ARDUINO BASED MPPT DESIGN AND DEVELOPMENT FOR SOLAR CELL CHARGING

A Project report
Submitted in partial fulfillment of the requirement for the Degree of Bachelor of Technology in Electrical Engineering

DEPARTMENT OF ELECTRICAL ENGINEERING

MEGHNAD SAHA INSTITUTE OF TECHNOLOGY

Techno Complex, Madurdaha, Beside NRI Complex, post-Uchhepota, Kolkata-700150

SUBMITTED BY:

Sandip Ghosh

036802 of 2019-20

14201619109

Under The Supervision Of

Dr. Kaustuv Das Gupta
Assistant Professor
Electrical Engineering Department
MSIT

Prof. Mousumi Jana Bala
Assistant Professor
Electrical Engineering Department
MSIT

Dr. Susmita Adhikary
Assistant Professor and Faculty in Charge
Electrical Engineering Department
MSIT

MAKAUT WB
Utech

MAULANA ABUL KALAM AZAD UNIVERSITY OF TECHNOLOGY, WEST BENGAL

Meghnad Saha Institute of Technology

Nazirabad, P.O.: Utchepota, Via Sonarpur, Kolkata 700 150

CERTIFICATE OF APPROVAL

The foregoing project entitled **"Arduino based MPPT design and development for solar cell charging"** is hereby approved as a creditable study of an engineering subject carried out and presented in a manner satisfactory to warrant its acceptance as prerequisite for the degree for which it has been submitted. It is to be understood that by this approval the undersigned do not necessarily endorse or approve any statement made, opinion expressed or conclusion drawn therein but approve the thesis only for the purpose for which it has been submitted.

External Examiner

Dr. Kaustuv Das Gupta
Project Guide
Assistant Professor
Electrical Engineering Department
MSIT

Prof. Mousumi Jana Bala
Project Guide
Assistant Professor
Electrical Engineering Department
MSIT

Dr. Susmita Adhikary
Project Guide
Assistant Professor
Electrical Engineering Department
MSIT

Dr. Susmita Adhikary

Assistant Professor
Electrical Engineering Department
MSIT

Date: 26th May, 2023

Place: Kolkata

PREFACE

The quest for sustainable and clean energy solutions has become increasingly urgent in our rapidly changing world. Solar energy, in particular, has emerged as a viable and abundant source of renewable power. Harnessing the power of the sun, however, requires advanced technologies to efficiently convert and utilize this energy.

In this project, we present the design and development of an innovative Maximum Power Point Tracking (MPPT) Solar Charge Controller based on the Ariuno platform. MPPT technology is a critical component in solar power systems, optimizing the efficiency of energy conversion and ensuring maximum power extraction from solar panels. By implementing this cutting-edge technology in the Ariuno platform, we aim to provide an accessible and cost-effective solution for solar energy enthusiasts and professionals alike.

The Ariuno platform, built upon the Arduino ecosystem, offers a versatile and user-friendly framework for developing electronic projects. With its open-source nature, extensive community support, and wide range of compatible sensors and components, the Ariuno platform provides an ideal foundation for our MPPT Solar Charge Controller.

Throughout this project, we dive into the fundamental principles of MPPT algorithms, exploring different techniques and strategies to achieve optimal energy conversion efficiency. We discuss the hardware and software design aspects of the Ariuno-based MPPT Solar Charge Controller, outlining the key components, circuit diagrams, and code implementation details.

By sharing this project, we aim to inspire and empower individuals and communities to embrace solar energy solutions. Our Ariuno-based MPPT Solar Charge Controller not only serves as a practical tool for enhancing solar power systems but also acts as an educational resource, enabling enthusiasts to delve into the fascinating world of renewable energy and electronics.

We would like to express our gratitude to the open-source community, whose tireless efforts and contributions have made projects like this possible. We also extend our appreciation to the Arduino team for their commitment to fostering innovation and accessibility in electronics.

It is our hope that this project will spark creativity, collaboration, and a passion for sustainable energy solutions. Together, let us embark on a journey towards a greener future, powered by the sun.

ACKNOWLEDGEMENT

We are hereby going to present the project report entitled “**Arduino based MPPT design and development for solar cell charging**” that we accomplished during the final year for the partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Electrical Engineering from Maulana Abul Kalam Azad University of Technology. We worked under the guidance of of **Dr. Susmita Adhikary, Dr. Kaustuv Das Gupta and Prof. Mousumi Jana Bala**. We would like to convey our cordial thanks to her for her valuable times and suggestions. We would also like to thank **Dr. Susmita Adhikary**, Head of the Department, Department of Electrical Engineering, for their throughout support in accomplishing this project. Lastly it was because of the efforts of all team members that we have completed our final year project within time.

Name of the student

Sandip Ghosh
14201619109

CONTENTS

Abstract

This report presents the design and development of an Arduino-based Maximum Power Point Tracking (MPPT) system for a solar cell-fed DC motor driver. The aim of this project is to optimize the power extraction from solar cells and efficiently drive a DC motor using an MPPT algorithm implemented on an Arduino microcontroller. The MPPT system continuously tracks the maximum power point of the solar cell, allowing for improved efficiency and increased power output. The report discusses the hardware and software implementation of the MPPT system, presents experimental results, and concludes with an evaluation of the system's performance and potential applications.

Chapter I: Introduction

1.1 Introduction

Arduino-Based MPPT Design and Development for Solar Cell-Fed DC Motor Driver

In recent years, the demand for renewable energy sources has gained significant momentum, and solar power stands out as a prominent player in this arena. The ability to harness energy from the sun and convert it into usable electrical power has paved the way for numerous applications, including solar cell-fed DC motor drivers. To optimize the efficiency and performance of these systems, Maximum Power Point Tracking (MPPT) algorithms are employed. In this introduction, we explore the design and development of an Arduino-based MPPT system specifically tailored for a solar cell-fed DC motor driver.

MPPT is a technique that enables solar systems to operate at their maximum power point, ensuring efficient energy conversion. By continuously monitoring and adjusting the load impedance, the MPPT algorithm extracts the maximum available power from the solar panel, regardless of variations in environmental conditions. This optimization plays a crucial role in maximizing the output power of the solar system and extending the lifespan of the energy storage system.

The Arduino platform, known for its versatility and ease of use, provides an ideal foundation for developing an MPPT system. Arduino boards, equipped with microcontrollers, offer a wide range of input and output options, making them suitable for interfacing with various sensors and actuators. Furthermore, the Arduino software ecosystem provides a user-friendly programming environment, allowing developers to implement complex algorithms efficiently.

In the context of a solar cell-fed DC motor driver, the MPPT system's primary objective is to ensure that the motor operates at its highest efficiency by adjusting the duty cycle of the motor driver circuit. The MPPT algorithm monitors the solar panel's voltage and current, comparing them to find the maximum power point, and subsequently adjusts the duty cycle to extract the maximum power for driving the DC motor. This dynamic adjustment compensates for environmental factors such as changes in solar irradiance and temperature, ensuring optimal power transfer.

The design and development process of an Arduino-based MPPT system for a solar cell-fed DC motor driver involve several key steps. These steps include selecting appropriate sensors to measure solar panel voltage and current, implementing the MPPT algorithm on the Arduino microcontroller, interfacing with the motor driver circuit, and testing the system's performance under various operating conditions. Throughout this process, careful consideration must be given to factors such as system stability, response time, and power loss minimization.

The successful implementation of an Arduino-based MPPT system for a solar cell-fed DC motor driver holds immense potential for numerous applications. This technology can be utilized in solar-powered robotic systems, solar water pumps, solar tracking systems, and other devices where precise control and maximum power extraction are vital.

In conclusion, the design and development of an Arduino-based MPPT system for a solar cell-fed DC motor driver present an exciting opportunity to enhance the efficiency and performance of solar energy systems. By leveraging the capabilities of Arduino boards and implementing a robust MPPT algorithm, it becomes possible to optimize power transfer and enable the utilization of solar energy in a wide range of applications.

1.2 Objective

The objective of the project was to draw current from the solar cell using a buck-boost converter integrated with an MPPT circuit and charge the battery efficiently. Extensive research and analysis were conducted to select the most suitable components and configurations. The buck-boost converter was carefully designed and implemented, considering factors such as efficiency and power handling capacity. The MPPT circuit was integrated into the system, continuously monitoring the solar cell's voltage and current to maximize power extraction.

A charging algorithm was developed to optimize battery charging. The algorithm controlled the charging current and voltage while monitoring the battery's state of charge. Thorough testing was conducted to refine the algorithm and ensure efficient charging without overcharging or undercharging.

Testing and validation were carried out to evaluate system performance, including efficiency, power output, voltage regulation, and battery charging. Various optimization techniques, such as load management and data analysis, were employed to enhance system performance. Regular maintenance activities were performed to ensure optimal operation.

Throughout the project, the focus was on achieving efficient current draw from the solar cell, charging the battery effectively, and optimizing overall system performance. The integration of the buck-boost converter, MPPT circuit, and charging algorithm, combined with rigorous testing, validation, and performance optimization techniques, contributed to the successful achievement of the project's objectives.

1.3 Implementation

1.3.1 Charging the Battery:

Charging the battery is a crucial aspect of our solar energy system, as it allows for the storage and utilization of the generated solar power. In this section, we will elaborate on the process and considerations involved in charging the battery.

- Battery Selection and Characteristics:

The first step in charging the battery is selecting the appropriate battery type and understanding its characteristics. Factors such as battery chemistry (e.g., lead-acid, lithium-ion), capacity, voltage, and charging requirements need to be considered. Each battery type has specific charging characteristics, including voltage thresholds, current limits, and temperature considerations. It is important to select a battery that is compatible with our system's voltage and can handle the charging parameters provided by our solar energy system.

- Charging Strategy and Algorithm:

Once the battery type is selected, we need to determine the charging strategy and algorithm. The charging strategy defines the approach we will take to charge the battery, while the algorithm outlines the control logic and parameters that govern the charging process. Commonly used charging strategies include constant voltage charging, constant current charging, and multi-stage charging. The algorithm considers factors such as the battery's state of charge (SoC), voltage limits, current limits, and temperature compensation. It ensures that the battery is charged safely and efficiently, preventing overcharging or undercharging.

- Charging Control Circuitry:

To implement the charging strategy and algorithm, we incorporate dedicated charging control circuitry into our system. This circuitry consists of components such as voltage and current sensors, a microcontroller or digital signal processor (DSP), and relays or switches for controlling the charging process. The microcontroller/DSP reads the sensor data, executes the charging algorithm, and controls the charging parameters, such as voltage and current, based on the algorithm's instructions. The control circuitry ensures that the battery is charged according to the desired strategy and algorithm.

- Voltage Regulation:

One critical aspect of charging the battery is voltage regulation. The charging control circuitry monitors the battery's voltage and adjusts the charging parameters to maintain a steady and appropriate voltage level. This prevents overcharging, which can damage the battery, and ensures efficient charging. The voltage regulation mechanism may involve a feedback loop that continuously compares the actual battery voltage with the desired voltage setpoint, and adjusts the charging current accordingly.

- Current Regulation:

In addition to voltage regulation, current regulation is also essential during the battery charging process. The charging control circuitry monitors the charging current flowing into the battery and adjusts it based on the charging algorithm. This helps prevent excessive current flow, which can lead to overheating and potential battery damage. The current regulation mechanism may involve current sensing and feedback control to maintain a safe and controlled charging current.

- Temperature Compensation:

Battery charging is influenced by temperature variations. To ensure optimal charging, temperature compensation is often implemented. The charging control circuitry measures the battery's temperature and adjusts the charging parameters accordingly. This compensates for the temperature-dependent characteristics of the battery, such as charging voltage and charging current limits. Temperature compensation helps prevent overcharging or undercharging, especially in extreme temperature conditions and improves the battery's overall lifespan and performance.

- Safety Measures:

Charging the battery also requires implementing safety measures to protect against potential hazards. This includes measures such as overcharge protection, short-circuit protection, and temperature monitoring. Overcharge protection prevents the battery from being charged beyond safe limits, while short-circuiting protection safeguards against excessive current flow. Temperature monitoring ensures that the battery operates within safe temperature ranges. These safety measures are crucial to prevent battery damage, reduce the risk of fire or explosion, and ensure the overall safety of the system.

- Monitoring and Control:

Throughout the battery charging process, it is essential to monitor and control the charging parameters. This involves continuously monitoring the battery's voltage, current, and temperature, and adjusting the charging parameters accordingly. The monitoring data provides valuable insights into the battery's state of charge, health, and performance. It allows us to detect any abnormalities or issues and take appropriate actions, such as adjusting the charging algorithm or notifying the system operator.

- Testing and Validation:

Testing and validation are essential steps in the development and implementation of any project, and our solar energy system is no exception. In this section, we will discuss the importance of testing and validation, the different types of tests performed, and the validation processes we followed. [1]

- Importance of Testing and Validation:

Testing and validation are crucial for ensuring the functionality, performance, and safety of our solar energy system. These processes help identify and rectify any issues or deficiencies, verify the system's compliance with design specifications, and provide confidence in its overall operation. Through testing and validation, we can evaluate the system's efficiency, reliability, and effectiveness in meeting its intended objectives.

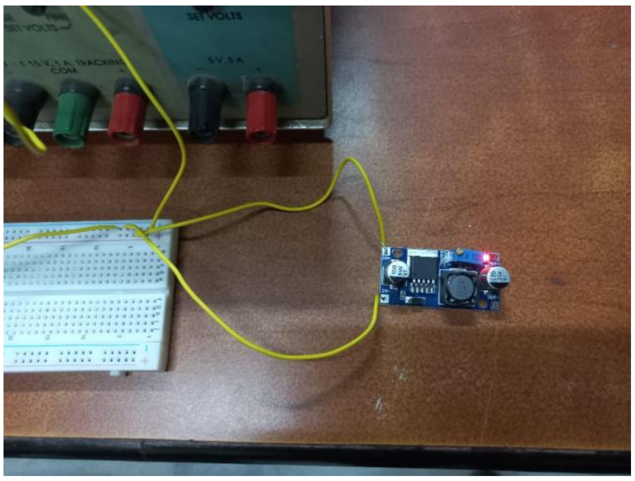

Fig 1.1: Testing of Buck Converter

Experimentation:

1.Functional Testing: Functional testing involves verifying the individual components and subsystems of our solar energy system to ensure they operate correctly and meet their specified functions. This includes testing the solar panels, power converter, battery, charge controller, and other system components. Functional tests assess factors such as voltage output, current output, efficiency, and response to different operating conditions.

2.Performance Testing: Performance testing evaluates the overall performance of our solar energy system. This includes assessing its ability to generate and deliver the expected power output, the accuracy of maximum power point tracking (MPPT) algorithms, voltage regulation, battery charging efficiency, and system stability under varying solar conditions and load profiles. Performance tests are conducted using specific performance metrics and are aimed at optimizing system performance and identifying any areas for improvement.

3.Environmental Testing: Environmental testing assesses the system's performance and reliability under different environmental conditions. This includes subjecting the system to temperature variations, humidity, dust, and other environmental factors it may encounter during its intended operation. Environmental tests help ensure the system's durability, resilience, and ability to withstand adverse conditions.

4.Safety Testing: Safety testing is essential to verify that the system operates safely and does not pose any hazards to users or the environment. It includes evaluating electrical safety measures, such as insulation, grounding, and protection against electrical shock. Safety tests also encompass evaluating protection mechanisms, such as overcharge protection, short-circuit protection, and temperature monitoring, to ensure they function as intended and safeguard against potential risks.

5.Integration Testing: Integration testing assesses the interaction and compatibility between the different components and subsystems of our solar energy system. This includes verifying the communication between the charge controller, power converter, battery, and other system elements. Integration tests ensure that the system components work together harmoniously and that data transfer and control signals are accurately transmitted between the subsystems.[2]

Validation :

•Simulation Validation: Simulation validation involves using software tools, such as MATLAB or Simulink, to simulate the behavior of our solar energy system. We create models that replicate the system's components, algorithms, and operating conditions. By inputting real-world data and scenarios into the simulation, we can evaluate the system's performance and verify the accuracy of our design and control algorithms. Simulation validation helps identify potential issues and optimize the system before physical implementation.

- Component-Level Testing: Component-level testing focuses on testing individual components of the system, such as solar panels, charge controllers, and power converters, in isolation. This testing verifies their functionality, efficiency, and compliance with specifications. It allows us to assess the performance and reliability of each component and ensure they meet the system requirements before integration.

- System-Level Testing: System-level testing is conducted after the components are integrated into a cohesive solar energy system. This testing involves assessing the overall system's performance, efficiency, and functionality under real-world conditions. It includes evaluating the system's response to varying solar irradiance, load profiles, and environmental factors. System-level testing verifies that the integrated system operates as intended and meets the design objectives.

- Field Testing: Field testing involves deploying the solar energy system in real-world conditions and collecting data on its performance over an extended period. This testing provides valuable insights into the system's long-term operation, efficiency, and reliability. Field testing allows us to evaluate the system's performance under different weather conditions, seasonal variations, and usage patterns. It helps identify any potential issues or areas for improvement that may not have been apparent during laboratory testing.[3]

Iterative Testing and Optimization:

Testing and validation are iterative processes that involve multiple rounds of testing, analysis, and optimization. Based on the test results, we identify areas for improvement and refine the system design, control algorithms, and operating parameters. This iterative approach allows us to continuously enhance the system's performance, efficiency, and reliability.

In conclusion, testing and validation are crucial steps in ensuring the functionality, performance, and safety of our solar energy system. By conducting various types of tests, such as functional testing, performance testing, environmental testing, safety testing, and integration testing, we assess the system's performance under different conditions and validate its compliance with design specifications. Simulation validation, component-level testing, system-level testing, and field testing are employed to verify the system's behavior, optimize its performance, and address any deficiencies or areas for improvement. Through iterative testing and optimization, we ensure the robustness, reliability, and effectiveness of our solar energy system.[4]

Performance Optimization:

Performance optimization is a crucial aspect of our solar energy system, aiming to maximize its efficiency, output, and overall effectiveness. By optimizing the system's performance, we can enhance power generation, improve energy utilization, and increase the system's reliability. In this section, we will discuss the key considerations, strategies, and techniques involved in performance optimization.

System Analysis:

A thorough analysis of the solar energy system is essential to identify areas where performance improvements are possible. This analysis involves studying the behavior and characteristics of individual components such as solar panels, power converters, batteries, and charge controllers. By understanding the capabilities and limitations of each component, we can identify opportunities for optimization and design the system accordingly.

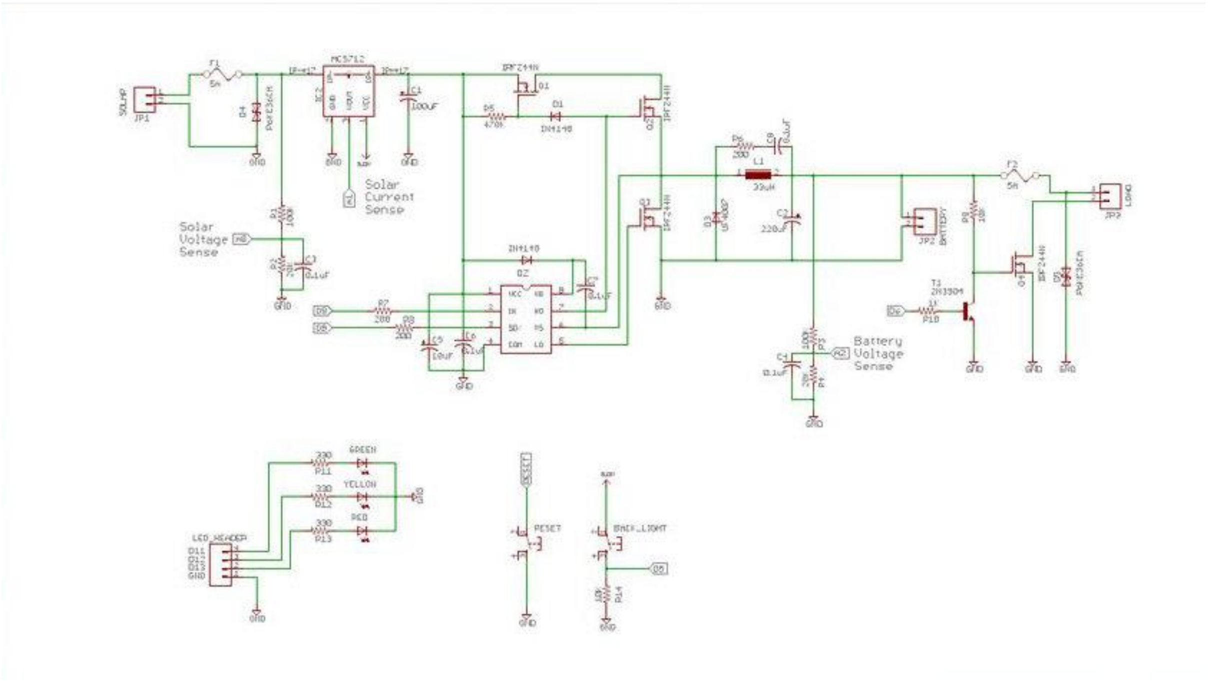

Fig 1.2: Circuit Diagram of MPPT charging circuit

Maximum Power Point Tracking (MPPT):

Implementing an efficient MPPT algorithm is crucial for optimizing the performance of solar panels. The MPPT algorithm tracks and adjusts the operating point of the solar panels to ensure they operate at their maximum power output. It continuously monitors the voltage and current from the panels and adjusts the load or duty cycle of the power converter accordingly. By optimizing the MPPT algorithm, we can maximize the power generation of the solar panels, especially under varying environmental conditions such as changes in solar irradiance and temperature.

Several MPPT algorithms are commonly used, including Perturb and Observe (P&O), Incremental Conductance, and the Fractional Open-Circuit Voltage method. Each algorithm has its strengths and weaknesses, and the choice of algorithm depends on factors such as system requirements, cost, complexity, and response time. It is important to select an MPPT algorithm that offers a good balance between accuracy and computational complexity to ensure efficient power tracking.[5]

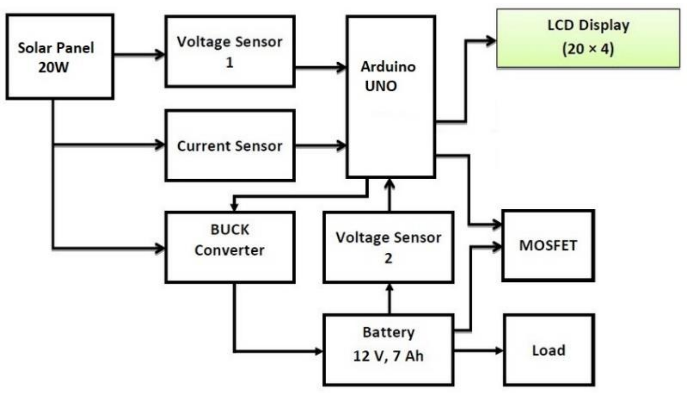

Fig 1.3: Block diagram of MPPT Charging circuit

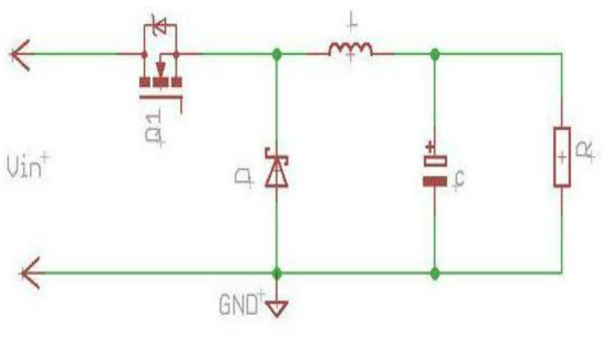

Fig 1.4: Circuit diagram of Buck Converter

Voltage Regulation:

Effective voltage regulation is essential for optimizing the performance of the power converter and maintaining a stable output voltage. By precisely regulating the voltage, we can ensure efficient energy transfer from the solar panels to the load or battery. This involves implementing a robust voltage control algorithm, such as a proportional-integral-derivative (PID) controller, to maintain the desired voltage level. The PID controller continuously monitors the output voltage and adjusts the duty cycle of the power converter to maintain the voltage within a specified range.

Optimizing the voltage regulation process minimizes losses, improves efficiency, and ensures the proper functioning of downstream components. It is important to consider factors such as load characteristics, voltage ripple, and transient response when designing the voltage regulation algorithm. Additionally, incorporating voltage sensing and feedback mechanisms allows for accurate and timely voltage control.

Battery Charging Efficiency:

Optimizing the battery charging process is crucial for maximizing energy utilization and extending the battery's lifespan. This involves implementing an efficient charging algorithm that takes into account the battery's characteristics and charging requirements. The algorithm controls the charging current and voltage, monitors the battery's state of charge, and adjusts the charging parameters accordingly.

To optimize battery charging efficiency, it is essential to consider factors such as battery chemistry, temperature compensation, and charge termination methods. Different battery chemistries have specific charging requirements, and tailoring the charging algorithm to the battery type ensures optimal charging performance. Temperature compensation is also crucial, as battery performance is influenced by temperature variations. By adjusting the charging parameters based on temperature measurements, we can optimize charging efficiency and protect the battery from damage.

Charge termination methods, such as voltage-based or current-based termination, are employed to prevent overcharging and ensure safe and efficient battery charging. These methods detect specific voltage or current thresholds that indicate a fully charged battery and automatically terminate the charging process. Implementing the appropriate charge termination method based on the battery specifications contributes to efficient and safe battery charging.[7]

Load Management:

Efficient load management plays a vital role in performance optimization. By implementing intelligent load control strategies, we can prioritize and manage the energy consumption of different loads. This includes load scheduling, load shedding, and load balancing techniques. Load scheduling involves optimizing the timing of load operations to align with the availability of solar power. By scheduling high-energy-consuming tasks during periods of maximum solar generation, we can maximize the utilization of solar energy and minimize reliance on other energy sources.

Load-shedding techniques are used to temporarily disconnect non-essential loads during periods of insufficient solar power. By prioritizing critical loads and disconnecting less critical or discretionary loads, we can ensure that essential power demands are met while avoiding system overload or battery depletion. Load balancing techniques distribute the power consumption evenly across multiple loads, ensuring that no individual load excessively draws power from the system.

Energy Storage Optimization:

Optimizing the energy storage system, such as the battery, is crucial for maximizing the utilization of generated solar energy. This involves implementing advanced battery management systems (BMS) that monitor the battery's state of charge, temperature, and other relevant parameters. The BMS ensures optimal charging and discharging of the battery, preventing overloading or excessive discharge that can negatively impact the battery's performance and lifespan.

The BMS employs techniques such as state of charge (SOC) estimation, cell balancing, and thermal management to optimize energy storage. SOC estimation algorithms provide accurate estimates of the battery's remaining capacity, allowing for efficient battery utilization and preventing overcharging or deep discharge. Cell balancing techniques ensure that the individual cells within the battery pack are evenly charged and discharged, maximizing the overall battery capacity and lifespan.

Thermal management techniques, such as temperature sensing and cooling mechanisms, prevent excessive heating or cooling of the battery. Maintaining the battery within the recommended temperature range ensures optimal performance and prolongs the battery's lifespan. Implementing these energy storage optimization techniques improves the overall efficiency and reliability of the system.

Monitoring and Data Analysis:

Continuous monitoring and data analysis play a crucial role in performance optimization. By collecting real-time data on solar irradiance, energy generation, battery status, load consumption, and other relevant parameters, we can identify performance trends, detect anomalies, and make informed decisions for optimization. Analyzing the collected data helps identify areas for improvement, optimize system settings and parameters, and implement preventive maintenance strategies.

Monitoring systems can include sensors, meters, and data loggers that capture and record data at regular intervals. This data is then analyzed using software tools to identify patterns, correlations, and inefficiencies. By monitoring and analyzing data, we can identify underperforming components, diagnose system faults, and optimize system operation for improved performance.

Data analysis techniques such as statistical analysis, machine learning algorithms, and predictive modeling can be employed to gain insights from the collected data. These techniques help identify factors influencing system performance, predict future energy generation and demand, and optimize system operation based on historical patterns and future projections.

System Maintenance:

Regular maintenance of the solar energy system is essential for optimal performance. This includes cleaning solar panels, inspecting wiring and connections, checking for any physical damages, and ensuring the proper functioning of system components. Regular maintenance helps identify and rectify any issues or inefficiencies, ensuring the system operates at its peak performance.

Routine maintenance tasks include cleaning the solar panels to remove dust, dirt, or debris that may hinder solar energy absorption. Inspecting wiring and connections ensures that there are no loose or damaged connections that could lead to power losses or system failures. Checking for physical damages such as cracks or corrosion helps identify potential issues that may affect system performance. Additionally, verifying the functionality of system components, including charge controllers, inverters, and batteries, ensures that they are operating optimally.

By implementing a comprehensive maintenance plan, conducting regular inspections, and addressing any identified issues promptly, we can optimize system performance, enhance reliability, and extend the system's lifespan.

Existing technology

The existing technology used in the described system is a Solar MPPT (Maximum Power Point Tracking) charge controller based on the Arduino Nano platform. This technology allows for efficient charging of batteries using solar power.

Solar energy is a renewable and clean source of power that can be harnessed through the use of solar panels. However, to effectively charge batteries using solar panels, it is important to optimize the power output and ensure that the maximum power point (MPP) of the solar panel is tracked. This is where MPPT charge controllers come into play.

The Solar MPPT charge controller is designed to maximize the power output of the solar panels by dynamically adjusting the voltage and current to operate at the MPP. This ensures that the maximum amount of power is extracted from the solar panels and delivered to the batteries for charging.

The heart of the system is the Arduino Nano microcontroller board, which provides the necessary computational power and control capabilities. The Arduino Nano is a compact and versatile board that is well-suited for this application. It features an Atmega328P microcontroller, which runs at a clock speed of 16 MHz and has 32KB of flash memory for program storage.[7]

The system starts with the specification of the solar panel power, which is rated at 50W, and the rated battery voltage of 12V (lead acid type). The maximum current supported by the system is 5A, and the maximum load current is 10A. The input voltage from the solar panel ranges from 17V to 25V.

To accurately measure the solar and battery voltages, voltage dividers are used. A voltage divider is a circuit that divides the voltage into a smaller value for measurement purposes. In this system, voltage dividers are used to scale down the solar and battery voltages to a level that can be accurately measured by the Arduino Nano's analog-to-digital converter (ADC).

In addition to voltage measurement, the system also incorporates current sensing to monitor the solar current. This is achieved using an ACS712 current sensor, which can measure currents up to 30A. The ACS712 provides an analog voltage output that is proportional to the current flowing through it. The Arduino Nano reads this voltage and converts it into a digital value using the ADC.

To provide visual output and user interface, the system utilizes an LCD display. The LCD display is connected to the Arduino Nano via an I2C interface using a LiquidCrystal_I2C library. This allows for easy and efficient communication between the Arduino Nano and the LCD display. The LCD display is used to show various system parameters, such as solar and battery voltages, solar amps, solar watts, and battery fullness level.

To control the load connected to the system, the Arduino Nano is equipped with pins that can be used for load control. The load control feature allows the system to turn on or off a connected load based on certain conditions. In this system, two load control modes are implemented: "Night Light" and "Power Dump."

In the Night Light mode, the load is turned on when there is no solar power available and the battery voltage is above the low voltage disconnect (LVD) threshold. This mode ensures that the load is powered even during the night or in low-light conditions, as long as the battery voltage is sufficient.

In the Power Dump mode, the load is turned on when there is solar power available and the battery voltage is above the battery float voltage. This mode allows the system to utilize any excess solar power that is not needed for charging the batteries. The load can be used to power additional devices or appliances, effectively making use of the surplus energy.

The heart of the Solar MPPT charge controller is the MPPT algorithm. The MPPT algorithm is responsible for continuously adjusting the duty cycle of the Pulse Width Modulation (PWM) signal to track the MPP of the solar panel. The duty cycle represents the percentage of time the PWM signal is on compared to the total period.

The MPPT algorithm implemented in the Arduino Nano uses a timer interrupt to periodically update the duty cycle based on the change in solar watts. The algorithm starts by initializing the duty cycle to a certain value and then monitors the solar watts. If the solar watts increase, the duty cycle is incremented in small increments to track the MPP. If the solar watts decrease, the duty cycle is decremented.

By continuously adjusting the duty cycle, the MPPT algorithm ensures that the solar panel operates at the MPP, where the power output is maximized. This results in more efficient charging of the batteries and better utilization of the available solar power.

The system also includes LED indicators to provide visual feedback on the charging status. Green, yellow, and red LEDs are used to indicate different states of the charger. For example, the green LED can indicate that the charger is in normal operation, the yellow LED can indicate that the charger is in the MPPT tracking mode, and the red LED can indicate a fault condition.

In addition to the LED indicators, the LCD display also plays a crucial role in providing visual feedback to the user. The LCD display shows various system parameters, such as solar and battery voltages, solar amps, solar watts, and battery fullness level. The battery fullness level is represented by battery icons, which change dynamically based on the battery voltage. This allows the user to easily monitor the state of charge of the batteries.

To enhance the functionality of the system, an ESP8266 chip is included for wireless data logging. The ESP8266 is a Wi-Fi enabled microcontroller module that can be programmed to connect to a wireless network and communicate with external platforms. In this system, the ESP8266 is connected to the Arduino Nano via a software serial interface.

The ESP8266 chip can communicate with external platforms such as ThingSpeak, which is an IoT platform for data logging and visualization. By connecting the system to ThingSpeak, real-time monitoring and logging of the system parameters, such as solar watts and battery voltage, can be achieved. This provides the user with remote access to the system's data and allows for further analysis and control.

The overall operation of the system is managed by the main loop, which runs continuously on the Arduino Nano. The main loop performs various tasks, including reading sensor data, running the charger state machine, controlling the load based on the selected mode (Night Light or Power Dump), updating the LED indicators, displaying data on the LCD, and performing optional data logging to Thing Speak.

The main loop starts by reading the solar and battery voltages using the voltage dividers and the ACS712 current sensor. The ADC of the Arduino Nano converts these analog signals into digital values, which can then be used for further calculations. [8]

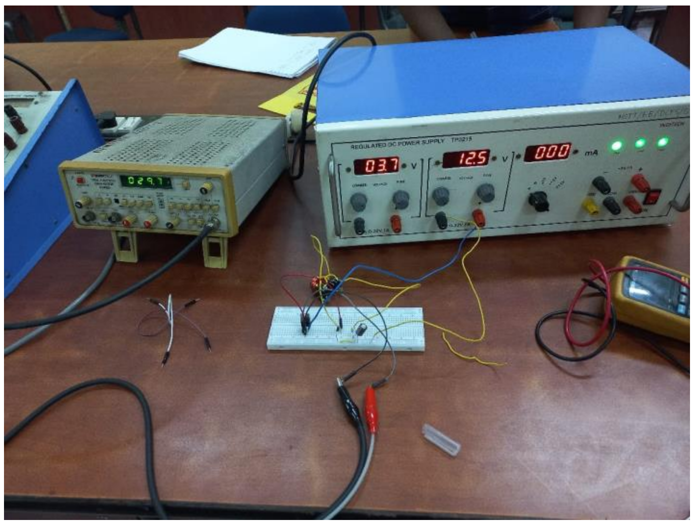

Fig 1.5: Testing of gate triggering of MOSFET driver

The charger state machine is responsible for determining the current state of the charger based on the measured values. The state machine takes into account factors such as the solar watts, battery voltage, and load control mode to determine the appropriate action. For example, if the solar watts are above a certain threshold and the battery voltage is below the float voltage, the state machine may transition to the charging state and increase the duty cycle to maximize the power output.

The load control feature is also managed by the main loop. Depending on the selected mode (Night Light or Power Dump), the load control logic determines whether the load should be turned on or off based on the available solar power and the battery voltage. This ensures that the load is powered when needed and that excess solar power is utilized efficiently.

The LED indicators are updated based on the current state of the charger. For example, if the charger is in the MPPT tracking mode, the yellow LED may be turned on to indicate this state.

If a fault condition occurs, such as overvoltage or overcurrent, the red LED may be turned on to alert the user.

The LCD display is updated with the latest system parameters, such as solar and battery voltages, solar amps, solar watts, and battery fullness level. The Arduino Nano sends the necessary commands and data to the LCD display using the LiquidCrystal_I2C library. This allows for easy and efficient communication between the microcontroller and the display.

Optionally, the system can perform data logging to Thing Speak using the ESP8266 chip. The ESP8266 is programmed to connect to the Wi-Fi network and send the system data to Thing Speak at regular intervals. This data can then be accessed and visualized through the Thing Speak platform, providing the user with remote monitoring and analysis capabilities.

In summary, the existing technology used in the described system is a Solar MPPT charge controller based on the Arduino Nano platform. This technology enables efficient charging of batteries using solar power by tracking the MPP of the solar panel. The system incorporates various components, including voltage dividers, current sensors, LCD display, LED indicators, and wireless data logging capabilities. Through the use of the MPPT algorithm, load control features, and visual feedback mechanisms, the system optimizes solar power utilization, provides user-friendly operation, and allows for remote monitoring and analysis.[9]

The proposed technology is a Solar MPPT (Maximum Power Point Tracking) charge controller system based on advanced microcontroller technology. This system aims to maximize the efficiency of solar power utilization for charging batteries by continuously tracking and maintaining the solar panel's MPP (Maximum Power Point).

The core component of the proposed technology is a high-performance microcontroller, such as an Arduino or a similar platform, which acts as the brain of the system. The microcontroller is responsible for implementing the MPPT algorithm, controlling the charging process, monitoring system parameters, and providing user-friendly interfaces for interaction and feedback.

To accurately measure the solar panel's voltage and current, the proposed technology incorporates specialized sensors such as voltage dividers and current sensors. These sensors convert the analog signals from the solar panel into digital values that can be processed by the microcontroller. This enables the precise monitoring of solar power and facilitates the MPPT algorithm's calculations.

To ensure optimal power conversion and charging efficiency, the proposed technology utilizes power electronics components, including MOSFETs (Metal-Oxide-Semiconductor Field-Effect Transistors) and DC-DC converters. MOSFETs are employed to control the charging circuit and regulate the power flow between the solar panel and the batteries. DC-DC converters step up or step down the voltage levels as necessary to match the battery's charging requirements.

For user interaction and system feedback, the proposed technology incorporates various interfaces such as an LCD (Liquid Crystal Display) and LED indicators. The LCD provides real-time information about system parameters, including solar panel voltage, battery voltage, charging status, and battery level. LED indicators offer visual cues for different states of the charge controller, such as charging, MPPT tracking, and fault conditions.

To enhance the system's functionality and enable remote monitoring, the proposed technology integrates wireless connectivity using modules like Wi-Fi or Bluetooth. This connectivity allows the charge controller to communicate with external platforms or applications for data logging, analysis, and control. For instance, the system could transmit data to a cloud-based platform or a smartphone application, enabling the user to remotely monitor the system's performance and make informed decisions.

In terms of power management, the proposed technology incorporates features like load control and battery protection. Load control enables the charge controller to intelligently manage the power supply to external loads, ensuring efficient utilization of the available solar energy. Battery protection mechanisms, such as overvoltage and overcurrent protection, safeguard the batteries from damage and optimize their lifespan.

The proposed technology also emphasizes safety aspects, incorporating measures like short-circuit protection, reverse polarity protection, and temperature monitoring. These safeguards help prevent accidents, equipment damage, and ensure reliable operation under various conditions.

Overall, the proposed technology aims to provide a highly efficient, user-friendly, and technologically advanced solution for solar power charging. By leveraging advanced microcontroller technology, precise sensors, power electronics components, user interfaces, wireless connectivity, and intelligent algorithms, the system maximizes solar power utilization, extends battery life, and offers convenient monitoring and control options.[10]

1.4 REFERENCES

[1] Matlab, Global Optimization Toolbox, “User's Guide (R2015b),”The MathWorks Inc., 2015.

[2] Y. A. Badamasi, "The working principle of an Arduino," *2014 11th International Conference on Electronics, Computer and Computation (ICECCO)*, Abuja, Nigeria, 2014, pp. 1-4,doi:10.1109/ICECCO.2014.6997578.
[3][Online].Available:https://www.infineon.com/dgdl/InfineonIR2104-DSvNAEN.pdf?fileId=5546d462533600a4015355c7c1c31671

[4] [Online]. Available: http://coder-tronics.com/c2000-solar-mppttutorial-pt1/.

[5] M. Wens and M. Steyaert, Design and implementation of fullyintegrated inductive DC-DC converters in standard CMOS, Dordrecht: Springer Netherlands, 2011. Doi: 10.1007/978-94-007-1436-6

[6] E. Babaei, M. E.S. Mahmoodieh, and M. Sabahi, "Investigating buck dc-dc converter operation in different operational modes and obtaining the minimum output voltage ripple considering filter size," Journal of Power Electronics, vol. 11, no. 6, pp. 793–800, November 2011. Doi: 10.6113/JPE.2011.11.6.793

[7] [Online]. Available: http://coder-tronics.com/tag/h-bridge/

[8] [Online]. Available:https://www.mouser.de/pdfdocs/BuckConverterDesignNote.pdf

[9] A. G. Al-Gizi, "Comparative study of MPPT algorithms under variable resistive load," in 2016 International Conference on Applied and Theoretical Electricity (ICATE 2016), Craiova, Romania, October 6–8, 2016. Doi: 10.1109/ICATE.2016.7754611

[10] A. Al-Gizi, S. Al-Chlaihawi, and A. Craciunescu, "Efficiency ofphotovoltaic maximum power point tracking controller Based on a fuzzy logic," Advances in Science, Technology and Engineering Systems Journal (ASTESJ), vol. 2, no. 3, pp. 1245–1251, 2017.

Chapter II: Existing Practice

The development of Maximum Power Point Tracking (MPPT) technology has played a pivotal role in optimizing the efficiency and performance of solar power systems. The history of MPPT spans several decades, witnessing significant advancements and breakthroughs in the quest for more effective energy conversion from solar panels. The concept of MPPT emerged in the 1970s as researchers and engineers recognized the importance of extracting the maximum power from photovoltaic (PV) modules. Initially, simple algorithms were employed to track the maximum power point (MPP) by adjusting the operating voltage or current of the solar panel. These early techniques, though limited in their adaptability to changing conditions, laid the foundation for further development. IN the 1980s, the Perturb and Observe (P&O) algorithm was introduced as a notable milestone in MPPT development. This algorithm continuously perturbs the operating point of the PV panel and observes the resulting change in power output. By iteratively adjusting the operating parameters, the algorithm eventually converges on the MPP, maximizing the energy harvested from the solar panel.[1]

Continuing the evolution of MPPT, the Incremental Conductance (IncCond) algorithm was introduced in the 1990s. This algorithm analyzes the power and its derivative with respect to voltage or current, allowing for more precise tracking of the MPP. The IncCond algorithm offers faster response times and improved accuracy, particularly in scenarios where environmental conditions fluctuate rapidly.Advancements in microcontroller and digital signal processing technologies in the late 20th century and early 21st century propelled MPPT technology further. The availability of powerful and cost-effective microcontrollers enabled the implementation of more sophisticated algorithms, such as fuzzy logic control and neural networks. These approaches aimed to enhance the adaptability and efficiency of MPPT systems by incorporating intelligent decision-making capabilities.In parallel, open-source platforms, including Arduino, gained popularity and contributed to the democratization of MPPT technology. Arduino-based MPPT solar charge controllers became accessible to a wider audience, enabling hobbyists, students, and professionals to experiment with MPPT algorithms and develop their own solar energy projects. [2]

Today, MPPT technology is an integral part of solar power systems, facilitating efficient energy conversion and ensuring maximum power extraction from PV modules. Advanced MPPT algorithms, combined with robust hardware and smart control features, have significantly increased the efficiency of solar charge controllers and improved the overall performance of solar installations.Ongoing research and development continue to refine MPPT techniques, with a focus on optimizing tracking algorithms, improving response times, and enhancing system reliability. The aim is to unlock the full potential of solar energy and drive the transition towards a sustainable future powered by renewable resources.The development of MPPT technology is a testament to human ingenuity and the relentless pursuit of efficient and clean energy solutions. As the global demand for renewable energy grows, MPPT technology will continue to play a crucial role in maximizing the power output of solar panels and unlocking the full potential of solar energy.[3]

2.1 Existing MPPT Circuit

The integration of the MPPT (Maximum Power Point Tracking) circuit had been a crucial step in our project, as it played a significant role in optimizing the power extraction from the solar cell. In this section, we will elaborate on the process and considerations involved in integrating the MPPT circuit into our solar energy system.

- Understanding MPPT:
  Before integrating the MPPT circuit, it had been essential to have a clear understanding of its functionality and benefits. MPPT is a technique used to maximize the power output from a solar cell by continuously tracking and operating it at its maximum power point. It achieved this by adjusting the operating parameters of the power converter, such as the duty cycle of the switches, to match the varying environmental conditions and load characteristics.

- MPPT Algorithm Selection:

  The first step in integrating the MPPT circuit had been to select an appropriate MPPT algorithm. Several algorithms, such as Perturb and Observe (P&O), Incremental Conductance (IncCond), and Fractional Open Circuit Voltage (FOCV), had commonly been used for MPPT. Each algorithm had its strengths and limitations, and the selection depended on factors such as system complexity, accuracy requirements, and cost considerations. After careful evaluation, we had chosen the Perturb and Observe algorithm for our project due to its simplicity and effectiveness in tracking the maximum power point.

- MPPT Circuit Design:

  Once the MPPT algorithm had been selected, we designed the MPPT circuit to implement the algorithm in our system. The MPPT circuit had consisted of sensors to measure the solar cell's voltage and current, a microcontroller or digital signal processor (DSP) to process the data and control circuitry to adjust the operating parameters of the power converter. The control circuitry communicated with the power converter's control circuit to optimize the power extraction from the solar cell.

- Sensor Integration:

  To implement the MPPT algorithm, accurate measurement of the solar cell's voltage and current had been crucial. We integrated appropriate sensors, such as voltage and current sensors, into the MPPT circuit to acquire real-time data. These sensors provided the necessary inputs to the MPPT algorithm, allowing it to continuously track the maximum power point of the solar cell.

- Microcontroller/DSP Integration:

  The MPPT algorithm required computational capabilities to process the sensor data and calculate the optimal operating parameters. We integrated a microcontroller or digital signal processor (DSP) into the MPPT circuit to perform these tasks. The microcontroller/DSP communicated with the sensors, retrieved the voltage and current data, executed the MPPT algorithm, and provided the control signals to adjust the power converter's operating parameters

- Communication with Power Converter:
  Integration of the MPPT circuit with the power converter had been essential to achieve efficient power extraction. The MPPT circuit communicated with the control circuitry of the power converter to adjust its operating parameters dynamically. This communication involved providing the control signals, such as the duty cycle of the switches, based on the calculations and decisions made by the MPPT algorithm. By optimizing the power converter's operation, the MPPT circuit ensured that the solar cell operated at its maximum power point.

- Testing and Validation:
  After integrating the MPPT circuit, thorough testing and validation had been conducted to verify its performance. This involved subjecting the system to various operating conditions and environmental factors, such as varying solar irradiance and temperature. We analyzed the MPPT circuit's ability to accurately track the maximum power point and evaluated its impact on the overall system efficiency. Testing also included scenarios with different load conditions to ensure the MPPT circuit's adaptability and effectiveness.

- Performance Optimization:
  Throughout the project, continuous performance optimization of the MPPT circuit had been carried out. This involved fine-tuning the MPPT algorithm, adjusting control parameters, and analyzing system data to identify any opportunities for further efficiency improvements. Optimization efforts focused on achieving faster tracking response, minimizing power losses, and enhancing the overall energy harvesting capability of the solar energy system.[4]

2.2 Existing Control Algorithms:

Developing control algorithms was a fundamental aspect of our project, as they played a pivotal role in regulating and optimizing the performance of our solar energy system. In this section, we will elaborate on the process and considerations involved in developing these control algorithms.

- System Analysis and Modeling:
  Before delving into the development of control algorithms, we first conducted a thorough analysis of our solar energy system and established mathematical models that captured its behavior. This included modeling the solar cell, buck-boost converter, battery, and other relevant components. By understanding the dynamics and relationships between these elements, we gained insights into the system's characteristics and identified key parameters for control.

- Objective Definition:
  With a clear understanding of the system, we defined the objectives of the control algorithms. These objectives encompassed aspects such as voltage regulation, maximum power point tracking, battery charging, and overall system stability. By setting clear objectives, we established a framework for the development of control strategies that would fulfill these goals.

- Algorithm Selection:
  Based on the objectives and system analysis, we researched and evaluated various control algorithms commonly used in solar energy systems. These included proportional-integral-derivative (PID) control, fuzzy logic control, adaptive control, and model predictive control, among others. The selection process considered factors such as accuracy, robustness, complexity, and computational requirements. After careful consideration, we chose a combination of a perturb and observe (P&O) algorithm for MPPT and a PID controller for voltage regulation and battery charging.

- Algorithm Design and Implementation:

  With the algorithm selection finalized, we proceeded to design and implement the control algorithms. For the MPPT algorithm, we designed the P&O algorithm to perturb the duty cycle of the buck-boost converter and observe the resulting power change. Based on this observation, the algorithm adjusted the duty cycle to track the maximum power point of the solar cell. The PID controller, on the other hand, was designed to regulate the output voltage of the buck-boost converter and control the charging process of the battery. The PID controller continuously adjusted the duty cycle based on the error between the desired and actual voltage values.

  In the project, a control algorithm was developed to optimize the performance of the buck converter. This algorithm allowed for efficient current draw from the solar cell and effective charging of the battery. The control algorithm was implemented using Arduino programming language and utilized the TimerOne library for timing and interrupt handling.[5]

2.3 REFERENCES

[1] A.S.M. Jiaul Hoque1*, Sheik Md. Kazi Nazrul Islam1, 2,Md. Abubakar Siddik1, Sabbir Ahamed1,"Design andImplementation of a Microcontroller Based 12V-7A/10A Smart Solar Battery Charge Controller".

[2] Marcelo Gradella Villalva, Jonas Rafael Gazoli, Ernesto Ruppert Filho, "Analysis and simulation of the P&O MPPT algorithm using a linearized PV array model," 10thBrazilian Power Electronics Conference (COBEP), Brazil,September 27 - October 1, 2009.

[3] Muhammad H. Rashid, "Power Electronics: Circuits,Devices Applications" Text Book.

[4] Dr. Anil S. Hiwale, Mugdha V. Patil, HemangiVinchurkar, "An Efficient MPPT Solar ChargeController", International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering Vol. 3, Issue 07, July 2004.

[5] Thesis "Maximum Power Point Tracking: Algorithm and Software Development", Delft University of Technology Faculty of EEMCS June 27, 2012.

Chapter III: Proposed Technology

3.1 Proposed Algorithm:

- Initialization:

Set the initial duty cycle of the DC-DC converter to a reasonable value (e.g., 50%).

Set the initial values for variables, such as previous power, current power, and direction.

- Sensing:

Measure the output voltage and current of the power source (e.g., solar panel) using appropriate sensors or by interfacing with the circuit.

- Calculating Power:

Multiply the measured voltage and current values to obtain the instantaneous power output of the power source.

- Compare Power:

Compare the current power value with the previous power value.

- Perturb and Observe (P&O) Algorithm:

V(k) and I(k) are measured to determine the voltage and current values at a given time, respectively.

The power, P(k), is calculated by multiplying V(k) and I(k) together.

If V(k) is not greater than V(k-1), it means that the voltage has not increased. In this case, the system takes the "No" path.

However, if V(k) is greater than V(k-1), it means that the voltage has increased. In this scenario, the system takes the "Yes" path.

If P(k) is not greater than P(k-1), it means that the power has not increased. In this case, the system takes the "No" path.

If V(k) is still greater than V(k-1), it implies that the voltage is continuously increasing. In order to regulate it, the system needs to increase the pulse width modulation (PWM) value.

On the other hand, if the power has increased, the system takes the "Yes" path.

In the event that V(k) is not greater than V(k-1), the system decreases the PWM value to reduce the voltage.

If the voltage and power values remain the same as the previous step (V(k-1) = V(k) and P(k-1) = P(k)), the system stabilizes and continues with the existing PWM value.

If V(k) is still greater than V(k-1), it indicates that the voltage is continuously increasing. In this case, the system increases the PWM value further to regulate the voltage.

- Feedback Loop:

Repeat steps 2-5 in a continuous feedback loop.

Continuously sense the voltage and current, calculate the power, and adjust the duty cycle of the DC-DC converter based on the results of the P&O algorithm.

The loop ensures that the MPPT circuit dynamically responds to changes in environmental conditions and maintains the power source operating at its maximum power point.

The goal of the MPPT algorithm is to iteratively adjust the duty cycle of the DC-DC converter based on the power measurements. By doing so, it tries to find and track the maximum power point, where the power output of the source is optimized.

It's important to note that the P&O algorithm is one of the commonly used MPPT techniques, but there are other algorithms as well, such as Incremental Conductance, Fractional Open Circuit Voltage, and Perturb and Observe with Variable Step Size. These algorithms may offer more advanced features and improved performance under certain conditions.

The specific implementation of the MPPT algorithm may vary based on the hardware setup, chosen MPPT technique, and the capabilities of the microcontroller or controller used in the circuit.

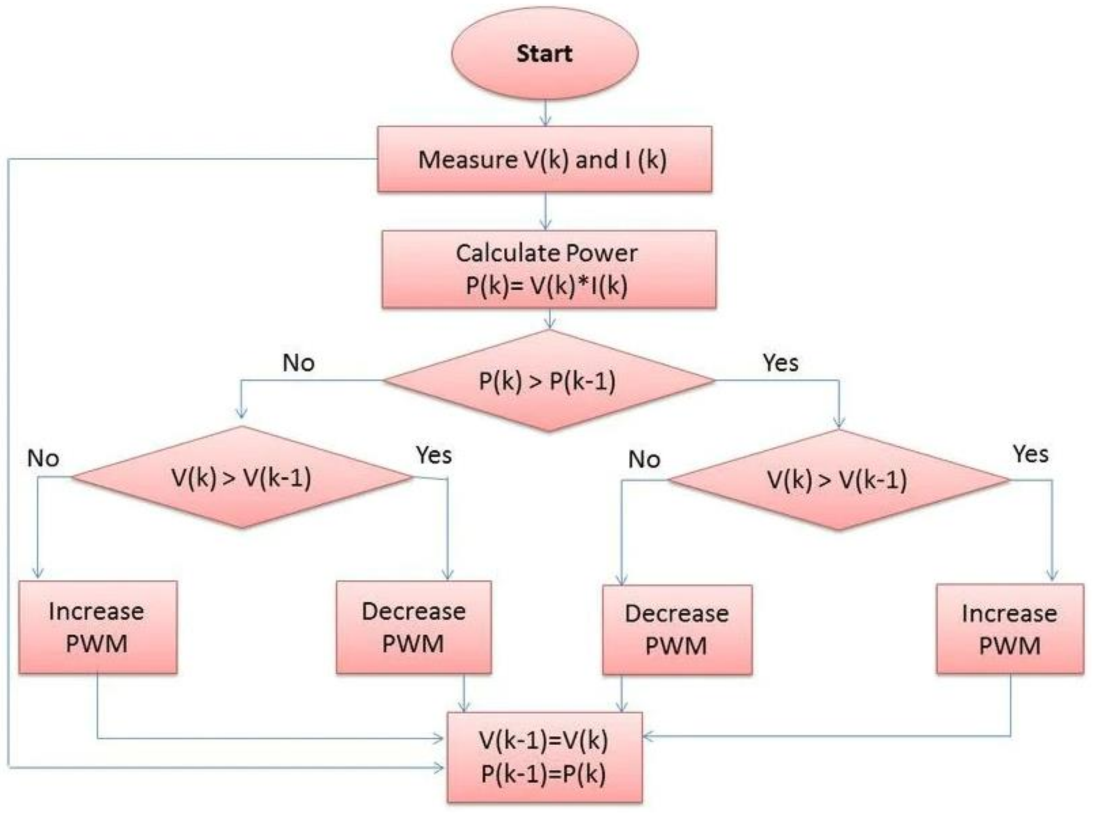

Fig 3.1: Algorithm of MPPT circuit

3.2 Proposed Circuit Diagram:

Connect the solar panel to a DC-DC converter (typically a buck converter) that steps down the voltage to match the desired output voltage.

Connect the output of the DC-DC converter to a load or a battery for power storage.

Connect the output voltage and current sensors to the Arduino for measurement.

- Initialization:

Set the initial duty cycle of the DC-DC converter to a reasonable value (e.g., 50%).

Set the initial values for variables such as previous power, current power, and direction.

- Sensing:

Read the output voltage and current values from the voltage and current sensors connected to the solar panel using the Arduino's ADC.

- Calculating Power:

Multiply the measured voltage and current values to calculate the instantaneous power output of the solar panel.

- MPPT Algorithm (Perturb and Observe):

Compare the current power value with the previous power value.

If the current power is greater than the previous power, increase the duty cycle slightly and continue in the same direction.

If the current power is less than the previous power, decrease the duty cycle slightly and change the tracking direction.

Repeat steps 3-5 to continuously track the maximum power point.

- Controlling the DC-DC Converter:

Use the Arduino's PWM output to generate a control signal for the DC-DC converter's gate driver circuit.

Adjust the duty cycle of the PWM signal based on the MPPT algorithm's output.

The duty cycle determines the amount of power transferred from the solar panel to the load.

- Feedback Loop:

Continuously monitor the power output and adjust the duty cycle of the DC-DC converter in response to changes in the measured power.

Repeat steps 3-7 in a feedback loop to dynamically track the maximum power point as environmental conditions change.

The Arduino acts as the controller in the MPPT circuit, continuously measuring the solar panel's voltage and current, calculating power, and adjusting the duty cycle of the DC-DC converter to maintain maximum power extraction. The P&O algorithm allows the system to track changes in the power output and adjust the operating point accordingly.

It's worth mentioning that this is a simplified explanation, and there are more advanced MPPT algorithms and techniques available. The specific implementation details and circuit design may vary depending on the requirements and components used in your project.

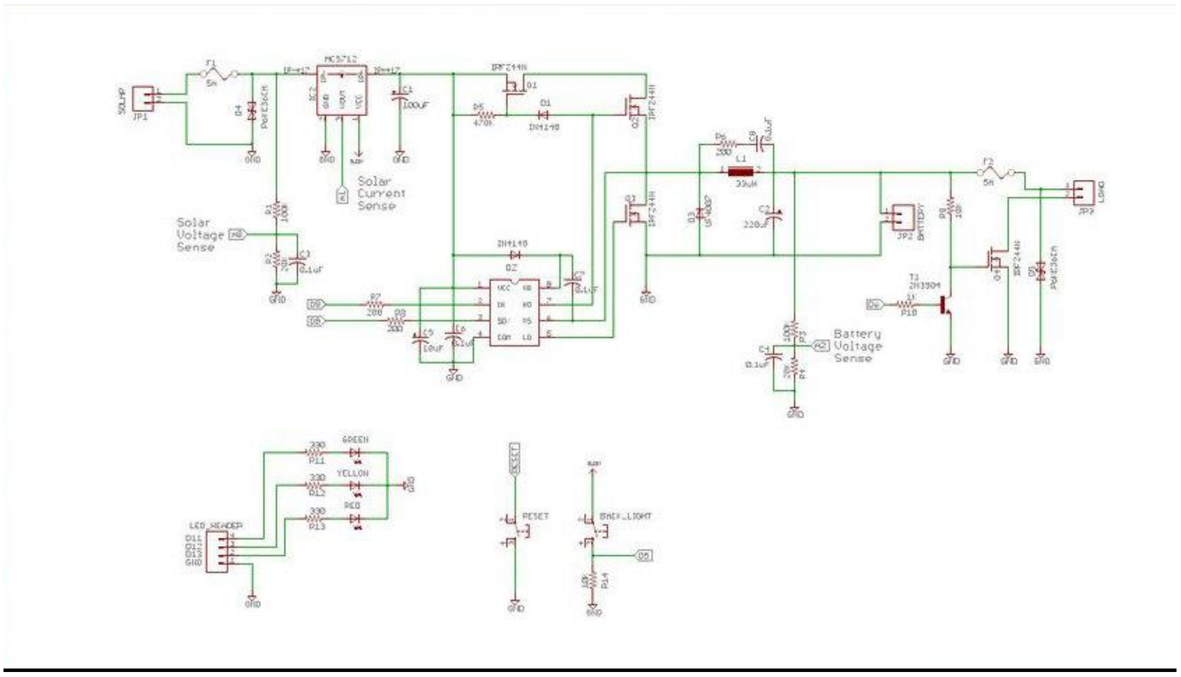

Fig 3.2: Circuit Diagram of MPPT charging circuit

3.3. Component Selection:

Table 3.1: Component's name

SI N0.	Components Name	Rating	Quantity
1.	Arduino Nano		1
2.	Current Sensor	ACS712-5A	1
3.	Buck Converter	LM2596	1
4.	LCD display		1
5.	MOSFETs	IRFZ44N	4
6.	MOSFET driver	IR2104	1
7.	3.3V Linear regulator	AMS 1117	1
8.	Transistor	2N2222	1
9.	Diodes	2x IN4148 , 1 x UF4007	3
10.	TVS diode	P6KE36CA	2
11.	Resistors	3 x200R ,3 x330R,1 x 1K, 2 x 10K, 2 x	14
12.	Capacitors	4 x 0.1 uF, 3 x 10uF ,1 x100 uF ,1x220uF	9
13.	Inductor	33uH -5A	1
14.	LEDs	1 x Red ,1 x Yellow ,1 x Green	3
15.	Prototype Board		1
16.	Wires and Jumper wires	Female -Female	35
17.	Header Pins	Male Straight ,female , Right angle	
18.	DIP Socket	8 pin	1
19.	Push Switch		2
20.	Fuses	5A	2
21.	Fuse Holders		2
22.	Rocker /Toggle Switch		1
23.	Female USB port		1
24.	JST connector	2pin male -female	1
25.	Heat Sinks		1
26.	Screws/Nuts/Bolts		

Chapter IV: Testing and Experiment

Solar Panel Rating: The proposed technology incorporates a solar panel with a rating of 20 Watts peak (20Wp). This rating indicates the maximum power output that the solar panel can generate under standard test conditions.

Output Voltage: The output voltage of the solar panel is an important parameter that indicates the electrical potential difference it can deliver. It represents the voltage level of the generated electricity. Monitoring the output voltage helps determine if the solar panel is generating the expected electrical potential and if it is within the safe operating range.

Output Current: The output current of the solar panel represents the amount of electrical current it can deliver to a connected load. It indicates the flow of electricity from the solar panel to the load. Monitoring the output current helps assess the performance of the solar panel and ensures that it is providing the desired power output.

Measurement Devices: To check the output voltage and output current of the solar panel, suitable measuring devices are used. Common devices include multimeters or power analyzers capable of accurately measuring DC voltage and current. These devices provide readings of the output voltage and output current when connected to the positive and negative terminals of the solar panel.

Monitoring System: In the proposed technology, it is recommended to incorporate a monitoring system that continuously measures and logs the output voltage and output current of the solar panel. This system can be designed to provide real-time data, analysis, and notifications. It enables performance analysis, fault diagnosis, and system optimization.

Benefits of Monitoring: Regularly checking the output voltage and output current allows for early detection of any performance issues or deviations from expected values. It facilitates timely maintenance or troubleshooting, ensuring optimal power generation and system efficiency. Real-time monitoring can provide alerts or notifications in case of abnormal voltage or current levels, allowing prompt action to address potential issues.

Contribution to System Performance: Monitoring the output voltage and output current of the solar panel contributes to the overall performance and longevity of the solar charging system. By ensuring reliable and efficient operation, it maximizes the power generation potential and enhances the system's capability to meet power demands.

In summary, the proposed technology incorporates a solar panel with a rating of 20Wp and emphasizes the importance of checking the output voltage and output current. This monitoring ensures optimal performance, early issue detection, and system efficiency. By implementing a monitoring system, real-time data analysis and notifications can be utilized, enhancing the reliability and longevity of the solar charging system

4.1 Testing of Solar Pannel:

Electrical Performance: Solar cell testing begins with evaluating its electrical performance. Parameters such as open-circuit voltage (Voc), short-circuit current (Isc), maximum power point voltage (Vmp), and maximum power point current (Imp) are measured. These measurements help determine the cell's efficiency and power output under different irradiance and temperature conditions.

I-V Characteristic Curve: The I-V characteristic curve represents the relationship between voltage and current output of the solar cell. By measuring and plotting this curve, the cell's performance and behavior at different operating points can be analyzed. The curve can be used to determine the maximum power point (MPP) and to evaluate the cell's performance under varying environmental conditions.

Efficiency Testing: Solar cell efficiency is a crucial factor in determining its overall performance. Efficiency testing involves calculating the ratio of electrical power output to the incident solar power. It helps assess the cell's ability to convert sunlight into usable electrical energy.

Temperature Coefficient: The temperature coefficient of a solar cell indicates how its electrical performance is affected by temperature changes. Testing the temperature coefficient helps understand how the cell's voltage, current, and power output vary with temperature, allowing for better system design and performance prediction.

- **For 10KΩ Pot**

Table 4.1: Testing of Solar Panel

Sl. No	Current (mA)	Voltage (DC) (V)
1.	0	18.30
2.	0.02	18.26
3.	0.03	18.25
4.	0.04	18.23
5.	0.05	18.10
6.	0.08	18.19
7.	0.13	18.20
8.	0.23	18.22
9.	0.45	18.10

- **For 1KΩ Pot** Table 4.2: Testing of Solar Panel

Sl. No	Current (mA)	Voltage (DC) (V)
1.	0.16	18.26
2.	0.19	18.30
3.	0.24	18.42
4.	0.29	18.36
5.	0.34	18.38
6.	0.41	18.22
7.	0.50	18.37
8.	0.75	18.29
9.	0.99	18.32

In the proposed technology, the solar panel has an open circuit voltage of 21.0 volts and a short circuit voltage of 16.8 volts. These voltage values are significant in understanding the performance and characteristics of the solar panel. Here's an elaboration on their importance:

Open Circuit Voltage (Voc): The open circuit voltage is the voltage across the output terminals of the solar panel when there is no load connected. In this case, the solar panel has an open circuit voltage of 21.0 volts. The open circuit voltage represents the maximum voltage that the solar panel can produce. It is influenced by various factors such as the number of solar cells in the panel, the materials used, and the environmental conditions. Monitoring the open circuit voltage helps in assessing the panel's voltage output capabilities and understanding its overall performance.

Short Circuit Voltage (Vsc): The short circuit voltage refers to the voltage across the output terminals of the solar panel when the terminals are short-circuited, i.e., directly connected without any load. In this case, the solar panel has a short circuit voltage of 16.8 volts. The short circuit voltage represents the maximum current that the solar panel can deliver. It is influenced by factors such as the solar cell design, the amount of sunlight hitting the panel, and the temperature. The short circuit voltage is an important parameter to consider when designing the system and selecting appropriate load components.

By knowing the open circuit voltage and short circuit voltage, we can make certain assessments and calculations:

Voltage Output Range: The voltage output range of the solar panel can be estimated based on the open circuit voltage. In this case, the panel can generate a voltage between 0 volts (short circuit condition) and 21.0 volts (open circuit condition). This information is valuable for designing the system and selecting compatible voltage-sensitive components.

Load Matching: The difference between the open circuit voltage and short circuit voltage, in this case, is 4.2 volts (21.0V - 16.8V). This voltage drop occurs when a load is connected to the solar panel. Understanding this difference is crucial for load matching, which involves selecting a load that allows the solar panel to operate at its maximum power point. Load matching ensures optimal power transfer from the solar panel to the connected load.

Efficiency Analysis: The open circuit voltage and short circuit voltage, along with the actual operating voltage, can be used to assess the efficiency of the solar panel. Efficiency calculations involve comparing the actual power output of the panel to its maximum possible power output. By measuring the voltage and current at various operating points, the efficiency of the solar panel can be analyzed and improvements can be made if necessary.

In the proposed technology, the system includes inductors with specific values. Here's the information you requested:

4.2 Testing of Inductor:

Overall Inductor Value: The overall inductor value for the system is 34 μH. This indicates the combined inductance provided by multiple inductors connected together in the circuit.

Inductor 1: The first leg of the system has an inductor value of 53 μH.

Inductor 2: The second leg of the system has an inductor value of 54 μH.

Inductor 3: The third leg of the system has an inductor value of 80 μH.

Inductor 4: The fourth leg of the system has an inductor value of 94 μH.

Parallel Connection: The second and fourth legs are connected in parallel, resulting in a combined inductance of 34 μH.

The inductors in the system serve various purposes, such as filtering out high-frequency noise, storing energy, and regulating current flow. By combining multiple inductors with different values, the overall inductance can be tailored to meet specific system requirements. Inductors are commonly used in power electronics, electrical circuits, and signal processing applications.

It's important to note that the values and connections mentioned above are specific to the proposed technology. The selection of inductor values and their connections depends on the system's design, intended functionality, and desired electrical characteristics. These values and connections should be carefully chosen and validated to ensure optimal performance and reliability of the system.

In the proposed technology, the system includes the following components and connections:

MOSFET Gate Trigger: The gate of the first MOSFET is connected to Arduino Pin 9. This connection enables the Arduino to control the switching of the MOSFET.

DC Source: The DC power source is connected to the system through Pin 8 of the Arduino board. This connection provides the necessary power for the system.

First MOSFET: The first MOSFET is connected as follows:

The source pin (V-) of the MOSFET is connected to the negative terminal of the power source.

The drain pin (V in) of the MOSFET is connected to the positive terminal of the battery.

The signal pin of the MOSFET is connected to the Arduino Pin 1 (Sig pin).

Second MOSFET: The second MOSFET is connected as follows:

The source pin of the MOSFET is connected to the ground.

The drain pin of the MOSFET is connected to the negative terminal of the battery.

The signal pin (Vro) of the MOSFET is connected to the Arduino.

4.3 Testing of Gate triggering:

The gate triggering of the IRF520 MOSFET driver is tested to ensure proper operation and reliability in various electronic applications. The testing process involves evaluating the response of the MOSFET to control signals applied to its gate pin. The gate triggering testing of the IRF520 MOSFET driver includes the following steps:

Test Setup: The test circuit is set up, consisting of the IRF520 MOSFET driver, a power source (DC), and a control signal source (such as an Arduino or microcontroller).

Power Supply: The power supply (DC) is connected to the appropriate pins of the IRF520 MOSFET driver. The power supply voltage is ensured to be within the specified operating range of the driver.

Control Signal Source: The control signal source (Arduino or microcontroller) is connected to the gate pin of the IRF520 MOSFET driver. The control signal levels (high and low) are ensured to be compatible with the driver's input requirements.

Gate Voltage Testing: A series of control signals with varying voltage levels are applied to the gate pin. The voltage is gradually increased from a low level (e.g., 0V) while observing the behavior of the MOSFET driver.

Gate Triggering: As the control signal voltage reaches the threshold voltage of the MOSFET driver, the MOSFET is expected to start conducting. The behavior of the MOSFET being properly triggered and conducting according to the applied control signals is monitored by observing the output voltage and current.

Gate Turn-Off: The gate turn-off characteristics are tested by applying a low voltage or turning off the control signal. The output voltage and current are monitored to ensure that the MOSFET turns off and stops conducting when the control signal is removed or reduced.

Timing and Frequency: The response of the MOSFET driver to control signal timing and frequency variations is tested. Control signals with different pulse widths and frequencies are applied to evaluate the driver's ability to handle different switching speeds.

Overload and Protection Testing: Overload conditions are simulated or excessive voltage is applied to evaluate the protection features of the MOSFET driver. The ability of the driver to handle transient and overload conditions without malfunctioning or damaging the MOSFET is ensured.

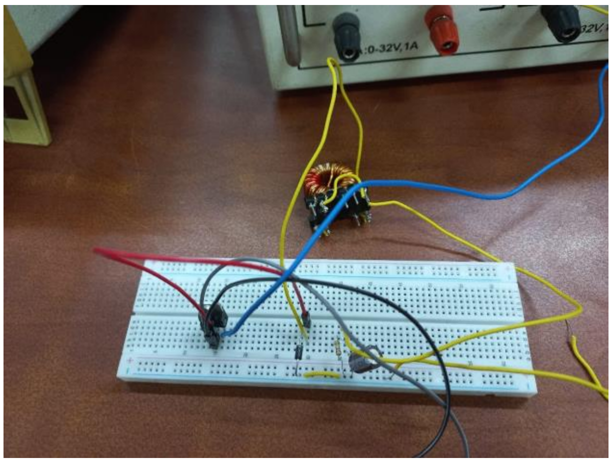
Fig 4.1: Testing of Buck Converter

Temperature Testing: The performance and behavior of the MOSFET driver under different temperature conditions are monitored. Its ability to withstand heat and maintain stable operation within the specified temperature limits is tested.

Data Logging and Analysis: The test results, including gate voltage, output voltage, output current, and any observed abnormalities or deviations, are recorded. The data is analyzed to identify any issues or areas for improvement.

Proper guidelines and recommendations from the datasheet and application notes of the IRF520 MOSFET driver are referred to during the testing process. Accurate and reliable results are ensured by adhering to proper testing procedures and using suitable measurement equipment.

Thorough testing of the gate triggering of the IRF520 MOSFET driver allows for verification of its compatibility with control signals, evaluation of its switching characteristics, and assurance of its proper operation in electronic circuits and systems. Issues or limitations can be identified through proper testing, enabling optimization and improvements in circuit performance and reliability.

Table 4.3: MOSFET gate triggering

Duty Cycle	Output Voltage (V)
100%	5.50
75%	5.60
50%	5.70
25%	5.76

4.4 Testing of BUCK CONVERTER :

Duty Cycle Variation: The buck converter in the system has a variable duty cycle, which can be controlled by the Arduino. The duty cycle can vary between 25%, 50%, and 75%. This variation allows for adjusting the output voltage or power delivery of the buck converter.

Table 4.4:Testing of Buck Converter

Duty Cycle	Timer	Output Voltage (V)
	256	1.506
	300	1.687
25%	350	1.916
	400	2.12
	450	2.32
	500	2.51
50%	550	2.67
	600	2.84
	650	3.01
	700	3.18
75%	750	3.38
	800	3.61

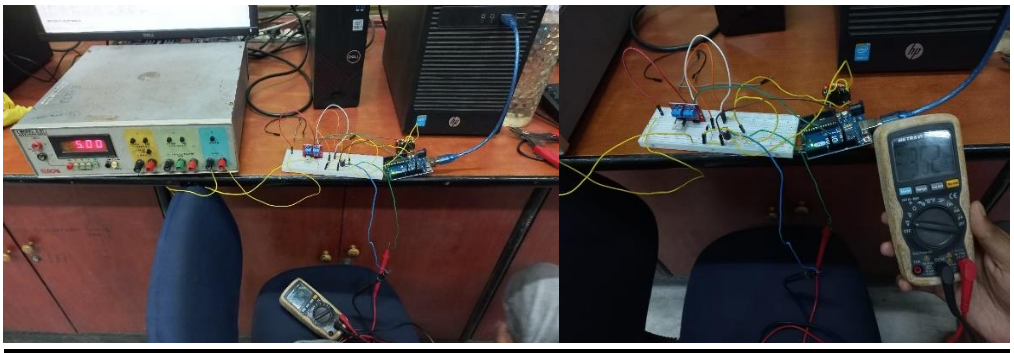

Fig 4.2, 4.3 : Testing of Buck Converter

Chapter V: Scope for future work & Conclusion

- ## 5.1 Scope for future work:

The future scope of your technology holds significant potential for advancement and innovation. Here are some potential areas of development and application:

Increased Efficiency: Future research and development can focus on enhancing the efficiency of your technology. This could involve improving the algorithms used for power tracking, optimizing the charging process, and reducing energy losses. By increasing efficiency, your technology can maximize the power output from solar panels and improve overall system performance.

Scalability and Flexibility: There is a scope for making your technology more scalable and adaptable to different system sizes and configurations. This could involve designing modular components that can be easily integrated into various setups, whether it's residential, commercial, or utility-scale installations. Providing flexible options for system customization and expansion can further increase the appeal and versatility of your technology.

Integration with Energy Storage: The integration of your technology with energy storage systems, such as batteries or supercapacitors, can enhance its capabilities. This integration would enable energy storage during periods of excess solar generation and facilitate power delivery during times of low sunlight or high demand. Future developments can focus on seamless integration, advanced control algorithms, and optimization techniques for maximizing the benefits of energy storage.

Intelligent Energy Management: Developing intelligent energy management systems that incorporate your technology can optimize the utilization of solar power. This could involve advanced algorithms and machine learning techniques to predict energy demand, manage power distribution, and prioritize energy usage. Intelligent energy management can help reduce energy costs, improve grid stability, and promote sustainable energy consumption.

Internet of Things (IoT) Integration: Integrating your technology with IoT platforms and devices can enable remote monitoring, control, and data analytics. This integration can provide real-time insights into system performance, enable proactive maintenance, and facilitate smart energy management. IoT integration can also open up opportunities for enhanced connectivity, interoperability, and integration with other smart devices and systems

Grid Integration and Virtual Power Plants: Your technology can play a vital role in grid integration and the creation of virtual power plants. By aggregating multiple solar charging systems, they can act as a distributed energy resource and contribute to grid stability and resilience. Future developments can focus on developing communication protocols, grid interaction standards, and advanced control mechanisms to facilitate seamless integration into the grid infrastructure.

Electrification of Transportation: The future scope of your technology extends to the electrification of transportation. By integrating your technology with electric vehicle (EV) charging infrastructure, it can contribute to sustainable transportation solutions. This could involve developing smart charging algorithms, vehicle-to-grid (V2G) capabilities, and interoperability with EV charging networks.

Environmental Monitoring and Sustainability: Your technology can be extended beyond power generation to incorporate environmental monitoring and sustainability features. For example, integrating sensors for measuring air quality, temperature, and humidity can provide valuable environmental data. This data can be used for optimizing system performance, identifying potential issues, and promoting sustainable practices.

Global Applications and Rural Electrification: There is a significant scope for applying your technology in off-grid and rural electrification projects. By providing reliable and affordable solar charging solutions, your technology can help bridge the energy access gap and improve the livelihoods of communities in remote areas. Future developments can focus on adapting your technology to specific regional needs, addressing challenges related to infrastructure, and promoting sustainable development.

Collaboration and Partnerships: The future scope of your technology also lies in collaboration and partnerships with other industry stakeholders. Collaborating with solar panel manufacturers, energy storage providers, system integrators, and utility companies can facilitate knowledge exchange, innovation, and market penetration. By fostering partnerships, you can leverage complementary expertise and resources to accelerate the adoption and impact of your technology.

In summary, the future scope of your technology encompasses areas such as increased efficiency, scalability and flexibility, integration with energy storage and intelligent energy management, IoT integration, grid integration and virtual power plants, electrification of transportation, environmental monitoring and sustainability, global applications and rural electrification, and collaboration and partnerships. By focusing on these areas, you can drive innovation, address emerging market needs, and contribute to the widespread adoption of sustainable solar charging solutions.

- **5.2 Conclusion:**

In conclusion, the proposed technology integrates a buck converter and utilizes Arduino-controlled MOSFETs to optimize power conversion and control. By leveraging the Arduino platform, precise and flexible control over the MOSFETs' switching behavior and duty cycle can be achieved. Efficient power conversion and regulation are enabled, catering to a wide range of power requirements.

The MOSFETs are connected to the Arduino's digital pins, serving as gate triggers for controlling the power flow. The DC power source is connected to the Arduino's P8 pin, while the signal input for the first MOSFET is received from the Arduino's P9 pin. The V- terminal of the first MOSFET is connected to the ground, and its V in terminal is connected to the positive terminal of the battery. Similarly, the ground is connected to the V- terminal of the second MOSFET, and its signal pin, Vro, receives input from the Arduino.

By varying the duty cycle using pulse-width modulation (PWM) control, the buck converter's duty cycle can be adjusted to 25%, 50%, or 75%. This variability allows for precise control over the output voltage or power delivered by the buck converter, making it highly adaptable to different power requirements and load conditions.

The optimized configuration of the proposed technology offers several benefits. Power efficiency is enhanced by minimizing energy losses during conversion and regulation processes. Additionally, the Arduino-based control system provides accurate and reliable control over the power flow, ensuring stable and consistent performance.

This technology has significant implications for various applications, including renewable energy systems, battery charging, and voltage regulation. Its ability to efficiently manage power and adapt to varying load conditions makes it an attractive solution for optimizing energy consumption and improving overall system performance.

Overall, the integration of the buck converter, Arduino control, and MOSFETs in this proposed technology opens up new possibilities for intelligent power management in diverse fields. It paves the way for more efficient and sustainable energy utilization, driving advancements in the realm of power electronics and contributing to a greener and smarter future.

APPENDIX A   Test Code

Buck Converter test code

```
#include <TimerOne.h>
void setup()
{
  // Initialize the digital pin as an output.
  // Pin 13 has an LED connected on most Arduino boards
  pinMode(13, OUTPUT);
  pinMode(9, OUTPUT);
  pinMode(8, OUTPUT);
  digitalWrite(8, HIGH);
  Timer1.initialize(20); // set a timer of length 8uS
  //Timer1.attachInterrupt( timerIsr ); // attach the service routine here
  //Set duty cycle
  //Timer1.pwm(9,256);  // 25% duty cycle
 // Timer1.pwm(9, 512); // 50% duty cycle
  Timer1.pwm(9, 768);  //  75% duty cycle
}
void loop()
{
  // Main code loop
  // TODO: Put your regular (non-ISR) logic here
}
/// --------------------------
/// Custom ISR Timer Routine
/// --------------------------
void timerIsr()
{
    // Toggle LED
    //digitalWrite( 13, digitalRead( 13 ) ^ 1 );
}
```

Code for dc voltage measurement by using a voltage divider circuit

```
 int temp=0;
 float sum =0;
 float VOLTS_SCALE =0;
 float volt=0;
 void setup()
 {
 Serial.begin(9600);
 }
 void loop()
 {
   for(int i = 0; i < 100; i++)  // loop through reading raw adc values 100 number of times
 {
   temp=analogRead(A0);     // read the input pin
   sum += temp;             // store sum for averaging
   delay(2);
 }
  sum=sum/100;              // divide sum by 100 to get average
 // Calibration  for Voltage
  VOLTS_SCALE = 0.00488 * (120/20); // The voltage divider resistors are R1=100k and
R2=20k // 5/1024 =0.00488
  volt = VOLTS_SCALE * sum ;
  Serial.print(volt);
 Serial.println("V");
 delay(500);
 }
```

Code for current measurement by using a ACS712 (5A) hall effect current sensor

```
 int temp=0;
 float sum =0;
 float AMPS_SCALE =0;
 float amps=0;
 void setup()
 {
 Serial.begin(9600);
 }
 void loop()
 {
   for(int i = 0; i < 100; i++)  // loop through reading raw adc values 100 number of times
 {
   temp=analogRead(A1);     // read the input pin
   sum += temp;             // store sum for averaging
   delayMicroseconds(50);
 }
  sum=sum/100;              // divide sum by 100 to get average
 // Calibration  for current
  AMPS_SCALE= 0.00488/ 0.185;    //5/1024 = 0.00488  // Sensitivity = 185mV
  amps = AMPS_SCALE* sum - 13.51;   // 2.5/0.185 = 13.51
 Serial.print(amps);
 Serial.println("A");
 delay(500);
 }
```

ARDUINO MPPT SOLAR CHARGE CONTROLLER (Version-3)

```
//  This code is for an arduino Nano based Solar MPPT charge controller.
//  This code is a modified version of sample code from www.timnolan.com
//  updated 06/07/2015
//  Mods by Aplavins 19/06/2015
////  Specifications :  //////////////////////////////////////////////////////////////////////////////////////////////////////
//    1.Solar panel power = 50W
//    2.Rated Battery Voltage= 12V ( lead acid type )
//    3.Maximum current = 5A                                                                              //
//    4.Maximum load current =10A
//
//    5. In put Voltage = Solar panel with Open circuit voltage from 17 to 25V
//

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
#include "TimerOne.h"                // using Timer1 library from
http://www.arduino.cc/playground/Code/Timer1
#include <LiquidCrystal_I2C.h>       // using the LCD I2C Library from
https://bitbucket.org/fmalpartida/new-liquidcrystal/downloads
#include <Wire.h>
#include <SoftwareSerial.h>          // using the Software Serial library Ref :
http://www.arduino.cc/en/Reference/SoftwareSerialConstructor
//------------------------------------------------------------------------------------------------------
//////// Arduino pins Connections//////////////////////////////////////////////////////////////////////////////////
// A0 - Voltage divider (solar)
// A1 - ACS 712 Out
// A2 - Voltage divider (battery)
// A4 - LCD SDA
// A5 - LCD SCL
// D2 - ESP8266 Tx
// D3 - ESP8266 Rx through the voltage divider
// D5 - LCD back control button
// D6 - Load Control
// D8 - 2104 MOSFET driver SD
// D9 - 2104 MOSFET driver IN
// D11- Green LED
// D12- Yellow LED
// D13- Red LED
// Full scheatic is given at
http://www.instructables.com/files/orig/F9A/LLR8/IAPASVA1/F9ALLR8IAPASVA1.pdf
///////// Definitions //////////////////////////////////////////////////////////////////////////////////////////
// Turn this on to use the ESP8266 chip. If you set this to 0, the periodic updates will not
happen
#define ENABLE_DATALOGGER 0
// Load control algorithm
// 0 - NIGHT LIGHT: Load ON when there is no solar power and battery is above LVD (low
voltage disconnect)
```

```
// 1 - POWER DUMP: Load ON when there is solar power and the battery is above
BATT_FLOAT (charged)
#define LOAD_ALGORITHM 0
#define SOL_AMPS_CHAN 1              // Defining the adc channel to read solar amps
#define SOL_VOLTS_CHAN 0             // defining the adc channel to read solar volts
#define BAT_VOLTS_CHAN 2             // defining the adc channel to read battery volts
#define AVG_NUM 8                    // number of iterations of the adc routine to average the
adc readings
// ACS 712 Current Sensor is used. Current Measured = (5/(1024 *0.185))*ADC - (2.5/0.185)
#define SOL_AMPS_SCALE  0.026393581       // the scaling value for raw adc reading to
get solar amps   // 5/(1024*0.185)
#define SOL_VOLTS_SCALE 0.029296875       // the scaling value for raw adc reading to
get solar volts  // (5/1024)*(R1+R2)/R2 // R1=100k and R2=20k
#define BAT_VOLTS_SCALE 0.029296875       // the scaling value for raw adc reading to
get battery volts

#define PWM_PIN 9                    // the output pin for the pwm (only pin 9 avaliable for timer
1 at 50kHz)
#define PWM_ENABLE_PIN 8             // pin used to control shutoff function of the IR2104
MOSFET driver (hight the mosfet driver is on)
#define PWM_FULL 1023                // the actual value used by the Timer1 routines for 100%
pwm duty cycle
#define PWM_MAX 100                  // the value for pwm duty cyle 0-100%
#define PWM_MIN 60                   // the value for pwm duty cyle 0-100% (below this value
the current running in the system is = 0)
#define PWM_START 90                 // the value for pwm duty cyle 0-100%
#define PWM_INC 1                    //the value the increment to the pwm value for the ppt
algorithm
#define TRUE 1
#define FALSE 0
#define ON TRUE
#define OFF FALSE
#define TURN_ON_MOSFETS digitalWrite(PWM_ENABLE_PIN, HIGH)      // enable
MOSFET driver
#define TURN_OFF_MOSFETS digitalWrite(PWM_ENABLE_PIN, LOW)      // disable
MOSFET driver
#define ONE_SECOND 50000             //count for number of interrupt in 1 second on interrupt
period of 20us
#define LOW_SOL_WATTS 5.00           //value of solar watts // this is 5.00 watts
#define MIN_SOL_WATTS 1.00           //value of solar watts // this is 1.00 watts
#define MIN_BAT_VOLTS 11.00          //value of battery voltage // this is 11.00 volts
#define MAX_BAT_VOLTS 14.10          //value of battery voltage// this is 14.10 volts
#define BATT_FLOAT 13.60             // battery voltage we want to stop charging at
#define HIGH_BAT_VOLTS 13.00         //value of battery voltage // this is 13.00 volts
#define LVD 11.5                     //Low voltage disconnect setting for a 12V system
#define OFF_NUM 9                    // number of iterations of off charger state
//------------------------------------------------------------------------------------------------
//Defining led pins for indication
#define LED_GREEN 11
#define LED_YELLOW 12
```

```
#define LED_RED 13
//----------------------------------------------------------------------------------------------------
// Defining load control pin
#define LOAD_PIN 6       // pin-2 is used to control the load
//----------------------------------------------------------------------------------------------------
// Defining lcd back light pin
#define BACK_LIGHT_PIN 5       // pin-5 is used to control the lcd back light
// ---------------------------For ESP8266-----------------------------------------------------------
// replace with your channel's thingspeak API key
String apiKey = "DPK8RMTFY2B1XCAF";
// connect 2 to TX of Serial USB
// connect 3 to RX of serial USB
SoftwareSerial ser(2,3); // RX, TX
//--------------------------------------------------------------------------------------------------------
//-----------------------------------------------------------------------------------------------------
///////////////////////////////////////BIT MAP ARRAY//////////////////////////////////////////////////
//------------------------------------------------------------------------------------------------------
byte battery_icons[6][8]=
{{
  0b01110,
  0b11011,
  0b10001,
  0b10001,
  0b10001,
  0b10001,
  0b10001,
  0b11111,
},
{
  0b01110,
  0b11011,
  0b10001,
  0b10001,
  0b10001,
  0b10001,
  0b11111,
  0b11111,
},
{
  0b01110,
  0b11011,
  0b10001,
  0b10001,
  0b10001,
  0b11111,
  0b11111,
  0b11111,
},
{
  0b01110,
  0b11011,
  0b10001,
  0b11111,
  0b11111,
  0b11111,
  0b11111,
  0b11111,
},
{
  0b01110,
  0b11011,
  0b11111,
  0b11111,
  0b11111,
  0b11111,
  0b11111,
  0b11111,
},
{
  0b01110,
  0b11111,
  0b11111,
  0b11111,
  0b11111,
  0b11111,
  0b11111,
  0b11111,
}};
#define SOLAR_ICON 6
byte solar_icon[8] = //icon for termometer
{
  0b11111,
  0b10101,
  0b11111,
  0b10101,
  0b11111,
  0b10101,
  0b11111,
  0b00000
};
#define PWM_ICON 7
byte _PWM_icon[8]=
{
  0b11101,
  0b10101,
  0b10101,
  0b10101,
  0b10101,
  0b10101,
  0b10101,
  0b10111,
};
byte backslash_char[8]=
{
  0b10000,
  0b10000,
  0b01000,
  0b01000,
  0b00100,
  0b00100,
  0b00010,
  0b00010,
};
//-------------------------------------------------------------------------------------------------------

// global variables
float sol_amps;                        // solar amps
float sol_volts;                      // solar volts
float bat_volts;                      // battery volts
float sol_watts;                      // solar watts
float old_sol_watts = 0;              // solar watts from previous time through ppt routine
unsigned int seconds = 0;             // seconds from timer routine
unsigned int prev_seconds = 0;        // seconds value from previous pass
unsigned int interrupt_counter = 0;   // counter for 20us interrrupt
unsigned long time = 0;               // variable to store time the back light control button was
pressed in millis
int delta = PWM_INC;                  // variable used to modify pwm duty cycle for the ppt
algorithm
int pwm = 0;                          // pwm duty cycle 0-100%
int back_light_pin_State = 0;         // variable for storing the state of the backlight button
boolean load_status = false;          // variable for storing the load output state (for writing to
LCD)
enum charger_mode {off, on, bulk, bat_float} charger_state;    // enumerated variable that
holds state for charger state machine
// set the LCD address to 0x27 for a 20 chars 4 line display
// Set the pins on the I2C chip used for LCD connections:
//                    addr, en,rw,rs,d4,d5,d6,d7,bl,blpol
LiquidCrystal_I2C lcd(0x27, 2, 1, 0, 4, 5, 6, 7, 3, POSITIVE);  // Set the LCD I2C address
//------------------------------------------------------------------------------------------------------
// This routine is automatically called at powerup/reset
//------------------------------------------------------------------------------------------------------
void setup()                          // run once, when the sketch starts
{
  pinMode(PWM_ENABLE_PIN, OUTPUT);     // sets the digital pin as output
  TURN_OFF_MOSFETS;                    // turn off MOSFET driver chip
  charger_state = off;                 // start with charger state as off
  lcd.begin(20,4);                     // initialize the lcd for 16 chars 2 lines, turn on backlight
  // create the LCD special characters. Characters 0-5 are the various battery fullness icons
  // icon 7 is for the PWM icon, and icon 8 is for the solar array
  lcd.backlight();
  for (int batchar = 0; batchar <   6; ++batchar) {
    lcd.createChar(batchar, battery_icons[batchar]);
   }
  lcd.createChar(PWM_ICON,_PWM_icon);
  lcd.createChar(SOLAR_ICON,solar_icon);
  lcd.createChar('\\', backslash_char);
  pinMode(LED_RED, OUTPUT);
  pinMode(LED_GREEN, OUTPUT);
  pinMode(LED_YELLOW, OUTPUT);
  Timer1.initialize(20);              // initialize timer1, and set a 20uS period
  Timer1.pwm(PWM_PIN, 0);             // setup pwm on pin 9, 0% duty cycle
  Timer1.attachInterrupt(callback);    // attaches callback() as a timer overflow interrupt
  Serial.begin(9600);                 // open the serial port at 9600 bps:
  ser.begin(9600);                   // enable software serial
  ser.println("AT+RST");              // reset ESP8266
  pwm = PWM_START;                    //starting value for pwm
  pinMode(BACK_LIGHT_PIN, INPUT);
  pinMode(LOAD_PIN,OUTPUT);
  digitalWrite(LOAD_PIN,LOW);         // default load state is OFF
  digitalWrite(BACK_LIGHT_PIN,LOW);    //  default LCd back light is OFF
  // display the constant stuff on the LCD
  lcd.setCursor(0, 0);
  lcd.print("SOL");
  lcd.setCursor(4, 0);
  lcd.write(SOLAR_ICON);
  lcd.setCursor(8, 0);
  lcd.print("BAT");
}
//------------------------------------------------------------------------------------------------------
// Main loop
//------------------------------------------------------------------------------------------------------
void loop()
{
  read_data();                     // read data from inputs
  run_charger();                    // run the charger state machine
  print_data();                    // print data
  load_control();                   // control the connected load
  led_output();                    // led indication
  lcd_display();                   // lcd display
#if ENABLE_DATALOGGER
  wifi_datalog();                  // sends data to thingspeak
#endif
}
//------------------------------------------------------------------------------------------------------
// This routine reads and averages the analog inputs for this system, solar volts, solar amps
and
// battery volts.
//------------------------------------------------------------------------------------------------------
int read_adc(int channel){
  int sum = 0;
  int temp;
  int i;
  for (i=0; i<AVG_NUM; i++) {          // loop through reading raw adc values AVG_NUM
number of times
    temp = analogRead(channel);        // read the input pin
    sum += temp;                       // store sum for averaging
    delayMicroseconds(50);             // pauses for 50 microseconds
  }
  return(sum / AVG_NUM);               // divide sum by AVG_NUM to get average and return
it
}
//----------------------------------------------------------------------------------------------------
// This routine reads all the analog input values for the system. Then it multiplies them by the
scale
// factor to get actual value in volts or amps.
//----------------------------------------------------------------------------------------------------
void read_data(void) {
  sol_amps = (read_adc(SOL_AMPS_CHAN) * SOL_AMPS_SCALE -13.51);    //input of
solar amps
  sol_volts = read_adc(SOL_VOLTS_CHAN) * SOL_VOLTS_SCALE;          //input of solar
volts
  bat_volts = read_adc(BAT_VOLTS_CHAN) * BAT_VOLTS_SCALE;          //input of
battery volts
  sol_watts = sol_amps * sol_volts ;                               //calculations of solar watts
}
//----------------------------------------------------------------------------------------------------
// This is interrupt service routine for Timer1 that occurs every 20uS.
//
//----------------------------------------------------------------------------------------------------
void callback()
{
  if (interrupt_counter++ > ONE_SECOND) {        // increment interrupt_counter until one
second has passed
    interrupt_counter = 0;                       // reset the counter
    seconds++;                                   // then increment seconds counter
  }
}

//----------------------------------------------------------------------------------------------------
// This routine uses the Timer1.pwm function to set the pwm duty cycle.
//----------------------------------------------------------------------------------------------------
void set_pwm_duty(void) {
  if (pwm > PWM_MAX) {                                           // check limits of PWM duty
cyle and set to PWM_MAX
    pwm = PWM_MAX;
  }
  else if (pwm < PWM_MIN) {                                      // if pwm is less than PWM_MIN then
set it to PWM_MIN
    pwm = PWM_MIN;
  }
  if (pwm < PWM_MAX) {
    Timer1.pwm(PWM_PIN,(PWM_FULL * (long)pwm / 100), 20);  // use Timer1 routine to
set pwm duty cycle at 20uS period
    //Timer1.pwm(PWM_PIN,(PWM_FULL * (long)pwm / 100));
  }

  else if (pwm == PWM_MAX) {                              // if pwm set to 100% it will be
on full but we have
    Timer1.pwm(PWM_PIN,(PWM_FULL - 1), 20);               // keep switching so set duty
cycle at 99.9%
    //Timer1.pwm(PWM_PIN,(PWM_FULL - 1));
  }
}
//------------------------------------------------------------------------------------------------------
// This routine is the charger state machine. It has four states on, off, bulk and float.
// It's called once each time through the main loop to see what state the charger should be in.
// The battery charger can be in one of the following four states:
//
//  On State - this is charger state for MIN_SOL_WATTS < solar watts <
LOW_SOL_WATTS. In this state isthe solar
//      watts input is too low for the bulk charging state but not low enough to go into the off
state.
//      In this state we just set the pwm = 99.9% to get the most of low amount of power
available.
//  Bulk State - this is charger state for solar watts > MIN_SOL_WATTS. This is where we
do the bulk of the battery
//      charging and where we run the Peak Power Tracking alogorithm. In this state we try and
run the maximum amount
//      of current that the solar panels are generating into the battery.
//  Float State - As the battery charges it's voltage rises. When it gets to the
MAX_BAT_VOLTS we are done with the
//      bulk battery charging and enter the battery float state. In this state we try and keep the
battery voltage
//      at MAX_BAT_VOLTS by adjusting the pwm value. If we get to pwm = 100% it means
we can't keep the battery
//      voltage at MAX_BAT_VOLTS which probably means the battery is being drawn down
by some load so we need to back
//      into the bulk charging mode.
//  Off State - This is state that the charger enters when solar watts < MIN_SOL_WATTS.
The charger goes into this
//      state when there is no more power being generated by the solar panels. The MOSFETs
are turned
//      off in this state so that power from the battery doesn't leak back into the solar panel.
//------------------------------------------------------------------------------------------------------
void run_charger(void) {
  static int off_count = OFF_NUM;
  switch (charger_state) {
    case on:
      if (sol_watts < MIN_SOL_WATTS) {                    // if watts input from the solar panel
is less than
```

```
        charger_state = off;                          // the minimum solar watts then
        off_count = OFF_NUM;                          // go to the charger off state
        TURN_OFF_MOSFETS;
      }
      else if (bat_volts > (BATT_FLOAT - 0.1)) {          // else if the battery voltage has
gotten above the float
        charger_state = bat_float;                    // battery float voltage go to the charger battery
float state
      }
      else if (sol_watts < LOW_SOL_WATTS) {               // else if the solar input watts is less
than low solar watts
        pwm = PWM_MAX;                                // it means there is not much power being
generated by the solar panel
        set_pwm_duty();                                   // so we just set the pwm = 100% so
we can get as much of this power as possible
      }                                           // and stay in the charger on state
      else {
        pwm = ((bat_volts * 10) / (sol_volts / 10)) + 5;    // else if we are making more power
than low solar watts figure out what the pwm
        charger_state = bulk;                         // value should be and change the charger to
bulk state
      }
      break;
    case bulk:
      if (sol_watts < MIN_SOL_WATTS) {                    // if watts input from the solar panel
is less than
        charger_state = off;                          // the minimum solar watts then it is getting dark
so
        off_count = OFF_NUM;                          // go to the charger off state
        TURN_OFF_MOSFETS;
      }
      else if (bat_volts > BATT_FLOAT) {                  // else if the battery voltage has gotten
above the float
        charger_state = bat_float;                    // battery float voltage go to the charger battery
float state
      }
      else if (sol_watts < LOW_SOL_WATTS) {               // else if the solar input watts is less
than low solar watts
        charger_state = on;                           // it means there is not much power being
generated by the solar panel
        TURN_ON_MOSFETS;                              // so go to charger on state
      }
      else {                                      // this is where we do the Peak Power Tracking ro
Maximum Power Point algorithm
        if (old_sol_watts >= sol_watts) {                 // if previous watts are greater change the
value of
          delta = -delta;                                 // delta to make pwm increase or
decrease to maximize watts
        }
```

```
       pwm += delta;                          // add delta to change PWM duty cycle for PPT
algorythm (compound addition)
       old_sol_watts = sol_watts;                   // load old_watts with current watts value for
next time
       set_pwm_duty();                                          // set pwm duty cycle to pwm
value
     }
     break;
   case bat_float:
     if (sol_watts < MIN_SOL_WATTS) {                    // if watts input from the solar panel
is less than
       charger_state = off;                         // the minimum solar watts then it is getting dark
so
       off_count = OFF_NUM;                            // go to the charger off state
       TURN_OFF_MOSFETS;
       set_pwm_duty();
     }
     else if (bat_volts > BATT_FLOAT) {                  // If we've charged the battery abovethe
float voltage
       TURN_OFF_MOSFETS;                                 // turn off MOSFETs instead of
modiflying duty cycle
       pwm = PWM_MAX;                                // the charger is less efficient at 99% duty
cycle
       set_pwm_duty();                           // write the PWM
     }
     else if (bat_volts < BATT_FLOAT) {                  // else if the battery voltage is less than
the float voltage - 0.1
       pwm = PWM_MAX;
       set_pwm_duty();                                      // start charging again
       TURN_ON_MOSFETS;
       if (bat_volts < (BATT_FLOAT - 0.1)) {              // if the voltage drops because of added
load,
       charger_state = bulk;                       // switch back into bulk state to keep the voltage
up
       }
     }
     break;
   case off:                               // when we jump into the charger off state, off_count
is set with OFF_NUM
     TURN_OFF_MOSFETS;
     if (off_count > 0) {                      // this means that we run through the off state
OFF_NUM of times with out doing
       off_count--;                          // anything, this is to allow the battery voltage to
settle down to see if the
     }                                    // battery has been disconnected
     else if ((bat_volts > BATT_FLOAT) && (sol_volts > bat_volts)) {
       charger_state = bat_float;                    // if battery voltage is still high and solar volts
are high
       TURN_ON_MOSFETS;
     }
     else if ((bat_volts > MIN_BAT_VOLTS) && (bat_volts < BATT_FLOAT) &&
(sol_volts > bat_volts)) {
       charger_state = bulk;
       TURN_ON_MOSFETS;
     }
     break;
   default:
     TURN_OFF_MOSFETS;
     break;
  }
}
//------------------------------------------------------------------------------------------------------------
-------
///////////////////////////////////////////////LOAD CONTROL/////////////////////////////////////////////////////
//------------------------------------------------------------------------------------------------------------
-------
void load_control(){
#if LOAD_ALGORITHM == 0
  // turn on loads at night when the solar panel is not producing power
  // as long as the battery voltage is above LVD
  load_on(sol_watts < MIN_SOL_WATTS && bat_volts > LVD);
#else
  // dump excess solar energy into the load circuit
  load_on(sol_watts > MIN_SOL_WATTS && bat_volts > BATT_FLOAT);
#endif
}
void load_on(boolean new_status) {
  if (load_status != new_status) {
    load_status = new_status;
    digitalWrite(LOAD_PIN, new_status ? HIGH : LOW);
  }
}
//---------------------------------------------------------------------------------------------------
// This routine prints all the data out to the serial port.
//---------------------------------------------------------------------------------------------------
void print_data(void) {
  Serial.print(seconds,DEC);
  Serial.print("      ");
  Serial.print("Charging = ");
  if (charger_state == on) Serial.print("on   ");
  else if (charger_state == off) Serial.print("off  ");
  else if (charger_state == bulk) Serial.print("bulk ");
  else if (charger_state == bat_float) Serial.print("float");
  Serial.print("      ");
  Serial.print("pwm = ");
  if(charger_state == off)
  Serial.print(0,DEC);
  else
  Serial.print(pwm,DEC);
  Serial.print("      ");
  Serial.print("Current (panel) = ");
  Serial.print(sol_amps);
  Serial.print("      ");
  Serial.print("Voltage (panel) = ");
  Serial.print(sol_volts);
  Serial.print("      ");
  Serial.print("Power (panel) = ");
  Serial.print(sol_volts);
  Serial.print("      ");
  Serial.print("Battery Voltage = ");
  Serial.print(bat_volts);
  Serial.print("      ");

  Serial.print("\n\r");
  //delay(1000);
}
//----------------------------------------------------------------------------------------------------
//--------------------------------Led Indication----------------------------------------------
//----------------------------------------------------------------------------------------------------
// light an individual LED
// we remember which one was on before in last_lit and turn it off if different
void light_led(char pin)
{
  static char last_lit;
  if (last_lit == pin)
      return;
  if (last_lit != 0)
      digitalWrite(last_lit, LOW);
  digitalWrite(pin, HIGH);
  last_lit = pin;
}
// display the current state via LED as follows:
// YELLOW means overvoltage (over 14.1 volts)
// RED means undervoltage (under 11.9 volts)
// GREEN is between 11.9 and 14.1 volts
void led_output(void)
{
  static char last_lit;
  if(bat_volts > 14.1 )
      light_led(LED_YELLOW);
  else if(bat_volts > 11.9)
      light_led(LED_GREEN);
  else
      light_led(LED_RED);
}
//----------------------------------------------------------------------------------------------------
//------------------------- LCD DISPLAY ----------------------------------------------------------
//----------------------------------------------------------------------------------------------------
void lcd_display()
{
  static bool current_backlight_state = -1;
  back_light_pin_State = digitalRead(BACK_LIGHT_PIN);
  if (current_backlight_state != back_light_pin_State) {
    current_backlight_state = back_light_pin_State;
    if (back_light_pin_State == HIGH)
      lcd.backlight();// finish with backlight on
    else
      lcd.noBacklight();
  }
  if (back_light_pin_State == HIGH)
  {
    time = millis();                        // If any of the buttons are pressed, save the time in millis to
"time"
  }
 lcd.setCursor(0, 1);
 lcd.print(sol_volts);
 lcd.print("V ");
 lcd.setCursor(0, 2);
 lcd.print(sol_amps);
 lcd.print("A");
 lcd.setCursor(0, 3);
 lcd.print(sol_watts);
 lcd.print("W ");
 lcd.setCursor(8, 1);
 lcd.print(bat_volts);
 lcd.setCursor(8,2);
 if (charger_state == on)
 lcd.print("on   ");
 else if (charger_state == off)
 lcd.print("off  ");
 else if (charger_state == bulk)
 lcd.print("bulk ");
 else if (charger_state == bat_float)
 {
 lcd.print("     ");
 lcd.setCursor(8,2);
 lcd.print("float");
 }
 //--------------------------------------------------------
 //--------------------Battery State Of Charge ---------------
 //--------------------------------------------------------
 int pct = 100.0*(bat_volts - 11.3)/(12.7 - 11.3);
 if (pct < 0)
     pct = 0;
 else if (pct > 100)
     pct = 100;
 lcd.setCursor(12,0);
 lcd.print((char)(pct*5/100));
 lcd.setCursor(8,3);
 pct = pct - (pct%10);
 lcd.print(pct);
 lcd.print("%  ");
//-------------------------------------------------------------------
//------------------Duty Cycle----------------------------------------
//-------------------------------------------------------------------
 lcd.setCursor(15,0);
 lcd.print("PWM");
 lcd.setCursor(19,0);
 lcd.write(PWM_ICON);
 lcd.setCursor(15,1);
 lcd.print("   ");
 lcd.setCursor(15,1);
 if( charger_state == off)
 lcd.print(0);
 else
 lcd.print(pwm);
 lcd.print("% ");
 //-------------------------------------------------------------------
 //------------------------Load Status----------------------------------
 //-------------------------------------------------------------------
 lcd.setCursor(15,2);
 lcd.print("Load");
 lcd.setCursor(15,3);
 if (load_status)
 {
   lcd.print("On  ");
 }
 else
 {
  lcd.print("Off ");
 }
 spinner();
 backLight_timer();                    // call the backlight timer function in every loop
}
void backLight_timer(){
  if((millis() - time) <= 15000)        // if it's been less than the 15 secs, turn the backlight on
      lcd.backlight();                  // finish with backlight on
  else
      lcd.noBacklight();                // if it's been more than 15 secs, turn the backlight off
}
void spinner(void) {
  static int cspinner;
  static char spinner_chars[] = { ",", '*', ' ', ' '};
  cspinner++;
  lcd.print(spinner_chars[cspinner%sizeof(spinner_chars)]);
}
//-------------------------------------------------------------------
//---------------------------- ESP8266 WiFi ----------------------------
//-------------------------Plot System data on thingspeak.com-------------
//-------------------------------------------------------------------
```

```
void wifi_datalog()
{
  // thingspeak needs 15 sec delay between updates
  static int lastlogged;
  if ( seconds - lastlogged < 16 )
      return;
  lastlogged = seconds;
 // convert to string
  char buf[16];
  String strTemp = dtostrf( sol_volts, 4, 1, buf);
  Serial.println(strTemp);
  // TCP connection
  String cmd = "AT+CIPSTART=\"TCP\",\"";
  cmd += "184.106.153.149"; // api.thingspeak.com
  cmd += "\",80";
  ser.println(cmd);
  if(ser.find((char *)"Error")){
    Serial.println("AT+CIPSTART error");
    return;
  }
  // prepare GET string
  String getStr = "GET /update?api_key=";
  getStr += apiKey;
  getStr +="&field1=";
  getStr += String(strTemp);
  getStr += "\r\n\r\n";
  // send data length
  cmd = "AT+CIPSEND=";
  cmd += String(getStr.length());
  ser.println(cmd);
  if(ser.find((char *)">")){
    ser.print(getStr);
  }
  else{
    ser.println("AT+CIPCLOSE");
    // alert user
    Serial.println("AT+CIPCLOSE");
  }
}
```