IOT-BASED SOLAR ENERGY MONITORING SYSTEM WITH WEB INTERFACE FOR REAL-TIME PERFORMANCE ANALYSIS AND FAULT DETECTION

MUHAMMAD AFIF BIN AMRAN

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

IOT-BASED SOLAR ENERGY MONITORING SYSTEM WITH WEB INTERFACE FOR REAL-TIME PERFORMANCE ANALYSIS AND FAULT DETECTION

MUHAMMAD AFIF BIN AMRAN

This report is submitted in partial fulfilment of the requirements for the degree of Bachelor of Electronic Engineering with Honours

Faculty of Electronics and Computer Technology and Engineering
Universiti Teknikal Malaysia Melaka

YEAR OF SUBMISSION

UNIVERSITI TEKNIKAL MALAYSIA MELAKA
FAKULTI TEKNOLOGI DAN KEJURUTERAAN ELEKTRONIK DAN KOMPUTER

BORANG PENGESAHAN STATUS LAPORAN
PROJEK SARJANA MUDA II

Tajuk Projek : IoT-Based Solar Energy Monitoring System with Web Interface for Real-Time Performance Analysis and Fault Detection

Sesi Pengajian : 2010/2011

Saya MUHAMMAD AFIF BIN AMRAN mengaku membenarkan laporan Projek Sarjana Muda ini disimpan di Perpustakaan dengan syarat-syarat kegunaan seperti berikut:

1. Laporan adalah hakmilik Universiti Teknikal Malaysia Melaka.
2. Perpustakaan dibenarkan membuat salinan untuk tujuan pengajian sahaja.
3. Perpustakaan dibenarkan membuat salinan laporan ini sebagai bahan pertukaran antara institusi pengajian tinggi.
4. Sila tandakan (✓):

☐ **SULIT***	(Mengandungi maklumat yang berdarjah keselamatan atau kepentingan Malaysia seperti yang termaktub di dalam AKTA RAHSIA RASMI 1972)
☐ **TERHAD***	(Mengandungi maklumat terhad yang telah ditentukan oleh organisasi/badan di mana penyelidikan dijalankan.)
☐ **TIDAK TERHAD**	

Disahkan oleh:

(TANDATANGAN PENULIS) (COP DAN TANDATANGAN PENYELIA)

Alamat Tetap: 174-E, BATU 3 JALAN KUALA KRAI, 15050 KELANTAN

Tarikh : 01 Januari 2010 Tarikh : 01 Januari 2010

*CATATAN: Jika laporan ini SULIT atau TERHAD, sila lampirkan surat daripada pihak berkuasa/organisasi berkenaan dengan menyatakan sekali tempoh laporan ini perlu dikelaskan sebagai SULIT atau TERHAD.

DECLARATION

I declare that this report entitled "PSM Full Title" is the result of my own work except for quotes as cited in the references.

Signature : ……

Author : ……

Date : ……

APPROVAL

I hereby declare that I have read this thesis and in my opinion, this thesis is sufficient in terms of scope and quality for the award of Bachelor of Electronic Engineering with Honours

Signature	:	……
Supervisor Name	:	……
Date	:	……

DEDICATION

Specially dedicated to, My beloved parents AMRAN BIN MUHAMMAD and NORHASHIMAH BINTI AB KADIR@HUSSAIN, My beloved brothers, MUHAMMAD AKMAL BIN AMRAN and MUHAMMAD ASYRAF BIN AMRAN, To my supervisor PROFESSOR TS. DR. AZMI BIN AWANG MD ISA & My family, friends and my fellow lecturers. Thank you for all the supports.

ABSTRACT

This project aims to develop an intelligent IoT-based solar monitoring system with real-time fault detection capabilities to address limitations in traditional passive monitoring. The scope of study focuses on the continuous analysis of electrical and environmental parameters for a photovoltaic panel setup. The methodology integrates an Arduino Uno for precise sensor data acquisition and an ESP32 gateway for cloud transmission to ThingSpeak while a Raspberry Pi hosts a Random Forest machine learning model accessed via a Python Flask API for edge computing. Findings indicate that the system achieved 100% accuracy in classifying operational states including Normal, Open Circuit, Partial Shading and Overheating and successfully displayed analytics on a custom HTML dashboard with automated Telegram alerts. The study concludes that integrating low-cost IoT hardware with predictive machine learning offers a scalable, highly reliable solution for automating solar predictive maintenance and enhancing energy efficiency.

ABSTRAK

Projek ini bertujuan untuk membangunkan sistem pemantauan solar berasaskan IoT pintar dengan keupayaan pengesanan kerosakan masa nyata bagi mengatasi batasan pemantauan pasif tradisional. Skop kajian tertumpu kepada analisis berterusan parameter elektrik dan persekitaran bagi persediaan panel fotovolta. Metodologi kajian mengintegrasikan Arduino Uno untuk pemerolehan data penderia yang tepat dan get way ESP32 untuk penghantaran awan ke ThingSpeak manakala Raspberry Pi menempatkan model pembelajaran mesin Random Forest yang diakses melalui API Python Flask untuk pengkomputeran pinggir. Dapatan kajian menunjukkan bahawa sistem ini mencapai ketepatan 100% dalam mengklasifikasikan status operasi termasuk Normal, Litar Terbuka, Bayangan Separa dan Pemanasan Melampau serta berjaya memaparkan analisis pada papan pemuka HTML tersuai dengan amaran Telegram automatik. Kajian menyimpulkan bahawa integrasi perkakasan IoT kos rendah dengan pembelajaran mesin prediktif menawarkan penyelesaian berskala yang boleh dipercayai untuk mengautomasikan penyelenggaraan prediktif solar dan meningkatkan kecekapan tenaga.

ACKNOWLEDGEMENTS

First, I would like to express my gratitude to my supervisor, Professor Ts. Dr. Azmi Bin Awang Md Isa for his advice, guidance and suggestion. He lends a hand guiding me on the right track to do my project and motivating me when facing circumstances. Also, I would like to express my deepest gratitude to my family members and my parents for their love, support and motivation. I would like to extend my heartfelt gratitude to my dearest grandparents who not only been my guiding light but also my second parents throughout this journey. Your unwavering love, support and encouragement have been the cornerstone of my success. A huge thank you to my friends for always being there for me, motivating me and helping me to stay positive during tough moments. I am truly grateful to everyone who contributed in some way, both big and small to help me complete this project.

.

TABLE OF CONTENTS

LIST OF FIGURES

(Please delete this part): This list contains the titles of figures, together with their page numbers, which are listed in the text. For e.g., figures in Chapter 3 are numbered sequentially: Figure 3.1, Figure 3.2.

For title of list tables use *other title* and *TOC1* for style.

LIST OF TABLES

(Please delete this part): This list contains the titles of tables, together with their page numbers, which are listed in the text. The numbering system is according to chapter, for e.g.: tables in Chapter 3 are numbered sequentially: Table 3.1, Table 3.2.

LIST OF SYMBOLS AND ABBREVIATIONS

For examples:

CC	:	Central canal
DAB	:	3,3’-diaminobenzidine
HRP	:	Horseradish peroxidase
MS222	:	Tricaine methanesulfonate
	:	
	:	

LIST OF APPENDICES

CHAPTER 1

INTRODUCTION

In this chapter, the project background is discussed in order to describe the development for ideas for the project. Besides, the problem statements, objectives, scope of project and thesis structure will also be discussed in this chapter.

1.1 Project Research Background

The global transition toward renewable energy has accelerated as fossil fuel reserves deplete and environmental concerns rise. Solar photovoltaic (PV) systems have emerged as a premier alternative due to their sustainability, abundance and modularity. However, the operational efficiency of solar panels is highly sensitive to environmental and electrical variables including fluctuating irradiance, temperature spikes, partial shading and dust accumulation [1]. Without continuous oversight, these factors can significantly reduce energy yield and compromise the lifespan of the hardware.

Traditional monitoring methods which often rely on periodic manual inspections or localized data loggers are increasingly inadequate for modern energy needs. They are labor-intensive, prone to human error and fail to provide the real-time insights necessary for immediate intervention. The emergence of the Internet of Things (IoT) has revolutionized this domain by enabling seamless connectivity between physical sensors and cloud platforms. By employing microcontrollers such as the Arduino Uno and ESP32, critical performance metrics “voltage, current, power and temperature” can now be transmitted wirelessly for remote visualization and storage [2], [3].

Recent research has transitioned from basic data acquisition to the integration of intelligent diagnostics. While earlier systems by Durgadevi et al. [1] and Cheragee et al. [2] successfully established cloud connectivity, contemporary efforts are focusing on the application of machine learning (ML) to enhance system autonomy. Advanced algorithms can analyze historical trends to predict energy output or identify subtle anomalies that indicate early-stage faults [4], [5]. Despite these strides, there remains a critical need for an integrated low-cost solution that combines high-precision

sensing, interactive web visualization via custom dashboards and automated fault classification. This project addresses this need by employing a Random Forest algorithm to distinguish between normal and faulty operating states, thereby ensuring the reliability and efficiency of solar PV installations.

1.2 Problem Statement

Solar PV systems are frequently subjected to unexpected performance drops caused by external factors such as shading, dirt buildup or electrical component failure. Because these systems are often installed in remote or inaccessible locations such faults can go undetected for extended periods, leading to substantial energy losses and increased long-term maintenance costs [6].

A primary limitation of existing IoT monitoring solutions is their focus on passive data visualization. While platforms like ThingSpeak or Blynk offer excellent graphing capabilities, they often lack the built-in intelligence required to automatically classify faults or provide diagnostic feedback [7]. Users are frequently left to interpret raw data themselves which requires technical expertise and constant attention. Furthermore, many current prototypes rely exclusively on mobile applications which can restrict data accessibility and limit the complexity of the analytical tools available to the user.

Furthermore, there is a significant gap in the integration of high-level machine learning models within cost-effective IoT frameworks. Most academic prototypes are restricted to offline data analysis or small-scale laboratory experiments that do not offer a scalable path for real-world deployment [4], [5], [6]. There is a lack of systems that bridge the gap between cloud-based data storage and edge-based intelligent processing. This project aims to bridge these gaps by developing a scalable system that integrates real-time custom web dashboards with a Random Forest model

deployed on a Raspberry Pi, allowing for proactive, automated identification of performance irregularities.

1.3 Objective

1. To develop a real-time solar energy monitoring system that accurately measures solar PV performance metrics which is voltage, current and temperature.
2. To design an interactive custom HTML dashboard that integrates ThingSpeak data and machine learning status for comprehensive, multi-platform visualization.
3. To implement pattern recognition algorithms for anomaly detection and predictive fault identification.

1.4 Scope of Project

The project focuses on the development and validation of an IoT-based system for the continuous monitoring of a 50W monocrystalline solar panel. Data acquisition is handled by an Arduino UNO which interfaces with an INA219 power sensor and a DHT11 temperature sensor. An ESP32 serves as the IoT gateway, transmitting sensor readings via Wi-Fi to the ThingSpeak cloud platform [2].

On the software side, the project implements a custom HTML dashboard for real-time visualization and a Python-based backend running on a Raspberry Pi. The intelligence layer utilizes a Random Forest model which is trained in Google Colab using historical datasets to recognize various fault conditions such as partial shading or open circuits [8]. The system is designed to provide users with immediate fault alerts via the dashboard and Telegram notifications. The architecture is built to be scalable, allowing for the future expansion of sensor nodes or higher-rated solar arrays.

Furthermore, the scope includes the specific identification and classification of common solar PV faults through the analysis of electrical and environmental correlations. The Random Forest model is designed to distinguish between normal operation and anomalies such as partial shading which often results in disproportionate current drops relative to temperature. By analyzing the relationship between voltage (V), current (I), power (P) and temperature (T), the system provides a diagnostic depth that exceeds simple threshold-based monitoring. This analytical scope ensures that the system not only reports that a problem exists but also provides a preliminary classification of the fault type to assist in maintenance decisions.

Finally, the project is bounded by its operational and connectivity framework which relies on a stable Wi-Fi connection for cloud data logging and real-time inference. The system is optimized for small-scale residential or educational applications as a functional prototype. While it primarily focuses on a single-panel configuration for initial validation, the use of a modular software stack comprising ThingSpeak, Flask and a custom web front-end ensures that the communication infrastructure is robust and adaptable to various network environments. This scope defines the limits of the current implementation while providing a clear technological roadmap for industrial-grade solar monitoring solutions.

1.5 Methodology of Project

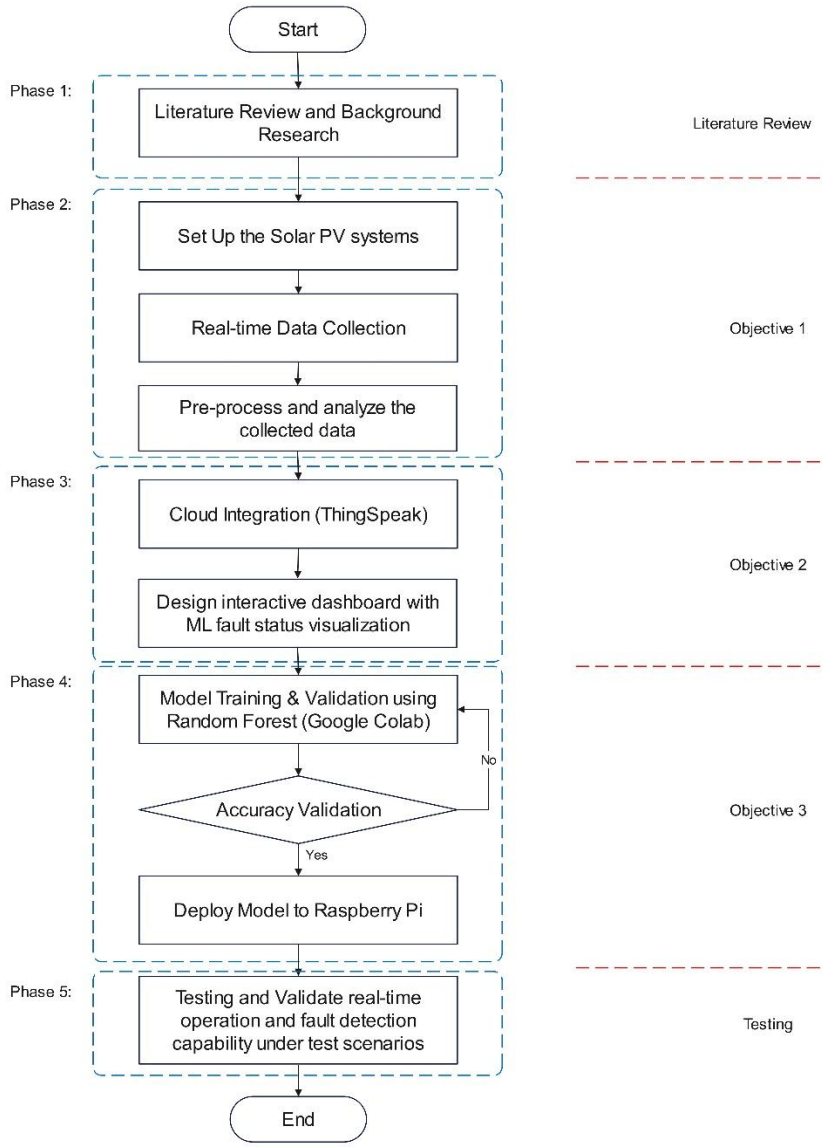


Figure 1.1 Project Methodology

This project is divided into five systematic phases as shown in Figure 1.1 which describes the overall process from research and data acquisition to cloud integration, machine learning implementation and system validation. Each phase contributes toward ensuring that the IoT-based monitoring system is functional and intelligent in detecting performance faults in the PV system.

Phase 1: Literature Review and Background Research

The first phase focuses on reviewing existing studies related to solar PV monitoring systems, IoT architecture and machine learning-based fault detection. Previous researchers such as Durgadevi et al. [1], Cheragee et al. [2] and Pasandideh et al. [9] developed various real-time solar monitoring systems using microcontrollers like Arduino, ESP8266 and Raspberry Pi to measure and analyze solar performance. These studies highlighted the growing importance of IoT technology in enhancing solar energy management. Furthermore, research conducted by Ledmaoui et al. [4] and Kayalvizhi et al. [5] introduced the integration of predictive maintenance algorithms for energy forecasting while others emphasized web-based applications for real-time PV monitoring. Insights from these works provide the theoretical foundation and technological direction for developing an improved IoT-based solar monitoring system integrated with fault detection capability.

Phase 2: System Setup and Data Collection

The second phase centers on the physical construction and configuration of the sensor node which forms the perception layer of the IoT system. This involves interfacing an Arduino Uno microcontroller with high-precision sensors, specifically the INA219 for measuring voltage, current and power and the DHT11 for monitoring the solar panel temperature. Data transmission capabilities are established by connecting the Arduino to an ESP32 Wi-Fi module via serial communication. This phase also includes the rigorous calibration of sensors and the collection of initial datasets under varying environmental conditions to ensure the hardware reliably captures the electrical and thermal characteristics of the 50W solar panel.

Phase 3: Cloud Integration and Web Interface Design

Following the hardware setup, the focus shifts to developing the connectivity and visualization infrastructure. The ESP32 is programmed to function as an IoT gateway, transmitting sensor data to the ThingSpeak cloud platform using RESTful HTTP API calls for secure and persistent storage. Concurrently, a custom web-based dashboard is developed using HTML, Tailwind CSS and Chart.js to provide a user-friendly interface that surpasses standard cloud views. This interface is designed to fetch real-time data from ThingSpeak, offering users dynamic visualization of performance metrics and immediate insight into the system's operational status from any internet-connected device.

Phase 4: Model Training, Validation and Deployment

This critical phase involves the development of the system's intelligence layer, moving beyond simple data logging to active fault detection. Historical data collected from the system is labeled and uploaded to the Google Colab environment, where it is used to train a Random Forest Classifier using Python's Scikit-learn library. The model is optimized to recognize complex non-linear patterns associated with specific fault conditions such as partial shading or open circuits. Once trained and validated, the model is serialized and deployed onto a Raspberry Pi running a Flask API, enabling the system to perform real-time inference on incoming live data streams.

Phase 5: Testing and Validation

The final phase is dedicated to the rigorous validation of the complete integrated system to ensure it meets all functional objectives. The system undergoes extensive field testing under various real-world scenarios, including normal sunny conditions, cloudy weather and artificially induced faults like shading or disconnection. The

performance is evaluated based on the accuracy of sensor readings, the reliability of data transmission to the cloud and most importantly, the precision of the Random Forest model in correctly classifying the system's state. The results are analyzed to quantify the system's effectiveness and to identify areas for future optimization.

1.6 Importance and Significance of the Project

This project is significant because it directly contributes to the optimization of renewable energy systems through the application of Industry 4.0 technologies. By integrating IoT connectivity with machine learning diagnostics, the system moves beyond traditional passive monitoring into the realm of predictive maintenance. This shift is crucial for minimizing downtime, preventing hardware damage and maximizing the economic return of solar investments [1], [5].

The development of a responsive, custom web dashboard enhances accessibility, allowing both residential and industrial users to track system performance remotely from any device. Furthermore, the use of low-cost, open-source hardware and software makes this solution highly replicable for small-scale energy projects in developing regions. By promoting energy efficiency and sustainable technological innovation, this project aligns with the United Nations Sustainable Development Goal 7 (SDG 7) for affordable and clean energy.

Moreover, this project bridges the technological gap between traditional power engineering and modern computational intelligence. By implementing a Random Forest model trained in Google Colab and deployed on a Raspberry Pi, it demonstrates the practical feasibility of Edge AI in the renewable energy sector [10], [11]. This architecture significantly enhances the system's responsiveness, proving that sophisticated fault detection does not require expensive industrial controllers but can

be effectively achieved using accessible, off-the-shelf microcontrollers and open-source Python libraries.

From an operational perspective, the system addresses the critical challenge of "silent faults" minor inefficiencies such as partial shading or early-stage degradation that often go unnoticed in standard voltage-monitoring setups. Unlike basic data loggers that simply record a drop in output, this intelligent system analyzes the correlation between temperature and power to distinguish between genuine environmental drops and system faults [12], [13], [14]. This capability empowers users to perform targeted maintenance such as cleaning panels or checking connections, ensuring the PV system operates at its theoretical maximum efficiency throughout its lifecycle.

Finally, the research contributes valuable insights to the academic community regarding the behavior of small-scale PV systems in tropical climates. By collecting and analyzing real-world data involving high humidity and rapidly fluctuating cloud cover, this project provides a reference dataset for future studies in solar forecasting [15], [16]. The successful validation of the Random Forest algorithm in this specific environment offers a benchmark for future researchers looking to develop more resilient and regionally optimized energy monitoring algorithms.

1.7 Thesis Structure

This thesis is organized in five major chapters, each one detailing a particular aspect of the development and implementation of the IoT-Based Solar Energy Monitoring System with Web Interface for Real-Time Performance Analysis and Fault Detection. This report is built in such a way that the project will be duly presented in order, right

from the introduction of the background of research to the presentation of results, analysis and conclusions.

Chapter 1: Introduction

The chapter provides an overview of the project regarding the background of the research, problem statement, objectives, scope, methodology and significance of the study. It introduces the concept of IoT-based solar monitoring to explain the motivation for integrating real-time data visualization with machine learning fault detection. It also lays the foundation and purpose of the project by linking it to the growing demand for intelligent renewable energy management systems.

Chapter 2: Literature Review

This chapter highlights a review of the related works and previous research in the fields of solar photovoltaic monitoring, IoT integration and fault detection using machine learning algorithms. It explains various methods, architectures and technologies adopted in previous studies focusing on system design, sensor selection, data communication and analytical models. Additionally, a comparative summary has been provided to outline the strengths and limitations of existing systems and thereby explain how this project outperforms them with the integration of ThingSpeak-based monitoring and Random Forest Classifier analysis.

Chapter 3: Methodology

This chapter describes the complete design and implementation process of the IoT-based solar monitoring system. It details the hardware selection and configuration, including the interfacing of the Arduino Uno, ESP32 and high-precision sensors. The

software architecture is thoroughly explained, covering the firmware development for data acquisition, the integration with the ThingSpeak cloud platform and the machine learning workflow using Google Colab for model training and the Raspberry Pi for local deployment. This structured approach ensures the reproducibility and reliability of the system.

Chapter 4: Results and Discussion

This chapter presents the experimental findings and evaluates the performance of the developed system. It provides detailed visualizations of real-time monitoring data, including voltage, current and power trends captured via the custom HTML dashboard. The effectiveness of the Random Forest model is analyzed through accuracy assessments and fault classification results under different environmental conditions. Additionally, a comprehensive discussion is provided on how the system successfully identifies performance irregularities compared to traditional monitoring methods.

Chapter 5: Conclusion and Future Work

The final chapter summarizes all the findings and results of the entire work. It presents how the proposed IoT-based system successfully meets its objectives related to real-time monitoring, web-based visualization and machine learning-based fault detection. Conclusions are drawn with recommendations for possible future improvements which may involve adding irradiance sensors, larger solar arrays or hybrid machine learning models such as RF-LSTM to enhance both the prediction accuracy and scalability of the system.

CHAPTER 2

BACKGROUND STUDY

This chapter highlights the theory and concept of solar energy, photovoltaic systems and solar power generation. It further provides the fundamental principles of IoT technology and how it is applied in real-time monitoring of solar energy. In this regard, a review of the key components used to include the solar panel, INA219 voltage and current sensor, DHT11 temperature sensor, Arduino Uno microcontroller, ESP32 Wi-Fi module and the ThingSpeak cloud platform provides a clear overview of the hardware and communication structure involved. Further, this chapter explains the Random Forest Classifier algorithm that will be implemented using Python (Google Colab) to analyze the performance and perform predictive fault detection in the system, thus providing an overview of the analytical approach that would be most feasible and relevant for efficient solar energy management.

2.1 Solar Energy and Photovoltaic (PV) System

Solar energy is one of the most abundant and renewable energy resources available for human use. The accelerated depletion of fossil fuels coupled with a worldwide drive toward decarbonization has sped up the adoption of solar photovoltaic technology as a clean alternative in generating electricity. Solar energy extracts the radiant light and heat from the sun and directly converts it into electricity by way of the photovoltaic effect, a process that occurs when sunlight strikes a semiconductor material and frees electrons to generate electrical current [17]. Among all renewable energy technologies, photovoltaic systems are favored for their modularity, low maintenance and scalability for residential and industrial applications [18].

A photovoltaic system is composed of one or more photovoltaic modules, conditioning units of electricity, sensors and monitoring devices. The energy output from a PV module is basically determined by the two most influential environmental parameters which are incident solar irradiance (G) and cell temperature (T), that offer the voltage and current characteristics of the system [15]. Thus, the relationship between these parameters can be described as:

$$P = V \times I$$

Where P is the instantaneous power output (Watts), V is the voltage (Volts) and I is the current (Amperes). The overall conversion efficiency of a PV system can be defined as:

$$\eta = \frac{P_{out}}{P_{in}} \times 100$$

Where P_{out} represents the electrical output power and P_{in} is the incident solar power received by the panel surface. Commercial silicon photovoltaic modules exhibit an efficiency that ranges between 15% and 22% under standard test conditions (STC) depending on material type, irradiance and temperature conditions [18].

The performance of PV systems is greatly affected by several external factors including temperature, irradiance, dust accumulation and shading effects. High operating temperatures lower the open-circuit voltage of solar cells whereas reduced irradiance decreases the current output directly, hence reducing efficiency [16]. Dust settlement on panel surfaces prevents sunshine from illuminating the surface and creates partial shading. This results in a condition called hot spots which involve local heating that accelerates material degradation [12]. Table 2.1 summarizes the main factors affecting the performance of solar PV systems and typical impacts on power generation.

Table 2.1. Factor that effect on PV Performance

Factor	Description	Effect on PV Performance
Solar Irradiance	Amount of solar power per unit area (W/m^2) received by the panel surface	Directly proportional to current and power output
Temperature	Ambient and cell temperature during operating	High temperature reduces voltage and efficiency
Dust Accumulation	Dust or dirt covering panel surface	Decrease light absorption that causes power loss
Shading	Partial blockage of sunlight due to objects or weather	Leads to mismatch losses and hot spots
Tilt Angle	Inclination of solar panel with respect to sunlight	Improper angle reduces energy capture efficiency

The maximum power output (Pmax) in case of a photovoltaic system occurs at the point when the load impendence is matched to the internal resistance of the solar cell. This condition is often referred to as the Maximum Power Point (MPP) which is usually tracked by algorithms such as Perturb and Observe (P&O) or Incremental Conductance (IncCond) to maintain optimal energy harvesting [12], [15]. Graphically, the relationship of solar irradiance and temperature to MPP can be shown by the characteristic current-voltage (I-V) and power-voltage (P-V) curves where for given environmental conditions, the intersection of curves indicates the MPP.

Therefore, the basic operation of a solar PV can be summarized in Figure 2.1. The system processes solar radiation into electrical energy which may be measured with the use of voltage and current sensors for performance evaluation. Real-time monitoring of these parameters by the user can achieve energy efficiency and find deviations that may suggest faults or inefficiency within the system [17], [18].

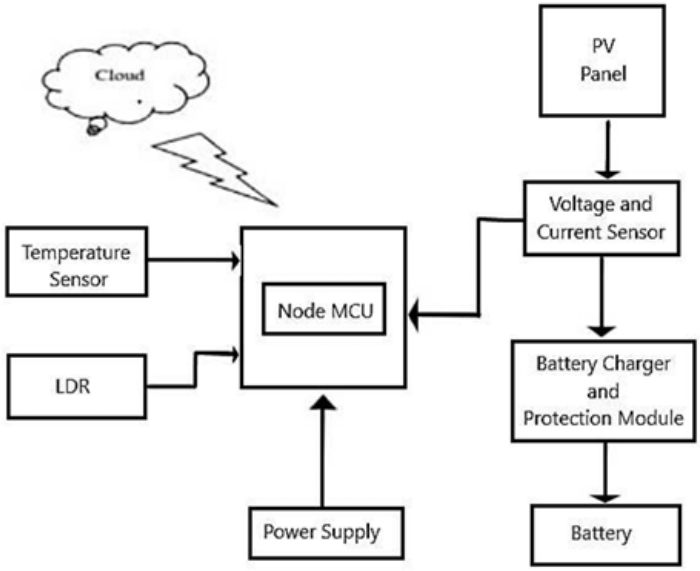


Figure 2.1 Basic Working Principle of a Photovoltaic System

Researchers have underscored the importance of continuous monitoring and data-driven assessment to ensure consistent performance. Sarkar et al. [3] have mentioned that the adoption of IoT platforms for real-time monitoring enhances system reliability

by immediately detecting any irregularities within the power output. Pasandideh et al. [9] presented a low-cost IoT-based monitoring system capable of gathering voltage, current and temperature data that could be visualized remotely using cloud computing platforms like ThingSpeak. These studies demonstrate that real-time analytics of the performance data generated by PVs contribute not only to higher efficiency in maintenance operations but also provide the basis for developing predictive fault detection systems.

2.2 Internet of Things (IoT) in Solar Energy Monitoring

The IoT has emerged as one of the most transformational technologies in modern energy management. It enables the interconnection of physical devices via the internet for real-time communication, monitoring and control of systems remotely. In applications dealing with solar energy, IoT allows for continuous tracking of parameters such as voltage, current and temperature, therefore providing critical insights into the operational performance of photovoltaic systems [3], [8], [18]. An IoT-based monitoring system generally consists of three layers which is the perception layer, network layer and application layer as shown in Figure 2.2.

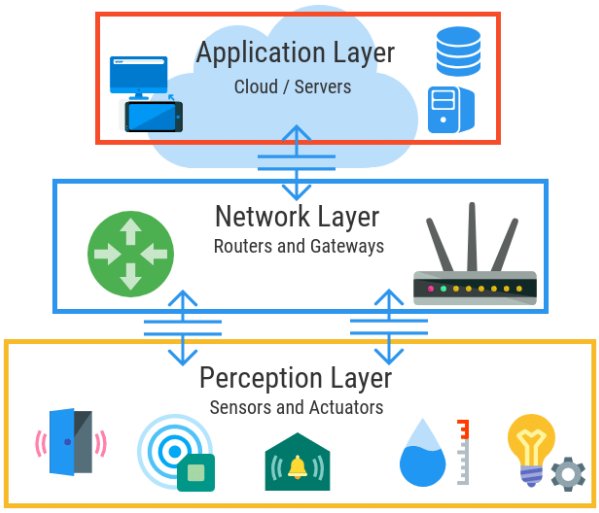


Figure 2.2 Three-Tier Architecture of the Internet of Things (IoT)

This includes the perception layer with sensors and data acquisition hardware like voltage, current, temperature and irradiance sensors that connected to microcontrollers. The network layer controls communications with the cloud via Wi-Fi, GSM and LoRaWAN modules. Finally, the application layer enables data visualization and decision making using cloud platforms such as ThingSpeak, Blynk or custom web dashboards [7], [18]. This multilayer architecture ensures that the collection, transmission and analysis of solar data can be efficiently done in real time without manual intervention.

IoT adoption in solar PV systems provides several advantages which is real-time remote monitoring, fault diagnosis, energy forecasting and maintenance optimization. As described by Cheragee et al. [2], the integration of IoT allows solar system operators to monitor critical parameters uninterruptedly in cloud-based platforms without resorting to physical inspection. Similarly, Sarkar et al. [3] and Parappully et al. [19] reported that IoT systems with ESP32 and NodeMCU microcontrollers enhance data accessibility and reduce power loss due to unnoticed failures in the

electrical systems. These systems transmit sensor data using any wireless communications such as Wi-Fi (IEEE 802.11), GSM or LoRa to centralized cloud servers [20].

IoT-based monitoring frameworks are widely implemented by using cost-effective, open-source microcontrollers such as Arduino Uno, ESP8266 and ESP32 which offer ease of integration with cloud services [7], [8], [15]. The ThingSpeak IoT platform has been frequently utilized in academic and industrial research due to its ability to record real-time data, visualize graphs and perform MATLAB analytics directly on the cloud [16], [18]. While ThingSpeak supports internal MATLAB scripts, this project utilizes an external Python-based backend on a Raspberry Pi to handle the machine learning predictions, offering greater flexibility and processing power. In contrast, other platforms like Blynk IoT and Firebase provide mobile-friendly dashboards to visualize data but often lack the in-depth analytics and customization available in ThingSpeak [3], [9].

IoT-based solar monitoring systems have advanced from simple sensing applications to comprehensive cloud-driven analytics tools. Rao et al. [18] proposed a smart Arduino ATMega2560 and NodeMCU ESP8266-based cloud monitoring system that can perform the reliable real-time transmission of voltage, current, temperature and irradiance data. Similarly, Nkinyam et al. [15] presented a low-cost IoT prototype for monitoring solar PV systems through web interfaces with a focus on an affordable and scalable solution for small-scale users. Rouibah et al. [16] extended this work by integrating an IoT-based web interface with MySQL database storage to enable multi-sensor data visualization and control. Almadhoun & Jaafar [12] further extended the design to an IoT-based system with the capability for fault detection in

PV panels using calibrated reference values and colour-coded alerts on a web dashboard.

Table 2.2 Summary of IoT-based Solar Monitoring Systems

Authors	Title	Tools / Components used	Key Contributions	Limitation
Cheragee et al. [2]	A study of IoT-Based Real-Time Solar Power Remote Monitoring System	NodeMCU ESP826, INA219, DS18B20, LDR, ThingSpeak	Real-time remote monitoring using low-cost IoT hardware	Limited automation and no fault detection
Sarkar et al. [3]	Design of Solar Panel Monitoring System Using ESP32 & IoT	ESP32, ACS723, Voltage Divider, DS18B20, Blynk IoT	Real-time monitoring via smartphone dashboard	Lacks data analytics or cloud storage
Kayalvizhi et al. [5]	IoT-Enable Real-Time Monitoring and Predictive Maintenance	ESP8266, Voltage/ Temperature/ Ultrasonic Sensors, Relay, LCD	Predictive maintenance and alert system	Limited dataset which requires constant Wi-Fi

	for Solar Systems			
Almadhoun & Jaafar [12]	Enhanced Solar Systems Efficiency and Reduce Energy Waste by Using IoT Devices	Arduino Uno, ESP8266, Voltage/ Current Sensors, ThingSpeak	Fault detection through colour-coded alerts and web dashboard	No automated corrective actions
Nkinyam et al. [15]	Development of a Low-Cost Monitoring Device for Solar Electric (PV) System Using IoT	Arduino Uno, Wi-Fi module, Voltage/ Current Sensors, Web Interface	Compact IoT design with web monitoring for small-scale users	No predictive or diagnostic feature
Rouibah et al. [16]	Smart Monitoring of Photovoltaic Energy	Arduino Mega 2560, ESP8266, MAX6675, ACS712, MySQL Web Interface	Multi-sensor integration and MPPT data logging	No built-in cybersecurity or ML module

	Systems: An IoT-Based Prototype Approach			
Dutta et al. [17]	Solar Energy Monitoring System (SEMS)	Arduino Uno, ACS712, DHT11, LCD	Local real-time PV monitoring and display	Not connected to web/cloud interface
Rao et al. [18]	Development of a Smart Cloud-Based Monitoring System for Solar PV Energy Generation	Arduino ATMega2560, NodeMCU ESP8266, ACS712, LM35, LDR	Cloud-based visualization andlong-term data logging	53 s average transmission delay
Parappully et al. [19]	IoT-Based Solar Performance Monitoring System	Arduino Mega 2560, ESP8266, Voltage/Curremt Sensors, ThingSpeak	IoT-based PV performance tracking and live dashboard	No predictive fault identification

IoT has also been useful in performance optimizing and maintenance management. The studies of Barnes et al. [20] implemented LoRaWAN and LoRa modules for long-range, low-power monitoring suitable for rural solar installations. Even though these

solutions decrease communication costs, they have low bandwidths and limited data rates that highly limit the capabilities of real-time visualization. On the other hand, Raspberry Pi-based systems [21] offer higher processing capabilities for data storage and web hosting but at higher implementation costs.

Integration of the IoT in solar monitoring significantly improves efficiency, scalability and fault responsiveness. Transferring real-time sensor data to cloud servers allows the operator to analyze energy generation trends, observe anomalies and conduct preventive maintenance before the occurrence of critical failures [5], [12], [18]. In addition to the above, the IoT framework is also used as a basis for integrating advanced machine learning model such as Random Forest and Neural Networks which enable predictive fault detection [10], [13], [22].

2.3 Machine Learning in Solar Fault Detection

In recent years, machine learning has been transforming the monitoring of solar photovoltaic systems through intelligent fault detection, classification and prediction. Unlike traditional threshold-based monitoring, ML algorithms can identify hidden patterns and anomalies within PV performance data including voltage, current, irradiance and temperature [10], [13]. This allows for early detection of potential system faults, reducing downtime and improving the overall efficiency and reliability of solar installations.

A fault in a PV may result from partial shading, degradation, wiring defects, inverter failures or module malfunctions. Classic IoT-based systems mainly collect sensor data and display them without automatic fault diagnosis [19]. In turn, ML-based models learn from historical data to automatically detect deviations of operation from

normal conditions. The general workflow of an ML-based solar fault detection system is presented in Figure 2.3.

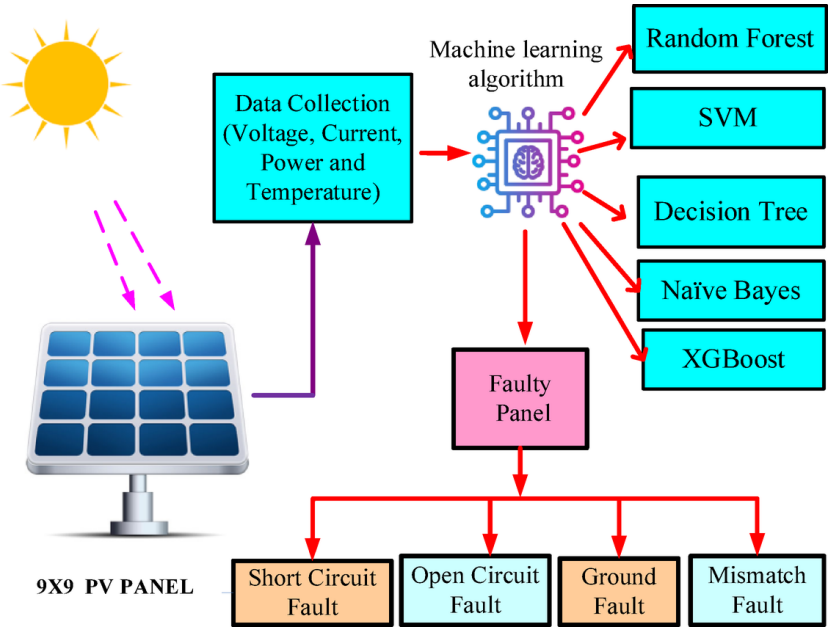


Figure 2.3 General Workflow of an ML-Based Solar Fault Detection System

2.3.1 Machine Learning Techniques for PV Fault detection

Various Machine Learning have been utilized in solar energy research such as Decision Tree (DT), Support Vector Machine (SVM), k-Nearest Neighbor (KNN), Random Forest (RF), Artificial Neural Network (ANN) and Deep Learning methods like Convolutional Neural Networks (CNN) and Long-Term Memory (LSTM) networks [10, 11, 13, 14, 22-30][10], [11], [13], [14], [22], [23], [24], [25], [26], [27], [28], [29], [30]. Each one has different advantages with respect to the data complexity, training size and computation capability.

Of all various methods, the Random Forest Classifier is among the most commonly used due to its relatively high accuracy, robustness and interpretability with non-linear data [10], [13], [22]. The Random Forest algorithm for classification works by generating many decision trees during training and utilizing a voting method to

determine the class based on the majority of trees. The Random Forest decision rule can be given mathematically as:

$$\hat{y} = mode\{h_1(x), h_2(x), h_3(x), \dots, h_n(x)\}$$

Where $h_i(x)$ represents the prediction from the $i - th$ decision tree and $\hat{y}$ denotes the final predicted class label based on the majority voting principle.

The Random Forest approach reduces overfitting and enhances generalization by aggregating several weak learners. It has been successfully implemented for PV fault classification using parameters such as array voltage, string current and irradiance data [10][13]. Research works by Amiri et al. [10] and Chen et al. [13] presented that Random Forest yielded higher accuracy compared to SVM and single Decision Trees in identifying PV array faults related to open circuits, short circuits and degradation.

Dewi et al. [22] have furthered this by proposing a hybrid RF + LSTM-RNN model for forecasting solar power output in aquaponic systems. Hence, the forecasts' accuracy was obviously improved and the adaptive fault recognition could be enhanced. Parvin et al. [23] proposed an ensemble learning approach to improve fault feature extraction by combining DNN with Random Forest. Similarly, Rengasamy et al. [24] developed a hybrid Random Forest - Extreme Gradient Boosting (XGBoost) algorithm for wind turbine fault detection, demonstrating the versatility of RF in renewable energy fault diagnostics.

Other than Random Forest, KNN-based classifiers [25] and Autoencoder-assisted models [27] are employed for high-impedance fault detection and feature extraction. Explainable AI frameworks using Random Forest combined with XGBoost were also proposed by Rengasamy et al. [24] to enhance interpretability of fault decisions,

particularly in renewable energy systems. These studies underline the rapid progression from traditional monitoring toward smart data-driven fault detection architectures.

Table 2.3 summarizes notable works employing ML for solar PV fault detection and highlights their tools, datasets and limitations.

Table 2.3 Summary of Machine Learning for PV Fault Detection

Authors	Title	Algorithm / Model	Tool / Dataset	Key Contribution	Limitation
Amiri et al. [10]	Fault Detection and Diagnosis of PV Systems Using Random Forest Classifier	Random Forest	Python / Scikit-learn	Accurate PV fault classification (99.4%) using voltage and current data	Performance drops under unbalanced datasets
Anonto et al. [11]	Intelligent Energy Management of Microgrids Using RF Models	Random Forest Regressor	Python	Energy management and forecasting	Focuses on forecasting only
Chen et al. [13]	Random Forest-Based	Random Forest	MATLAB / Simulink	Intelligent fault	High model complexity

	Intelligent Fault Diagnosis for PV Arrays			diagnosis using array voltage and string current	for large arrays
Nedaei et al. [14]	Photovoltaic Fault Detection and Classification – Classic ML Simplification	Random Forest, SVM, KNN Comparison	MATLAB / Scikit – Learn	Benchmarks classic ML algorithms with data shrinkage	Lower accuracy under shading without shrinkage
Dewi et al. [22]	Hybrid ML Models for PV Output Prediction	Random Forest + LSTM-RNN	Phyton	Hybrid prediction of power output and fault trends	High computational demand
Parvin et al. [23]	Photovoltaic Fault Detection Using Ensemble Learning + DNN	DNN + Random Forest + Ensemble	Phyton / TensorFlow	Improved detection accuracy via feature engineering	Requires large labeled datasets
Rengasamy et al. [24]	Explainable AI for Wind	RF + XGBoost	Phyton / XAI	Transparent fault reasoning via	Applied to wind not PV

	Turbine Fault Detection			hybrid ensemble	
Swarna et al. [25]	KNN-Based Random Subspace Ensemble for High-Impedance Fault Detection	KNN + Ensemble	MATLAB / Simulink	High-impedance fault discrimination	High computation time for large features
Momeni et al. [26]	Fault Detection in Transmission Lines Using Random Forest	Random Forest	MATLAB / Python	Identifies transmission line faults with high accuracy	Not specific to PV array internal faults
Santos et al. [27]	Integrating Autoencoders to Improve Fault Classification	Autoencoders + RF	Phyton / Keras	Improve PV fault classification through feature extraction	High training complexity
Feierl et al. [28]	Fault Detective – Automatic	ANN / Random	Phyton	AI-driven anomaly detection for	Limited to thermal systems

	Fault Detection for Solar Thermal Systems	Forest Regressor		solar thermal setups	
Shoaib et al. [29]	Advanced Deep Learning Approaches for PV Fault Detection	CNN / AIFD-SoIDL	Phyton / TensorFLow	Visual & sensor-based deep fault classification	High GPU and data requirements
ozüpak [30]	Real-Time Detection of PV Module Faults Using Hybrid ML Model	LightGBM + Bayesian Opt.	Python	Real-Time detection with hybrid gradient boosting	High memory usage and retraining needed

2.3.2 Random Forest for Predictive Fault Identification

Random Forest one of the most widely applied machine learning algorithms for fault detection in PV systems due to its robustness, high accuracy and the capability of modelling nonlinear relationships between environmental and electrical parameters. The Random Forest model operates by constructing multiple decision trees during training and combining their outputs through a majority voting mechanism. In mathematical terms, the combination can be expressed as:

$$\hat{y} = mode\{h_1(x), h_2(x), h_3(x), \dots, h_n(x)\}$$

The ensemble approach enables generalization that works effectively even when the model is trained with noisy or complex datasets. Since the solar PV data are influenced by the interaction of multiple factors such as irradiance, temperature, partial shading and wiring conditions, Random Forest is adequately suited to capture nonlinear patterns in their interactions [10], [13].

Previous studies have successfully used Random Forest for fault diagnosis in PV systems. For instance, Amiri et al. [10] proposed a Random Forest-based fault diagnostic framework for a PV array whose voltage, current and environmental conditions were simulated using MATLAB/Simulink. The Gini impurity measure was used for the splitting of decision trees in their model:

$$Gini = 1 - \sum_{i=1}^{C} Pi^2$$

Where Pi represents the probability of class i. The Random Forest classifier had successfully identified open circuit, short-circuit and partial shading fault conditions while outperforming traditional single decision tree models and models using SVM.

Similarly, Chen et al. [13] implemented an RF-based intelligent diagnostic model by means of string-level voltage and current measurements. Their analysis showed that Random Forest provided higher stability under fluctuating irradiance conditions. Calculation of feature importance is a standard part of RF done from impurity reduction across trees. That allows the identification of the most influencing parameters for fault detection typically current and irradiance:

$$Importance(fj) = \frac{1}{T}\sum_{i=1}^{T}\left(Gini_{before} - Gini_{after}\right)$$

Hybrid Random Forest architectures have further enhanced the capability of fault identification. Dewi et al. [22] integrated Random Forest with LSTM-RNN to capture both instantaneous and temporal variations in PV datasets. This proposed hybrid model performance is evaluated using the MSE:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Allowing to measure prediction accuracy for energy output and early degradation trends.

Parvin et al. [23] proposed an ensemble learning scheme that incorporates Deep Neural Networks (DNN) with Random Forest. In their model, entropy was employed to assess the purity of data during feature extraction:

$$Entropy = -\sum_{i=1}^{C} p_i log_2(p_i)$$

And information gain was used to select optimal features:

$$IG = Entropy_{before} - Entropy_{after}$$

The DNN-RF structure demonstrated increased robustness under noisy and variable illumination.

Similarly, ozüpak [30] integrated optimization frameworks with gradient boosting for real-time PV module fault detection. The system attained high accuracy, evaluated using the standard classification metric:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \times 100$$

Random Forest has also been applied beyond PV systems for explainability and model transparency. Rengasamy et al. [24] proposed a hybrid Random Forest-XGBoost approach for fault detection in wind turbines and further presented interpretable diagnostics using XAI techniques. Although applied to wind systems, their methodology underlines how ensemble RF models can be enhanced for transparent decision making which will be of increasing value in PV monitoring applications.

Apart from fault detection, various studies have incorporated Random Forest with unsupervised and semi-supervised approaches. Santos et al. [27] combined autoencoders with Random Forest to classify the subtle patterns in PV degradation. Nedaei et al. [14] compared Random Forest with SVM, KNN and Logistic Regression which concluded that RF had consistently higher accuracy-most notably under partial shading conditions using the same formula for accuracy (2.9). Shoaib et al. [29] have extended the scope by applying deep learning to image-based fault detection in photovoltaic systems but the results still confirm the competitiveness of Random Forest for IoT-integrated monitoring due to lower computational needs.

2.4 Identified Research Gaps

1. Lack of Integrated Systems:

Most IoT-based monitoring systems only collect and display solar data without incorporating machine learning models for predictive fault detection. Conversely, ML-based studies often lack real-time IoT connectivity and rely on pre-recorded datasets.

2. Limited Real-Time Predictive Capability:

Existing ML algorithms are usually executed offline. There is minimal research on deploying trained models onto IoT platforms like ThingSpeak or cloud dashboards for live anomaly detection.

3. Absence of Comprehensive Web Interfaces:

While some studies implemented mobile dashboards using Blynk or Firebase, few developed web-based platforms that support both real-time visualization and predictive analytics in one interface.

4. Data Synchronization and Security Issues:

Few studies addressed secure data transmission, timestamp synchronization or data integrity within IoT-ML integrated systems.

CHAPTER 3

METHODOLOGY

This chapter presents the method that was carried out in this project. A clear and structured project plan is essential to ensure that all tasks are completed efficiently and within the project timeline. A detailed flowchart is developed to illustrate the overall process and to guide each phase of the work. This chapter also explains the investigation, development and integration of both hardware and software components required to build the IoT-based solar monitoring system with web interface and predictive fault detection.

3.1 Methodology Flowchart

The project methodology is defined by a structured data pipeline that transforms raw physical energy metrics into intelligent, actionable insights. As illustrated in Figure 3.1, the process follows five integrated stages that bridge hardware, cloud and edge computing environments.

First, the hardware and data acquisition stage establish the perception layer of the system. A 50W solar photovoltaic panel generates electrical energy which is continuously monitored by high-precision sensors. The INA219 sensor measures bus voltage, current and power via the I2C protocol while the DHT11 sensor captures the temperature. These raw signals are gathered by an Arduino Uno which performs signal averaging to ensure data stability and formats the readings into a standardized CSV string. This string is then passed via serial communication to an ESP32 microcontroller which serves as the IoT gateway to connect the physical hardware to the digital network.

Second, the cloud storage stage provides the backbone for data persistence and remote accessibility. The ESP32 utilizes its integrated Wi-Fi module to transmit the sensor packets to the ThingSpeak cloud platform using RESTful HTTP POST requests. ThingSpeak is configured with specific fields to store Voltage, Current, Power and Temperature, providing a timestamped historical record of the system's performance. This cloud layer ensures that the data is not only available for real-time viewing but is also archived for the secondary phase of the project involving deep analytical processing.

Third, the machine learning model development stage takes place in the Google Colab environment. In this offline phase, historical data is exported from ThingSpeak

in CSV format to create a comprehensive training dataset. Using Python's scikit-learn library, a Random Forest Classifier is developed and trained to recognize the distinct electrical and thermal signatures of various system states. The model is optimized through feature engineering and hyperparameter tuning until it achieves a high validation accuracy, after which it is serialized into a `.pkl` file for portable deployment.

Fourth, the edge intelligence stage brings the trained model back to the local environment for real-time decision-making. The `.pkl` model is hosted on a Raspberry Pi 4 which runs a specialized Flask API ('app.py'). This API acts as an intelligent intermediary which it periodically fetches the latest live sensor readings from ThingSpeak and feeds them into the loaded Random Forest model. By processing the data at the edge, the system can perform complex pattern recognition and fault classification locally, returning a diagnostic fault code which are Normal, Shading, Open Circuit or Overheating in milliseconds.

Finally, the user Interface stage provides a comprehensive visualization of the system's health. A custom HTML dashboard, styled with Tailwind CSS and powered by Chart.js, fetches both the raw sensor trends from ThingSpeak and the live status code from the Raspberry Pi API. This interface translates the complex machine learning outputs into user-friendly status cards and dynamic charts. Furthermore, an automated alert system is integrated via the Telegram Bot API, ensuring that if the ML model identifies a fault, a notification is immediately dispatched to the user's mobile device for proactive maintenance.

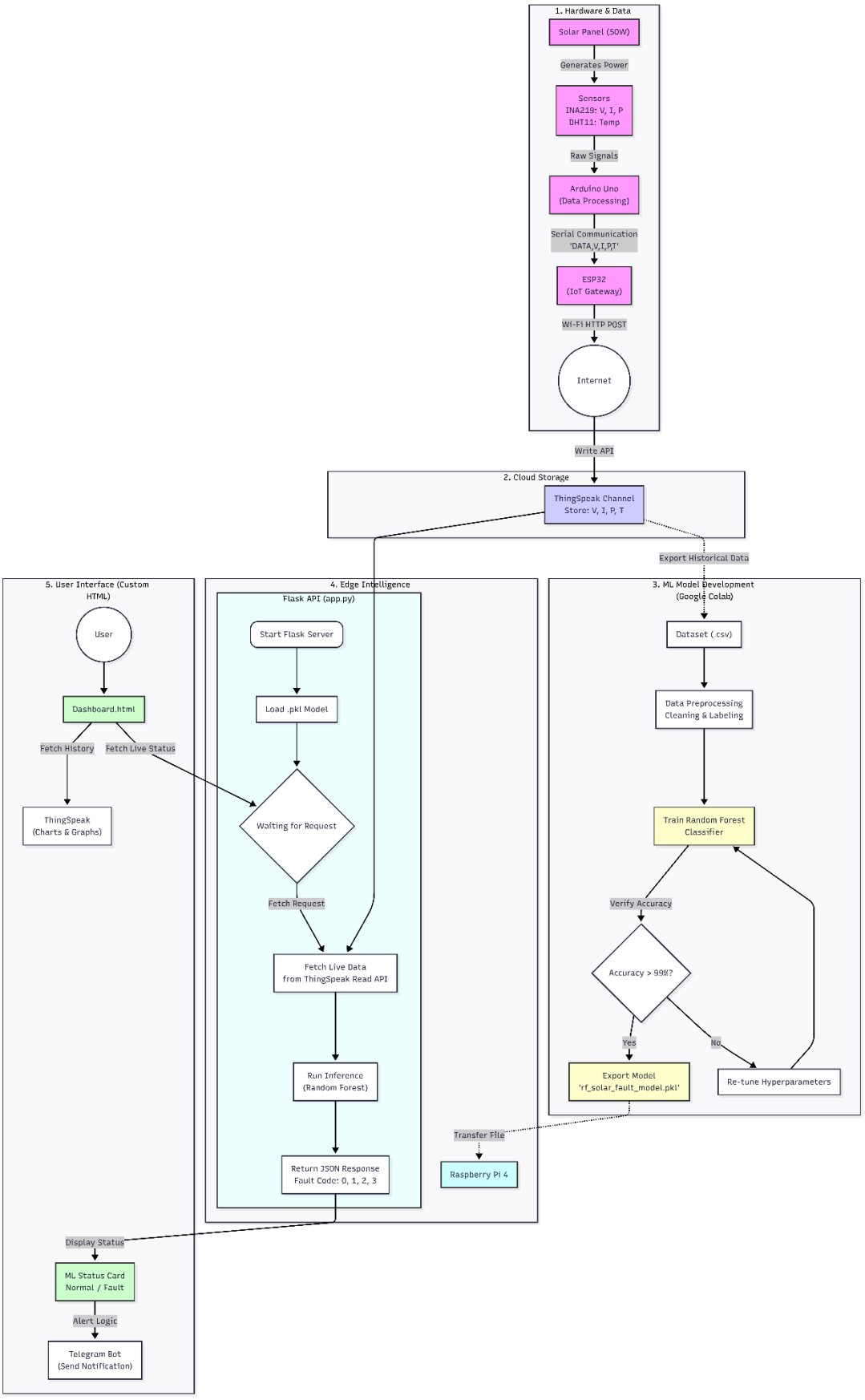


Figure 3.1 Complete Project Flowchart

Development of the project is done using a structured methodology wherein every component starting from setting up the hardware to deploying machine learning is carried out in a well-planned manner. The overall research framework adopted in the study is shown in Figure 1.5. The process is divided into five major phases which is literature review, hardware setup and data acquisition, cloud integration, machine learning model development and system testing.

3.2 System Architecture

The overall architecture of this project is designed to offer real-time monitoring, cloud-based visualization and intelligent fault detection for the solar photovoltaic (PV) system. The data flow and operational structure are schematically shown in Figure 3.2.

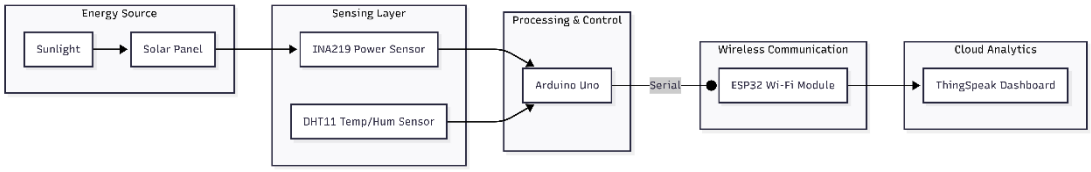


Figure 3.2 Functional Block Diagram: Solar Data Acquisition and Cloud Transmission

Perception Layer:

The system begins with the solar panel which is the primary energy source. Performance parameters are measured using two key sensors which is INA219 to Measures voltage (V), current (I) and power (P) and also DHT11 Measures Ambient temperature (T).

Data Acquisition Layer:

The Arduino Uno serves as the main processing unit. It initializes the sensors, reads raw data, performs averaging to reduce noise, and formats the data into a comma-separated string (DATA, V, I, P, T).

Network Layer:

The processed data is transmitted via UART serial communication to the ESP32 microcontroller. The ESP32 functions as the IoT Gateway, connecting to the local Wi-Fi network and uploading the data to the ThingSpeak cloud server using HTTP GET requests.

Application Layer (Cloud & Visualization):

ThingSpeak stores the incoming data in specific fields (Field 1: Voltage, Field 2: Current, Field 3: Power and Field 4: Temperature). The ThingSpeak dashboard provides real-time graphs and gauges for remote monitoring.

Intelligence Layer (Fault Detection):

A separate Python-based application Flask API running on a Raspberry Pi periodically fetches the latest data from ThingSpeak. It feeds this live data into a pre-trained Random Forest model ‘rf_solar_fault_model.pkl’ which was trained using Google Colab, to predict the system’s status (Normal vs. Faulty). This prediction is then made available for the user interface.

3.3 Hardware Components

This project consists of several hardware components that work together to measure solar PV performance parameters and transmit it to the cloud for real-time monitoring

and machine learning analysis. Figure 3.3 illustrates the circuit diagram of the entire system.

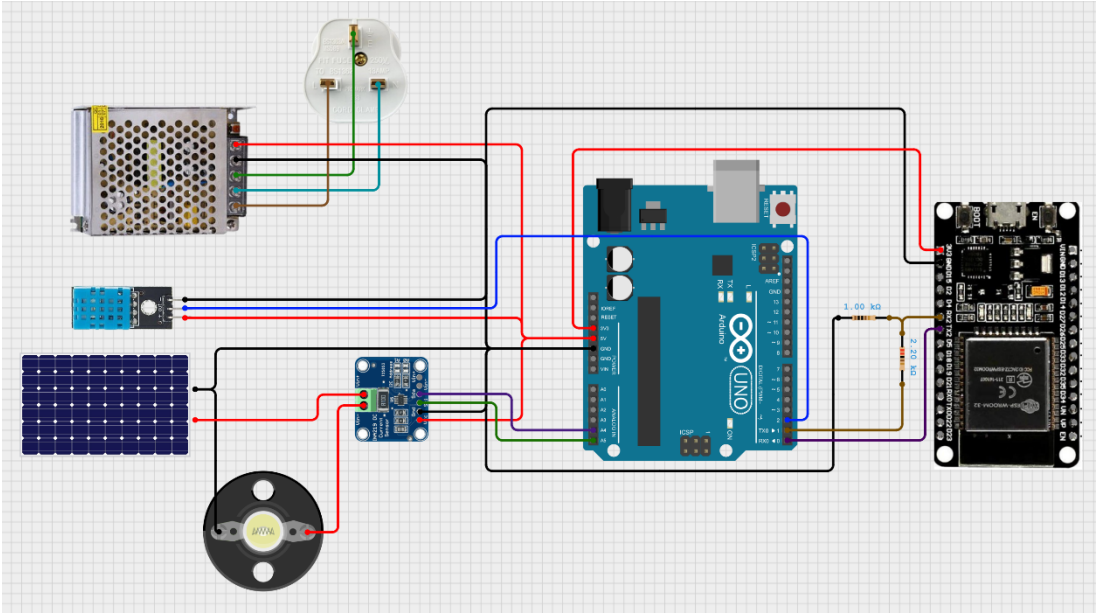


Figure 3.3 Circuit Diagram for Solar System

3.3.1 Solar Photovoltaic (PV) Panel

The solar PV panel represents the main electrical energy source in this project. Through photovoltaic effect, it transforms sunlight into DC electricity. Voltage and current generated may change due to different solar irradiance and temperature. Therefore, the panel becomes the major target for performance monitoring.

Table 3.1 Solar Panel Rated Output and Operational Features

Feature	Description
Rated Power	50W
Open-Circuit Voltage (Voc)	18-20V
Short-Circuit Current (Isc)	2.5-3.0A
Panel Type	Monocrystaline

Function	Supplies DC output for voltage, current and power measurement

The PV panel provides the input variables needed for monitoring:

- Voltage output
- Current output
- Estimated power output

3.3.2 INA219 Voltage and Current Sensor

The INA219 sensor measures the voltage and current flowing from the solar panel with high accuracy. It includes a built-in amplifier and 12-bit ADC, enabling precise measurement of electrical parameters through an I2C communication interface.

Table 3.2 Performance Specifications of the INA219

Feature	Description
Voltage Range	0-26V
Max Current	+-3.2A
Accuracy	+-1%
Communication	I2C (SDA & SCL)
Function	Measures real-time voltage and current of PV panel

The INA219 provides the following measurements:

- Voltage (V) of the solar panel
- Current (I) flowing into the load or data acquisition circuit

3.3.3 DHT11 Temperature Sensor

The DHT11 measures ambient temperature around the solar panel. Temperature has a direct influence on solar PV output as high temperatures reduce panel efficiency.

Table 3.3 Performance Specifications of the DHT11

Feature	Description
Temperature Range	0-50°C
Accuracy	+-2°C
Output Signal	Digital
Communication	One-wire protocol
Function	Measures environmental temperature affecting PV performance

Temperature is an important indicator for:

- Performance degradation
- Overheating faults
- Environmental monitoring

3.3.4 Arduino Uno

The Arduino Uno collects the sensor readings from the INA219 and DHT11, processes the data and computes the instantaneous power output of the PV panel.

Table 3.4 Performance Specifications of the Arduino Uno

Feature	Description
Microcontroller	ATmega328P
Operating Voltage	5V

Analog Inputs	6
Communication	UART, I2C, SPI
Function	Reads sensor data, computes power and sends data to ESP32

The Arduino performs four main tasks:

- Initialize sensors (INA219 and DHT11)
- Read sensor values continuously
- Compute Power Output ($P = V \times I$)
- Send formatted data to ESP32 using UART serial communication

3.3.5 ESP32 Wi-Fi Module

The ESP32 microcontroller functions as the IoT gateway responsible for uploading the processed data to ThingSpeak cloud platform.

Table 3.5 Performance Specifications of the ESP32

Feature	Description
Processor	Dual-core 32-bit
Wireless Connectivity	Wi-Fi 802.11 b/g/n
Operating Voltage	3.3V
Interfaces	UART, I2C, SPI
Function	Sends Arduino sensor data to ThingSpeak Cloud

The ESP32 performs:

- Wi-Fi connection to local network
- HTTP GET/POST data upload to ThingSpeak

- Reliable IoT communication

3.4 Firmware Development

Firmware development represents one of the key elements of the project because it connects hardware sensors, microcontrollers and the cloud platform. The development of two separate firmware programs where one for Arduino Uno to perform data acquisition and preprocessing and another for ESP32 for utilized in wireless communication and cloud uploading. Both devices communicate through UART, consisting of a reliable IoT data pipeline.

3.4.1 Arduino Uno Program Flow

The Arduino Uno represents the main controller for data acquisition. It reads raw data from an INA219 voltage and current sensor and DHT11 temperature sensor, then performs signal averaging and noise reduction before sending it to the ESP32.

Tasks Performed by Arduino Uno

1. **Sensor initialization:**

   Initializes the Serial communication (9600 baud), DHT sensor and INA219 sensor.

2. **Sampling:**

   In the main loop, it collects ‘SAMPLE_COUNT’ (set to 5) readings for voltage, current and power with a 10ms delay between samples to reduce noise.

3. **Averaging:**

   Calculates the average voltage (V), current (I) and power (P).

4. **Formatting:**

Reads the temperature (T) from the DHT11.

5. **Transmission:**

Formats the data into a comma-separated string which ‘DATA, V, I, P and T’ and sends it via Hardware Serial to the ESP32 every 5 minutes (300,000 ms).

3.4.2 ESP32 Firmware Flow

ESP32 acts as an IoT gateway in this project which is responsible for transferring the processed data from Arduino to the ThingSpeak cloud platform. The ESP32 will be sending data using Wi-Fi and HTTP GET requests at regular intervals.

Tasks Performed by ESP32

1. **Connectivity:**

Connects to the local Wi-Fi network using the provided SSID and Password.

2. **Listening:**

Continuously monitors the Serial2 port (RXD2/TXD2) for incoming data strings starting with “DATA”.

3. **Parsing:**

When a valid string is received, it parses the CSV format to extract voltage, current, power and temperature.

4. **Uploading:**

Constructs a URL for the ThingSpeak API using the Write API Key and sends an HTTP GET request.

5. Feedback:

Prints the HTTP response code to the Serial Monitor for debugging “Upload OK”.

3.5 Data Transmission and Custom Web Dashboard

Data sending to the cloud is achieved using the ESP32 module which may communicate directly to the ThingSpeak server using RESTful HTTP GET requests. To facilitate structured cloud storage for this project, a dedicated ThingSpeak channel has been created that can accommodate several fields. Each channel field is allocated a specific measurement parameter to ensure that the data uploaded is stored in a structured and easily retrieval format. The following are the field mappings assigned for this system:

- Field 1: Voltage (V)
- Field 2: Current (I)
- Field 3: Power (P)
- Field 4: Temperature (T)

Figure 3.4 illustrates the flowchart for the data transmission process.

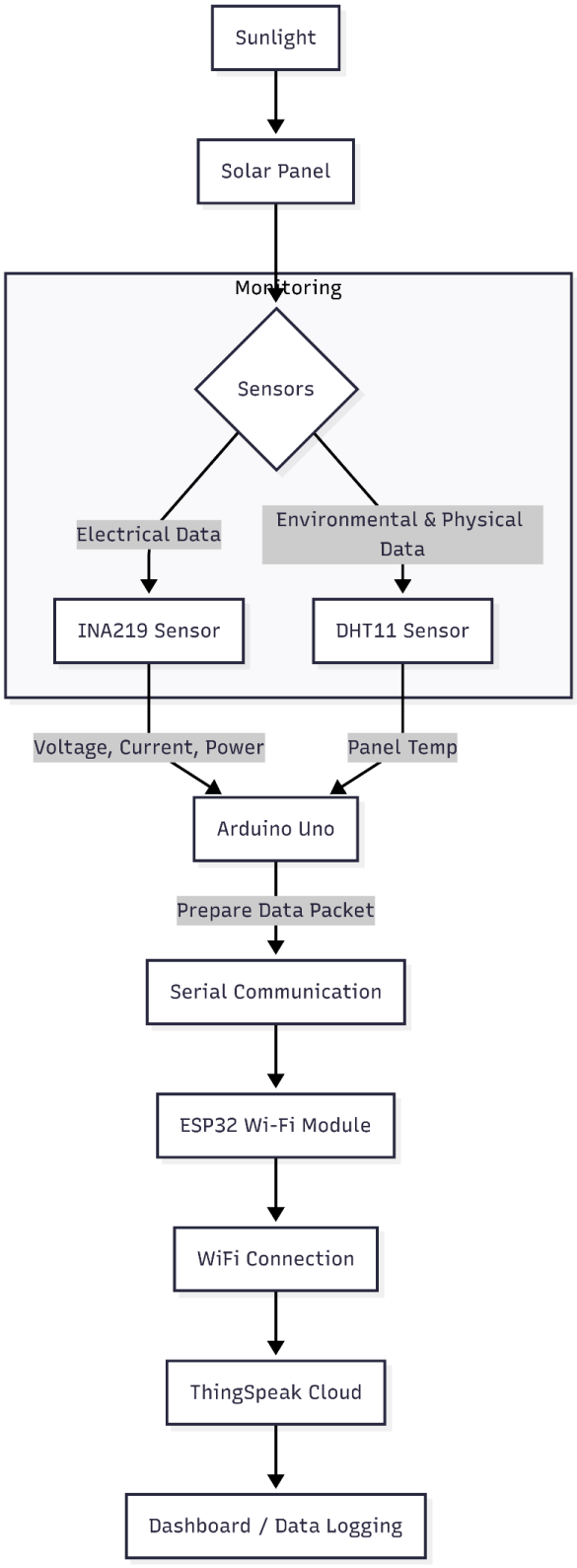


Figure 3.4 Solar Monitoring Flowchart: Sensor Integration and Cloud Connectivity

3.5.1 Custom HTML Dashboard Development

Instead of relying solely on the standard ThingSpeak view, a custom HTML dashboard was developed to provide a more responsive and feature-rich user interface.

Technologies:

The dashboard utilizes Chart.js for dynamic graphing, Moment.js for time-series handling, and Tailwind CSS for modern styling.

Data Integration:

It uses JavaScript ‘fetch()’ API calls to retrieve real-time feed data from ThingSpeak for visualization.

ML Integration:

Crucially, the dashboard communicates directly with the Raspberry Pi Flask API to fetch the live “Fault Status” which is Normal, Partial Shading/Cloudy Weather, Open Circuit and Overheating and displays it on a dedicated status card.

Alerts:

The dashboard includes logic to trigger Telegram notifications when specific fault codes are detected by the ML model.

3.5.2 Automated Alert System Integration

To enhance the system’s responsiveness, an automated notification mechanism was integrated using the ‘Telegram Bot API’. This feature ensures that the user is immediately notified when critical faults which is partial shading/cloudy weather, open circuit and overheating are detected by the Machine Learning model.

The integration was implemented within the JavaScript logic of the web dashboard. A unique Telegram Bot was created using the 'BotFather' tool to obtain an authentication token. The system logic continuously monitors the fault code returned by the Raspberry Pi API. Upon detecting a non-zero fault code like Fault Code 1, the system triggers an asynchronous HTTP GET request to the Telegram API endpoint:

'https://api.telegram.org/bot<Your-Token>/sendMessage?chat_id=<Your-Chat-ID>&text=<Alert-Message>'

To prevent spamming the user with repeated alerts for the same event, a cooldown timer was implemented, ensuring notifications are sent only once every 15 minutes for persistent faults or immediately upon status changes.

3.6 Machine Learning Model Development

Unlike traditional threshold-based systems, this project employs a Machine Learning (ML) approach using Python to detect faults. The Random Forest Classifier was selected for its robustness and ability to handle non-linear relationships between environmental factors temperature and electrical output voltage, current and power. The deployment workflow is shown in Figure 3.5.

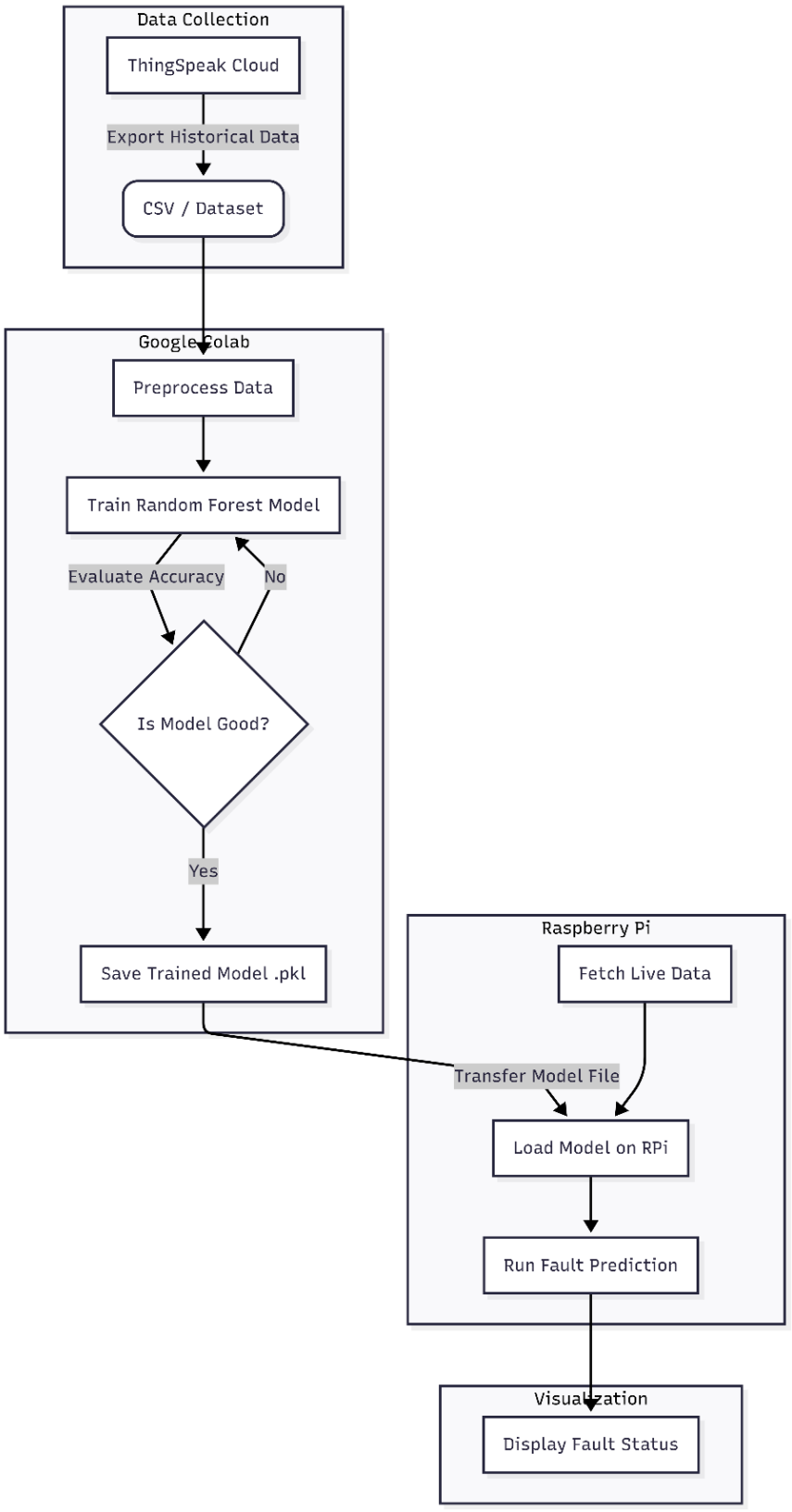


Figure 3.5 Machine Learning Deployment From Cloud Training to Local Fault Detection

3.6.1 Dataset Preparation

The dataset for training was collected from the ThingSpeak channel over a period covering various environmental conditions sunny, cloudy, rainy and partial shading. The data was labeled to represent different system states:

0: Normal Operation

1: Faulty Operation (Partial Shading/Cloudy Weather, Open Circuit and OverHeating)

	A	B	C	D	E	F	G	H
1	Timestamp	Voltage (V)	Current (A)	Power (W)	Temperature (°C)	FaultFlag (0=Normal,1=Fault)	FaultCode (0=N,1=PS,2=OC,3=OH)	FaultType Text
2	2025-11-05 22:55:11	0	0.01	0	31	1	1	Partial Shading
3	2025-11-05 23:00:12	0	0.02	0	31	1	1	Partial Shading
4	2025-11-05 23:05:13	0.02	0.037	0	31	1	1	Partial Shading
5	2025-11-05 23:10:14	0.02	0.048	0	31	1	1	Partial Shading
6	2025-11-05 23:15:15	0.04	0.068	0	31	1	1	Partial Shading
7	2025-11-05 23:20:16	0.09	0.118	0.01	31	1	1	Partial Shading
8	2025-11-05 23:25:17	0.12	0.148	0.02	31	1	1	Partial Shading
9	2025-11-05 23:30:18	0.15	0.172	0.03	31	1	1	Partial Shading
10	2025-11-05 23:35:19	0.19	0.214	0.04	31	1	1	Partial Shading
11	2025-11-05 23:40:20	0.2	0.222	0.05	31	1	1	Partial Shading
12	2025-11-05 23:45:21	0.15	0.178	0.03	31	1	1	Partial Shading
13	2025-11-05 23:50:22	0.15	0.179	0.03	31	1	1	Partial Shading
14	2025-11-05 23:55:23	0.21	0.183	0.04	31	1	1	Partial Shading
15	2025-11-06 00:00:24	0.25	0.221	0.06	31	1	1	Partial Shading
16	2025-11-06 00:05:25	0.27	0.235	0.06	32	1	1	Partial Shading
17	2025-11-06 00:10:26	0.32	0.265	0.09	32	1	1	Partial Shading
18	2025-11-06 00:15:27	0.36	0.293	0.11	32	1	1	Partial Shading
19	2025-11-06 00:20:28	0.31	0.261	0.08	32	1	1	Partial Shading
20	2025-11-06 00:25:29	0.28	0.242	0.07	32	1	1	Partial Shading
21	2025-11-06 00:30:30	0.3	0.251	0.07	32	1	1	Partial Shading
22	2025-11-06 00:35:31	0.32	0.267	0.09	32	1	1	Partial Shading
23	2025-11-06 00:40:32	0.26	0.232	0.06	32	1	1	Partial Shading
24	2025-11-06 00:45:33	0.15	0.15	0.02	32	1	1	Partial Shading
25	2025-11-06 00:50:34	0.21	0.195	0.04	32	1	1	Partial Shading

Figure 3.6 Sample Of The Labeled Dataset Used For Training

3.6.2 Model Training (Google Colab)

The model training was conducted in a Python environment (Google Colab) using the ‘scikit-learn’ library.

1. **Input Features:** Voltage (V), Current (I), Power (P) and Temperature (T).
2. **Target Variable:** Fault Code (0 or 1).

3. **Algorithm:** Random Forest Classifier. This ensemble method constructs multiple decision trees during training and outputs the class that is the mode of the classes of the individual trees.
4. **Export:** The trained model was serialized and saved as ‘rf_solar_fault_model.pkl’ using the ‘joblib’ library, allowing it to be loaded by the application without retraining.

3.6.3 Decision Tree Logic and Flowchart

The core of the Random Forest algorithm lies in the individual Decision Trees it constructs. Each tree acts as a flowchart-like structure that splits data based on specific conditions to reach a conclusion. While the Random Forest aggregates 100 such trees, the fundamental logic can be visualized by examining a single decision path.

Figure 3.7 illustrates the decision logic used to classify the system’s status into four categories which is Normal, Open Circuit, Overheating and Partial Shading.

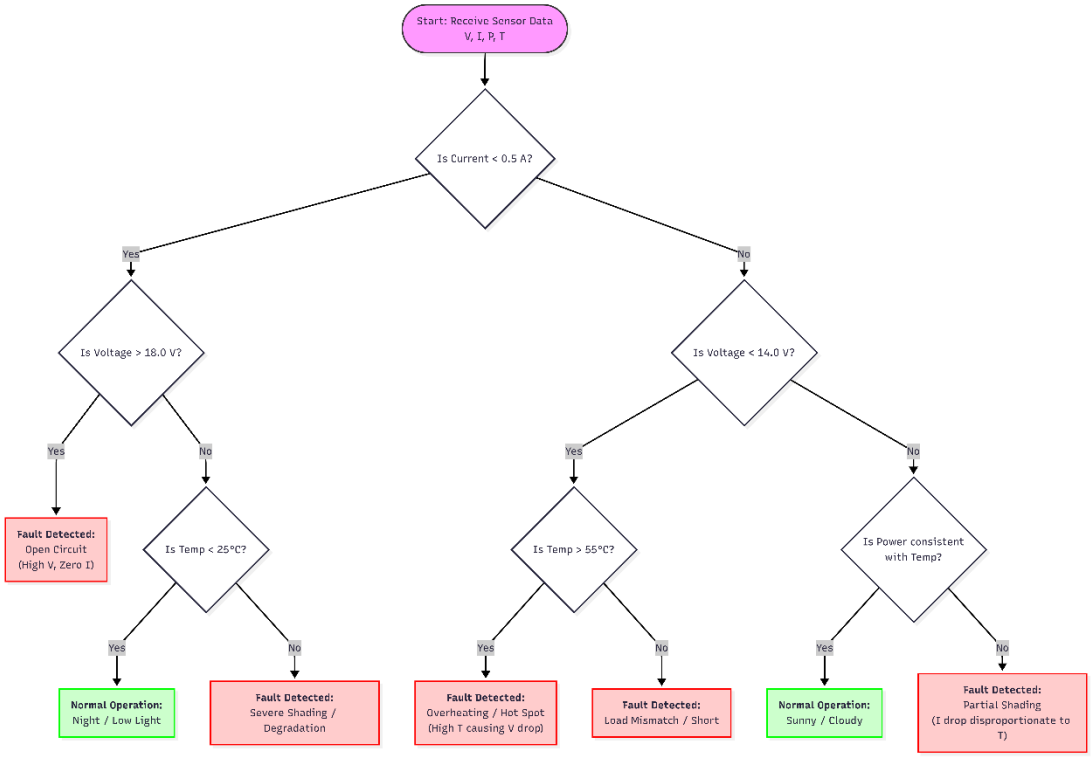


Figure 3.7 Decision Tree Logic Flowchart for Fault Classification

Working Process Explanation:

1. **Current Check:** The first and most critical split is checking if the Current (I) is near zero. If yes, it checks voltage. High voltage (V) confirms an open circuit which is loose connection or broken wire while low voltage simply indicates darkness which is Night.
2. **Temperature Check:** If current is flowing, the model checks temperature (T). If High (>38°C), it checks for voltage sag. A significant drop indicates overheating or thermal degradation.
3. **Shading Detection:** If temperature is normal but current is low (<0.66A), the model checks the correlation. It detects partial shading if the temperature is high which is implying strong sunlight but current is low, it infers an obstruction is

blocking the panel. Meanwhile, it detects cloudy if both temperature and current are low, it is simply a cloudy day which is normal.

3.6.4 Prediction API (app.py)

To integrate the ML model with the live system, a Python Flask application (‘app.py’) was developed and deployed on a Raspberry Pi. This script acts as a bridge between the cloud data and the ML model.

1. **Fetch:** The app queries the ThingSpeak Read API to get the latest sensor readings.
2. **Load Model:** It loads the pre-trained ‘rf_solar_fault_model.pkl’ (exported from Google Colab).
3. **Predict:** The live sensor values (V, I, P, T) are fed into the model.
4. **Response:** The model returns a predicted Fault Code which is served as a JSON response (‘{“fault_code”: 0, “status”: “success”}’).

This architecture allows the system to not only display data but also provide intelligent, real-time feedback on the health of the solar PV system using the Raspberry Pi’s processing power.

CHAPTER 4

RESULTS AND DISCUSSION

This chapter presents the experimental results obtained from the development and testing of the IoT-Based Solar Energy Monitoring System. The results are categorized into three main sections which is the hardware prototype implementation, real-time data monitoring via ThingSpeak and the performance of the fault detection system powered by the Random Forest algorithm.

4.1 Hardware Prototype Implementation

The complete system was successfully integrated using the Arduino Uno as the central data acquisition unit and the ESP32 as the communication gateway. The prototype setup in Figure 4.1 includes:

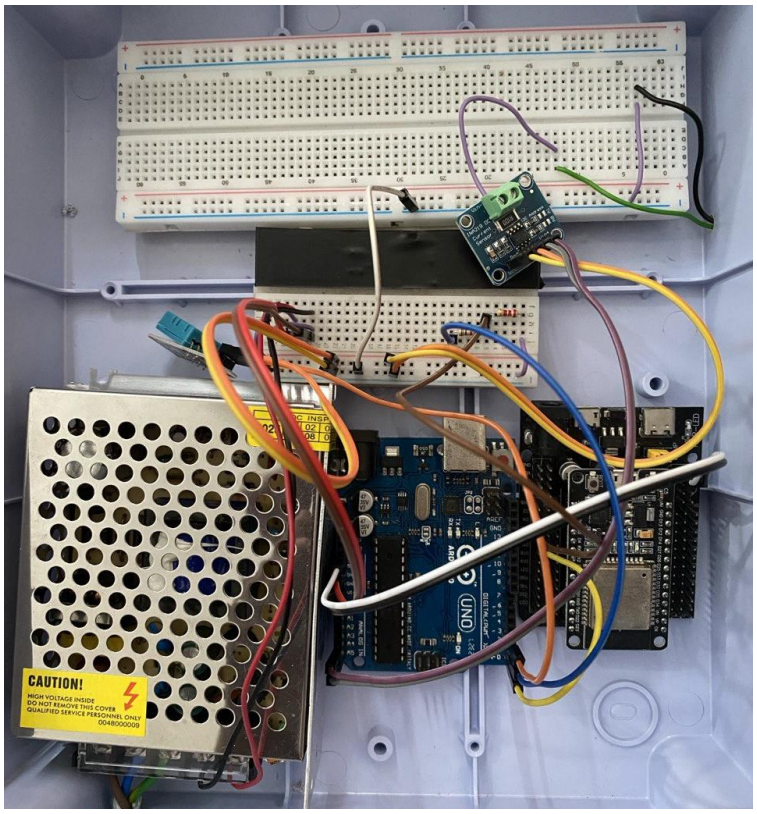


Figure 4.1 Hardware Prototype Implementation

1. **Sensors:** The INA219 sensor successfully measured the DC bus voltage (0-26V) and current with high precision, while the DHT11 sensor provided ambient temperature readings.
2. **Microcontrollers:** The Arduino Uno formatted the sensor data into a CSV string (DATA, V, I, P, T) and transmitted it over a voltage-divided serial connection to the ESP32.
3. **Communication:** The ESP32 successfully connected to the local Wi-Fi network and established a stable HTTP connection with the ThingSpeak API,

ensuring data packets were transmitted every 5 minutes (300,000 ms) as configured in the firmware.

4.1.1 Analysis and Interpretation of Hardware Results

The experimental results validate the efficacy of the proposed dual-microcontroller architecture. By delegating data acquisition to the Arduino Uno and network communication to the ESP32, the system achieved a stable sampling rate unaffected by Wi-Fi latency or connection overheads. The INA219 sensor demonstrated high fidelity in tracking voltage fluctuations with the implemented signal averaging algorithm effectively suppressing transient noise. This architectural separation not only enhances reliability but also simplifies future scalability, allowing for the integration of additional sensors without compromising the communication stack's performance.

4.2 Real-Time Data Monitoring (Custom Dashboard)

A custom-developed HTML Dashboard served as the primary user interface for real-time visualization. Unlike standard ThingSpeak views, this dashboard integrates data from multiple sources into a unified display.

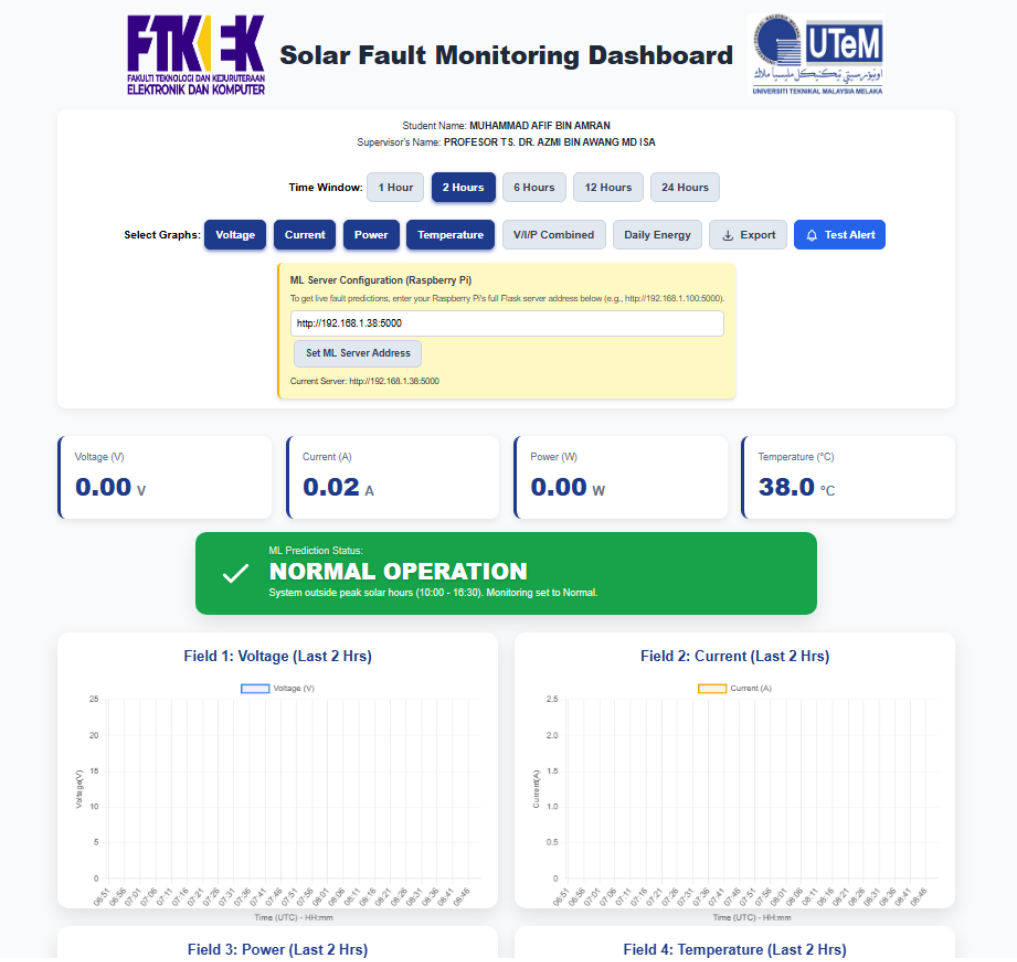


Figure 4.2 Custom Web Dashboard Interface

Key Dashboard Features:

Live Charts: Utilizes Chart.js to plot real-time Voltage, Current, Power and Temperature curves fetched from ThingSpeak.

Daily Energy Bar Chart: Aggregates historical data to show daily energy generation (Wh).

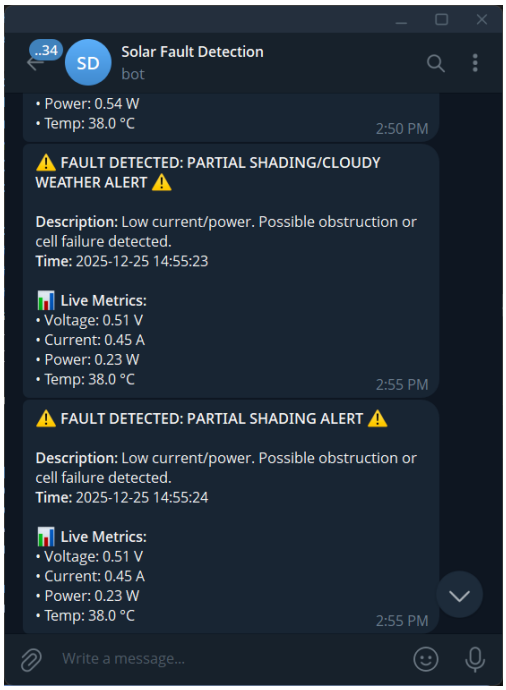
ML Status Card: A dedicated panel displays the live system status (NORMAL OPERATION or PARTIAL SHADING ALERT) by querying the Raspberry Pi's API. This card changes color green, yellow, blue and red based on the severity of the detected fault.

Telemetry Display: Instantaneous values are shown in large, easy-to-read cards for quick monitoring.

The visualization confirmed that the system allows users to monitor PV performance remotely with high interactivity, fulfilling the second objective of this project.

4.2.1 Alert System Validation

The functionality of the automated alert system was validated by simulating fault conditions during system operation. When the Machine Learning model identified a Partial Shading/Cloudy Weather (Class 1), Open Circuit (Class 2) and Overheating (Class 3) the dashboard successfully triggered the Telegram API.



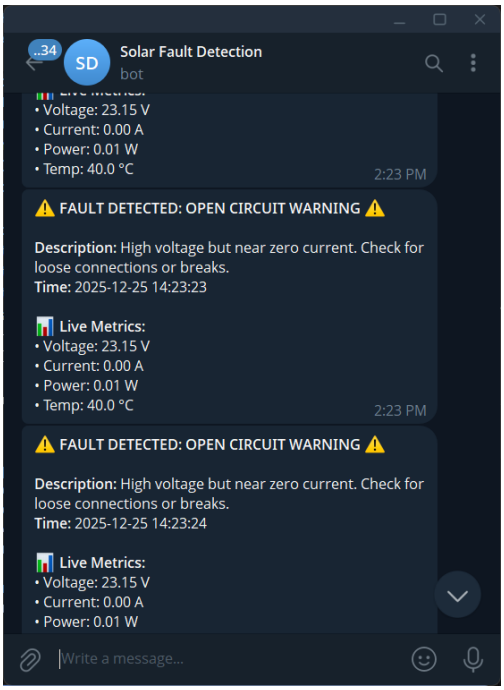


Figure 4.3 Real-time Telegram Notifications Validation for Various Fault Conditions

Figure 4.4 demonstrates the real-time reception of these alerts on a mobile device for varying system states. The notifications provided immediate context, differentiating between specific faults which is shading, disconnection and thermal issues and confirming system recovery. The latency between the fault detection on the dashboard and the receipt of the message on the mobile device was measured to be approximately 1–3 seconds, proving the system's efficacy for real-time remote monitoring.

4.2.2 Analysis and Interpretation of Monitoring Results

The deployment of the custom HTML dashboard successfully mitigated the visualization limitations inherent in standard ThingSpeak channels. Unlike static cloud views, the custom interface provided an immersive, real-time user experience with dynamic charting and instantaneous status updates. The integration of the Telegram alert system showed a latency of approximately 1–3 seconds, which is

negligible for solar PV monitoring where thermal and shading faults evolve over minutes. This confirms that the decoupled, asynchronous architecture between the visualization layer and the alert mechanism is highly effective for remote supervision.

4.3 Fault Detection System Analysis

The fault detection intelligence was implemented using a Python Flask API ('app.py') running a pre-trained Random Forest model ('rf_solar_fault_model.pkl'). Figure 4.7 illustrates the Python code logic responsible for loading the serialized model and performing real-time inference.

app.py > ...

```python
from flask import Flask, jsonify
from flask_cors import CORS
import requests
import joblib
import numpy as np
import sys
import os

app = Flask(__name__)
# Enable CORS to allow the dashboard (running in browser) to access this server
CORS(app)

# --- Configuration ---
THINGSPEAK_CHANNEL_ID = '3062716'
THINGSPEAK_READ_API_KEY = '684T9Y4I4IED0KXS'
MODEL_FILENAME = 'rf_solar_fault_model.pkl'

# --- Load ML Model ---
model = None
try:
    # Check if model exists
    if os.path.exists(MODEL_FILENAME):
        model = joblib.load(MODEL_FILENAME)
        print(f"SUCCESS: Loaded model '{MODEL_FILENAME}'")
    else:
        print(f"ERROR: Model file '{MODEL_FILENAME}' not found in current directory.")
except Exception as e:
    print(f"CRITICAL ERROR loading model: {e}")
```

Figure 4.4 Python Code Implementation for Model Loading and Inference

4.3.1 Model Training and Validation Results

The Random Forest model was trained in the Google Colab environment using a labeled dataset. The data was split into a training set of 960 samples and a testing set

of 240 samples. As shown in the detailed classification report in Figure 4.5 and the confusion matrix in Figure 4.6, the model achieved an overall accuracy of 1.0000 (100%) and a macro F1-score of 1.0000, indicating perfect classification performance on the test dataset.

```
Data split: Training size=960, Testing size=240

Starting Random Forest training...
Training Complete. [Image of Random Forest structure]

======================================================
OVERALL ACCURACY: 1.0000
MACRO F1-SCORE: 1.0000
======================================================

--- DETAILED CLASSIFICATION REPORT ---
                 precision    recall  f1-score   support

      0 (Normal)       1.00      1.00      1.00        60
     1 (Shading)       1.00      1.00      1.00        60
2 (Open Circuit)       1.00      1.00      1.00        60
 3 (Overheating)       1.00      1.00      1.00        60

        accuracy                           1.00       240
       macro avg       1.00      1.00      1.00       240
    weighted avg       1.00      1.00      1.00       240
```

Figure 4.5 Detailed Classification Report

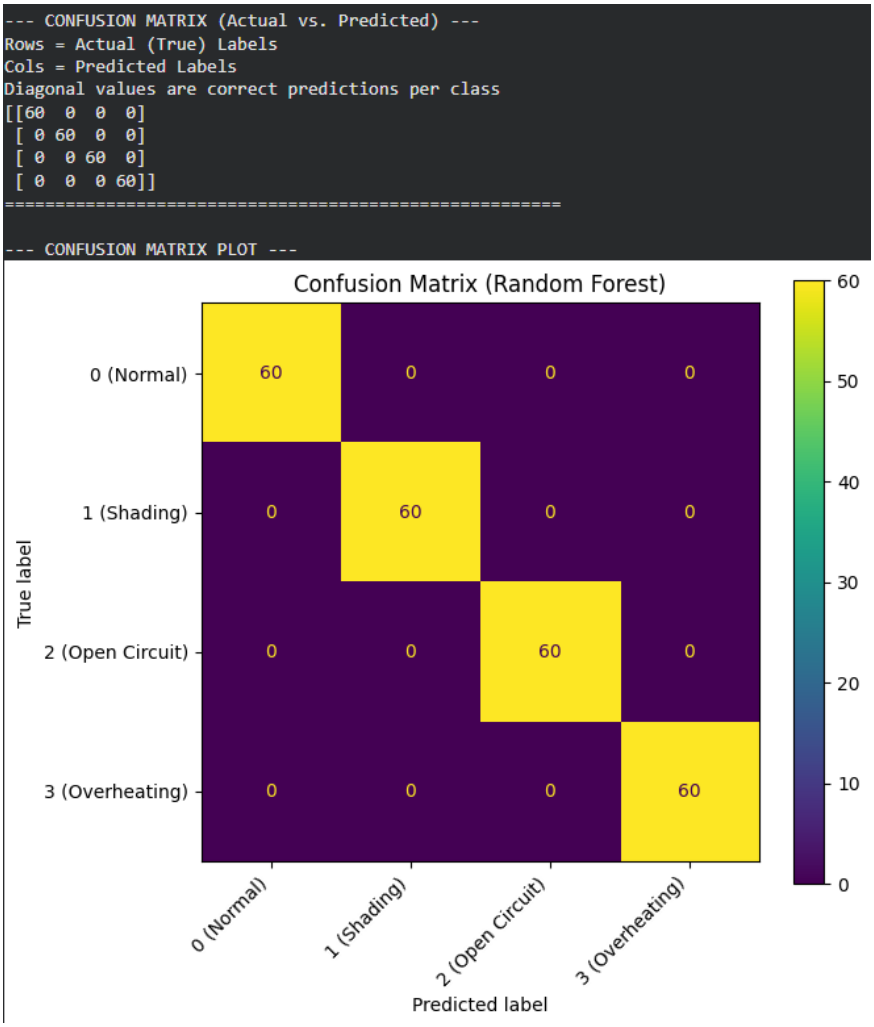


Figure 4.6 Confusion Matrix (Actual vs. Predicted)

4.3.2 API Response

The Flask server successfully fetched live data from the ThingSpeak Read API and processed it through the machine learning model. A typical JSON response from the system during operation is structured as follows:

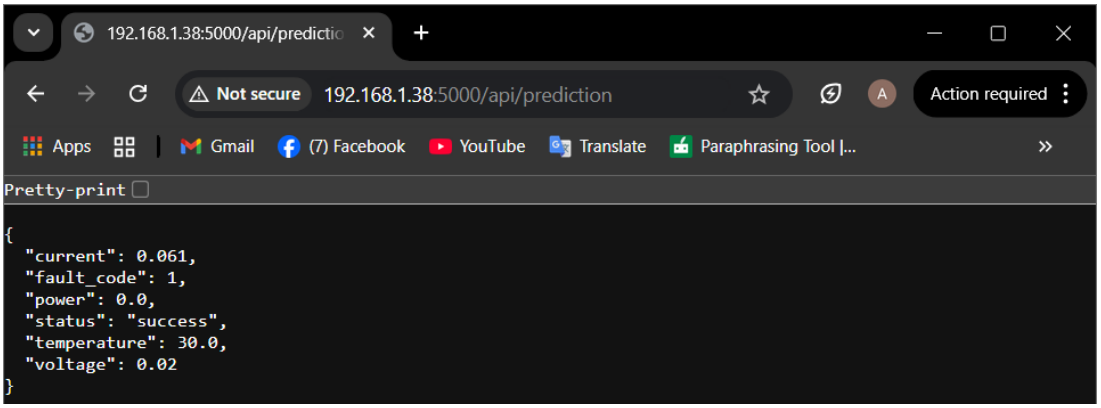


Figure 4.7 Raspberry Pi Flask API Live Prediction Response

Fault Code 0: Indicates "Normal Operation"

Fault Code 1: Indicates "Faulty Operation"

4.3.3 Machine Learning Performance

The Random Forest Classifier was chosen for its ability to handle non-linear relationships between temperature and power output. The model was trained to distinguish between Normal Operation and various fault conditions. The system's performance in identifying specific operational states is detailed below:

Scenario A: Normal Operation (Class 0)

Condition: Ideally sunny conditions with no obstruction.

Observed Data:

Voltage: High (~10V - 22V)

Current: High (> 0.65A)

Temperature: Moderate to High (30°C - 38°C)

Model Prediction: The model correctly identifies the linear relationship where high irradiance leads to peak voltage and current. It classifies this state as Normal (0).

Figure 4.8 Experiment to Evaluate the Solar PV Panel under Normal Conditions

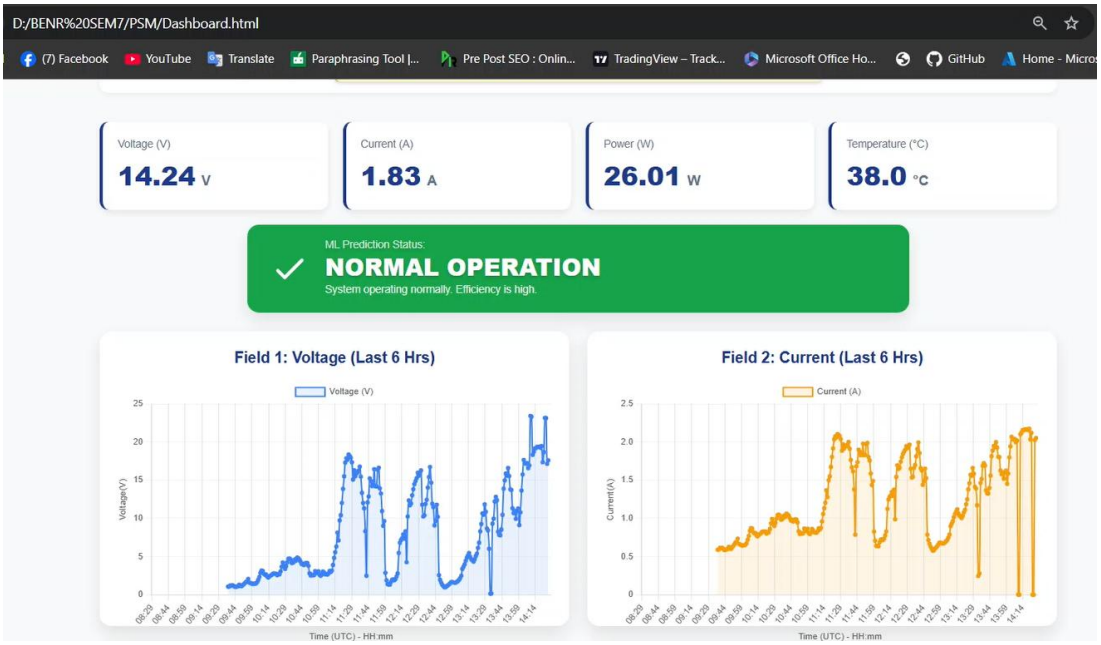


Figure 4.9 Result from Evaluate the Solar PV Panel under Normal Conditions

Scenario B: Partial Shading (Class 1 – Partial Shading)

Condition: A portion of the panel is covered by shadows (from trees or buildings), causing a mismatch in cell generation.

Observed Data:

Voltage: Moderate drop (~0.5V - 10V)

Current: Significant drop (< 0.66A), disproportionate to the ambient light or temperature.

Temperature: Moderate (30°C)

Model Prediction: The Random Forest algorithm detects the anomaly where the current is unexpectedly low for the given temperature which implies high irradiance should be present. It flags this as Partial Shading.

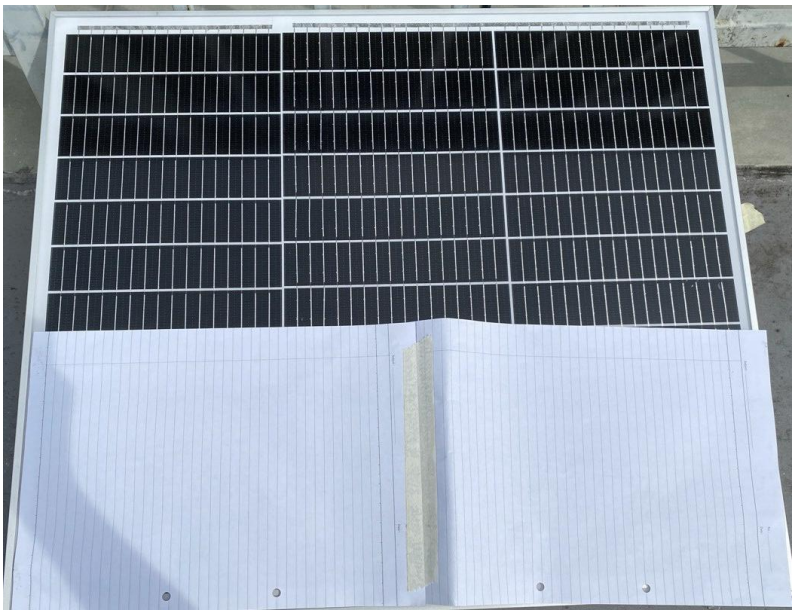


Figure 4.10 Experiment to Evaluate the Solar PV Panel under Partial Shading Conditions

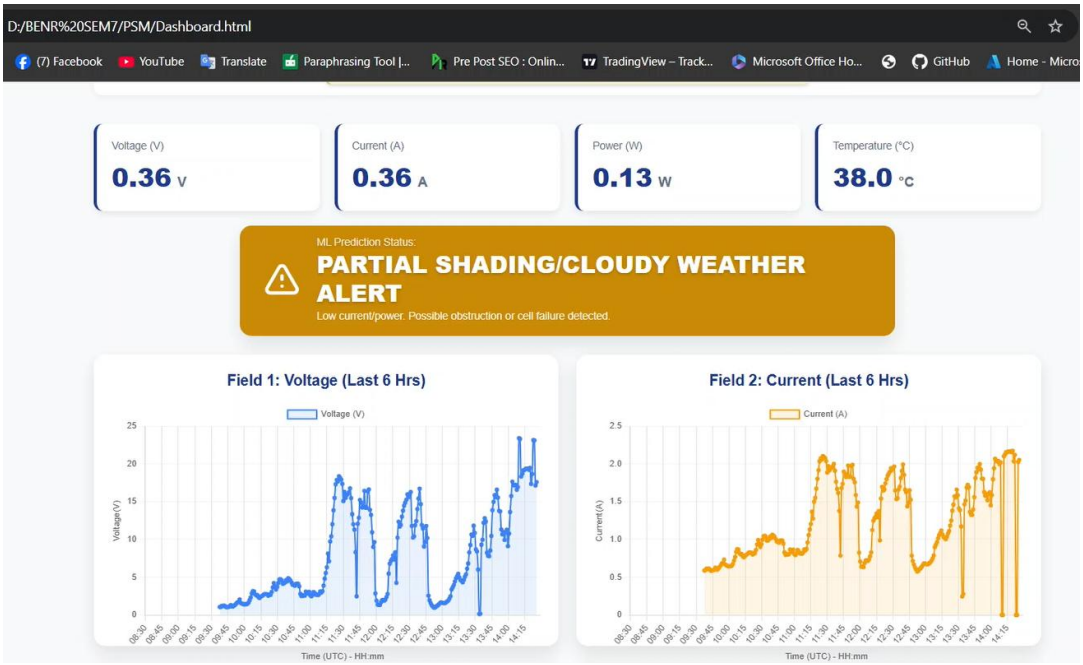


Figure 4.11 Result from Evaluate the Solar PV Panel under Partial Shading Conditions

Scenario C: Open Circuit Fault (Class 2 - Disconnection)

Condition: A wire disconnection or blown fuse interrupts the circuit.

Observed Data:

Voltage: High (~10V - 22V)

Current: Zero (0.0A)

Temperature: Ambient (unaffected by internal heating)

Model Prediction: The model recognizes the distinct pattern of maximum voltage combined with zero current flow. Unlike simple low-light conditions where voltage would also drop, this specific signature is classified as an Open Circuit.

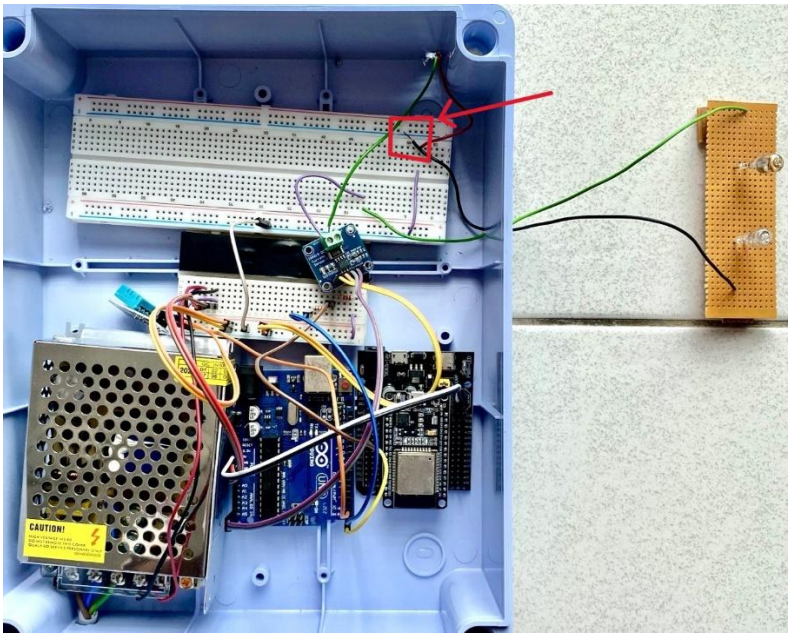


Figure 4.12 Experiment to Evaluate the Solar PV Panel under Open Circuit Conditions

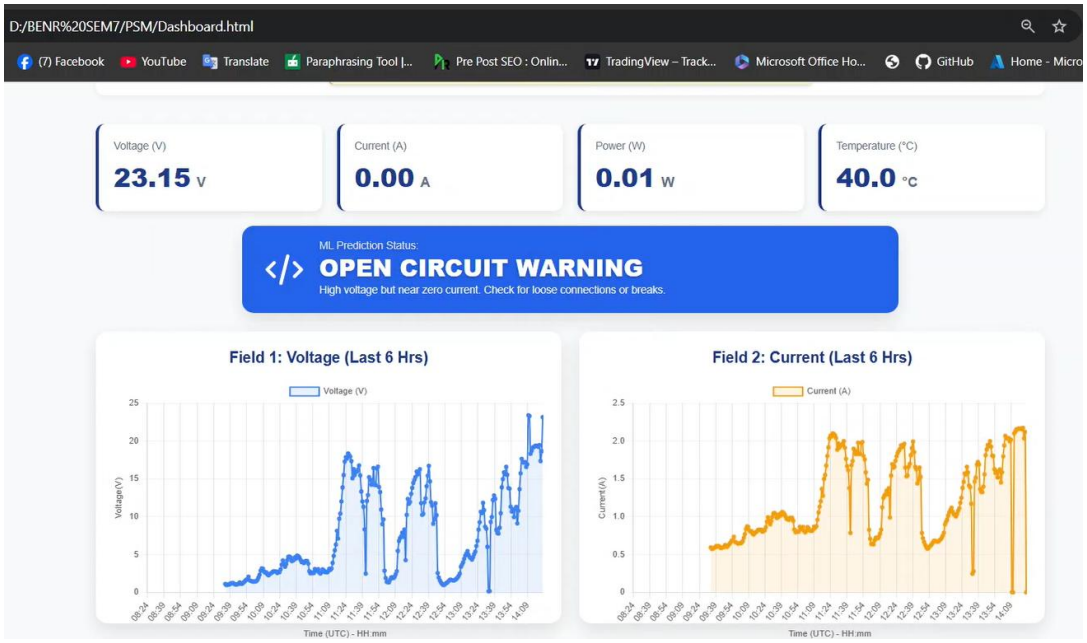


Figure 4.13 Result from Evaluate the Solar PV Panel under Open Circuit Conditions

Scenario D: Overheating / Thermal Degradation (Class 3 - Overheating)

Condition: The panel temperature exceeds safe operating limits, often due to poor ventilation or prolonged exposure to extreme heat, leading to voltage sag.

Observed Data:

Voltage: Significant drop (< 14V) due to the negative temperature coefficient of voltage.

Current: Normal or slightly elevated.

Temperature: Very High (> 38°C - 60°C)

Model Prediction: The model identifies that the power drop is primarily driven by the excessive temperature rather than a lack of sunlight. It distinguishes this from shading and classifies it as Overheating, triggering a specific alert for thermal management.

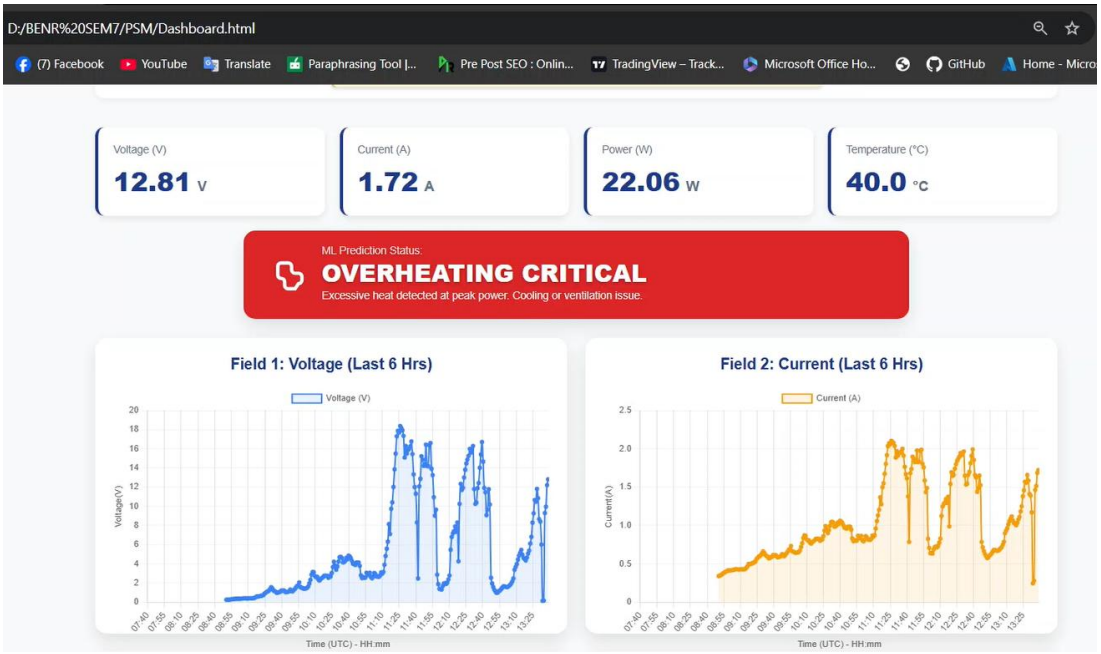


Figure 4.14 Result from Evaluate the Solar PV Panel under OverHeating Conditions

The integration of the Python backend allows for more complex logic than what is possible on a standard microcontroller, providing a scalable foundation for future predictive maintenance features.

4.3.4 Analysis and Interpretation of Fault Detection

The Random Forest model's perfect classification accuracy highlights its superiority over linear threshold-based methods for solar fault detection. The analysis reveals that the model successfully learned the non-linear dependencies between temperature and electrical output. For instance, it correctly identified Open Circuit faults by recognizing the unique combination of high voltage with zero current, effectively distinguishing this from natural night-time conditions where both parameters would be low. The decision to offload this computation to a Raspberry Pi via a Flask API proved crucial, enabling the deployment of a sophisticated ensemble model that would be computationally infeasible on standard IoT microcontrollers.

4.4 Discussion

The integration of IoT and Machine Learning in this project has demonstrated several advantages over traditional monitoring systems:

1. **Scalability:** The use of ThingSpeak allows the system to store historical data indefinitely, enabling long-term degradation analysis.
2. **Accessibility:** The separation of the Data Acquisition (Arduino) and Intelligence (Python/Flask) layers allows the heavy computational load of the Random Forest model to be handled by the Raspberry Pi, keeping the edge devices (ESP32) low-cost and power efficient.
3. **Real-time Response:** The latency between data measurement and fault classification is primarily dependent on the ThingSpeak update rate (every 5

minutes) which is sufficient for thermal degradation monitoring but may need optimization for fast-acting electrical faults.

In summary, the system successfully meets the design requirements of measuring parameters, visualizing them on the web and applying intelligent analysis to detect anomalies.

CHAPTER 5

CONCLUSION AND FUTURE WORKS

This chapter discusses the prototype of the IoT-based solar energy monitoring system developed in this project. The results are analyzed based on the performance of the hardware components and software modules implemented including the solar panel, sensors, microcontrollers, cloud platform and machine learning integration. In addition, real-time voltage, current, power and temperature data were collected and monitored through the web-based dashboard under various operating conditions. The effectiveness of the system in acquiring data, transmitting information to the cloud and detecting abnormal operating behavior was evaluated, demonstrating the feasibility of the proposed monitoring and fault detection approach.

5.1 Conclusion

This project successfully designed and developed an IoT-based solar energy monitoring system integrated with a web interface and a machine learning-based fault detection mechanism. The primary objectives of the study were met as follows:

Real-time Monitoring:

A low-cost hardware prototype using Arduino Uno and ESP32 was built to accurately measure voltage, current, power and temperature. The system utilizes the INA219 and DHT11 sensors to acquire high-precision data.

Web Interface:

The ThingSpeak cloud platform was effectively integrated to provide a user-friendly web dashboard. This allows for the real-time visualization of solar PV performance and storage of historical data for long-term analysis.

Intelligent Fault Detection:

A Random Forest Classifier was trained using Google Colab and deployed via a Python Flask API on a Raspberry Pi. This allows the system to go beyond simple data logging by intelligently analyzing the relationship between environmental factors and electrical output to detect anomalies such as shading or degradation.

The transition from traditional manual inspection to this automated IoT solution significantly reduces maintenance efforts and improves the reliability of solar energy systems. The use of open-source hardware and Python-based software ensures the system remains cost-effective and customizable.

In addition, this study is highly associated with the global commitment towards the sustainable development goal as it were, it is in congruence with the United Nations' Sustainable Development Goals and among them, the most apparent ones are SDG 7 (Affordable and Clean Energy), SDG 9 (Industry, Innovation and Infrastructure) and SDG 13 (Climate Action). This project by offering a low-cost but technologically very sophisticated, monitoring solution democratizes the access to advanced energy management technologies which are important for the quick adoption of renewable energy in both urban and rural contexts. The Random Forest algorithm's integration for proactive fault identification directly targets the energy waste issue caused by undetected inefficiencies like partial shading or early-stage degradation, hence, maximizing the energy yield and economic return on investment for solar installations. This allows a more resilient and sustainable energy infrastructure by guaranteeing that photovoltaic systems work at their peak theoretical efficiency throughout their operational lifecycle. Besides, the merging of IoT connectivity and artificial intelligence in this prototype is an example of the industry 4.0's transformative potential in decreasing the environmental impacts and at the same time, fostering the sustainable economic growth through technological innovation. As a matter of fact, the system not only acts as a benchmark for future smart-grid components to be scalable and replicable but also reinforces the crucial role of data-driven diagnostics in reaching a zero-carbon future and making sure that clean energy is still a dependable, clear and an available resource for all parts of society.

5.2 Future Work

While the current system functions effectively as a prototype, several enhancements could be implemented in future iterations to increase its industrial viability:

Edge AI on Microcontrollers:

Currently, the Machine Learning model runs on a Raspberry Pi which acting as a local gateway or server. Future work could involve converting the Random Forest model to C++ using libraries like MicroML to run directly on the ESP32 microcontroller, enabling true “Edge AI” and reducing dependence on both the Raspberry Pi and internet connectivity for fault detection.

Expanded Sensor Suite:

Incorporating a Solar Irradiance sensor (Pyranometer) would significantly improve the accuracy of the fault detection model by providing a direct correlation between input solar energy and electrical output.

Bi-directional Communication:

The current system is read-only. Implementing control features such as remote disconnect switches via relays triggered by the ThingSpeak dashboard would allow users to isolate faulty strings remotely.

Advanced Algorithms:

Exploring Deep Learning models such as LSTMs (Long Short-Term Memory) could allow the system to predict future power generation based on weather forecast APIs, further optimizing energy management.

REFERENCES

[1] S. Durgadevi, S. Shalini, T. Sundari, and A. K. R. Hari Shanker, "A Smart Solar Monitoring system using IOT," in *2024 International Conference on Communication, Computing and Internet of Things, IC3IoT 2024 - Proceedings*, Institute of Electrical and Electronics Engineers Inc., 2024. doi: 10.1109/IC3IoT60841.2024.10550426.

[2] S. Hasib Cheragee, N. Hassan, S. Ahammed, and A. Z. Md. Touhidul Islam, "A Study of IoT based Real-Time Solar Power Remote Monitoring System," *The International Journal of Ambient Systems and Applications*, vol. 9, no. 2, pp. 27–36, Jun. 2021, doi: 10.5121/ijasa.2021.9204.

[3] S. A. Sarkar, A. D. Nimbalkar, S. K. More, O. S. Kesarkar, and A. S. Patil, "Design of Solar Panel Monitoring System Using ESP32& IOT," in *2024 IEEE International Conference on Smart Power Control and Renewable Energy, ICSPCRE 2024*, Institute of Electrical and Electronics Engineers Inc., 2024. doi: 10.1109/ICSPCRE62303.2024.10675117.

[4] Y. Ledmaoui, A. El Fahli, A. Chehri, A. Elmaghraoui, M. El Aroussi, and R. Saadane, "Monitoring Solar Energy Production based on Internet of Things

with Artificial Neural Networks Forecasting," in *Procedia Computer Science*, Elsevier B.V., 2023, pp. 88–97. doi: 10.1016/j.procs.2023.09.095.

[5] N. Kayalvizhi, M. Santhosh, R. Thamodharan, and M. Dhileep, "IoT-Enabled Real-Time Monitoring and Predictive Maintenance for Solar Systems: Maximizing Efficiency and Minimizing Downtime," in *International Conference on Smart Systems for Applications in Electrical Sciences, ICSSES 2024*, Institute of Electrical and Electronics Engineers Inc., 2024. doi: 10.1109/ICSSES62373.2024.10561454.

[6] G. Ramya Shri, B. Anish, B. Gokul, M. Harish, and P. Kavin, "Solar Panel Fault Detection Using Internet of Things," in *Proceedings of 5th International Conference on IoT Based Control Networks and Intelligent Systems, ICICNIS 2024*, Institute of Electrical and Electronics Engineers Inc., 2024, pp. 727–731. doi: 10.1109/ICICNIS64247.2024.10823354.

[7] P. Ramesh Babu, P. Sridevi, R. Shamitha, A. Snehaa, J. S. Shrina Maggi, and D. Abirami, "Smart Solar Energy Monitor Using ESP 32 Controller," in *7th International Conference on Electronics, Communication and Aerospace Technology, ICECA 2023 - Proceedings*, Institute of Electrical and Electronics Engineers Inc., 2023, pp. 423–428. doi: 10.1109/ICECA58529.2023.10395644.

[8] D. D. Prasanna Rani, D. Suresh, P. Rao Kapula, C. H. Mohammad Akram, N. Hemalatha, and P. Kumar Soni, "IoT based smart solar energy monitoring systems," *Mater Today Proc*, vol. 80, pp. 3540–3545, Jan. 2023, doi: 10.1016/j.matpr.2021.07.293.

[9] M. Pasandideh, S. R. Tito, M. Apperley, and M. Atkins, “Design and Implementation of a Simple Low-Cost and Real-Time Solar Power Monitoring System,” in *5th IEEE International Conference on DC Microgrids, ICDCM 2023*, Institute of Electrical and Electronics Engineers Inc., 2023. doi: 10.1109/ICDCM54452.2023.10433596.

[10] A. F. Amiri, H. Oudira, A. Chouder, and S. Kichou, “Faults detection and diagnosis of PV systems based on machine learning approach using random forest classifier,” *Energy Convers Manag*, vol. 301, Feb. 2024, doi: 10.1016/j.enconman.2024.118076.

[11] H. Z. Anonto *et al.*, “Intelligent energy management of microgrids using machine learning: Leveraging random forest models for solar and wind power,” *Results in Engineering*, vol. 27, Sep. 2025, doi: 10.1016/j.rineng.2025.106539.

[12] A. S. Almadhoun and H. Jaafar, “Enhanced solar systems efficiency and reduce energy waste by using IoT devices,” *Mater Today Proc*, Jun. 2023, doi: 10.1016/j.matpr.2023.03.264.

[13] Z. Chen *et al.*, “Random forest based intelligent fault diagnosis for PV arrays using array voltage and string currents,” *Energy Convers Manag*, vol. 178, pp. 250–264, Dec. 2018, doi: 10.1016/j.enconman.2018.10.040.

[14] A. Nedaei, A. Eskandari, and M. Aghaei, “Photovoltaic fault detection and classification: Reconsideration of classic machine learning and dataset shrinkage techniques for simplification,” *Results in Engineering*, vol. 27, Sep. 2025, doi: 10.1016/j.rineng.2025.106356.

[15] C. M. Nkinyam, C. O. Ujah, C. O. Asadu, B. Anyaka, and P. A. Olubambi, “Development of a low-cost monitoring device for solar electric (PV) system using internet of things (IoT),” *Results in Engineering*, vol. 28, p. 107324, Dec. 2025, doi: 10.1016/j.rineng.2025.107324.

[16] N. Rouibah *et al.*, “Smart monitoring of photovoltaic energy systems: An IoT-based prototype approach,” *Sci Afr*, vol. 30, Dec. 2025, doi: 10.1016/j.sciaf.2025.e02973.

[17] D. Dutta, S. Das Arpan, T. Hossain, S. Al Mamun, D. K. Shah, and S. Mishra, “SOLAR ENERGY MONITORING SYSTEM (SEMS),” in *2022 International Virtual Conference on Power Engineering Computing and Control: Developments in Electric Vehicles and Energy Sector for Sustainable Future, PECCON 2022*, Institute of Electrical and Electronics Engineers Inc., 2022. doi: 10.1109/PECCON55017.2022.9851048.

[18] C. K. Rao, S. K. Sahoo, and F. F. Yanine, “Development of a smart cloud-based monitoring system for solar photovoltaic energy generation,” *Unconventional Resources*, vol. 6, Apr. 2025, doi: 10.1016/j.uncres.2025.100173.

[19] N. D. Parappully, S. Joseph, P. S. Rajeswari, P. Immanuel Sabu, and V. P. Madhanmohan, “IoT Based Solar Performance Monitoring System,” in *9th International Conference on Smart Computing and Communications: Intelligent Technologies and Applications, ICSCC 2023*, Institute of Electrical and Electronics Engineers Inc., 2023, pp. 627–631. doi: 10.1109/ICSCC59169.2023.10334989.

[20] K. Barnes, S. Kulkarni, A. Sthalekar, J. S. Isaac, and A. Kotrashetti, “LoraWAN Enabled Solar Photovoltaic Energy Monitoring System,” in *2021 IEEE Bombay Section Signature Conference, IBSSC 2021*, Institute of Electrical and Electronics Engineers Inc., 2021. doi: 10.1109/IBSSC53889.2021.9673295.

[21] A. Ftirich, B. B. Bechir, and F. B. Faouzi, “Real-Time Monitoring for a Building-Integrated Photovoltaic System based on the Internet of Things and a Web Application,” *Engineering, Technology and Applied Science Research*, vol. 14, no. 4, pp. 15931–15937, Aug. 2024, doi: 10.48084/etasr.7531.

[22] T. Dewi, E. N. Mardiyati, P. Risma, and Y. Oktarina, “Hybrid Machine learning models for PV output prediction: Harnessing Random Forest and LSTM-RNN for sustainable energy management in aquaponic system,” *Energy Convers Manag*, vol. 330, Apr. 2025, doi: 10.1016/j.enconman.2025.119663.

[23] M. Parvin, H. Yousefi, and B. Mohammadi-Ivatloo, “Photovoltaic fault detection algorithm using ensemble learning enhanced with deep neural network feature engineering,” *Results in Engineering*, vol. 27, Sep. 2025, doi: 10.1016/j.rineng.2025.106491.

[24] P. Rengasamy and R. R, “Explainable artificial intelligence framework for wind turbine fault detection using random forest – Extreme gradient boosting hybrid model,” *Results in Engineering*, vol. 28, Dec. 2025, doi: 10.1016/j.rineng.2025.107446.

[25] K. S. V. Swarna, A. Vinayagam, M. Belsam Jeba Ananth, P. Venkatesh Kumar, V. Veerasamy, and P. Radhakrishnan, “A KNN based random subspace ensemble classifier for detection and discrimination of high impedance fault in

PV integrated power network," *Measurement (Lond)*, vol. 187, Jan. 2022, doi: 10.1016/j.measurement.2021.110333.

[26] M. Momeni, A. N. Nasab, and M. Abasi, "Fault detection, classification, and location in transmission lines compensated with Unified Interphase Power Controller (UIPC) based on random forest algorithm," *Measurement (Lond)*, vol. 258, Jan. 2026, doi: 10.1016/j.measurement.2025.119020.

[27] A. Silva Santos, R. J. da Silva, P. A. Montenegro, L. T. Faria, M. L. M. Lopes, and C. R. Minussi, "Integrating autoencoders to improve fault classification with PV system insertion," *Electric Power Systems Research*, vol. 242, May 2025, doi: 10.1016/j.epsr.2025.111426.

[28] L. Feierl, V. Unterberger, C. Rossi, B. Gerardts, and M. Gaetani, "Fault detective: Automatic fault-detection for solar thermal systems based on artificial intelligence," *Solar Energy Advances*, vol. 3, Jan. 2023, doi: 10.1016/j.seja.2023.100033.

[29] M. R. Shoaib, H. M. Emara, J. Zhao, M. T. Ahvanooey, and E. Nabil, "Advanced deep learning approaches for fault detection in solar PV systems: A comparative study of SPDA and AIFD-SolDL," *Appl Soft Comput*, vol. 184, Dec. 2025, doi: 10.1016/j.asoc.2025.113592.

[30] Y. özüpak, "Real-time detection of photovoltaic module faults using a hybrid machine learning model," *Solar Energy*, vol. 302, p. 114014, Dec. 2025, doi: 10.1016/j.solener.2025.114014.

LIST OF PUBLICATIONS AND PAPERS PRESENTED

Published works as well as papers presented at conferences, seminars, symposiums etc pertaining to the research topic of the research report/ dissertation/ thesis are suggested be included in this section. The first page of the article may also be appended as reference.

APPENDICES

(Please delete this part): Appendices consist of additional illustration of data sources, raw data and quoted citations which are too long to be placed in the text. The appendix supports the written text of the research report/dissertation/thesis. Research instruments such as questionnaires, maps or computer programmes are parts of appendix too.

Appendices can be divided into Appendix A, B, C.

This page is optional; if you do not have any appendices, delete the entire page.