar 29, 2025
solar_fail
solar fail pls check

1:29 PM Mar 29, 2025
Online

The image shows a mobile app interface for a "Dc Micro Grid" system, displaying real-time voltage readings.

CHAPTER 7
CONCLUSION AND FUTURE SCOPE

7.1 Conclusion

The IoT-based DC microgrid automation system developed in this project significantly enhances the performance of traditional DC microgrids by integrating hybrid power generation and advanced automation features. The system leverages solar panels and a DC geared motor (emulating wind power) to prioritize renewable energy sources, reducing reliance on non-renewable alternatives and promoting sustainable energy usage for urban applications, such as EV charging stations. The incorporation of voltage sensors, an ESP32 microcontroller, and a relay module enables precise monitoring and control of power distribution, ensuring efficient source switching and load management.

The Blynk IoT application plays a pivotal role in automating the microgrid, providing users with real-time data visualization of voltage levels from solar, wind, and battery sources. By allowing users to set customizable threshold limits for voltage and load capacity, the system facilitates proactive fault detection and rapid load isolation during overvoltage or overcurrent conditions. The generation of alert messages via the Blynk app notifies users instantly of system status, eliminating the need for manual intervention and reducing response times. This automation enhances system reliability, as faults are addressed with minimal delay, protecting components and ensuring operational continuity.

The successful integration of hardware (solar panel, DC motor, battery, sensors, ESP32, relays) and software (Arduino IDE, Blynk app) demonstrates the potential of IoT in addressing challenges in DC microgrid management. The system's ability to dynamically switch between power sources based on real-time conditions and user-defined parameters underscores its adaptability and efficiency. This project, conducted at [Institution Name], establishes a robust framework for automated microgrid operation, contributing to the advancement of smart energy solutions in urban environments.

7.2 Future Scope

The DC microgrid automation system presented in this work offers a scalable foundation that can be extended to address broader applications and challenges in both DC and AC microgrids. Several avenues for future development are proposed to enhance the system's functionality and applicability:

- **Adaptation for AC Microgrids**: The current system, designed for DC power distribution, can be modified to support AC microgrids by incorporating inverters and AC-compatible relays. This would enable integration with grid-tied systems or AC loads, expanding the system's utility for residential and commercial applications.
- **Enhanced Fault Detection**: The system addresses overvoltage and overcurrent faults effectively but can be upgraded to detect additional DC microgrid faults, such as arc faults and short circuits. By integrating arc-fault detection circuits or current sensors (e.g., ACS712), the ESP32 can monitor for irregular current patterns and trigger protective measures. Firmware updates would enable real-time analysis of fault signatures, improving system safety and reliability.
- **Energy Storage Optimization**: The battery unit can be enhanced with advanced charge management algorithms to optimize charging and discharging cycles, extending battery lifespan. Integration of lithium-ion batteries with a battery management system (BMS) could provide precise state-of-charge monitoring, communicated via the Blynk app, to improve energy efficiency.
- **Scalability and Multi-Source Integration**: The system can be scaled to include additional renewable sources, such as hydroelectric or fuel cells, by expanding the sensor and relay network. A modular design would allow the ESP32 to manage multiple sources, with the Blynk app providing a unified interface for monitoring and control.
- **Edge Computing and AI**: Incorporating edge computing capabilities into the ESP32 or an auxiliary module could enable predictive maintenance and fault forecasting using machine learning algorithms. By analyzing historical voltage and load data, the system could anticipate faults or optimize source selection, further enhancing performance.

APPENDEX

Arduino Code:

```
#define BLYNK_TEMPLATE_ID "TMPL357G79pKS"
#define BLYNK_TEMPLATE_NAME "DC Micro Grid"
#define BLYNK_AUTH_TOKEN "UrTfNgnXg3gJI1WiEIdARCIOQ7_pfujJ"

#include <BlynkSimpleEsp32.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

bool myflag = false;
bool myflag1 = false;
bool myflag2 = false;

// LCD I2C address (change if needed)
LiquidCrystal_I2C lcd(0x27, 20, 4);  // 0x27 or 0x3F for 20x4 or 16x2 LCD

#define SOLAR_PIN 33   // ADC pin for solar voltage
#define BATTERY_PIN 35 // ADC pin for battery voltage
#define WIND_PIN 32    // ADC pin for wind voltage

#define RELAY_SOLAR  19  // Relay for solar power
#define RELAY_BATTERY 18 // Relay for battery power
#define RELAY_WIND  5   // Relay for wind power

#define DISHCHARGE 23   // Relay for wind power
```

```
#define VOLTAGE_DIVIDER_RATIO 10.0  // Adjust based on resistor values
#define ADC_RESOLUTION 4095.0  // 12-bit ADC
#define REF_VOLTAGE 5.0       // ESP32 ADC reference voltage

#define BATTERY_FULL_VOLTAGE 12.6 // Change based on battery type

// WiFi credentials (for online mode)
char ssid[] = "Anand";   // WiFi SSID
char password[] = "Anand123";  // WiFi Password

// Blynk Authorization Token (from your Blynk app)
char auth[] = BLYNK_AUTH_TOKEN;

// Function to read voltage from ADC
float readVoltage(int pin) {
   int adcValue = analogRead(pin);
   float voltage = (adcValue / ADC_RESOLUTION) * REF_VOLTAGE *
VOLTAGE_DIVIDER_RATIO;
   return voltage;
}
void setup() {
   Serial.begin(115200);
   analogReadResolution(12);
   Blynk.begin(auth, ssid, password);
   pinMode(RELAY_SOLAR, OUTPUT);
   pinMode(RELAY_BATTERY, OUTPUT);
   pinMode(RELAY_WIND, OUTPUT);
   pinMode(DISHCHARGE, OUTPUT);
```

```
  digitalWrite(RELAY_SOLAR, LOW);
  digitalWrite(RELAY_BATTERY, LOW);
  digitalWrite(RELAY_WIND, LOW);
  digitalWrite(DISHCHARGE, LOW);

  lcd.init();
  lcd.backlight();
  lcd.setCursor(0, 0);
  lcd.print("DC Microgrid");
  lcd.setCursor(0, 1);
  lcd.print("Battery Mgmt Sys");
  delay(1500);
  lcd.clear();
}

void loop() {
  float solarVoltage = readVoltage(SOLAR_PIN);
  float batteryVoltage = readVoltage(BATTERY_PIN);
  float windVoltage = readVoltage(WIND_PIN);

  lcd.setCursor(0, 0);
  lcd.print("Bat:");
  lcd.print(batteryVoltage);
  lcd.print("V");

  lcd.setCursor(0, 1);
  lcd.print("sol:");
  lcd.print(solarVoltage);
  lcd.print("V");
  lcd.setCursor(0, 2);
  lcd.print("win:");
  lcd.print(windVoltage);
  lcd.print("V");

  if(solarVoltage >=2 && batteryVoltage <2 && windVoltage <2){ //// solar on
     digitalWrite(RELAY_SOLAR, HIGH);
     lcd.setCursor(11, 1);
     lcd.print("source   ");
     lcd.setCursor(11, 2);
     lcd.print("solar ");
  }

  if(batteryVoltage >=2 && solarVoltage <2 && windVoltage <2){ // battery  on
     digitalWrite(RELAY_BATTERY, HIGH);
     lcd.setCursor(11, 0);
     lcd.print("source   ");
     lcd.setCursor(11, 1);
     lcd.print("bat    ");
  }

  if(windVoltage >=2 && solarVoltage <2 && batteryVoltage <2){ //wind on
     digitalWrite(RELAY_WIND, HIGH);
     lcd.setCursor(11, 0);
     lcd.print("source   ");
     lcd.setCursor(11,
```

REFERENCES

[1] Y. Murai, Y. Kawase, K. Ohashi, K. Nagatake & K. Okuyama, “Torque ripple improvement for brushless DC miniature motors” IEEE Transactions on Industry Applications, vol. 25, no.3, pp 441-450,May/Junl989.

[2] P. Pillay and R. Krishnan, “Modeling, simulation, and analysis of permanent- magnet motor drives. II. The brushless DC motor drive”, IEEE Transactions on Industry Applications, pp. 274–279, 1989.

[3] R. Krishnan and G.-H. Rim, “Modeling, simulation, and analysis of variable-speed constant frequency power conversion scheme with a permanent magnet brushless dc generator”, IEEE Transactions on Industrial Electronics, 37(4), 291–296, 1990.

[4] R. Carlson, M. Lajoie-Mazenc & J. C. D. S. Fagundes, “Analysis of torque ripple due to phase commutation in brushless DC machines”, Industry Applications, IEEE Transactions on, vol. 28no.3, pp 632-638, May/Jun 1992.

[5]. R. Krishnan, “Electric Motor Drives”, Prentice Hall, Upper Saddle River, NJ, 2001.

[6] Q. Zhao, and F.C. Lee, “High-Efficiency, High Step-Up DC–DC Converters”, IEEE Trans. on Power Electr., vol. 18, issue 1, pp. 65-73, January 2003.