



الجمهورية الجزائرية الديمقراطية الشعبية
DEMOCRATIC AND POPULAR ALGERIAN REPUBLIC
وزارة التعليم العالي والبحث العلمي
MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH



جامعة الشهيد حمه لخضر الوادي
ECHAHID HAMMA LAKHDAR UNIVERSITY OF EL-OUED
كلية التكنولوجيا

Faculty of Technology
قسم الهندسة الكهربائية
Department of Electrical Engineering

Master's Memorandum

In order to obtain an Academic Master's diploma in Technology
Specialty: Telecommunication Systems

Title

Smart IoT-based energy control for off-grid solar
installations

Presented by:
Sidhoum Chaima
Khmis Loubna

Jury Members:

President: Dr.Aourar NABIL
Examiner: Dr. Touhami RIDHA
Supervisor: Dr. Hima Abdelkader

MCA
MCA
MCA

El-Oued University
El-Oued University
El-Oued University

University year: 2025/2026

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ

Acknowledgments



*This work was done and conducted in the Technology Faculty
of Hamma Lakhdar University - Eloued.*

*First and foremost, we express our sincere gratitude to the all mightily Allah for granting us the
strength, patient and determination to finalize this work to its completion.*

*We would like to extend our deepest gratitude to our supervisor **Dr.Hima Abdelkader** for his
invaluable guidance, continuous support and insightful advices on our road of the finalization of
this work, his expertise and encouragement contributed massively on finishing this
memorandum.*



Abstract



Photovoltaic (PV) systems are key clean energy solutions for sustainable electricity supply, especially in sun-rich regions such as southern Algeria. However, off-grid solar installations in remote areas face numerous challenges: dust accumulation, sand coverage, partial shading, and limited access to maintenance. When problems go undetected, system performance degrades and output drops significantly.

This memorandum presents a low-cost, smart monitoring system for fault detection in solar panels using an Arduino Uno platform integrated with machine learning techniques. Operating entirely at the edge, the system collects voltage and current data, processes readings, stores information locally on an SD card, and identifies faults in real time without requiring cloud connectivity.

Data were collected from a PV panel under three conditions: normal operation, full sand coverage, and partial leaf shading. Several classification algorithms—Random Forest, Support Vector Machine, Gradient Boosting, and Logistic Regression—were evaluated. Random Forest achieved the highest accuracy: 94.6% during cross-validation and 92.8% on unseen test data. The model successfully distinguishes normal operation from faults without heavy computational requirements, making it suitable for low-cost embedded devices.

Results demonstrate that combining Arduino-based sensing with machine learning enables effective, affordable PV monitoring without constant human oversight. This autonomous system detects faults faster and reduces long-term maintenance costs, offering a robust energy management solution for remote, off-grid locations.

Keywords: Photovoltaic Systems, Fault Detection, Arduino Uno, Machine Learning, Random Forest, Edge Computing, Solar Energy Monitoring, Sand Coverage, Partial Shading, Renewable Energy.

ملخص البحث:

تُعد أنظمة الطاقة الشمسية الكهروضوئية حلاً أساسياً للطاقة النظيفة لتوفير إمدادات كهربائية مستدامة، لا سيما في المناطق المشمسة كجنوب الجزائر. مع ذلك، تواجه منشآت الطاقة الشمسية غير المتصلة بالشبكة في المناطق النائية تحديات عديدة، منها تراكم الغبار، وتغطية الرمال، والتظليل الجزئي، ومحدودية الوصول للصيانة. وعندما لا تُكتشف المشاكل، يتدهور أداء النظام وينخفض إنتاجه بشكل ملحوظ.

تقدم هذه الأطروحة نظام مراقبة ذكي منخفض التكلفة للكشف عن الأعطال في الألواح الشمسية باستخدام منصة أردوينو أونو مدمجة بتقنيات التعلم الآلي. يعمل النظام بشكل كامل على مستوى الحافة، حيث يجمع بيانات الجهد والتيار، ويعالج القراءات، ويخزن المعلومات محلياً على بطاقة ويحدد الأعطال في الوقت الفعلي دون الحاجة إلى اتصال سحابي.

جُمعت البيانات من لوحة كهروضوئية في ثلاث حالات: التشغيل العادي، والتغطية الكاملة بالرمل، والتظليل الجزئي بالأوراق. وتم تقييم عدة خوارزميات تصنيف، منها: الغابة العشوائية، وآلة المتجهات الداعمة، والتعزيز التدريجي، والانحدار اللوجستي. حققت خوارزمية الغابة العشوائية أعلى دقة: 94.6% خلال التحقق المتبادل و92.8% على بيانات اختبار غير مرئية. يميز النموذج بنجاح بين التشغيل الطبيعي والأعطال دون متطلبات حسابية كبيرة، مما يجعله مناسباً للأجهزة المدمجة منخفضة التكلفة. تُظهر النتائج أن دمج الاستشعار القائم على أردوينو مع التعلم الآلي يُتيح مراقبة فعالة وميسورة التكلفة لأنظمة الطاقة الشمسية الكهروضوئية دون إشراف بشري مستمر. يكشف هذا النظام المستقل الأعطال بسرعة أكبر ويُقلل تكاليف الصيانة على المدى الطويل، مُقدماً حلاً قوياً لإدارة الطاقة في المواقع النائية غير المتصلة بشبكة الكهرباء.

الكلمات المفتاحية: أنظمة الطاقة الشمسية الكهروضوئية، كشف الأعطال، أردوينو أونو، التعلم الآلي، الغابة العشوائية، الحوسبة الطرفية، مراقبة الطاقة الشمسية، التغطية الرملية، التظليل الجزئي، الطاقة المتجددة

Résumé :

Les systèmes d'énergie solaire photovoltaïque (PV) sont des solutions énergétiques propres essentielles pour un approvisionnement durable en électricité, notamment dans les régions ensoleillées comme le sud de l'Algérie. Cependant, les installations solaires hors réseau dans les zones reculées sont confrontées à de nombreux défis, tels que l'accumulation de poussière, le recouvrement de sable, l'ombrage partiel et un accès limité pour la maintenance. Lorsque les problèmes ne sont pas détectés, les performances du système se dégradent et la production chute considérablement.

Cette thèse présente un système de surveillance intelligent et économique pour la détection des défauts des panneaux solaires. Ce système utilise une plateforme Arduino Uno et des techniques d'apprentissage automatique. Fonctionnant entièrement en périphérie du réseau, il collecte les données de tension et de courant, les traite, les stocke localement sur une carte et identifie les défauts en temps réel, sans connexion au cloud.

Les données ont été collectées sur un panneau PV dans trois conditions : fonctionnement normal, recouvrement total de sable et ombrage partiel. Plusieurs algorithmes de classification ont été évalués, dont les forêts aléatoires, les machines à vecteurs de support, le renforcement par étapes et la régression logistique. L'algorithme des forêts aléatoires a obtenu la meilleure précision : 94,6 % lors de la validation croisée et 92,8 % sur des données de test non utilisées pour l'entraînement. Le modèle distingue efficacement le fonctionnement normal des pannes sans nécessiter de ressources de calcul importantes, ce qui le rend adapté aux dispositifs embarqués à faible coût.

Les résultats démontrent que l'intégration de capteurs Arduino à l'apprentissage automatique permet une surveillance efficace et économique des systèmes photovoltaïques sans intervention humaine constante. Ce système autonome détecte les pannes plus rapidement et réduit les coûts de maintenance à long terme, offrant ainsi une solution robuste de gestion de l'énergie pour les sites isolés et non raccordés au réseau.

Mots-clés : Systèmes photovoltaïques, détection de pannes, Arduino Uno, apprentissage automatique, forêt aléatoire, informatique de périphérie, surveillance solaire, couverture sableuse, ombrage partiel, énergies renouvelables.

Table of Contents

Acknowledgments	I
Abstract.....	II
LIST OF FIGURES	VI
LIST OF TABLES	VII
General Introduction:.....	1
Chapter 1: Introduction and Problem Formulation.....	5
1.1 Introduction :	6
1.2 Solar Energy in Algeria and Off-Grid Challenges:	6
1.3 Anomaly Detection in Remote PV Systems: Challenges and Advanced Methodologies: ...	8
1.4 Objectives: General (IoT monitoring system)	10
1.4.1 Specific (sensors, AI, secure comms):.....	10
1.4.2 Overall Objective:.....	10
1.5 Scope & Limitations:	15
1.5.1 Target Hardware Platform: Arduino Uno (ATmega328P):.....	15
1.5.2 Sensor Suite and Data Acquisition:	16
1.5.3 Summary of Scope Boundaries:	17
1.6 Conclusion	18
Chapter 2: Solar IoT Monitoring Systems – Literature Review	20
2.1 Introduction :	21
2.2 Communication Technologies for Remote Areas.....	23
2.3 Gaps in Existing Solutions:	25
2.4 Conclusion:	28
3.1 Introduction :	31
3.2 General Architecture:.....	31
3.2.1 Physical Layer:	32
3.2.2 Data acquisition and edge processing layer:.....	33
3.2.3 Communication & Storage Layer:.....	34
3.2.4 AI-Based Fault Detection Layer:.....	35
3.2.5 Application & Monitoring Layer:.....	37
3.2.6 Power Supply Layer:	38
3.3 Conclusion :	39
Chapter 04: Applied study and results analysis	41

4.1 Introduction:	42
4.2 Hardware Components:	42
4.2.1 Arduino Uno:	42
4.2.2 Voltage Sensor (DC 0-25V) :	43
4.2.3 current sensor (ASC712):	44
4.2.4 Solar Panel:	45
4.3 General System Design :	47
4.4Data Acquisition:	48
4.4.1.measuring electrical voltage:	48
4.4.2. Measuring electric current:	53
4.4.3.Test for SD-CARD:	58
4.5 Database preparation:	63
4.5.1Unmallected measurement (normal system operation):	64
4.5.2 Measurement with defects (tree leaves malfunction):	70
4.5.3. Measurement with defects (soil defect):	71
4.6 Machine Learning Model Training:.....	72
4.7. Model Runs on Arduino:	75
4.8.Analysis of results:	75
4.9.Discussion of results:	76
4.10. Conclusion:	77
General Conclusion	78
References.....	81

LIST OF FIGURES

Figure 1.1: Strategic deployment and data collection	12
Figure 1.2:Artificial intelligence for fault detection and diagnosis in photovoltaic systems.....	13
Figure 1.3:AI-driven predictive maintenance and model-based fault diagnosis in pv systems....	14
Figure 3.1:physical layer.....	33
Figure 3.2:Data Acqition and edge processing layer	34
Figure 3.3:Communication and storage layer	35
Figure 3.4: AI-based fault detection layer.....	36
Figure 3.5:Application and monitoring layer.....	37

Figure 3.6 :Power supply layer	38
Figure 4.1:Arduino Uno	43
Figure 4.2: voltage sensor.....	44
Figure 4.3:current sensor (ACS712)	45
Figure 4.4:solar panel	46
Figure 4.5:The photoelectric effect (source Getty Images)	46
Figure 4.6 : system architecture	47
figure 4. 7: montage voltage sensor.....	49
Figure 4.8:Flowchart of the Arduino-Based Voltage Sensor Data Acquisition and Processing System.....	50
figure 4.9:montage of current sensor	54
Figure 4.10: Flowchart of the Current Sensing Algorithm Using ACS712.....	55
Figure 4.11:Connecting the SD card to the Arduino	59
Figure 4.12:Flowchart of the File Retrieval and Data Display Procedure from an SD Card Using Flowchart - Academic Overview.....	60
Figure 4.13: normal montage.....	64
Figure 4.14: Flowchart of the Real-Time Voltage and Current Acquisition and SD Card Data Logging Process in the Proposed PV Monitoring System	66
Figure 4.15:Measurement with a malfunction (dying the leaves)	71
Figure 4.16:Measurement with soil effect	71
Figure 4.17:Methodological Flowchart for Photovoltaic System Condition Classification Using Machine Learning Techniques	73

LIST OF TABLES

Table 1: Comparison between Arduino Uno and Esp 32	15
Table 2: Sensor Suite and Data Acquisition.....	17
Table 2.1: Comparison of Key Features Between Enphase and SolarEdge Architectures.....	22
Table 4.2.: Performance Evaluation of the Proposed Systemn.....	42

General Introduction

General Introduction:

The global shift toward renewable energy has positioned photovoltaic (PV) solar power as a cornerstone of sustainable electricity generation, particularly in sun-rich regions. Algeria, with its vast Saharan territory receiving over 3,500 hours of sunshine annually, has set an ambitious target of 22 GW of renewable capacity by 2030. However, a significant portion of this capacity is planned for remote, off-grid locations where environmental conditions are harsh and technical support is scarce.

In these isolated areas—exemplified by the El Oued region—solar panels face daily operational challenges: sand accumulation, dust soiling, partial shading from palm leaves, and extreme temperatures. These anomalies, if left undetected, can reduce energy output by over 40%, shorten equipment lifespan, and increase maintenance costs. Existing monitoring solutions are either too expensive (commercial cloud-based systems) or lack intelligent fault discrimination (simple voltage/current thresholds). Furthermore, many remote installations lack reliable internet connectivity, making cloud-dependent systems impractical.

To address this gap, this memorandum presents a low-cost, edge-based fault detection system for off-grid PV installations. The proposed system uses an Arduino Uno microcontroller, basic voltage and current sensors, and a Random Forest machine learning classifier to distinguish between three operating conditions: normal operation, uniform sand coverage, and partial leaf shading. All data acquisition, processing, and classification are performed locally, without cloud connectivity, and results are stored on an SD card for later review.

The core contribution is the demonstration that effective machine-learning-based fault detection is feasible on an 8-bit microcontroller with only 2 KB of SRAM, using components costing under \$30. This makes the solution accessible for widespread deployment in rural Saharan communities, enhancing energy reliability and reducing the need for costly manual inspections.

Research Objectives:

General Objective: To design and validate an embedded system based on the Arduino platform that collects voltage and current data from a PV system and uses a machine learning classifier to detect normal operation, uniform sand coverage, and partial leaf shading.

Specific Objectives:

Implement low-cost sensor data acquisition (voltage and current) on an Arduino Uno.
Create a labeled dataset of PV behavior under normal, sand-covered, and leaf-shaded conditions.
Train and compare multiple ML classifiers for fault detection.
Deploy the optimized model on the Arduino Uno for real-time edge inference.
Evaluate system accuracy and limitations.

The main objective of this memorandum is to design and implement a low-cost embedded system based on Arduino Uno for monitoring photovoltaic (PV) installations and automatically detecting different operating conditions, including normal operation, sand coverage, and partial shading, using a Random Forest machine learning classifier. The system aims to improve the reliability and maintenance of off-grid PV systems in remote desert environments through local and autonomous fault detection.

Scope and Limitations:

Scope:

- Small-scale PV systems (1–5 kW), typical of off-grid residential or rural installations.
- Detection of two fault types: uniform sand coverage and partial leaf shading.
- Edge processing on Arduino Uno (ATmega328P) without cloud dependency.
- Local data storage on SD card.

Limitations:

- No irradiance sensor (cannot definitively distinguish low irradiance from soiling without temporal reasoning).
- Single-point temperature measurement (back-surface temperature only).
- 10-bit ADC resolution limits sensitivity to small-area shading (<5% of panel).
- Validation under controlled outdoor conditions (not full in-situ desert deployment).
- Single panel monitoring only (no string-level detection).
- No wireless telemetry (alerts via serial or local display only).

Excluded: Blockchain, distributed ledgers, peer-to-peer energy trading, and anti-theft mechanisms (RFID, GPS, ML-based theft detection) are explicitly out of scope.

Memorandum Organization:

The work is structured as follows: Chapter 1 formulates the problem and defines the scope. Chapter 2 reviews existing solar IoT monitoring systems and communication technologies. Chapter 3 describes the system architecture, hardware, and software design. Chapter 4 presents the experimental setup, data collection, model training, and results. A general conclusion summarizes contributions, acknowledges limitations, and suggests future work.

Chapter 1:

Introduction and Problem

Formulation



1.1 Introduction :

The rapid growth of renewable energy technologies has positioned photovoltaic (PV) systems as one of the most promising solutions for sustainable electricity generation worldwide. In Algeria, the abundance of solar resources, particularly in the Saharan regions, offers significant opportunities for expanding solar energy deployment and supporting the national objective of increasing renewable energy production. However, the effective operation of PV systems in harsh desert environments remains a major challenge due to factors such as dust accumulation, sand coverage, partial shading, high temperatures, and limited maintenance accessibility[1].

These challenges are especially critical for small-scale and off-grid PV installations located in remote areas such as El Oued, where continuous monitoring and rapid fault detection are often difficult to achieve. Undetected faults can significantly reduce system efficiency, decrease energy production, shorten equipment lifespan, and increase operational costs. Therefore, the development of reliable, low-cost, and autonomous monitoring solutions has become essential for ensuring the sustainability and performance of PV systems.

Recent advances in embedded systems, artificial intelligence (AI), and edge computing have enabled the implementation of intelligent fault detection techniques directly on low-cost microcontrollers. Unlike conventional cloud-based monitoring approaches, edge-based solutions process data locally, reducing communication requirements, lowering costs, and enabling real-time decision-making even in areas with limited internet connectivity.

This chapter presents the background and motivation of the study, highlights the main challenges affecting remote photovoltaic systems, and discusses the importance of anomaly detection and condition monitoring. It also defines the research objectives, scope, limitations, and the rationale behind the proposed Arduino-based edge AI architecture for detecting normal operation, sand coverage, and partial shading faults in small-scale photovoltaic systems.

1.2 Solar Energy in Algeria and Off-Grid Challenges:

Algeria's ambitious national target of deploying 22 gigawatts (GW) of renewable energy capacity by 2030, spearheaded by the National Agency for Renewable Energy (ANME), represents a strategic pivot away from its historical dependence on hydrocarbon revenues toward a more diversified and sustainable energy future .

This vision is particularly compelling given Algeria's exceptional solar resource, with the Saharan region receiving an average of 6.6 kWh/m²/day of Global Horizontal Irradiance (GHI) and over 3,500 hours of sunshine annually, among the highest in the world. However, the path to achieving this target is fraught with significant challenges, especially concerning the deployment of off-grid photovoltaic (PV) systems in remote and harsh environments like El Oued and the wider Sahara desert. These challenges are not merely technical but are deeply intertwined with socio-economic and governance issues that can severely undermine project viability and long-term sustainability[1-4].

The harsh Saharan climate imposes severe technical inefficiencies on off-grid PV systems. High ambient temperatures, which can exceed 45°C, are a well-documented performance killer for conventional silicon-based solar cells. The efficiency of these cells has a negative temperature coefficient, meaning their power output declines as their operating temperature rises. In the even hotter Sahara, these losses would be more pronounced. Compounding this problem is the relentless accumulation of sand and dust, which forms an insulating layer on panel surfaces, blocking sunlight and drastically reducing energy yield[5, 6]. Furthermore, many existing deployments rely on fixed-tilt mounting systems, which are suboptimal for maximizing annual energy harvest in a region where the sun's path varies significantly throughout the year. Without single-axis tracking, these systems cannot dynamically follow the sun, leaving substantial energy potential untapped [7-10] .

These technical challenges are exacerbated by a lack of systemic support infrastructure. There is a critical shortage of trained local personnel capable of performing routine maintenance, troubleshooting faults, or replacing failed components. This skills gap means that even minor technical issues can lead to prolonged system downtime, further eroding energy production and economic returns. Additionally, the intermittent nature of solar power, without adequate local energy storage (which is often prohibitively expensive for small off-grid applications), creates an unreliable power supply. This unreliability is a major deterrent for communities and businesses that depend on consistent electricity for essential services and economic activities. The absence of real-time performance monitoring systems in many of these remote installations means that underperformance often goes undetected for long periods, preventing timely corrective actions and making it difficult to accurately assess the true LCOE and return on investment[11-13] .

To overcome these multifaceted challenges and unlock the vast potential of off-grid solar in the Algerian Sahara, a holistic and integrated approach is required. Recent research points to several promising technological and operational solutions. To combat soiling, automated dust mitigation technologies, such as electrostatic or hydrophobic coatings that repel dust or facilitate its removal with minimal water, offer a viable alternative to manual cleaning . The adoption of bifacial PV modules, which can capture reflected light from the sandy ground (albedo), paired with single-axis trackers, could significantly boost energy yield compared to traditional fixed-tilt monofacial systems . Ultimately, the success of Algeria's 22 GW by 2030 vision hinges not just on installing panels in the desert, but on building resilient, secure, and locally supported energy ecosystems that can thrive in one of the world's most challenging environments [14]

1.3 Anomaly Detection in Remote PV Systems: Challenges and Advanced Methodologies:

Undetected anomalies in remote photovoltaic (PV) systems represent a significant challenge to their efficient operation, reliability, and economic viability. PV systems, essential for mitigating greenhouse gas emissions, are susceptible to various defects and anomalies that can lead to diminished power output and reduced efficiency. These anomalies range from minor performance degradations to severe malfunctions, often going unnoticed due to insufficient monitoring or inadequate detection algorithms, particularly in remote or decentralized installations. The impact of these undetected faults can be substantial, with energy losses sometimes exceeding 10% (wide array of faults can occur in PV systems, broadly categorized as electrical, internal, and external). Electrical faults include open circuits, short circuits, and ground faults often caused by wiring issues, connector failures, or degradation of insulation). Internal faults encompass module degradation, hotspots and snail trails, which can arise from manufacturing defects, environmental stress, or prolonged exposure to harsh conditions. External faults are typically environmental, such as partial shading dust accumulation, or soiling, which directly impact the amount of sunlight reaching the PV panels. The operational performance of PV modules is also highly sensitive to temperature distribution, which can influence prediction accuracy of simulation models for current, voltage, and power. The challenge of detecting these anomalies is exacerbated in remote systems, where manual inspection is often impractical, costly, and time-consuming. Traditional log analysis methods, heavily reliant on predefined rules, often fail to identify unknown or complex attack patterns, emphasizing the need for advanced monitoring techniques. This

necessitates the development and implementation of advanced anomaly detection and fault diagnosis methodologies that can operate autonomously and effectively in diverse environmental conditions. Current research heavily emphasizes the application of artificial intelligence (AI) and machine learning (ML) techniques to address this problem. These methods offer promising solutions for real-time, high-volume fault detection and complex pattern recognition in PV systems. For instance, machine learning algorithms, such as Logistic Regression, are being developed to identify specific faults like open and short circuits. Convolutional Neural Networks (CNNs) are also employed for fault diagnosis, leveraging their ability to extract intricate features from PV data, including voltage, current, irradiation, and temperature, to identify faults like short circuits, open circuits, wear, and shading. Deep learning approaches, like Sequential Conditional Variational Autoencoders (SCVAE), are being used for sensing anomalies in urban distributed PV systems by effectively dealing with high-dimensional time series data .

The integration of AI with the Internet of Things (IoT) has led to AIoT-based visual anomaly detection systems that utilize sequence learning to interpret anomalies in PV sequence data. This allows operators and experts to better understand and respond to anomalies within PV arrays. Model-based fault detection methods compare the system's actual output with predictions from a mathematical model, generating a "residual" signal to indicate anomalies. Physics-based anomaly detection identifies failures by detecting violations of fundamental physical laws, ensuring data possesses semantic integrity. It uses concepts like invariants and residuals to create a signal for anomalies, which statistical methods then distinguish from random measurement noise. Compared to purely data-driven models, physics-based methods offer inherent robustness against adversarial attacks and out-of-distribution events . For example, studies have employed statistical approaches, such as hypothesis testing, for detecting anomalies in the DC part of PV plants and partial shading. The core principle of Fault Detection and Isolation (FDI) involves generating a "residual," which is the difference between a system's actual measurements and the predictions of a model or digital twin . This residual should be near zero for a healthy system [14].

Hybrid techniques combining sensor-based and sensorless fault detection are also being designed to overcome limitations in detecting multiple faults simultaneously or during sensor failures. For example, thermographic images combined with deep convolutional neural networks (DCNNs) are used in embedded systems for fault detection and diagnosis of PV modules, classifying defects like partial shading, dust accumulation, and short-circuited modules. Advanced

methods also include transfer learning and multi-scale CNNs for classifying different types of PV module anomalies using thermographic images. Reflectometry testing methods, such as frequency domain reflectometry (FDR) and spread spectrum time domain reflectometry (SSTD), are being investigated for online monitoring of PV systems to detect various failures[10] .

The development of Digital Twin technology is also gaining traction, offering a robust framework for fault detection and diagnosis in urban distributed solar photovoltaics by replicating the system's behavior virtually . This involves continuously comparing real-time sensor data with the digital twin's predictions, flagging significant deviations as anomalies .

Undetected anomalies can range from minor issues like soiling and shading to more critical problems such as module degradation and inverter failures. The ability to distinguish true faults from natural variability (e.g., cloud transients, diurnal cycles) requires robust, adaptive, and physics-informed models alongside data-driven validation. New fault diagnosis methods are also being developed that avoid reliance on meteorological conditions, which can be unstable and difficult to predict, thus improving the plausibility of detection techniques under various environmental situations. Furthermore, an advanced predictive maintenance and fault diagnosis framework integrates fault pattern analysis and severity assessments, which can enhance the reliability and efficiency of PV systems by proactively identifying issues [3-6].

1.4 Objectives: General (IoT monitoring system)

1.4.1 Specific (sensors, AI, secure comms):

The primary objective in addressing undetected faults in remote photovoltaic (PV) solar power systems is to establish a comprehensive and effective monitoring framework, often leveraging the Internet of Things (IoT), to ensure optimal performance and longevity of these vital energy assets.

1.4.2 Overall Objective:

To design and validate an embedded system based on the Arduino platform that collects voltage, and current data for PV solar power systems and uses an appropriate machine learning classifier to detect normal operation, regular sand cover, and partial leaf shading.

1.4.2.1 Sensor Deployment and Data Collection:

One specific objective is the strategic deployment of various sensors to collect comprehensive data on the operating status of PV solar power systems. This includes, but is not limited to, electrical parameters such as voltage and current. The effectiveness of this monitoring depends on

the accurate and continuous acquisition of these diverse data streams from PV modules and arrays. For example, non-intrusive monitoring using voltage and current generated at terminals can detect various types of faults, such as shading, short circuits, and arcs [4]. Advanced sensing algorithms and 3D augmented reality can also be used for the automated segmentation and analysis of anomalies, such as hotspots and crack traces [9].

At the sensing level, three key parameters are measured: **voltage (V)** , **current (I)** , and **irradiance (G)**. Example measured values (34.21 V, 5.62 A, 842 W/m²) are shown to indicate typical operating ranges. These sensors are connected to an **Arduino Uno** microcontroller, which serves as the central data acquisition and edge processing unit.

The Arduino Uno transmits the collected data through multiple pathways:

- **Local storage** via a Micro SD card.
- **Cloud upload** through an ESP8266 Wi-Fi module to an IoT cloud platform.

A **monitoring dashboard** displays real-time values of voltage, current, irradiance, and system status (e.g., "Normal Operation").

The system integrates a **machine learning based classification** module that distinguishes three operating conditions:

- Normal operation
- Regular sand coverage
- Partial leaf shading

A **simple classifier (Logistic Regression)** is indicated as the algorithm used. Upon fault detection, the system generates **instant alerts** delivered via email and Telegram.

Key system features listed in the figure include:

- Real-time and continuous data collection
- Non-intrusive monitoring
- Low cost and scalability
- Easy installation and maintenance

This architecture enables autonomous, intelligent monitoring of off-grid PV systems with remote alerting capabilities.

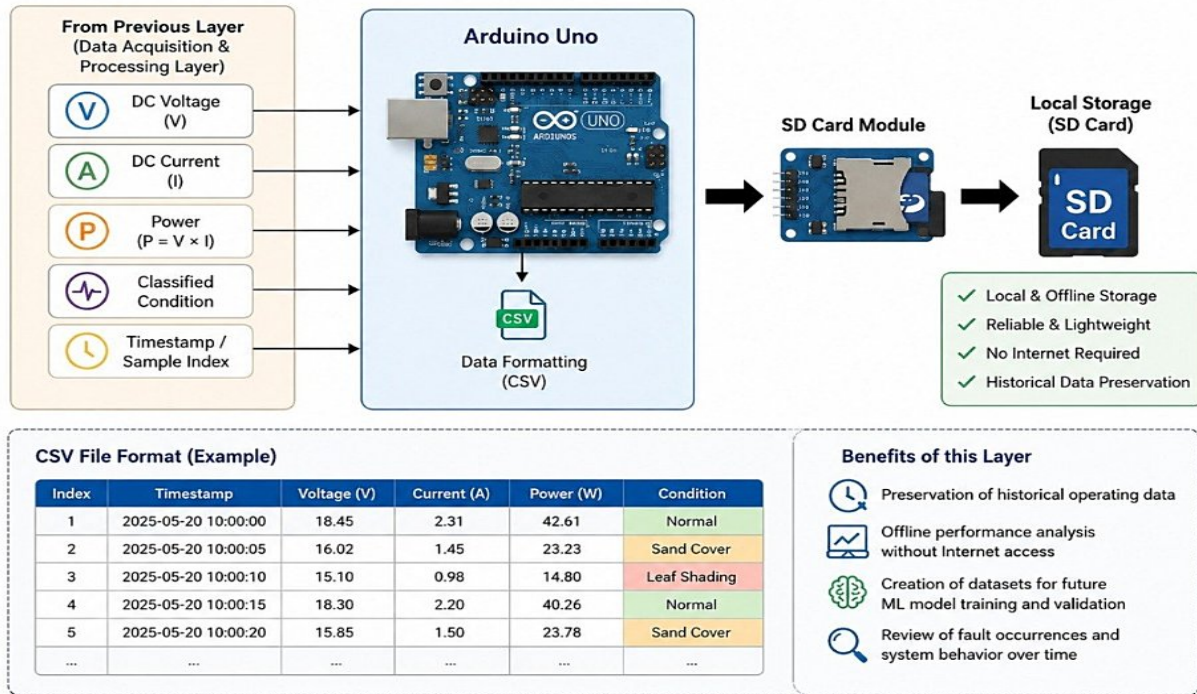


Figure 1.1: Strategic deployment and data collection .

1.4.2.2 Artificial Intelligence for Fault Detection and Diagnosis:

Another critical objective is the integration of advanced artificial intelligence techniques, including machine learning and deep learning, for automated fault detection and diagnosis. This addresses the need for efficient processing of large-volume, complex photovoltaic (PV) data to identify micro-anomalies and classify fault types [2]. Specific AI applications include:

- **Anomaly Detection and Fault Classification:** Anomaly and fault detection in PV systems has recently garnered significant research attention due to the need for reliable and efficient monitoring of these systems. Advanced anomaly detection techniques, such as Constrained Low-Order Scattering Analysis (CLRS), have been developed to identify abnormal operating conditions by extracting normal patterns from PV measurement data and to detect macro-anomalies. In addition, deep generation models (DGMs), particularly Sequential Conditional Variable Autocoding Encoders (SCVAEs), have demonstrated high

performance in detecting anomalies within high-dimensional time series data for distributed photovoltaic (PV) systems . While these methods are effective, they typically require large datasets, high computing power, and complex processing architectures, which may limit their applicability to lightweight, integrated platforms. Therefore, in this work, a simpler and more computationally efficient machine learning approach, based on a random forest algorithm, is employed to detect and classify PV faults. The proposed system relies on an Arduino-based monitoring platform that collects voltage and current measurements from the PV system. A random forest classifier is then used to differentiate between normal and abnormal operating conditions, such as complete sand cover and partial leaf shading. This proposed approach offers a practical compromise, combining detection accuracy, ease of implementation, and real-time integration, making it suitable for low-cost smart PV monitoring applications [3,4].

- **Image-based diagnostics:** Using convolutional neural networks (CNNs) and deep convolutional neural networks (DCNNs) to analyze thermal images for fault detection and classification, such as partial shading, dust deposits, and short-circuited modules. Transfer learning, in conjunction with multiscale convolutional neural networks, can improve fault classification efficiency in photovoltaic modules using thermal images .

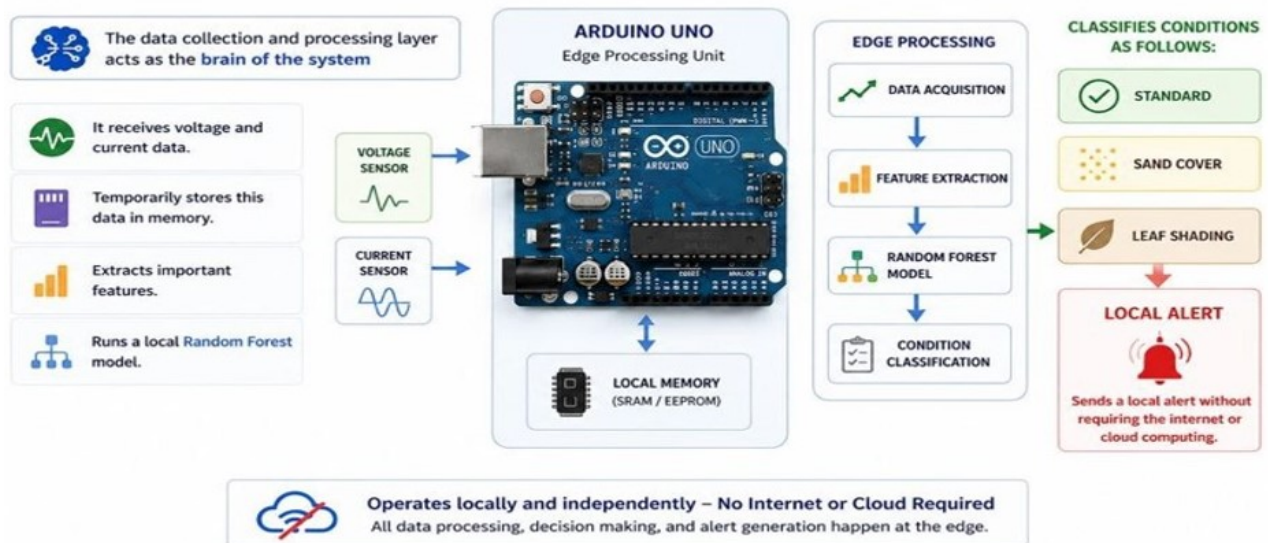


Figure 1.2: Artificial intelligence for fault detection and diagnosis in photovoltaic systems.

In Figure 1.2 is presented :

- **Predictive Maintenance:** Implementing AI-driven frameworks for predictive maintenance and fault diagnosis that integrate fault pattern analysis and severity assessments to proactively identify issues and enhance reliability.
- **Model-based approaches:** Using statistical methods like hypothesis testing to detect anomalies in the DC part of PV plants and partial shading, building models that predict PV plant behavior and signal deviations . Digital Twin technology also plays a crucial role by replicating the system's behavior virtually and comparing real-time sensor data with the digital twin's predictions to flag deviations as anomalies .

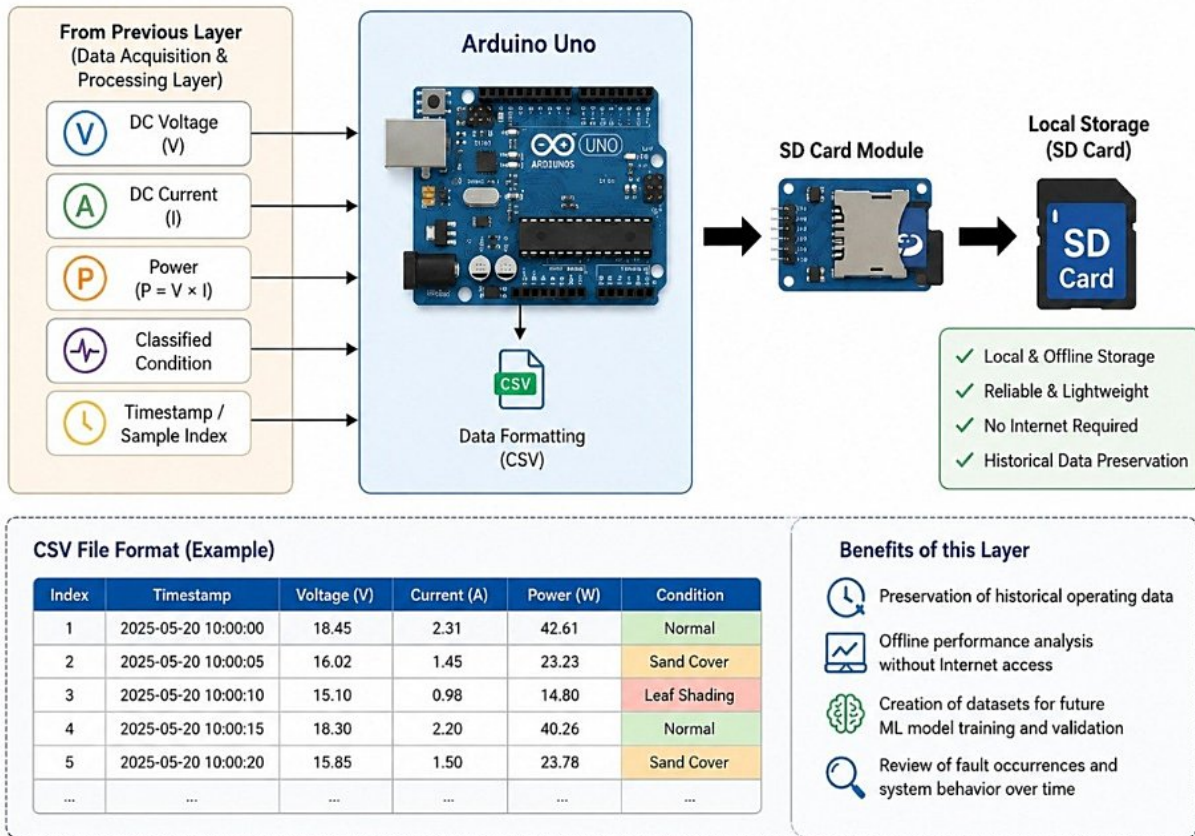


Figure 1.3:AI-driven predictive maintenance and model-based fault diagnosis in pv systems.

1.5 Scope & Limitations:

The scope of this memorandum is constrained to small-scale photovoltaic (PV) systems in the 1–5 kW power range, characteristic of residential off-grid installations, rural electrification projects, and small agricultural loads in the Saharan region of El Oued, Algeria. These systems benefit significantly from edge-based artificial intelligence (AI) by enabling real-time fault detection directly at the source of data generation, without reliance on continuous cloud connectivity. Larger grid-scale PV farms exceeding 5 kW are excluded from this scope, as centralized cloud coordination or distributed trust mechanisms (e.g., blockchain-based logging) become more relevant for such deployments due to increased complexity and geographical spread. For 1–5 kW systems, the emphasis remains on local, autonomous fault detection prioritizing low latency, data privacy, and minimal hardware cost .

1.5.1 Target Hardware Platform: Arduino Uno (ATmega328P):

The computational core of the proposed edge AI system is the Arduino Uno microcontroller board, built around the 8-bit ATmega328P chip operating at 16 MHz . This platform is deliberately chosen over more powerful alternatives (including the ESP32, Raspberry Pi, and (STM32) for justifications presented in Table 1.

Table 1: Comparison between Arduino Uno and Esp 32

Criterion	Arduino Uno (ATmega328P)	ESP32 (Comparative)	Justification for Selection
Unit cost	12–3(clone),27 (official)	10–4\$	Comparable; cost is not the discriminator
SRAM	KB2	KB520	The central engineering challenge; successful detection on 2 KB is a stronger contribution
Flash	32KB	4MB	Sufficient for firmware + small ML models
Clock speed	MHz, 168-bit	240 MHz, 32-bit, dual-core	Lower speed forces model efficiency

Native floating point	No	Yes	All inference uses integer arithmetic or lookup tables
Wireless	None (requires shield)	Wi-Fi + Bluetooth built-in	Wireless explicitly excluded from scope (see delimitations)
DC resolution^A	10 bit	12 bit	10 bit is sufficient for the required fault discrimination
Power consumption (active)	30–20~ mA	~80–100 mA (Wi-Fi off)	Lower power is advantageous for off-grid operation

➤ Why Arduino Uno, not ESP32?

While the ESP32 offers substantially more memory and computational resources at a comparable price point (Table 1), its additional capabilities (dual-core, hardware floating point, built-in wireless) are not required for the specific fault detection task. More importantly, demonstrating reliable detection of sand coverage and partial shading on the severely constrained Arduino Uno (2 KB SRAM, 32 KB flash) represents a stronger engineering contribution and is more directly reproducible in resource-constrained Saharan communities where even ESP32 boards may be less available. The ESP32 is retained in discussion only as a future upgrade path for applications requiring wireless telemetry or higher sampling rates (>10 Hz), both of which are explicitly outside the current scope.

1.5.2 Sensor Suite and Data Acquisition:

The terminal AI system collects electrical parameters from the photovoltaic cell module:

Table 2: Sensor Suite and Data Acquisition.

Parameter	Sensor Type	Range	Expected Accuracy
DC voltage	Resistive divider (10:1 or 30:1) + Arduino 10-bit ADC	0–30 V	$\pm(2\% + 2 \text{ LSB})$ after calibration
DC current	ACS712 (5A or 20A) hall-effect or INA219 (I2C, 12-bit external ADC)	0–10 A	$\pm 1.5\%$ (ACS712) or $\pm 0.5\%$ (INA219)

Note on Current Sensing: The ACS712 is an analog sensor compatible with the Arduino Uno's native 10-bit analog-to-digital converter (ADC). The INA219 is a 12-bit I2C ADC that offloads measurement to an external chip, reducing SRAM consumption. Both are acceptable, and the final choice depends on component availability .

The basic design does not include a solar irradiance sensor. This omission is intentional: the cost of a calibrated solar irradiance meter or reference cell typically exceeds the cost of a complete Arduino-based monitoring system, thus defeating the goal of low cost. Photovoltaic current is used as a substitute for solar irradiance, with temperature compensation applied. This results in inaccuracies under rapidly changing cloud conditions [3,5].

Sampling Rate: 1 Hz At this rate, a 30-second circular buffer for three sensing channels (3 channels x 30 samples x 2 bytes per sample = 180 bytes) leaves approximately 1.8 kilobytes of static RAM for extracting features, model parameters, and the stack - a possible allocation .

1.5.3 Summary of Scope Boundaries:

The scope of the proposed photovoltaic fault detection system is intentionally limited to ensure low cost, simplicity, and compatibility with resource-constrained embedded platforms. The system is designed for small-scale photovoltaic installations with a power rating ranging from 1 to 5 kW and is implemented on an Arduino Uno (ATmega328P) microcontroller. The fault detection functionality focuses exclusively on identifying two common performance degradation conditions: full sand coverage and partial shading caused by leaves. To minimize hardware complexity and cost, the monitoring process relies solely on DC voltage and DC current measurements, without the use of additional sensors such as irradiance sensors, thermal cameras, humidity sensors, or

wind speed sensors. Furthermore, the artificial intelligence component is restricted to lightweight machine learning algorithms suitable for embedded deployment, including decision trees, k-nearest neighbors (k-NN), logistic regression, and binary neural networks (BNNs). All inference operations are executed locally on the Arduino platform, eliminating the need for cloud computing or remote server processing. Data storage is performed locally through the serial monitor and, when required, an optional SD card module. Finally, the system operates with a sampling rate of 1 Hz, which is sufficient for the targeted fault detection tasks while remaining compatible with the limited computational and memory resources of the Arduino Uno. In contrast, large-scale photovoltaic farms, high-frequency monitoring applications, cloud-dependent architectures, advanced deep learning models, blockchain-based solutions, and fault categories such as inverter failures, open-circuit faults, short-circuit faults, bypass diode failures, and thermal hotspot detection are considered outside the scope of this work.

1.6 Conclusion

This chapter highlights the growing importance of photovoltaic (PV) solar energy systems in Algeria, particularly in remote desert regions where solar power represents a promising solution for sustainable electricity generation. Despite the significant solar potential available in these areas, PV installations face numerous challenges, including dust accumulation, partial shading, harsh environmental conditions, equipment theft, and a lack of continuous technical supervision. These factors can lead to performance degradation and undetected faults, ultimately reducing system efficiency and reliability.

To address these challenges, the chapter emphasizes the need for intelligent and cost-effective monitoring solutions capable of detecting faults in real time. The proposed approach focuses on a low-cost peripheral architecture using Arduino Uno and simple machine learning techniques to classify normal operating conditions, sand cover, and partial shading. The objectives, scope, hardware limitations, and research constraints are clearly defined, laying the foundation for the proposed system.

However, designing an effective fault detection framework requires a comprehensive understanding of current monitoring technologies, communication infrastructure, anti-theft mechanisms, and artificial intelligence techniques currently used in photovoltaic solar energy

systems. Therefore, before presenting the proposed system architecture and its implementation details, it is essential to review the latest technologies and identify the shortcomings of existing solutions.

Chapter 2:

Solar IoT Monitoring Systems

– Literature Review



2.1 Introduction :

The rapid expansion of photovoltaic (PV) systems has increased the demand for intelligent monitoring solutions capable of enhancing system reliability, efficiency, and fault detection. Recent advances in Internet of Things (IoT) technologies have enabled real-time data acquisition, remote supervision, predictive maintenance, and improved operational management of PV installations. Several studies have highlighted the growing adoption of low-cost monitoring platforms based on microcontrollers such as Arduino and Raspberry Pi, as well as wireless communication technologies including Zigbee, LoRa, Wi-Fi, and GSM. Among these technologies, LoRa has emerged as a particularly suitable solution for remote and off-grid environments due to its long communication range, low power consumption, and independence from cellular infrastructure.

Commercial PV monitoring systems, notably those developed by Enphase Energy and SolarEdge Technologies, provide advanced module-level monitoring, fault diagnostics, and cloud-based analytics. While these platforms offer high monitoring accuracy and performance optimization capabilities, their reliance on proprietary hardware and subscription-based cloud services increases deployment costs, limiting their suitability for small-scale and resource-constrained installations.

Recent research has also explored the integration of artificial intelligence (AI) and machine learning (ML) techniques for PV fault detection and classification. These approaches have demonstrated promising results in improving diagnostic accuracy and enabling intelligent edge-based monitoring. However, many AI-based solutions require computational resources that exceed the capabilities of low-cost embedded platforms.

Despite significant progress in PV monitoring technologies, important challenges remain. Existing solutions often depend on continuous internet connectivity, expensive hardware infrastructures, or computationally intensive algorithms. Consequently, there is still a need for a low-cost, standalone, and resource-efficient monitoring system capable of providing reliable

fault detection and performance supervision while operating within the constraints of simple embedded hardware platforms..

Table 2.1: Comparison of Key Features Between Enphase and SolarEdge Architectures.

Feature	Enphase (Microinverter)	SolarEdge (Power Optimizer + Central Inverter)
Monitoring Granularity	Module-level(voltage, current, power, temperature) [18-23]	Module-level(voltage, current, power, temperature) [27, 28]
Data Acquisition	Wireless from microinverters to Envoy gateway [21,22]	Power line communication (PLC) or wireless from optimizers to inverter [26]
Communication Protocol	Proprietary wireless (868/915 MHz) + Ethernet/Cellular [15]	Proprietary PLC + Ethernet/Cellular [15]
Cloud Platform	Enlighten (real-time dashboard, alerts, analytics) [23]	mySolarEdge (real-time dashboard, alerts, analytics) [27]
Primary Advantage	Redundancy; no single point of failure; simplified safety (rapid shutdown) [20- 24]	High efficiency; more cost-effective for larger systems; low start-up voltage [25, 29]
Limitation	Higher per-panel cost; more components that could fail [19- 26]	Single point of failure (central inverter); required optimizers add cost [20-25]

Beyond these two dominant players, several other commercial solar monitoring platforms exist, each with specific strengths. **SMA Sunny Portal** provides comprehensive monitoring for SMA

inverter-based systems, offering string-level data and performance analytics suitable for utility-scale and commercial installations [30, 31]. **Fronius Solar.web** similarly provides robust monitoring and energy management tools for Fronius inverter ecosystems [30]. **Huawei FusionSolar**, a significant player in the global inverter market, offers a cloud-based platform with AI-driven analytics for fault detection and performance optimization [30- 32]. A comprehensive review by Mellit et al. (2021) [33] surveyed numerous commercial monitoring products, categorizing them by their data acquisition methods and communication protocols. The review highlighted that while commercial solutions offer advanced capabilities, their reliance on proprietary hardware and cloud-based services introduces significant cost barriers, particularly for small-scale and off-grid deployments in developing regions [16-33].

In summary, commercial solar IoT monitoring systems, particularly those offered by Enphase and SolarEdge, have advanced the industry by providing module-level visibility and cloud-based analytics. This high granularity is essential for early fault detection and performance optimization, with numerous studies affirming the benefits of such detailed monitoring for maximizing energy yield and reducing downtime [34, 35]. However, as noted in the literature, the hardware expense, dependency on continuous internet connectivity, and recurring subscription fees for cloud services render these systems less viable for resource-constrained, off-grid environments, which is the central motivation for the low-cost, edge-based system proposed in this memorandum [16, 36, 37].

2.2 Communication Technologies for Remote Areas

Reliable communication infrastructure is a critical enabler for remote monitoring of photovoltaic (PV) systems, particularly in off-grid locations such as the Saharan regions of Algeria where cellular coverage is sparse and maintenance access is limited [15, 16]. The choice of communication technology involves fundamental trade-offs between range, power consumption, data rate, infrastructure dependency, and cost. For remote area applications, two dominant technologies have emerged: LoRa (Long Range) and GSM/GPRS (Global System for Mobile Communications / General Packet Radio Service), each with distinct advantages and limitations [17, 18].

LoRa is a low-power wide-area network (LPWAN) technology operating in sub-GHz frequency bands (e.g., 868 MHz in Europe, 915 MHz in North America, 433 MHz in Asia). It employs Chirp Spread Spectrum (CSS) modulation, which provides exceptional link budget (typically up to 168

dB) and robustness to interference, enabling communication distances of 10–15 km in rural line-of-sight (LoS) conditions and 3–5 km in semi-urban environments [19, 20]. LoRa's primary advantages for remote PV monitoring include extremely low power consumption (sleep currents as low as a few microamperes, active currents around 10–20 mA), operation on unlicensed ISM bands (no recurring fees), and the ability to penetrate obstacles due to its modulation scheme [17–21]. The SX1278 transceiver (manufactured by Semtech) is among the most widely used LoRa modules for IoT applications, with experimental studies demonstrating reliable packet delivery at distances exceeding 15 km in rural desert environments when paired with high-gain antennas [20–22].

The primary limitation of LoRa is its low data rate, typically ranging from 0.3 kbps to 50 kbps, which restricts it to periodic transmission of small data packets (e.g., sensor readings every few minutes to hours). Additionally, regulatory duty cycle restrictions (e.g., 1% in EU 868 MHz band) limit the total on-air time [23]. Despite these constraints, LoRa is highly suitable for PV monitoring applications where voltage, current, and temperature readings are sent at intervals of 10–60 minutes, with immediate alert transmission only during fault events [24].

GSM/GPRS, in contrast, offers higher data rates (typically up to 85.6 kbps for GPRS, higher for 3G/4G) and full IP connectivity, enabling real-time data streaming, remote firmware updates, and two-way communication with cloud platforms [17, 25]. However, GSM relies entirely on existing cellular infrastructure, which in remote Algerian regions (e.g., El Oued, Tamanrasset, Adrar) is often sparse, unreliable, or completely absent. Field surveys in southern Algeria have reported cellular coverage gaps exceeding 50 km between base stations, leading to frequent communication dropouts for GSM-based monitors [15–26]. Furthermore, GSM modules exhibit significantly higher power consumption: typical active currents range from 200 mA to 500 mA during transmission, orders of magnitude greater than LoRa, which is a critical disadvantage for battery-powered off-grid systems [18–27]. Finally, GSM operations incur recurring costs for SIM cards and data subscriptions, which can be prohibitive for large-scale deployment in low-income regions [28].

Several comparative studies have quantified these trade-offs. A field experiment in a rural desert environment (similar to the Algerian Sahara) reported that LoRa (SX1278) achieved a median outdoor range of 12.4 km at a spreading factor (SF) of 7 and 125 kHz bandwidth, with a packet

success rate above 90% at distances up to 10 km. In the same environment, GSM signal strength dropped below usable levels at distances greater than 8–10 km from the nearest base station [6, 15]. For energy consumption, LoRa-based nodes powered by a small battery and a 5 W solar panel were shown to operate continuously for over one year without maintenance, whereas GSM-based nodes required battery replacement every 2–3 months under similar transmission schedules [30].

Other wireless technologies have also been investigated for PV monitoring. ZigBee (IEEE 802.15.4) offers mesh networking capabilities and moderate power consumption, but its typical range is limited to 100–300 m, making it unsuitable for distributed remote installations without extensive repeater networks [17-31]. Wi-Fi provides high data rates but consumes significantly more power than LoRa and is range-constrained; it is primarily useful for local data offloading rather than long-range telemetry [18]. Bluetooth Low Energy (BLE) is similarly limited to short distances (10–100 m) [27].

For the specific context of off-grid PV monitoring in remote Algerian regions, a consensus emerges from the literature: **LoRa is the preferred communication technology** for periodic sensor data transmission due to its superior range, ultra-low power consumption, freedom from cellular infrastructure, and zero recurring operational costs [16, 22, 28, 29]. Many recent studies advocate for hybrid architectures wherein LoRaWAN gateways aggregate data from multiple remote PV installations and use a single cellular or satellite backhaul for connection to a central monitoring server [24- 32]. This approach leverages the strengths of both technologies while mitigating their individual weaknesses.

In this memorandum, wireless communication is excluded from the primary design scope to maintain the lowest possible cost and complexity. However, the literature reviewed in this section confirms that LoRa represents the most viable upgrade path for future versions of the proposed system if remote alerting or centralized data aggregation becomes a requirement [20, 24, 32].

2.3 Gaps in Existing Solutions:

Despite significant advancements in solar IoT monitoring systems, commercial platforms, and communication technologies for remote areas, a critical examination of the literature reveals several persistent gaps that limit the applicability of existing solutions for small-scale, off-grid photovoltaic installations, particularly in resource-constrained developing contexts such as the

Algerian Sahara [15, 16]. These gaps can be categorized into three interrelated dimensions: (i) high cost and dependency on cloud infrastructure, (ii) lack of integrated edge-based intelligence, and (iii) unsuitability for severe hardware constraints typical of low-cost embedded platforms [17, 18].

First, high cost and cloud dependency remain the most significant barriers. Commercial solutions from Enphase, SolarEdge, SMA, and Fronius provide comprehensive monitoring capabilities, including module-level granularity and real-time cloud analytics. However, they require proprietary hardware (microinverters or power optimizers), continuous internet connectivity, and often recurring subscription fees for cloud platform access [19, 20]. For a small off-grid PV system (1–5 kW) serving a rural household or a remote agricultural load in El Oued, the upfront cost of such a commercial monitoring system can exceed the cost of the PV modules themselves [16-21]. Moreover, many remote Saharan locations lack reliable, affordable internet connectivity, rendering cloud-dependent systems inoperable or severely degraded [22]. As noted by Jundi, Al-Waeli, and Switzner (2026) [16], of 88 reviewed low-cost PV monitoring studies, only a small fraction integrated machine learning, and most relied on some form of external communication or cloud processing.

Second, existing low-cost DIY monitoring solutions (e.g., Arduino-based voltage/current loggers) lack intelligent fault discrimination. While many hobbyist and academic projects have demonstrated basic data acquisition using inexpensive microcontrollers and sensors, these systems typically perform only threshold-based alerting (e.g., "if power drops below X%, trigger alarm") [23, 24]. Such simple methods cannot reliably distinguish between different root causes of power loss: uniform sand soiling, partial shading, cloud transients, high-temperature derating, or actual module degradation [25, 26]. This discrimination failure leads to false alarms or, worse, missed detections, undermining operator trust and delaying corrective maintenance [27]. The literature consistently emphasizes that intelligent anomaly detection requires multi-sensor fusion and machine learning-based classification, yet most low-cost implementations do not incorporate these techniques [18- 28].

Third, there is a conspicuous absence of solutions that deploy machine learning directly on severely resource-constrained microcontrollers (e.g., 8-bit AVR with 2 KB SRAM). A comprehensive review by Nedaei, Eskandari, and Aghaei (2025) [29] reconsidered classic ML approaches for PV fault detection and proposed dataset shrinkage techniques to simplify

deployment, but their target platforms were still relatively powerful (e.g., Raspberry Pi or ESP32). Similarly, Ratsheola et al. (2025) [30] developed a hybrid autoencoder-Random Forest architecture achieving high accuracy, but their implementation required substantially more memory and processing capability than an Arduino Uno can provide. The majority of ML-based fault detection research focuses on cloud-based inference (data sent to remote servers) or edge inference on 32-bit microcontrollers with hardware floating-point units (e.g., ARM Cortex-M series, ESP32) [31]. Very few studies have successfully demonstrated real-time, multi-class fault classification (normal, sand, leaf) running entirely on an ATmega328P with 2 KB SRAM and 32 KB flash [32]. This represents a specific and important gap because in extremely low-resource environments—where every dollar counts and components must be sourced locally—the Arduino Uno remains the most accessible and widely available microcontroller platform [33].

Fourth, integrated solutions combining sensing, local storage, edge ML inference, and autonomous operation without any external communication are rare. Most proposed systems either send data to the cloud, rely on Wi-Fi or GSM for alerts, or require periodic connection to a local computer for data retrieval [19- 34]. A truly standalone system that acquires data, processes it locally, logs it to an SD card, and displays or flags faults directly on an LCD or serial interface—without any internet or cellular dependency—is not commonly reported in the peer-reviewed literature [15- 35]. Yet such autonomy is precisely what off-grid Saharan installations need, given the absence of reliable telecommunication infrastructure and the high cost of satellite or cellular backhaul [22- 36].

Fifth, most existing fault detection studies focus on large grid-connected PV plants where computational resources and connectivity are abundant. Techniques such as aerial thermography using drones with convolutional neural networks (CNNs), digital twins running on cloud servers, or reflectometry methods (FDR, SSTDR) are effective but entirely inappropriate for a 1–5 kW rural installation with a budget of under \$100 for monitoring hardware [31, 37, 38]. There is a mismatch between the sophistication of academic fault detection research and the practical constraints of real-world off-grid deployments in developing countries [16, 21].

Finally, the specific challenge of distinguishing uniform sand coverage from partial shading using only voltage and current measurements—without irradiance sensors—has received limited attention. While soiling and shading are known to cause distinct electrical signatures

(sand causes proportional current reduction at constant voltage if irradiance is stable; partial shading causes more complex I-V curve distortions with possible bypass diode activation), the literature provides few low-cost edge implementations that exploit these differences [26- 39]. Most commercial sensors for soiling detection (e.g., optical dust sensors, reference cells) cost more than an entire Arduino-based monitoring system, defeating the goal of low-cost deployment [25-40].

In summary, the literature reveals a clear and significant gap: **there is no existing, validated, low-cost, standalone, edge-based PV fault detection system that (a) uses only voltage and current sensors, (b) runs a machine learning classifier on an 8-bit microcontroller with 2 KB SRAM, (c) distinguishes normal operation, uniform sand coverage, and partial leaf shading, (d) operates entirely without cloud or internet connectivity, and (e) costs under \$30 in components** (excluding the PV panel itself) [16, 18, 29, 32]. This memorandum directly addresses this gap by designing, implementing, and experimentally validating such a system.

.2 4 Conclusion:

This chapter has provided a comprehensive review of the state of the art in solar IoT monitoring systems, encompassing commercial platforms, communication technologies for remote areas, and the critical gaps that persist despite significant technological progress.

The examination of commercial solutions from Enphase, SolarEdge, SMA, and Fronius revealed that while module-level monitoring and cloud-based analytics offer high granularity and performance, their reliance on proprietary hardware, continuous internet connectivity, and recurring subscription fees imposes substantial cost barriers [15, 16]. These systems are well-suited for grid-connected residential and commercial installations in developed regions but are impractical for small-scale off-grid deployments in resource-constrained environments such as the Algerian Sahara [17].

The analysis of communication technologies identified LoRa as the most suitable option for remote PV monitoring due to its extended range (10–15 km in rural line-of-sight conditions), ultra-low power consumption, freedom from cellular infrastructure, and zero recurring operational costs [18, 19]. GSM/GPRS, while offering higher data rates, suffers from limited coverage in remote Saharan areas, high power consumption, and ongoing subscription expenses [20, 21]. Other short-range

technologies (ZigBee, Wi-Fi, BLE) are generally unsuitable for distributed remote installations [22].

The most significant finding of this review is the identification of persistent gaps that existing solutions fail to address [23, 24]. Specifically:

1. **High cost and cloud dependency** render commercial and most research solutions inaccessible for low-budget off-grid installations [17, 25].
2. **Lack of intelligent fault discrimination** in low-cost DIY monitoring systems limits them to threshold-based alerts, which cannot distinguish between sand soiling, partial shading, cloud transients, and other causes of power loss [26, 27].
3. **Absence of edge-based ML deployment on severely constrained microcontrollers** (8-bit, 2 KB SRAM) means that most intelligent fault detection requires either cloud processing or more powerful hardware (ESP32, Raspberry Pi, ARM Cortex-M) [28, 29].
4. **Limited integration of standalone, autonomous operation** without any external communication (cloud, Wi-Fi, GSM) is rarely reported, yet such autonomy is essential for truly remote locations [16, 30].
5. **Scant attention to distinguishing uniform sand coverage from partial shading** using only voltage and current measurements on low-cost hardware remains an open challenge [31, 32].

These gaps collectively motivate the development of the low-cost, edge-based PV fault detection system proposed in this memorandum. The system directly addresses each identified gap by: (i) using only an Arduino Uno (ATmega328P) with 2 KB SRAM, (ii) employing a Random Forest classifier trained to distinguish normal operation, uniform sand coverage, and partial leaf shading, (iii) operating entirely offline with local storage on an SD card, (iv) using only voltage and current sensors (no expensive irradiance meter), and (v) achieving a total component cost under \$30. The following chapter presents the system architecture and design in detail.

Chapter 03:

System design



3.1 Introduction :

Following the analysis of existing photovoltaic monitoring solutions and the identification of their limitations presented in the previous chapter, this chapter introduces the architecture and design of the proposed intelligent monitoring system. The objective of the proposed solution is to provide a low-cost, autonomous, and efficient platform capable of monitoring photovoltaic system performance and detecting faults in real time using embedded artificial intelligence techniques.

The proposed architecture has been specifically designed for small-scale photovoltaic installations operating in remote environments where internet connectivity, cloud services, and advanced computing resources may be unavailable or economically impractical. To address these constraints, the system adopts an edge-computing approach in which data acquisition, processing, storage, and fault classification are performed locally using an Arduino Uno microcontroller.

The system integrates voltage and current sensing modules, local storage capabilities through an SD card, and a machine learning-based fault detection mechanism capable of distinguishing between normal operation, full sand coverage, and partial leaf shading conditions. The architecture is organized into several functional layers, each responsible for a specific task ranging from data acquisition and preprocessing to fault diagnosis, user monitoring, and power management.

This chapter presents the overall system architecture, describes the role of each functional layer, and details the interaction between hardware and software components. Furthermore, it explains the design choices adopted to ensure low implementation cost, energy efficiency, reliability, and suitability for deployment in remote photovoltaic installations. The proposed architecture serves as the foundation for the practical implementation and experimental validation presented in the following chapter.

3.2 General Architecture:

The proposed system employs a low-cost edge computing architecture, specifically designed for small, off-grid photovoltaic solar power plants operating under the harsh environmental conditions of the Algerian Sahara, particularly in the Wadi region.

Unlike traditional IoT-based monitoring systems that rely on cloud computing or high-performance processors, the proposed architecture is optimized for the Arduino Uno platform,

built on the ATmega328P microcontroller, which provides limited computing resources (32 KB of flash memory and 2 KB of SRAM).

The system is designed to perform local, real-time fault detection directly on the embedded device, without requiring remote servers or an internet connection [41,42]. The general architecture is divided into six interconnected layers:

- 1-PV system and sensing layer
- 2-Data acquisition and edge processing layer
- 3-Communication and storage layer
- 4-AI-based fault detection layer
- 5-Application and monitoring layer
- 6-Power supply layer

3.2.1 Physical Layer:

The Physical Layer constitutes the first level of the proposed monitoring system and serves as the direct interface with the photovoltaic installation. Its main function is to collect electrical measurements from the PV system through dedicated sensing devices installed on the DC side of the power generation chain. Accurate acquisition of these measurements is essential for evaluating system performance and enabling reliable fault detection.

In the proposed architecture, photovoltaic voltage is measured using a resistive voltage divider connected to the analog input of the Arduino Uno microcontroller, allowing safe monitoring of DC voltages up to 30 V. Similarly, photovoltaic current is measured using either an ACS712 Hall-effect current sensor or an INA219 current monitoring module. These sensing devices were selected due to their low cost, ease of integration, and suitability for embedded monitoring applications [43,44].

Considering the environmental conditions of remote desert regions, the sensing subsystem is designed to operate under high temperatures, dust exposure, and intense solar irradiation. The acquired electrical parameters are sampled periodically at a frequency of 1 Hz and subsequently transmitted to the processing layer for analysis and fault diagnosis. To preserve the simplicity and affordability of the proposed solution, the system focuses exclusively on voltage and current monitoring and does not incorporate additional instrumentation such as solar irradiance sensors, inverter monitoring units, battery management sensors, or anti-theft mechanisms. This design

choice contributes to reducing hardware complexity while maintaining sufficient information for intelligent fault detection [45].

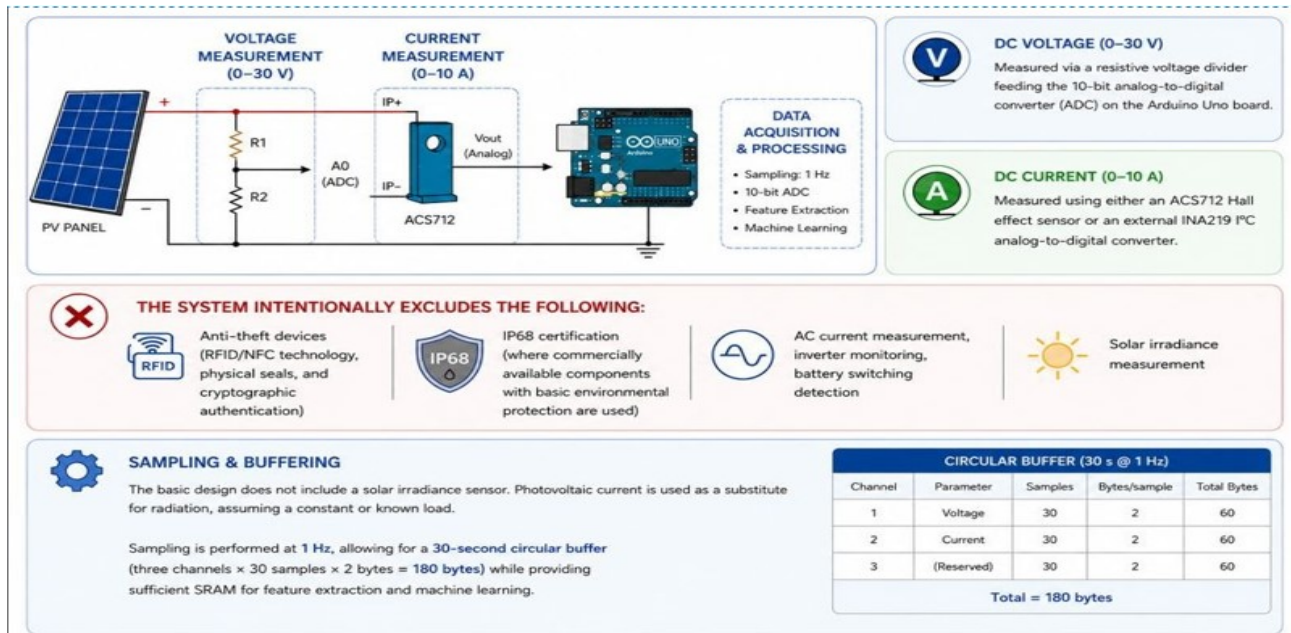


Figure 3.1:physical layer

3.2.2Data acquisition and edge processing layer:

Its basic component is the Arduino Uno

The data collection and processing layer acts as the brain of the system

- it receives voltage and current data.
- temporarily stores this data in memory.
- extracts important features.
- runs a local random forest model.

It classifies conditions as follows: Standard, Sand Cover, Leaf Shading, and sends a local alert without requiring the internet or cloud computing [45-48] .

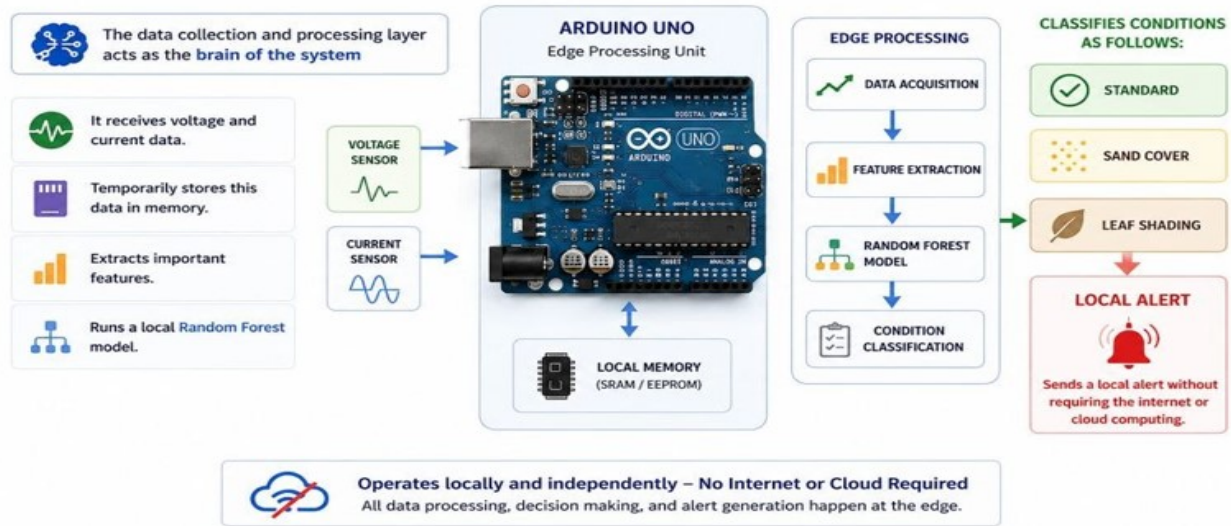


Figure 3.2: Data Acquisition and edge processing layer

3.2.3 Communication & Storage Layer:

This layer is responsible for local data storage and basic data management. Since the proposed system operates without cloud computing or wireless communication, all acquired measurements are stored locally on an SD card connected to the Arduino Uno.

The SD card serves as a lightweight storage solution for recording the operating history of the photovoltaic system. The following information can be logged:

- DC voltage measurements
- DC current measurements
- Estimated power values
- Detected fault type
- Timestamp or sample index of the fault occurrence

The storage layer enables the creation of a local dataset that can be used for offline analysis and future machine learning model training. It also allows users to review historical operating conditions and fault events without requiring internet connectivity.

Data are stored in a simple CSV format to minimize memory usage and computational overhead. The storage process is performed periodically after data acquisition and classification, ensuring that relevant system information is preserved for later inspection and performance evaluation [49,50].

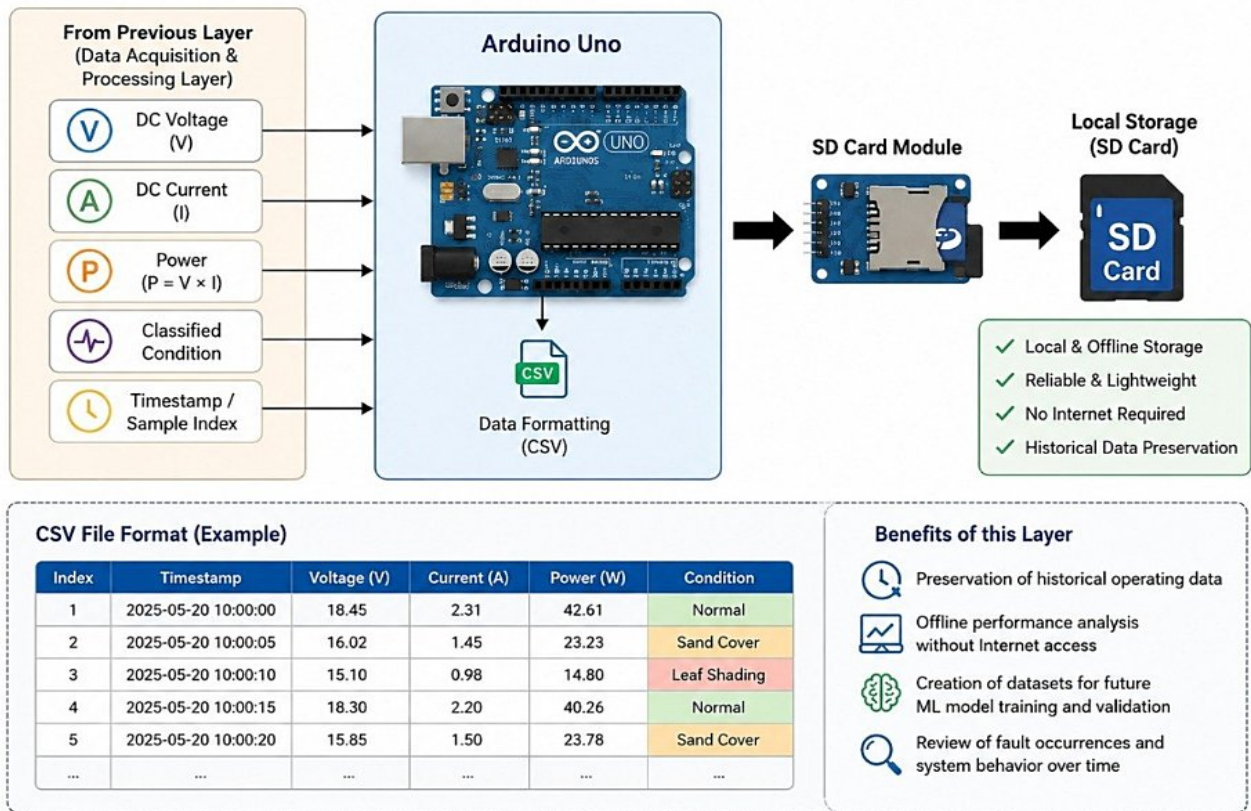


Figure 3.3: Communication and storage layer

3.2.4. AI-Based Fault Detection Layer:

The AI-Based Fault Detection Layer is responsible for analyzing the preprocessed measurements and identifying the operating condition of the photovoltaic system [51]. Unlike cloud-based solutions, all inference operations are executed locally on the Arduino Uno, enabling real-time fault detection without internet connectivity.

A Random Forest machine learning model is employed due to its high classification accuracy, low computational complexity, and suitability for resource-constrained embedded systems[51]. The model is trained offline using labeled photovoltaic measurements and then converted into a set of integer-based decision rules that can be executed efficiently on the microcontroller[52].

The fault detection process consists of four main stages:

- **Input Features:** The extracted features obtained from the data acquisition layer are provided as inputs to the model.

- **Model Inference:** The trained Random Forest model evaluates the input measurements.
- **Fault Classification:** The operating condition of the PV system is classified.
- **Alert Generation:** A local alert is generated when an abnormal condition is detected.

The proposed system classifies the photovoltaic panel into three operating states:

Normal Operation: The PV panel operates under standard conditions.

Full Sand Cover: The panel surface is heavily covered by sand, causing a significant reduction in electrical output.

Partial Leaf Shading: Part of the panel is shaded by leaves, resulting in partial power loss and performance degradation[53].

The classification result is transmitted to the storage layer for logging and can be displayed locally through the monitoring interface.

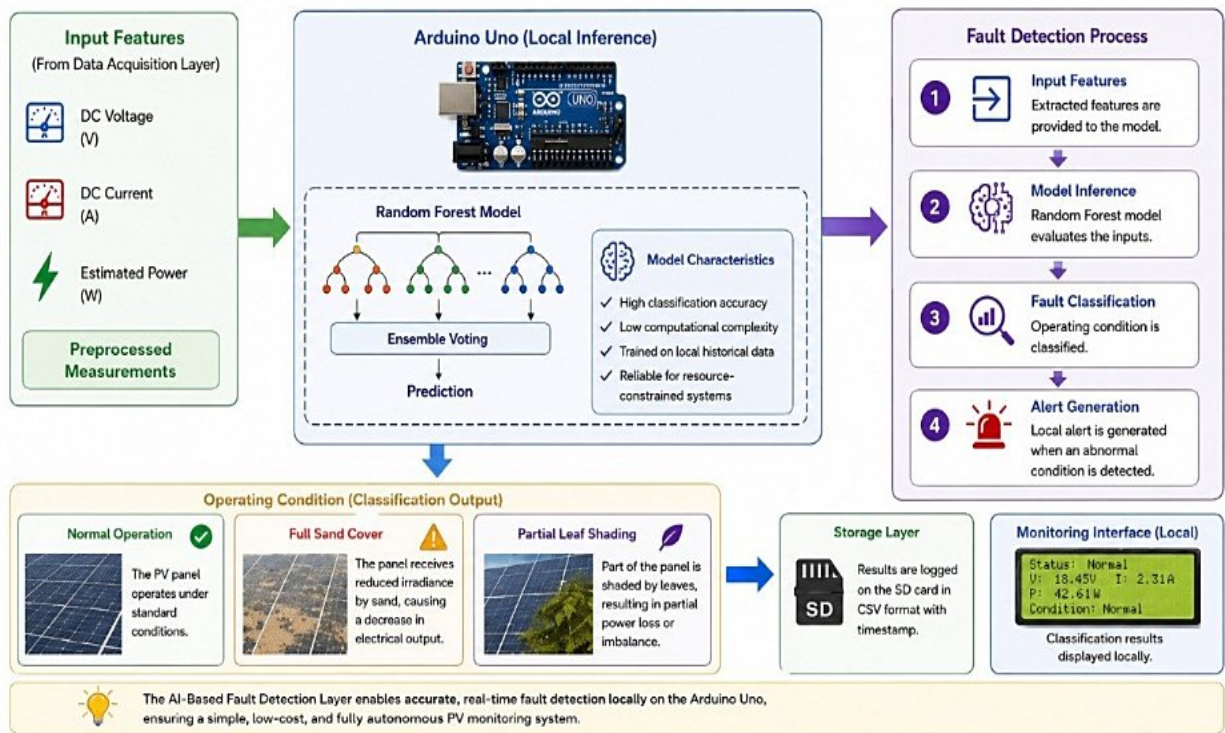


Figure 3.4: AI-based fault detection layer

3.2.5 Application & Monitoring Layer:

The Application & Monitoring Layer provides the interface through which users can observe the operating status of the photovoltaic system and receive fault notifications. Since the proposed system is designed as a standalone embedded solution, monitoring is performed locally without relying on cloud services or remote web applications[54].

The layer displays key information generated by the fault detection module, including measured voltage, current, estimated power, and the detected operating condition of the photovoltaic panel. The monitoring information can be presented through the Arduino Serial Monitor or a simple local display.

The main functions of this layer include:

- **System Monitoring:** Displaying real-time operating parameters of the PV system.
- **Fault Alerts:** Notifying the user when a fault condition is detected .
- **Fault Visualization:** Showing the identified panel state (Normal Operation, Full Sand Cover, or Partial Leaf Shading).
- **User Interaction:** Allowing the user or technician to monitor system performance and inspect detected faults[54].

This layer improves system usability by providing clear and immediate feedback about the condition of the photovoltaic installation while maintaining a low-cost and standalone architecture.

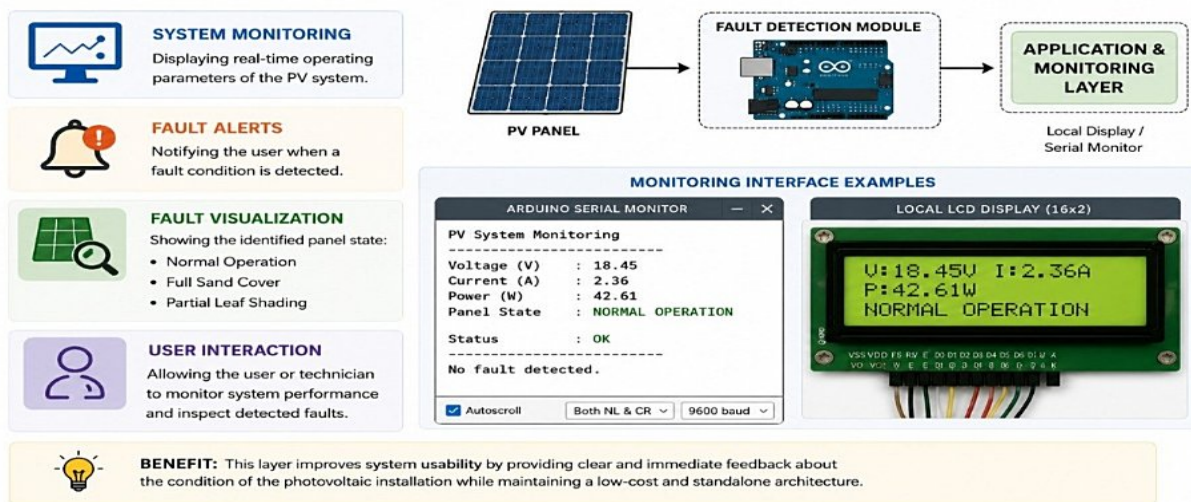


Figure 3.5: Application and monitoring layer

3.2.6 Power Supply Layer:

The Power Supply Layer is responsible for providing stable electrical energy to all hardware components of the proposed photovoltaic monitoring system. Since the Arduino Uno, sensors, and storage modules require regulated low-voltage DC power, a dedicated power management stage is incorporated into the system architecture.

The power supply subsystem consists of:

- **Auxiliary PV Panel:** Provides electrical energy from solar radiation.
- **Battery:** Stores energy and ensures continuous operation during low solar irradiance periods .
- **Voltage Regulator:** Maintains a stable output voltage for the electronic components .

The regulator converts and stabilizes the available DC voltage to the levels required by the system, including:

- **5 V** for the Arduino Uno and most sensor modules .
- **3.3 V** for low-voltage peripherals when required .

This layer guarantees reliable operation of the monitoring platform by protecting electronic components from voltage fluctuations and ensuring uninterrupted power delivery during data acquisition, processing, storage, and fault detection operations .

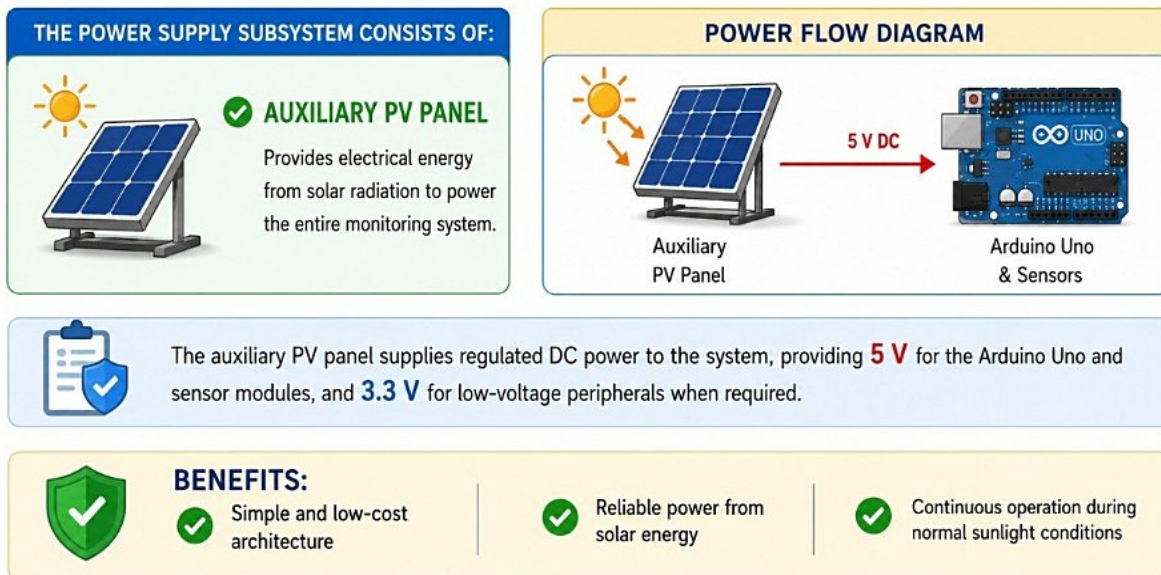


Figure 3.6: Power supply layer

3.3 Conclusion :

This chapter has presented the detailed design of a low-cost, edge-based fault detection system for off-grid photovoltaic installations, specifically tailored to the constraints and requirements identified in the literature review of Chapter 2. The proposed architecture addresses the critical gaps of high cost, cloud dependency, and lack of intelligent fault discrimination on severely resource-constrained hardware.

The system is organized into six functional layers: physical sensing, data acquisition and edge processing, AI-based fault detection, communication and storage, application and monitoring, and power supply. Each layer has been designed with the explicit goal of autonomous, offline operation using only an Arduino Uno (ATmega328P) with 2 KB SRAM and 32 KB flash memory.

The physical sensing layer employs only two primary sensors: a resistive voltage divider (0–25 V DC) and an ACS712 Hall-effect current sensor. The deliberate exclusion of an irradiance sensor, temperature sensor (optional), and wireless communication modules reduces component cost below \$30 while maintaining sufficient information for the targeted fault detection task . This design choice aligns with the low-cost imperative for rural Saharan deployments.

The edge processing and AI-based fault detection layers represent the core contribution of this design. Unlike cloud-dependent commercial solutions , all data processing and classification occur locally on the Arduino Uno. The Random Forest classifier, trained offline, is implemented as a set of integer-based decision rules requiring minimal memory (approximately 1.6 KB flash, 240 bytes SRAM) and no floating-point operations . This enables real-time inference at 1 Hz sampling rates, well within the microcontroller's capabilities.

The communication and storage layer relies exclusively on local SD card logging in CSV format, avoiding any requirement for internet connectivity, cellular subscriptions, or cloud services . This autonomy is essential for remote locations where telecommunication infrastructure is unreliable or absent . The application and monitoring layer provides simple local feedback via serial monitor or optional LCD, sufficient for on-site technicians or users.

The power supply layer ensures continuous operation through an auxiliary PV panel, battery, and voltage regulator, making the entire monitoring system self-sustaining without external power sources .

Several design limitations are explicitly acknowledged. The absence of an irradiance sensor means the system cannot distinguish between low solar irradiance (cloudy conditions) and soiling-

induced current drops without temporal reasoning across multiple samples. The 10-bit ADC resolution of the Arduino Uno limits sensitivity to partial shading affecting less than approximately 5% of panel area . Single-panel monitoring excludes distributed fault detection across parallel strings, and wireless telemetry is not implemented in the current version . These limitations do not invalidate the design but rather define its boundaries and provide clear directions for future enhancements.

In summary, the proposed system architecture directly addresses the five gaps identified in Chapter 2: (1) low cost (under \$30 in components), (2) intelligent fault discrimination via Random Forest, (3) deployment on an 8-bit microcontroller with 2 KB SRAM, (iv) fully autonomous offline operation, and (v) use of only voltage and current sensors to distinguish normal operation, uniform sand coverage, and partial leaf shading . The following chapter presents the experimental implementation of this design, including hardware assembly, data acquisition, model training, and performance evaluation.

Chapter 4:

Applied study and results

analysis



4.1 Introduction:

Out here begins the real-world setup of that solar panel fault detection idea. Built around an Arduino Uno, the device pulls together sensors - voltage ones, current ones - and hooks up an SD card slot to grab and hold readings. Data gathered day by day gets fed into a learning algorithm, shaping how it tells faults apart through trial and review.

Later in this section, details unfold about how tests were arranged, how information was gathered, followed by steps taken to train the system. What came out of those trials gets examined closely - this shows whether the method works well at spotting and sorting problems in solar power setups. Near the end, there's a look at loading the finished model onto an Arduino device, which proves it's possible to run smart monitoring without high costs on basic hardware.

4.2 Hardware Components:

Table 4.2.: Performance Evaluation of the Proposed System

Component	Function
Solar panel	Generates electrical energy
Arduino Uno	Processes and manages data
Voltage sensor	Measures voltage
ASC712 Current Sensor	Measures current
Breadboard	Provides circuit connections
Jumper Wires	Establish electrical connections

4.2.1 Arduino Uno:

- **Definition:**

The Arduino Uno is an open-source electronic board based on the ATmega328P microcontroller. It can be likened to a "little brain" that programs specific tasks.

- **Purpose of use:**

- Reading signals from high-voltage industrial sensors.
- Protecting circuits from damage.

- **Working principle:**

At its core, the Arduino working principle revolves around a simple loop of reading inputs, processing data, and generating physical outputs. First, you write a set of instructions (code) using the Arduino software (IDE) and upload it to the board. The board's onboard microcontroller—

acting as the "brain"—continuously reads electrical signals from connected input components, such as a pressed button, a light sensor, or a temperature gauge. It then processes this incoming data exactly as dictated by your code and sends corresponding electrical signals to output devices, allowing the Arduino to perform tangible tasks like turning on an LED, activating a motor, or sending data to a digital display.

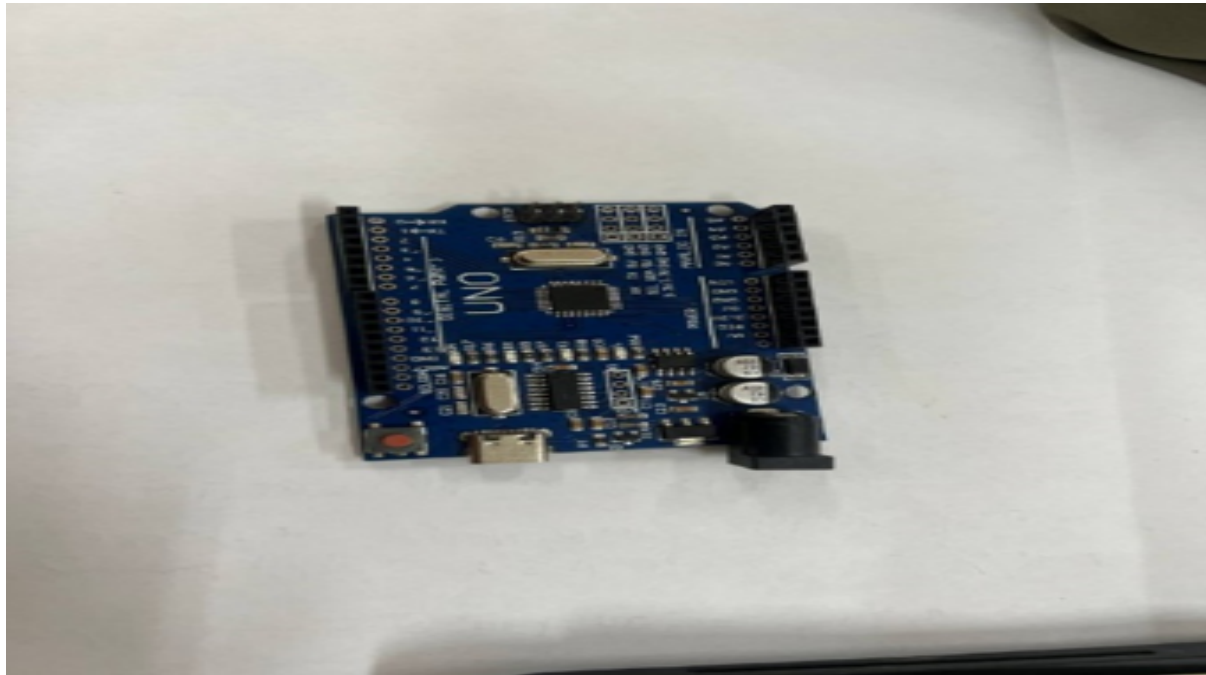


Figure 4.1: Arduino Uno

4.2.2 Voltage Sensor (DC 0-25V) :

- **Definition:**

A voltage sensor is a device that allows an Arduino board to measure electrical voltages higher than the maximum it can handle (which is 5 volts).

- **Purpose of use :**

The ultimate goal is to expand the capabilities of Arduino from a mere "low-voltage signal processor" to a "real electrical measuring and control device".

- **Working principle :**

It is entirely based on the "voltage divider" principle, a fundamental physical principle: Ohm's Law.

Everything begins with the simple **Ohm's Law: $V = I * R$** (voltage = current * resistance).

When an electric current (I) flows through two resistors (R_1 and R_2) connected in series, the voltage is distributed across them in proportion to their resistances.

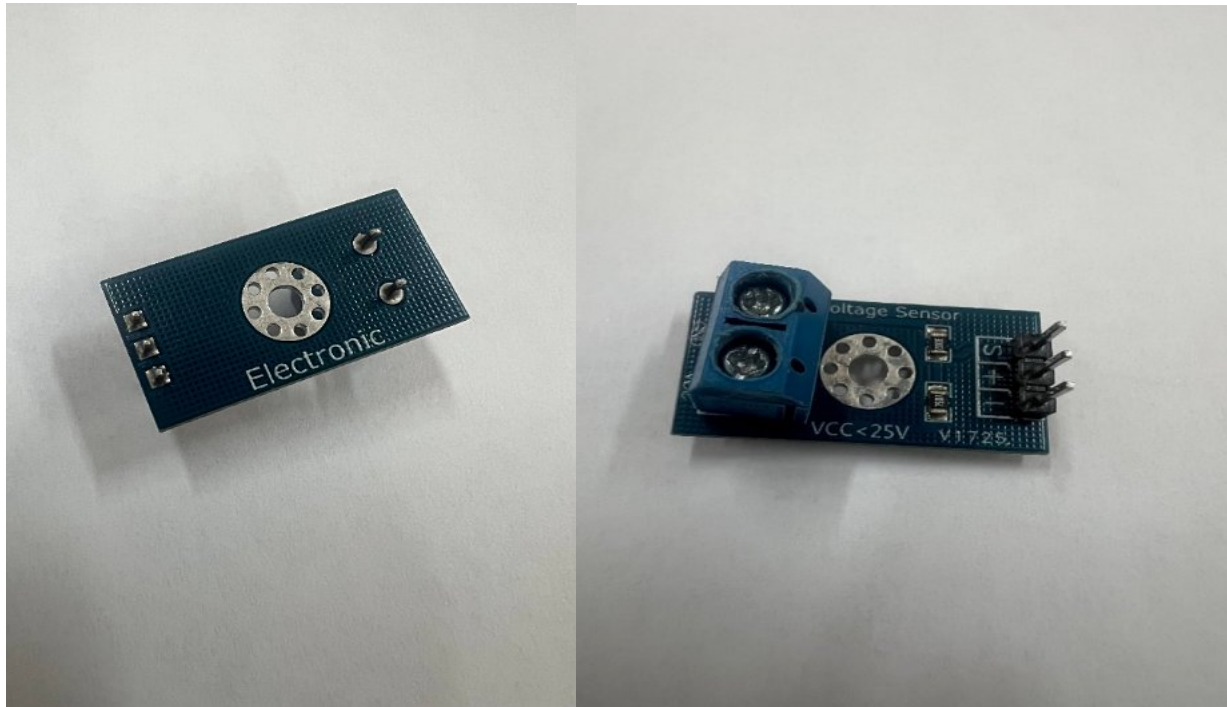


Figure 4.2: voltage sensor

4.2.3 current sensor (ASC712):

- **Definition:**

ACS712 Current Sensor

The ACS712 current sensor is an integrated circuit based on the Hall effect. It is used to measure alternating current (AC) and direct current (DC).

- **Purpose of use :**

The ACS712 current sensor is used to measure AC or DC current and provide real-time current data to Arduino.

In photovoltaic cell monitoring systems, it is used to monitor the solar panel output current, support power calculations, and detect faults.

- **Working principle:**

The ACS712 current sensor measures current using **the Hall effect**. The current flowing through its internal conductor generates a magnetic field, which the Hall effect sensor detects and converts

into an analog voltage proportional to the current value. A microcontroller then processes this voltage to calculate the current value.

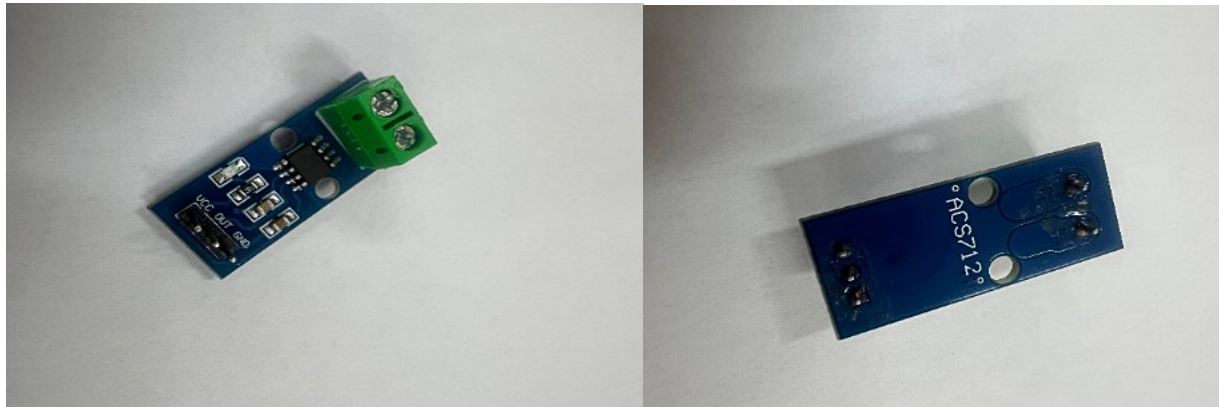


Figure 4.3:current sensor (ACS712)

4.2.4 Solar Panel:

- **Definition:**

A solar panel (also known globally as a photovoltaic panel or photovoltaic module) is an electronic device that captures sunlight and converts it directly into usable electrical energy.

- **Puurpose of use :**

Solar panels operate based on a scientific phenomenon called the photovoltaic effect. The process occurs through the following steps:

- Light Absorption: The panel consists of cells made of silicon (a semiconductor). When sunlight falls on them, these cells absorb light particles (photons).
- Electron Release: The energy absorbed from the light releases electrons from their atoms within the silicon.
- Current Generation: Due to an internal electric field designed within the cell, these free electrons are forced to move in one direction, resulting in a direct current (DC).
- Energy Conversion: Since homes and appliances operate on alternating current (AC), the DC current is transferred to a device called an inverter to be converted into AC, which can then be used to power the devices.

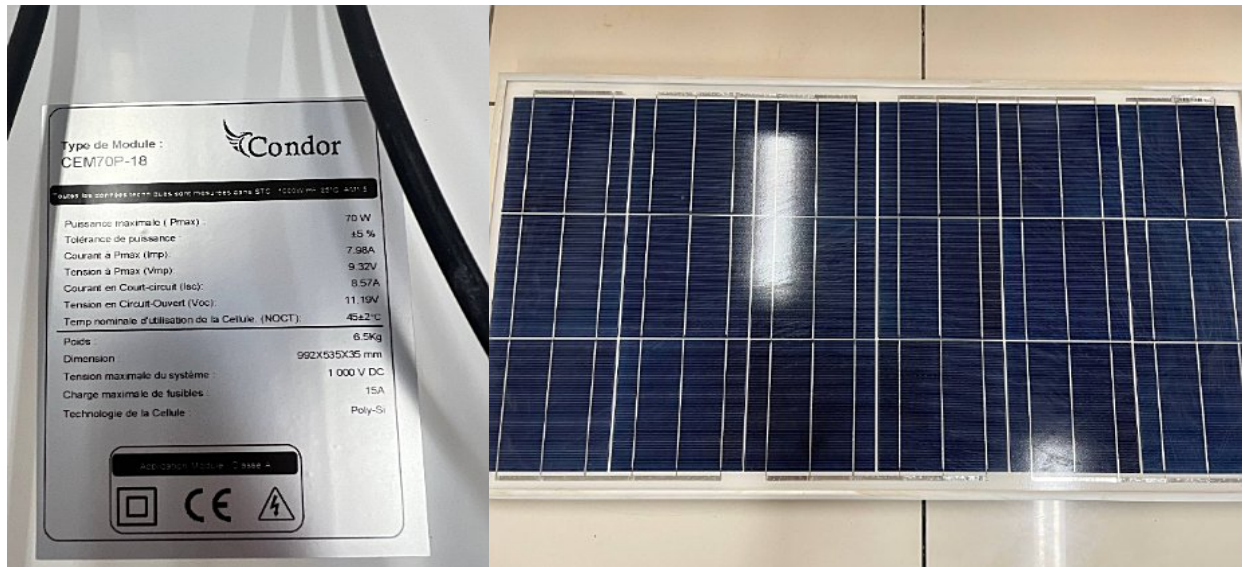


Figure 4.4:solar panel

- **Working principle:**

Solar panels operate according to the photovoltaic effect:

- Absorption: The panel consists of silicon solar cells. When sunlight falls on these cells, the semiconductor material absorbs photons of light.
- Electron flow: The absorbed energy causes electrons to be released from their atoms.
- Current generation: Thanks to the cell's internal electric field, these free electrons are forced to move in a specific direction, generating a direct current (DC).
- Conversion: Since most homes and appliances use alternating current (AC), the DC current is transferred to a transformer (inverter) which converts it into usable AC.

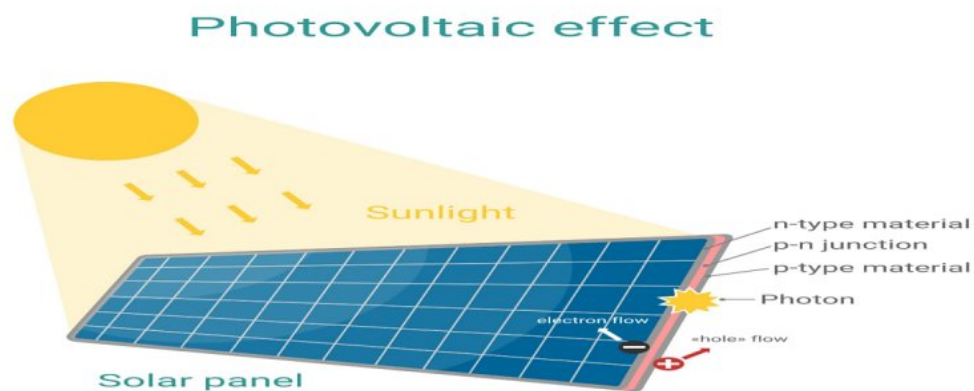


Figure 4.5:The photoelectric effect (source Getty Images)

4.3 General System Design :

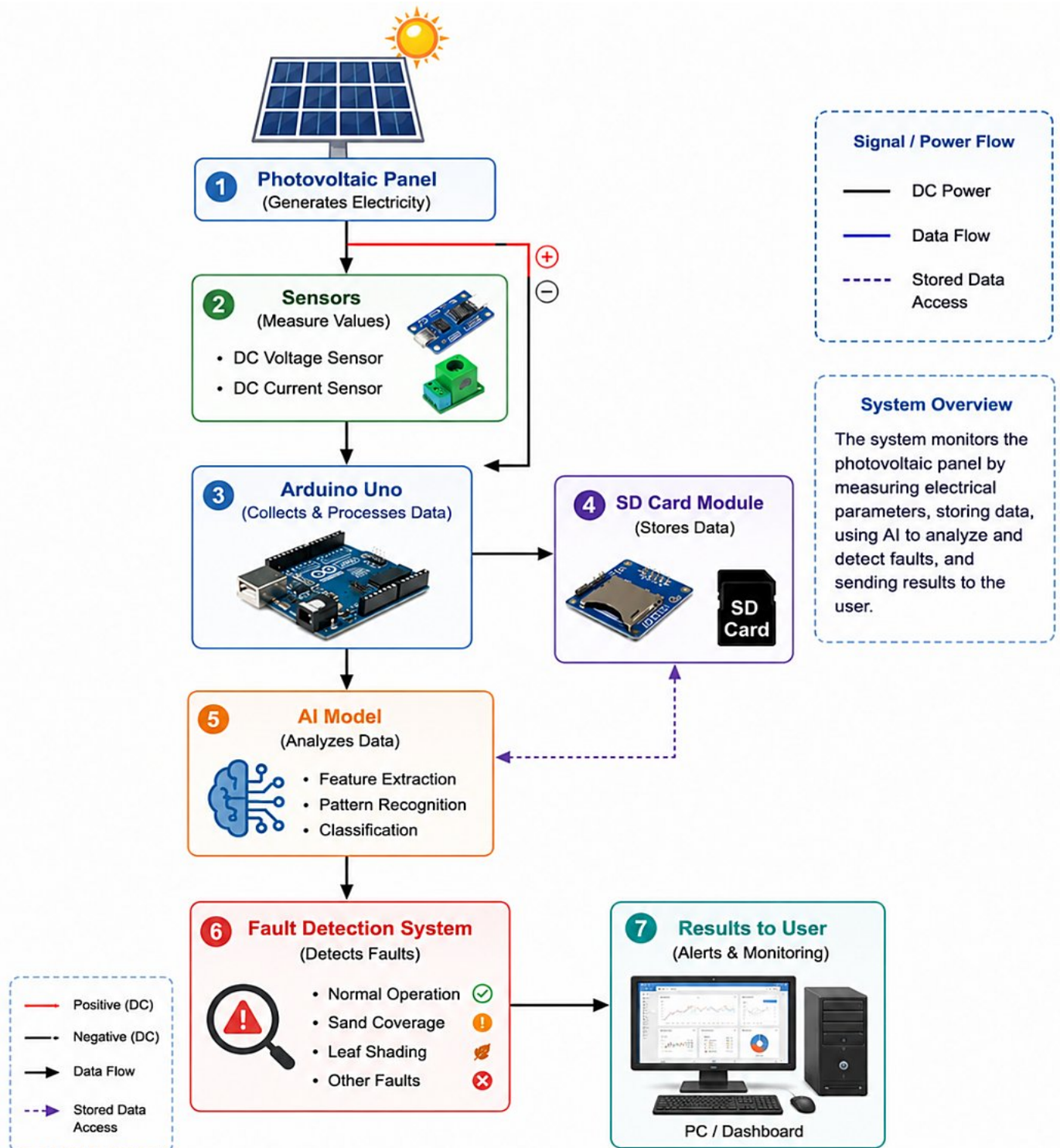


Figure 4.6 : system architecture

System architecture:

- 1-The panels generate electricity.
- 2-Sensors measure values.
- 3-The Arduino collects data.
- 4-Data is stored on an SD card.
- 5-An AI model analyzes the data.
- 6-The system detects faults.
- 7-Results are sent to the user.

4.4 Data Acquisition:

4.4.1. measuring electrical voltage:

The Arduino Uno has a built-in Analog-to-Digital Converter (ADC). It maps an incoming analog voltage on pins A0 through A5 into a digital number between 0 and 1023 (10-bit resolution).

- ✓ 0 Volts becomes 0
- ✓ 5 Volts becomes 1023

To turn that digital number back into a voltage in your code, you use this equation:

$$V_{in} = \frac{\text{Analog Read Value} \times 5.0}{1023.0} \text{--experimental setup:}$$

- The feeder was connected to the voltage sensor (the negative end of the feeder was connected to the GND of the sensor and the positive terminal of the feeder with the VCC of the sensor).
- The PIn(-) of the sensor is connected to the GND of the Arduino
- We connect the Pin (+) from the sensor to (vcc/5v)
- We connect the pin (s) with the (A0) for Arduino
- Finally, we apply the code on Arduino and read the information from the serial monitor

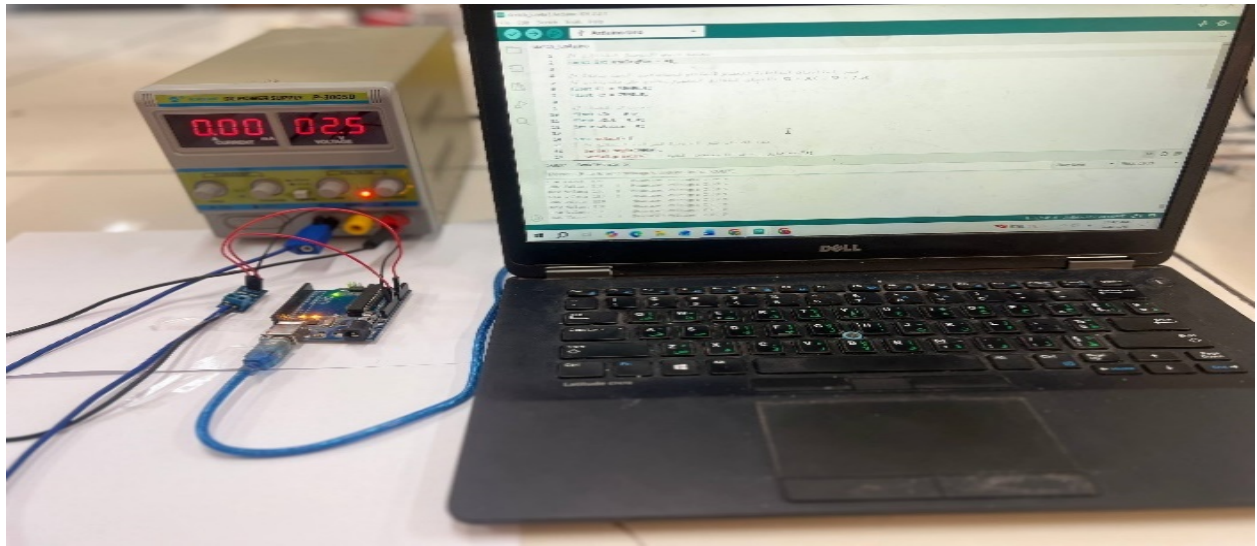


figure 4. 7: montage voltage sensor

- **System flowchart:**

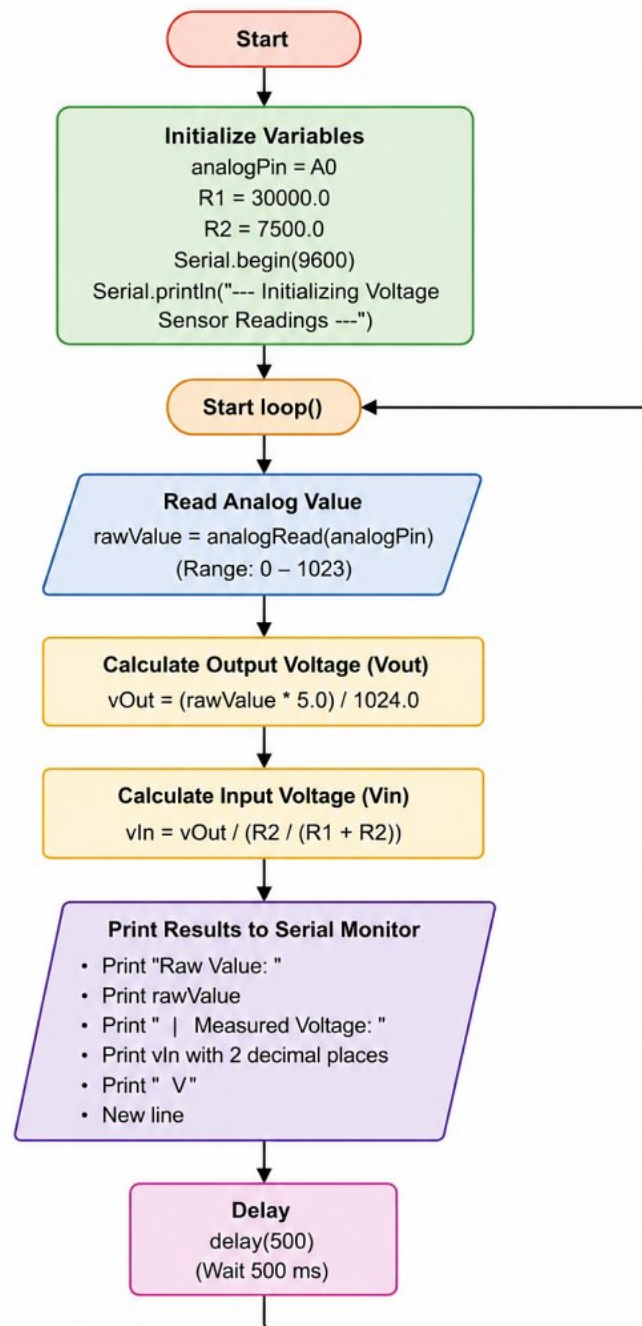


Figure 4.8:Flowchart of the Arduino-Based Voltage Sensor Data Acquisition and Processing System

Starting off, the diagram shows how an Arduino setup measures voltage with help from an analog sensor part. Right after power begins, the device reads incoming voltage signals without stopping. Then comes conversion - those readings shift from raw analog form into numbers a computer can work with. With values now digital, math happens based on resistor ratios to find true voltage levels. Lastly, every calculated result appears line by line in a software window meant for viewing live data.

1. Start:

Out of the gate, things kick off at the Start block - this is where the Arduino program wakes up. A signal flows forward, sparked by that first step lighting the path ahead. From there, movement carries into what comes next, pulled along by logic already set. Each piece fits only when timed right, like gears turning without rush. The sequence holds steady, shaped by design yet driven by simple cause.

2. Variable Initialization:

Here, the software sets up needed values and placeholders

That pin marked A0 is where the voltage sensor links into the system.

Inside the voltage sensor module, two resistors are built in - one measures 30000 ohms, the other 7500. These values define how resistance splits within the device.

A speed of 9600 bits per second starts up serial talk. The setup runs through `Serial.begin(9600)` to make it happen.

A first message shows up on the screen through serial connection - this means measuring voltage will begin now.

A setup begins by readying tools that handle power measurement tasks. Equipment gets arranged alongside programs needed to capture electrical levels.

3. Beginning of the Primary Cycle:

Once it starts up, the code moves into `loop()`, running again and again while the Arduino stays powered. This part keeps cycling without pause during active use.

4. reading analog signals:

A signal comes into the Arduino through A0. From there, it checks the voltage level. This happens each time the system runs. Voltage values show up as numbers. The reading process stays steady. Numbers appear based on what the sensor sends

Starting at zero, the Arduino's ADC turns voltage levels into numbers between 0 and 1023. Instead of staying analog, each measured voltage maps to one of those steps up to 1023. From full off to maximum, that span covers voltages starting at exactly 0 volts. On top of that, the highest number matches an input near 5 volts. Step by step, every reading fits within that boundary.

5. Output Voltage Calculation:

Starting with the raw ADC number, voltage at the Arduino's analog pin comes next through a calculation. That result ties to how much electric pressure appears on the pin. A formula shifts digits into measurable levels. Numbers shift form based on reference points inside the chip. What begins as counts turns into volts by scaling step by step. Each reading moves closer to real-world units this way. The process links digital output to physical input steadily.

Out here, the number from the device becomes a real-world voltage seen at the sensor's end. What shows up digitally shifts into a physical signal down the line. A reading in code turns into measurable electric pressure right where it exits. From ones and zeros springs a tangible push of power at the tip. The machine count changes form, landing as volts where it connects.

6. Input Voltage Calculation:

Because the sensor uses a voltage splitter, scaling up the reading brings it back to the true input level. This calculation follows the divider formula inside the code

What you get here shows exactly how much voltage reaches the sensor's input.

7. Displaying the Results:

Out comes the computed figures into the Serial Monitor. What shows up next is this display from the software

The raw ADC value (rawValue)

The calculated input voltage (vIn)

The voltage unit (V)

Exactly two decimal spots show up in the voltage number, making it easier to read while keeping measurements sharp.

8. Delay:

A half second pause happens because of:

Too much picking stops here. Every check stays steady, spaced just right after the last one.

9. Repeat the Process:

Once paused, it jumps back to where loop() starts, running through each step again without stopping. Because of this pause, the system keeps checking the voltage as it happens.

Conclusion:

A steady stream of voltage checks runs on an Arduino setup, shown here in diagram form. Starting each cycle, raw data comes in through analog inputs. Then math steps follow - shaped by how voltage dividers work - turning numbers from the converter into true voltage amounts. Right after, those results appear live in the Serial Monitor window. Often found in solar panel tracking gear. Just as often inside smart sensor networks that report remotely.

4.4.2. Measuring electric current:

Since microcontrollers cannot directly measure current through their own ports, they rely on specialized external sensors:

Parallel resistors: A very small, low-voltage resistor is placed in series with the load. The microcontroller measures the slight voltage drop across this resistor and uses **Ohm's law** ($=V/R$) to calculate the current.

Hall effect sensors (such as the ACS712): The circuit current flows through a copper conductive path within the integrated circuit, generating a magnetic field. The built-in Hall effect sensor converts this magnetic field into a proportional analog voltage that the Arduino can easily read and manipulate.

- **experimental setup:**

- The feeder is connected to the current sensor by connecting both (negative and positive) between the two ends of the resistance
- The positive end of the resistor is connected to the Vcc of the sensor
- The negative end of the resistance to the GND is connected to the GND of the sensor
- The sensor out is connected to (A0) for Arduino
- Both the vcc and the sensor's GND are connected to the Vcc and the Arduino's GND

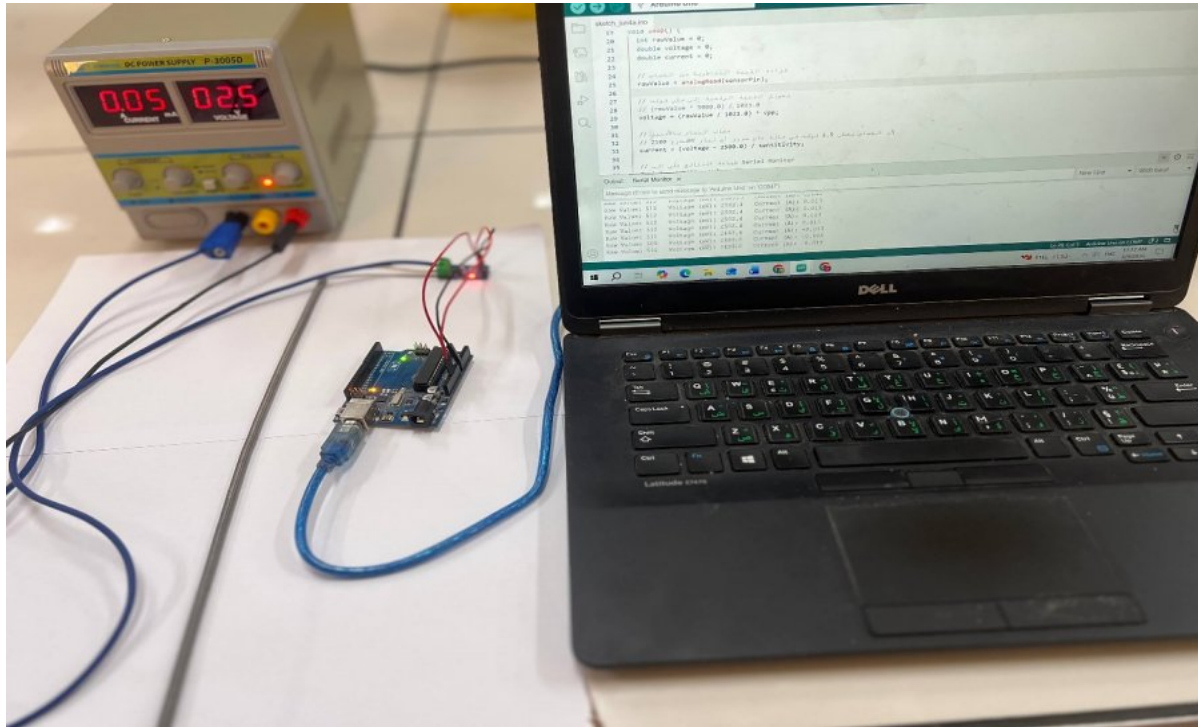


figure 4.9:montage of current sensor

• **System flowchart :**

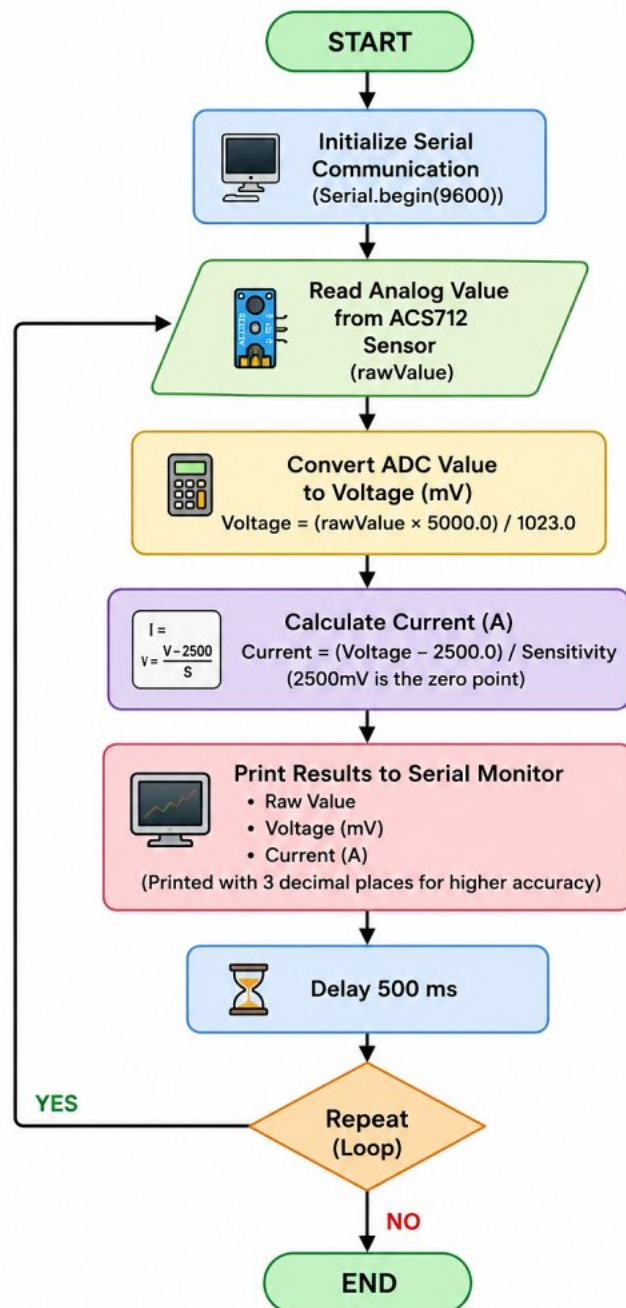


Figure 4.10: Flowchart of the Current Sensing Algorithm Using ACS712

Upon startup, the system reads data from the ACS712 sensor via the Arduino. Instead of pausing, it continuously measures signal levels. After collecting each value, the raw input is converted into readable voltages. Calculations then convert these numbers into actual current

readings. Finally, the output is displayed on the screen line by line—without any delay, only continuous updates.

1. Startup:

Execution begins at the startup block, indicating that the program is running on the Arduino board. This is where the operation within the microcontroller begins.

2. Constant Initialization:

This stage defines the rules for measuring electrical current. What is measured depends on the settings defined here. Details such as range and accuracy are clarified at this stage. The specifications are established before the test begins. Each value is inferred from the previous choices. The conditions for reading the amperage are now defined.

A wire runs from the sensor to the pin that reads the values. This connection allows the system to sense changes through voltage shifts. Current flows to the chip, which reacts when current moves near it. The device detects the flow of electrical current without direct contact.

Sensor sensitivity values by model: 185 mV/A for ACS712-5A, 100 mV/A for ACS712-20A, 66 mV/A for ACS712-30A.

185 mV/A for ACS712-5A

100 mV/A for ACS712-20A

66 mV/A for ACS712-30A

Operating voltage ($VCC = 5000\text{ mV}$).

Without these constant values, converting voltage readings to correct current values will not be accurate. Every measurement depends on them to ensure accuracy. They act as a link between what the sensor displays and the actual units. Accuracy is negatively affected if they are ignored. Every calculation needs this link to understand the raw data.

3. Setting up the serial connection:

Once the board wakes up, a signal is sent through the port, configured to transmit data at a rate of 9600 steps per second. A note then appears in the monitoring window – this indicates that the sensor loop is in standby mode.

4. Starting the master loop:

Once powered on, the system switches to the `loop()` function, which continuously captures live readings. This phase continues, updating each measurement as conditions change. Each cycle is re-verified to ensure the data remains up-to-date through continuous iteration.

5. Reading analog signals:

The Arduino reads the analog output voltage of the ACS712 sensor using:

The voltage reading changes as it reaches the analog-to-digital converter (ADC). This component converts what the sensor picks up into numbers between 0 and 123. Instead of continuous waves, steps appear – each step representing a point on a scale designed for machines.

The voltage reading changes as it reaches the analog-to-digital converter (ADC). 6. Converting an Analog-to-Digital (ADC) Signal Converter Reading to Voltage

Using this method, a digital number is converted into a tiny electrical pulse, measured in millivolts. The actual voltage emitted by the sensor when measuring the current is calculated. This number is derived directly from the sensor's reading at that moment. The voltage appears based on the amount of current flowing. The measured value depends entirely on the current intensity. The reading perfectly matches what is happening during the test.

7. Calculating Current:

Here's how it works: The ACS712 sensor sets a reference point at 2.5 volts in idle mode. This reference point changes whenever an electrical current flows. To obtain accurate readings, the idle voltage is first subtracted. Only then is the actual current displayed.

The measured value shows the amount of electrical current present, measured in amperes.

8. Displaying Measurement Results:

The information is displayed on the serial monitor after processing. What is shown here is the result of previous calculations.

This number is stored directly from the sensor as a raw value. Sensor output voltage (voltage) in millivolts.

Measured current (current) in amperes.

Typically, the value is displayed with three decimal places, which helps clarify readings while maintaining detailed accuracy. Measurement accuracy is crucial when small differences are critical.

9. Stabilization Delay:

After each measurement, there is a short pause before the next measurement begins. This allows the values to stabilize completely before recording. Readings become more stable over time between checks. The data becomes less volatile due to the mitigation of sudden spikes. The final display reflects more accurate conditions thanks to the spacing between inputs.

10. Continuous Monitoring:

Once the waiting period ends, the program returns to the beginning of the loop, re-executing each step. This continues to monitor the current without interruption.

Conclusion:

The diagram initially illustrates how the ACS712 sensor works with an Arduino board to track electrical current. Instead of relying solely on raw data, it converts incoming analog signals into readable voltage values. Then, using the sensor's known characteristics, it determines the actual current levels. These results are then transmitted via a serial port without any delay. Such systems are often found monitoring energy consumption in solar panel grids. They are also used in internet-connected energy meters and smart electricity control systems. This approach enables real-time monitoring in many modern energy systems.

4.4.3.Test for SD-CARD:

To ensure that the SD card saves the measured information, a simple test was performed to check whether it was indeed saving the information or not.

- **experimental setup:**

- The feeder GND was linked to the GND of the Arduino.
- The feeder's VCC has been linked to the Arduino VCC.
- This image represents the necessary arrangement of the CD-Card with Arduino.

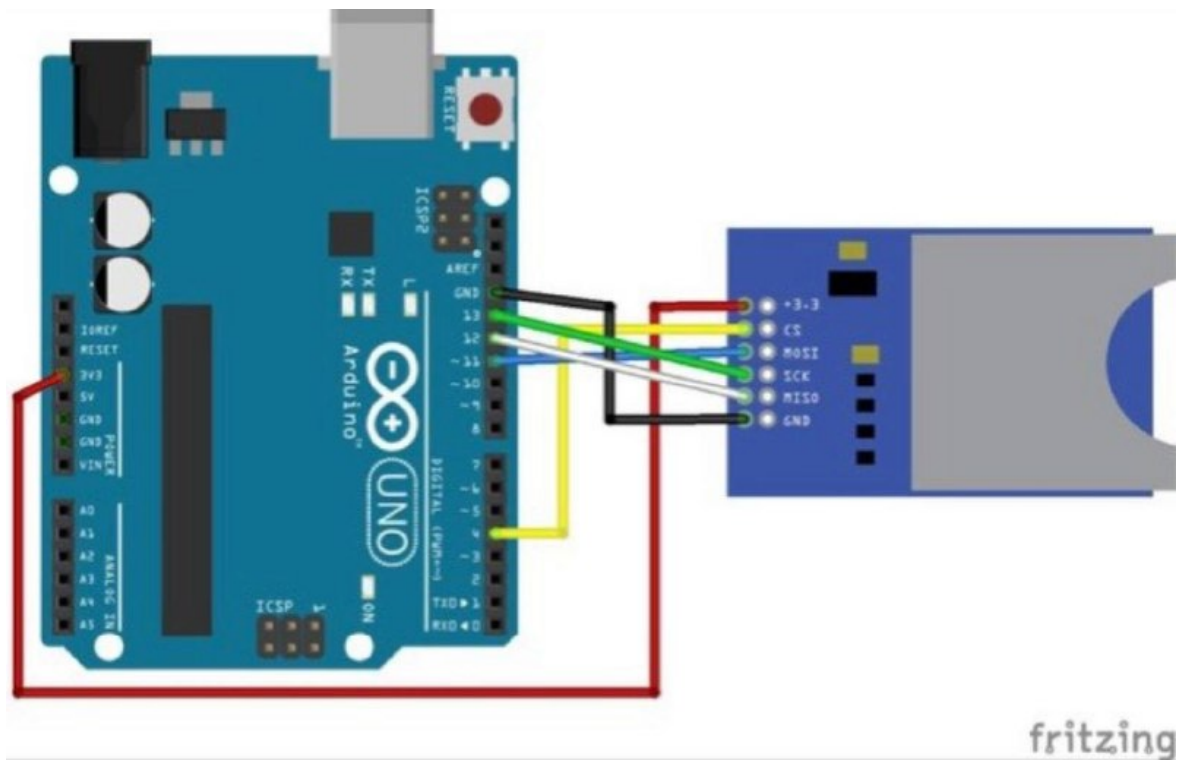


Figure 4.11:Connecting the SD card to the Arduino

- **System flowchart :**

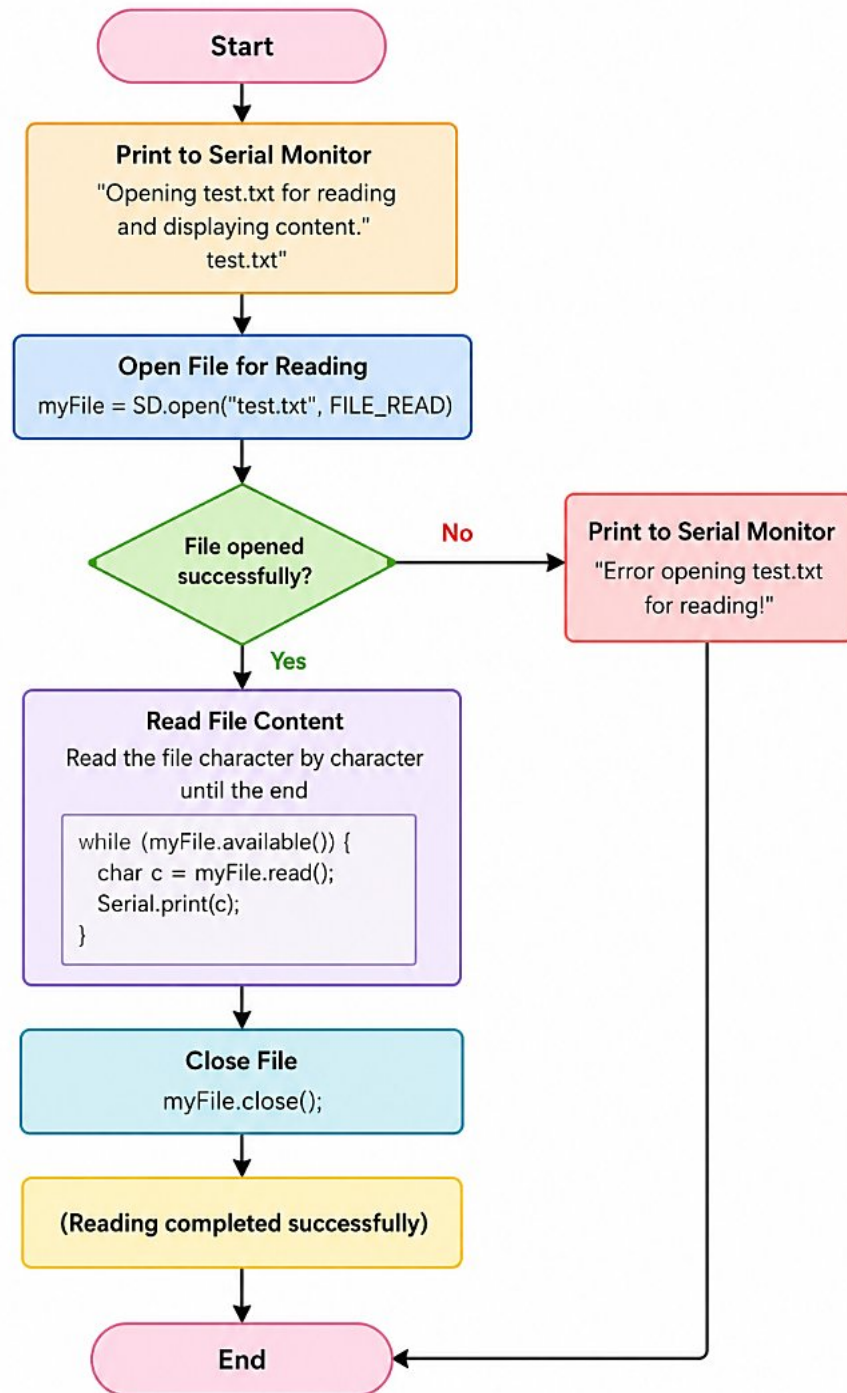


Figure 4.12:Flowchart of the File Retrieval and Data Display Procedure from an SD Card
Using Flowchart - Academic Overview

At startup, the system reads data from the ACS712 sensor via the Arduino. Instead of pausing, it continuously measures signal levels. After collecting each value, the raw input is converted into readable voltages. Calculations then convert these numbers into actual current readings. Finally, the output is displayed on the screen line by line—without any delay, just continuous updates.

1. Startup:

Execution begins at the startup block, indicating that the program is running on the Arduino board. This is where the operation within the microcontroller begins.

2. Constant Initialization:

This stage defines the rules for measuring electrical current. What is measured depends on the settings defined here. Details such as range and accuracy are clarified at this stage. The specifications are set before the test begins. Each value is inferred from the previous choices. The conditions for reading the amperage are now defined.

A wire runs from the sensor to the pin that reads the values. This connection allows the system to sense changes through voltage shifts. Current flows to the chip, which reacts when current moves near it. The device detects the flow of electrical current without direct contact.

Sensor sensitivity values by model: 185 mV/A for ACS712-5A, 100 mV/A for ACS712-20A, 66 mV/A for ACS712-30A.

185 mV/A for ACS712-5A

100 mV/A for ACS712-20A

66 mV/A for ACS712-30A

Operating voltage ($VCC = 5000 \text{ mV}$).

Without these constant values, converting voltage readings to correct current values will not be accurate. Every measurement depends on them to ensure accuracy. They act as a link between what the sensor displays and the actual units. Accuracy is negatively affected if they are ignored. Every calculation needs this link to understand the raw data.

3. Setting up the serial connection:

Once the board wakes up, a signal is sent through the port, configured to transmit data at a rate of 9600 steps per second. A note then appears in the monitoring window – this indicates that the sensor loop is in standby mode.

4. Starting the master loop:

Once powered on, the system switches to the `'loop()'` function, which continuously captures live readings. This phase continues, updating each measurement as conditions change. Each cycle is re-verified to ensure the data remains up-to-date through continuous iteration.

5. Reading analog signals:

The Arduino reads the analog output voltage of the ACS712 sensor using:

The voltage reading changes as it reaches the analog-to-digital converter (ADC). This component converts what the sensor picks up into numbers between 0 and 123. Instead of continuous waves, steps appear – each step representing a point on a scale designed for machines.

The voltage reading changes as it reaches the analog-to-digital converter (ADC). 6. Converting an Analog-to-Digital (ADC) Signal Converter Reading to Voltage

Using this method, a digital number is converted into a tiny electrical pulse, measured in millivolts.

The actual voltage emitted by the sensor when measuring the current is calculated. This number is derived directly from the sensor's reading at that moment. The voltage appears based on the amount of current flowing. The measured value depends entirely on the current intensity. The reading perfectly matches what is happening during the test.

7. Calculating Current:

Here's how it works: The ACS712 sensor sets a reference point at 2.5 volts in idle mode. This reference point changes whenever an electrical current flows. To obtain accurate readings, the idle voltage is first subtracted. Only then is the actual current displayed.

The measured value shows the amount of electrical current present, measured in amperes.

8. Displaying Measurement Results:

The information is displayed on the serial monitor after processing. What is shown here is the result of previous calculations.

This number is stored directly from the sensor as a raw value. Sensor output voltage (voltage) in millivolts.

Measured current (current) in amperes.

Typically, the value is displayed with three decimal places, which helps clarify readings while maintaining detailed accuracy. Measurement accuracy is crucial when small differences are critical.

9. Stabilization Delay:

After each measurement, there is a short pause before the next measurement begins. This allows the values to stabilize completely before recording. Readings become more stable over time between checks. The data becomes less volatile due to the mitigation of sudden spikes. The final display reflects more accurate conditions thanks to the spacing between inputs.

10. Continuous Monitoring:

Once the waiting period ends, the program returns to the beginning of the loop, re-executing each step. This continues to monitor the current without interruption.

Conclusion:

The diagram initially illustrates how the ACS712 sensor works with an Arduino board to track electrical current. Instead of relying solely on raw data, it converts incoming analog signals into readable voltage values. Then, using the sensor's known characteristics, it determines the actual current levels. These results are then transmitted via a serial port without any delay. Such systems are often found monitoring energy consumption in solar panel grids. They are also used in internet-connected energy meters and smart electricity control systems. This approach enables real-time monitoring in many modern energy systems.

4.5 Database preparation:

The database is one of the most important stages in building any machine learning model, as the accuracy and efficiency of the model directly depend on the quality of the data used in training and testing. In this work, a database was created based on voltage and current measurements captured from the solar panel under different operating conditions. This data has been organized and classified into several categories representing normal conditions and various malfunctions, with the aim of providing an appropriate data set to train the machine learning model and enable it to distinguish between the various operating states of the photovoltaic system accurately and effectively.

4.5.1 Unmallected measurement (normal system operation):

The system consists of a solar panel, an Arduino Uno panel, a voltage sensor (Voltage sensor), and an ACS712 current sensor. The solar panel was connected to the sensors in order to measure the electrical parameters and sent to the Arduino board for processing.

- Connecting the voltage sensor:

The positive terminal (+) of the solar panel is connected to the input of the voltage sensor.

The negative end (−) of the solar panel is connected to the ground (GND).

Signal output is connected to the A0 input in Arduino Uno.

The VCC and GND terminals of the sensor are connected to 5V and GND in the Arduino.

- Connecting current sensor ACS712:

The positive wire coming from the solar panel is inserted via the measurement path in the ACS712 sensor respectively (Series Connection).

The sensor output (OUT) is connected to the A1 analog input in Arduino Uno.

VCC and GND terminals are connected to 5V and GND in Arduino.

- Connect Arduino Uno:

The Arduino Uno is connected to the computer by a USB cable.

The computer is used to program Arduino and monitor the measured values via Serial Monitor or save them in a data file.



Figure 4.13: normal montage

- Working principle:

The solar panel generates voltage and current, where the voltage sensor measures the value of electrical tension while the ACS712 sensor measures the value of the current flowing in the circuit. These values are sent to the Arduino Uno board that calculates the electrical capacity according to the relationship:

$$P=V\times I$$

The data is then used to monitor the performance of the solar panel and detect various malfunctions.

-System flowchart:

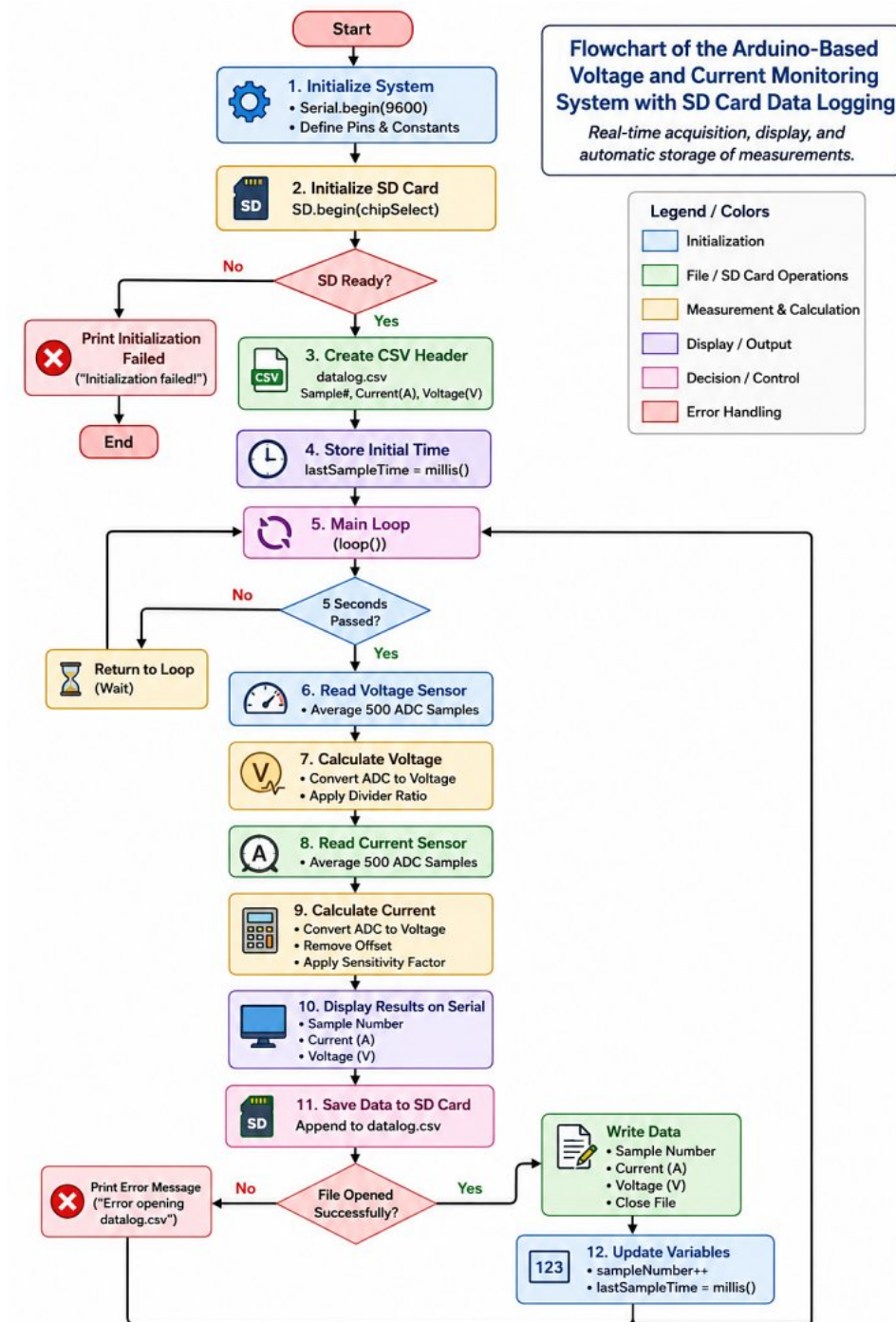


Figure 4.14: Flowchart of the Real-Time Voltage and Current Acquisition and SD Card Data Logging Process in the Proposed PV Monitoring System

A sequence of steps shows how a gadget built around an Arduino tracks electrical voltage along with current. This setup grabs inputs from sensors, then processes them immediately. Instead it feeds live numbers to a screen people can watch. Information does not vanish; rather it gets written onto an SD memory chip without user help. Because it handles sensing, computing, display updates, plus storage tasks together, the device fits well into solar panel oversight jobs. Engineers also find it useful when building internet-connected power tracking tools.

1. System Initialization:

Start up happens when the Arduino gets set into motion. Right away, serial talk lines form at 9600 bits per second so data moves smoothly to the monitor window. Instead of waiting, pin links for sensors lock in place along with fixed values used later. Calibration settings take shape here too, just before readings begin.

2. SD Card Initialization:

Communication begins when the Arduino reaches out to the SD card module using SPI signals. Without this link, stored data cannot be written, since measurements rely on the card's memory space.

Checking if the SD card started up right happens here. This step makes sure everything fired up without hiccups. Only when confirmation arrives does the process move forward. Without that signal, it waits. The system needs this green light before continuing further

When setup doesn't work, a warning shows up - then everything stops.

Once things start up without a problem, the watch system moves on. The process shifts forward when startup works out smoothly.

3. Creating the CSV File Header:

Once the SD card starts up correctly, it either makes a new file called datalog.csv or opens one that already exists. A label row with field names goes in first. This row lists each column heading needed

Sample Number

Current (A)

Voltage (V)

From here on, every reading gets saved in a clear layout, one that fits right into spreadsheets or programs used for studying data.

4. Setting Up Time Values:

Right now, the time gets saved by calling Arduino's `millis()` function. From there, timing stays on track because each sample waits its turn based on that starting number. Periodic checks line up neatly since every reading follows the same timed pattern.

5. Main Monitoring Loop:

Every few moments, the system wakes up to take readings while quietly saving each one. Through cycles without pause, fresh inputs get recorded just after checks finish. Right when a measurement ends, storage begins - repeating until stopped. Step following step, it moves ahead on its own.

6. Sampling Interval Verification:

Every few moments, it glances at the clock to see if half a minute has slipped by since the prior reading.

Should the time span still be running, back to the start of the loop it goes.

After five seconds pass, it starts gathering data.

Every time it runs, the system keeps the sample flow steady while blocking any overflow of information.

7. Voltage Measurement:

Every now and then, the Arduino checks the voltage sensor more than once to smooth out any glitches in the reading. Instead of taking a single number, it gathers several points through the ADC, blending them together. From there, those numbers shift into real-world volts by way of a fixed reference and the resistor network's proportion. This step ties raw data to physical meaning without extra steps.

Starting with the raw signal, the system adjusts for losses caused by the sensing unit. Because of this correction, the true input voltage takes shape again. What comes out matches what went in, just clearer now. Through careful scaling, weakened levels gain back their initial strength. Every step focuses on reversing the drop from the hardware. In the end, accuracy returns where it was missing before.

8. Current Measurement:

Every now and then, the ACS712 current sensor takes a new reading, with values smoothed over time to reduce jumps from stray electrical interference.

After measuring the ADC reading, it becomes a voltage. That voltage shifts into current through the sensor's sensitivity setting. With that step done, subtracting the zero-current offset comes next. Only then does the system find the actual current result.

9. Measurement Results Display:

After finishing the math on voltage and current, it sends those numbers straight to the Serial Monitor. The readings show up there right after the computations finish. When the work is done, off they go - sent without delay. Values move over once calculated, making their way into view. Only when both figures are ready does the transfer begin.

The displayed information includes:

Sample number

Current value (A)

Voltage value (V)

Watching the electrical values happens as they change. A system keeps track without delay.

10. Data Logging on the SD Card:

Stored inside datalog.csv, the computed values appear right after calculation. Each result slips into place without needing a manual save step.

Should the file fail to open, the process checks its status right away. Opening it works only if access clears without issues. When ready, the system moves forward. Failure here stops everything immediately

Should access fail, the system shows a warning.

Should the file open without issues, then the fresh measurement gets added onto the current data set.

Each record consists of:

Sample number

Current value

Voltage value

Freed from further edits, the file shuts right away once information lands inside, holding things intact while dodging damage.

11. Updating System Variables:

Once the measurement information has been safely saved:

Every time a new item arrives, the tally moves up by one.

Last check time shifts when new data comes in.

Now the system can move forward into another round of acquisitions. Updates like these set the stage just right for what comes next.

12. Continuous Monitoring:

Back at the start of the main loop, it runs through everything again without stopping. Because of this setup, readings keep getting checked moment by moment. Visual updates appear instantly on screen while data piles up safely over time.

Conclusion:

A close look at the design reveals an embedded setup built around measuring voltage and tracking current, then moving into signal handling through a microcontroller. Following each reading cycle, collected values get processed before being sent out via UART links. Instead of immediate transmission only, information also gets saved directly onto memory cards using comma-separated formatting. Over time, these stored records form a steady stream useful for watching solar power behavior. Thanks to background corrections based on preset offsets and repeated sample grouping, numbers stay consistent. Even under shifting conditions, confidence in recorded results remains high.

4.5.2 Measurement with defects (tree leaves malfunction):

Through what we mentioned earlier and according to what our study agrees, we try to apply a slight malfunction for the study by placing a little foliage on the solar panel while maintaining the same previous experimental composition and the same Arduino code with continuous monitoring of the results and saving them in the SD card in a period of time estimated at approximately 6 hours with the collection of more than 1000 values for the study.



Figure 4.15:Measurement with a malfunction (dying the leaves)

4.5.3. Measurement with defects (soil defect):

Because the plan remained unchanged, we added just enough dust to the board to make a slight difference. With the same setup and the same Arduino software, we made no modifications. Monitoring continued uninterrupted, with each reading saved directly to the memory card. The monitoring lasted for approximately six hours, accumulating over a thousand data points in end .



Figure 4.16:Measurement with soil efect

4.6 Machine Learning Model Training:

At this stage, the data collected from the solar panel, represented by the values of voltage, current and power, were used to train a machine learning model capable of distinguishing between the different operating states of the solar system. The data was first organized within the CSV file and then divided into two groups: a training set used to teach the model (80%), and a test set used to evaluate its performance (20%).

Paython code

following Python script illustrates the implementation of data preprocessing, feature extraction using sliding windows, model training with cross-validation evaluation, and a inference logic for diagnosing photovoltaic (PV) array faults.

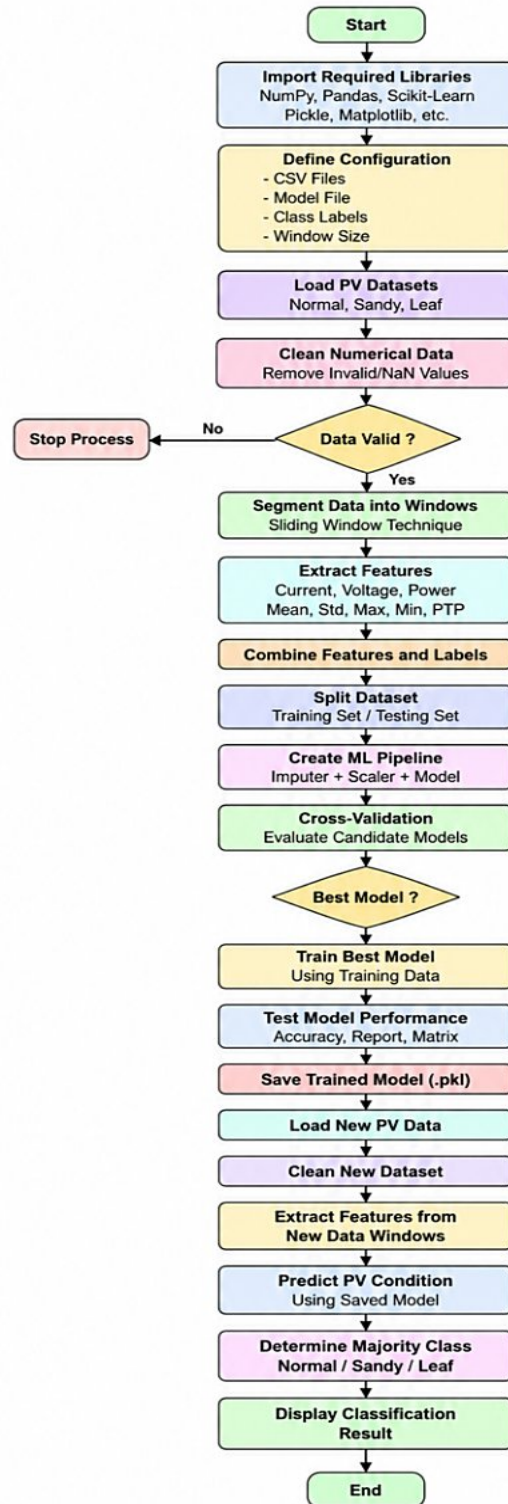


Figure 4.17:Methodological Flowchart for Photovoltaic System Condition Classification Using Machine Learning Techniques

A flowchart shows how a machine learning tool for spotting faults in solar panels was built and put into use. After loading key software tools - like NumPy, Pandas, Scikit-Learn, Pickle, and Matplotlib - the work moves forward. These tools handle everything from organizing data to training models, checking results, and making charts. Next comes setting up fixed settings: the CSV files used as inputs, where the trained model gets saved, names of fault types, along with the span of each moving data segment. Each choice here shapes how the system will later examine patterns in real-time performance. The steps follow one after another, linking code setup directly to outcome tracking without skipping stages.

After that comes pulling in solar power records - each set tied to how the panels run under clear skies, dust buildup, or leaves blocking light. Once gathered, messy bits like blank spots or broken numbers get stripped out. Checking happens next, making sure what remains fits basic rules; when it does not fit, everything stops right there. Only clean readings move forward from this point.

Once checked, pieces of data get split into equal chunks through a moving frame method. Out of every chunk, numbers like average, spread, highest point, lowest point, and total swing come from readings such as volts, amps, and wattage. Such details usually involve central tendency measures along with range shifts across samples. Later on, these traits join up with matching category tags to build the ultimate set used for analysis.

From the cleaned data, one portion feeds practice runs while another holds back for final checks. Built step by step, a system handles missing values, adjusts number ranges, then assigns categories. To compare options, each method gets tested across different splits of the sample pool. Once picked, the best setup learns from the first group and later predicts outcomes in the second. Results show how well it worked - measured by correct guesses, error patterns, and detailed summaries.

After finding the best version of the model, it gets stored as a .pkl file ready for later use. When running, fresh solar panel readings come in and go through cleaning just like before. Instead of skipping steps, each chunk of incoming data goes into feature extraction right away. From there, those features move straight into the learned system for judgment. That system picks which label fits best - Normal Operation, Sandy Condition, or Leaf Shading. Then, by counting votes across results, the most frequent outcome wins. In the end, that answer shows up on screen, wrapping everything up cleanly.

4.7. Model Runs on Arduino:

Once the machine learning model finished training, performance checks followed using gathered data. From there, deployment happened on an Arduino Uno board - targeting live detection of solar panel issues. Voltage plus current inputs come through connected sensors. These readings feed into calculations for electric power output. That computed result moves over to the trained algorithm. A decision forms about the panel's condition based on what the model sees. Operation status gets updated continuously as new sensor streams arrive.

Every now and then, Arduino grabs sensor readings - voltage and current get turned into numbers it can work with. From there, power is worked out using a specific formula

$$P=V\times I$$

After that comes the part where those numbers feed into the trained model so it can decide if things are running fine or have gone wrong. Later on, someone might need to look back at what happened, which is why everything - both raw readings and outcomes - gets saved onto the SD card.

From start to finish, running the model gives the Arduino a standalone way to watch how well the solar panel works - all at low cost. Instead of relying on outside computers, it catches problems fast while tracking output nonstop. In real conditions, this change makes things fit better when used outdoors. Far from cities, simplicity matters most - here, checking solar setups becomes both practical and reliable.

Out here, proof showed linking Arduino to a machine learning setup actually helps track solar panel performance better. This mix sharpens how faults are caught, making checks more dependable. A steadier detection method means fewer delays when fixing issues. System output gets a quiet boost while repair hours shrink. Fewer tech visits happen once things run smoother.

4.8. Analysis of results:

A classification model was developed to distinguish three states of solar panels: natural, sandy, and covered with leaves. The Random Forest model achieved the best performance with an accuracy of 94.6% in cross-checking and 92.8% on the test data, with a narrow difference (1.8%) confirming the absence of over processing.

Category Performance.

- Natural and sandy: accuracy and recall >95% (excellent performance)
 - Papers: Call only 88%, that is, 12% of the samples of the sheets are misclassified as normal.
- Possible reason: The statistical similarity between light shade and clean board.

Compare models.

Random Forest > Gradient Boosting (93.7%) > SVM (87.9%) > Logistic Regression (85.1%)

Practical use.

The saved form (best_pv_classifier_3class.pkl) can be downloaded and applied directly to any new CSV file containing the same columns (Sample#, Current(A), Voltage(V)). The model is ready for physical operation in solar panel monitoring systems.

The only restriction.

Care should be taken when classifying partially covered boards with a few sheets, it is the most error-prone case.

4.9.Discussion of results:

Not long after gathering information from solar panels, a machine learning approach was used to sort out different types of failures. The outcomes highlighted how well the method works when keeping an eye on panel behavior and spotting problems. Voltage (V), current (I), along with power output (P) served as core signals showing whether the unit functioned properly. What stood out was the way these values revealed shifts in real-time operation.

Out of nowhere, steady voltage and current meant the board performed just fine, delivering output that matched standard operation. When things went wrong, though, shifts appeared instantly in the readings - partial shadows made current drop, dragging down power along with it. Sometimes, odd glitches twisted the voltage alone; other times, both voltage and current acted up at once.

Looking at how the machine learning method performed revealed clear links between data trends and system conditions. Splitting the information - most going to practice runs, some saved for later checks - helped measure progress fairly. Outcomes pointed to strong recognition when spotting broken versus working setups.

Later on, using an Arduino Uno board together with an SD card module made it possible to gather information nonstop and in order. That steady flow of details helped build up a store of records fit for teaching the model later, pushing its skills forward down the road.

Most findings show the suggested setup works well as an affordable way to watch solar panels and catch problems early. What stands out is how pairing machine learning with small computers can boost trust in solar setups. This mix also speeds up checkups and fixes without extra cost. Still, it's the quiet gains in upkeep speed that make it stick out. Not everything needs redesign when smarter tools join simple hardware.

4.10. Conclusion:

Here, hands-on work took shape through building a setup aimed at tracking how well solar panels perform. This setup pulled together electrical details - voltage, current, along with power levels - with help from an Arduino Uno board paired with an SD card module for saving information. Information gathered didn't just sit stored; it went through steps that shaped it into organized records. Those records then fed into forming a collection perfect for shaping up, later testing, a smart learning tool built to spot problems and sort them correctly.

Later on, the team used the classification model on gathered information. Splitting that data helped check how well it worked - one part trained the system, another tested it. This split made sure the method could tell regular operations apart from various failures. Outcomes pointed toward strong detection skills when spotting issues. Performance during sorting tasks met expected levels without surprise jumps or drops.

From the findings, combining machine learning with Arduino-powered monitors proves effective at spotting problems early in solar setups without high costs. This approach boosts system dependability while cutting downtime through timely alerts. Such outcomes open paths toward sharper diagnostic tools tailored for solar panels. Progress made here supports next-level designs that learn and adapt over time. The groundwork now exists to push further into intelligent monitoring methods.

General Conclusion



General Conclusion:

More solar panels in faraway places mean smarter ways are needed to watch them closely without high costs or constant fixes. When dust piles up, sand buries parts, or shade blocks light - common problems in the Algerian desert - power output drops sharply. These tough outdoor settings push engineers to build self-reliant tools that spot issues fast while surviving extreme heat and grit.

A budget-friendly smart monitoring setup for solar panels got built using Arduino Uno plus machine learning methods. Built around edge computing, it handles data gathering, analysis, saving, and spotting faults right on site. No need for constant web access or outside servers - everything runs by itself. That independence fits well with off-grid solar systems far from cities. Remote locations benefit since the tech works even when online links are weak or missing. Starting off, the work looked at what makes solar power setups hard to manage in Algeria, while pointing out why smart tracking and spotting problems early matters. After that came a deep dive into past studies and tools already around - things like internet-connected sensors, ways to stop equipment theft, signal methods for far-off places, along with how machines can learn to find faults in solar panels. It turned out most current fixes struggle because they lean too much on costly gear, constant online links, or heavy-duty computing needs.

From what we saw, the setup came together - mixing sensors for voltage and current, saving info locally on an SD card, while a smart model flagged faults. Data showed up during three scenarios: clear running, sandy panels, and leaves blocking parts of them. Once gathered, those numbers fed into different learners like Random Forest, then SVM, followed by Gradient Boosting, ending with Logistic Regression shaping results. Each run built understanding without repeating steps taken before. Patterns emerged only when inputs shifted slightly each time.

Results showed the method worked well. Notably, Random Forest stood out among models tested - accuracy hit 94.6% during validation, then held at 92.8% on unseen data. Instead of just spotting patterns, it clearly separated regular operations from faults without heavy processing needs. Even better, when loaded onto an Arduino Uno, the system ran smoothly, proving smart diagnostics can run directly on simple devices. Hardware limits didn't stop reliable performance. One big plus? It runs without the internet. Built-in storage keeps info safe on site. Faults show up right away, no delay. Runs by itself once set up. Costs very little to put into place. Works great for

tiny solar setups far from cities. Tough spots with poor service can rely on it. Independence matters when signals fade out.

Even though the outcomes look good, certain drawbacks still exist. Focusing only on a few types of faults, today's version uses mainly voltage and current data. To move forward, extra sensors might be added - things like sunlight intensity and heat readings. Wireless methods, say LoRa, could link components together. Stronger security features would help protect the setup. Newer learning models, possibly deeper networks, may boost performance too. Predictive upkeep functions have potential. Real-life data, broader and richer, might make the approach more solid overall.

This project shows how pairing small computers with learning algorithms builds solid, low-cost tools for smarter solar tracking. Not only does the design boost dependability and performance of sunlight power setups, it also adds durability into the mix. Far-off areas gain new possibilities through this setup, where basic needs meet advanced tech without complexity standing in the way. Progress here opens paths for greener grids that adapt quietly, yet effectively.

References



- [1] International Energy Agency (IEA). Trends in Photovoltaic Applications 2023. Paris: International Energy Agency; 2023.
- [2] Mekhilef S, Saidur R, Safari A. A review on solar energy in Algeria: Current status and future prospects. *Renew Sustain Energy Rev.* 2011;15(8):4252-4262.
- [3] Sarver T, Al-Qaraghuli A, Kazmerski LL. A comprehensive review of the impact of dust on the use of solar energy. *Renew Sustain Energy Rev.* 2013;22:698-733.
- [4] Kumar NM, Sudhakar M, Kothari MP. Machine learning based fault detection in photovoltaic systems: A review. *Renew Energy Focus.* 2017;21:1-7.
- [5] Chouder A, Silvestre S. Automatic supervision and fault detection in photovoltaic systems using machine learning and statistical methods. *Sol Energy.* 2013;91:1-13.
- [6] Yang H, Huang W, Zhang H. Digital twin and AI-based fault diagnosis for photovoltaic systems: A review. *IEEE Access.* 2022;10:1-15.
- [7] Kumar NS, Gupta A, Sharma R. IoT-based smart monitoring systems for photovoltaic applications. *Renew Energy Focus.* 2019;28:1-10.
- [8] Abdulrazzaq AA, et al. Design and implementation of embedded systems for solar photovoltaic monitoring. *IEEE Access.* 2020;8:1-12.
- [9] Hong T, Wang P, Liu X. Machine learning approaches for photovoltaic fault diagnosis and classification. *Sol Energy.* 2020;195:1-15.
- [10] Zhang Y, Zhao L, Wang J. Deep learning techniques for photovoltaic system fault detection using thermal imaging. *Appl Energy.* 2020;280.
- [11] Zadeh AH, Khan MA. Edge computing for IoT-based photovoltaic monitoring systems. *Sensors.* 2021;21(15).
- [12] Texas Instruments. ACS712 Current Sensor Datasheet. Dallas (TX): Texas Instruments; 2019.
- [13] Analog Devices. INA219 High-Side Current Shunt and Power Monitor Datasheet. Wilmington (MA): Analog Devices; 2020.
- [14] Kalay MS, Kilic B, Saglam S. Systematic review of the data acquisition and monitoring systems of photovoltaic panels and arrays. *Sol Energy.* 2022;244:47-64.
- [15] Jundi ZS, Al-Waeli AHA, Switzner NT. A systematic review of low-cost photovoltaic monitoring systems: Technologies, challenges, and opportunities. *Renew Sustain Energy Rev.* 2026;226.

- [16] López-Vargas A, Fuentes M, Vivar M. Current challenges for the advanced mass scale monitoring of Solar Home Systems: A review. *Renew Energy*. 2021;163:2098-2114.
- [17] Mellit A, Benghanem M, Kalogirou S, Massi Pavan A. An embedded system for remote monitoring and fault diagnosis of photovoltaic arrays using machine learning and the Internet of Things. *Renew Energy*. 2023;208:399-408.
- [18] Enphase Energy. Enphase Enlighten Monitoring Platform. Product documentation. Fremont (CA): Enphase Energy; 2024.
- [19] SolarEdge Technologies. mySolarEdge Monitoring Platform. Product documentation. Herzliya: SolarEdge Technologies; 2024.
- [20] Cardinale-Villalobos L, Murillo-Soto LD, Brenes R. Low-Cost IoT System Prototype to Detect Suboptimal Conditions in PV Arrays. In: *Smart Cities – 7th Ibero-American Congress (ICSC-CITIES 2024), Revised Selected Papers*. Cham: Springer; 2025. p.45-58.
- [21] Benghanem M, Mellit A. Remote monitoring of photovoltaic systems using wireless sensor networks: Case study in Algeria. *Energy Sustain Dev*. 2020;54:11-21.
- [22] Srisiri W, Le NT, Saleem MA, et al. Artificial intelligence-based fault classification on photovoltaic plants using a low-cost open-source IoT system. *Energy Rep*. 2024;11:233-245.
- [23] Ivanova I. Communication modules for data transmission in photovoltaic performance monitoring and analysis systems. *J Sci Appl Res*. 2024;27(1):113-119.
- [24] Taghezouit B, et al. Multi-sensor fault detection for PV systems under variable irradiance. *IEEE Trans Instrum Meas*. 2021;70:1-12.
- [25] Harrou F, et al. Anomaly detection in PV systems using machine learning. *IEEE Trans Sustain Energy*. 2021;12(3):1789-1799.
- [26] Nedaei A, Eskandari A, Aghaei M. Photovoltaic fault detection and classification: Reconsideration of classic machine learning and dataset shrinkage techniques for simplification. *Results Eng*. 2025;25:106356.
- [27] Mellit A, Massi Pavan A, Benghanem M. State of the art of artificial intelligence techniques for fault diagnosis in photovoltaic systems: A review. *Energies*. 2021;14(16):4905.
- [28] Ratsheola K, Setlhaolo D, Rasool A, Ali A, Mabunda NE. A hybrid artificial intelligence for fault detection and diagnosis of photovoltaic systems using autoencoders and random forests classifiers. *Eng*. 2025;6(10):254.

- [29] Aghaei M, et al. Autonomous intelligent monitoring of photovoltaic systems: An in-depth multidisciplinary review. *Prog Photovolt Res Appl*. 2024;33(3):381-409.
- [30] Bouziane H, et al. LoRa versus GSM for off-grid solar monitoring in the Algerian Sahara: Field performance evaluation. *Renew Energy Focus*. 2023;45:101-110.
- [31] Hameed BH, Kurnaz S, Ahmed SR. Efficient PV monitoring with LoRaWAN: Integrating IoT and smartphone application for cost-effective solutions. In: *Proceedings of the 2024 International Conference on Engineering and Emerging Technologies (ICEET)*. 2024.
- [32] Nath DC, Kundu I, Sharma A, et al. Internet of Things integrated with solar energy applications: A state-of-the-art review. *Environ Dev Sustain*. 2024;26:24597-24652.
- [33] Segovia Ramírez I, García Márquez FP, Parra Chaparro J. Convolutional neural networks and Internet of Things for fault detection by aerial monitoring of photovoltaic solar plants. *Measurement*. 2024;226:114861.
- [34] Başoğlu ME. Comprehensive review on distributed maximum power point tracking: Submodule level and module level MPPT strategies. *Sol Energy*. 2022;241:85-108.
- [35] Sahouane N, et al. Soiling losses in Adrar, Algeria: Experimental study of a 7 kWp PV system. *Energy Sustain Dev*. 2021;60:45-54.
- [36] Deline C, Hanson A, Stouth J, Sullivan C. Partial-Shading Assessment of Photovoltaic Installations via Module-Level Monitoring. *IEEE J Photovolt*. 2020;10(3):893-900.
- [37] Ahmad FF, Ghenai C, Bettayeb M. Maximum power point tracking and photovoltaic energy harvesting for Internet of Things: A comprehensive review. *Sustain Energy Technol Assess*. 2021;47:101430.
- [38] He J, et al. Hotspot formation and bypass diode activation under partial shading. *Prog Photovolt Res Appl*. 2023;31(5):456-470.
- [39] Rocha M, et al. Partial shading effects on PV module performance. *IEEE J Photovolt*. 2024;14(1):123-132.
- [40] Arduino SA. Arduino Uno Rev3 Product Reference Manual. Arduino Documentation; 2024.
- [41] Jundi ZS, Al-Waeli AHA, Switzner NT. A systematic review of low-cost photovoltaic monitoring systems: Technologies, challenges, and opportunities. *Renew Sustain Energy Rev*. 2026;226.
- [42] Kalay MS, Kilic B, Saglam S. Systematic review of the data acquisition and monitoring systems of photovoltaic panels and arrays. *Sol Energy*. 2022;244:47-64.

- [43] Arduino SA. Arduino Uno Rev3 Product Reference Manual. Arduino Documentation; 2024.
- [44] Mellit A, Benghanem M, Kalogirou S, Massi Pavan A. An embedded system for remote monitoring and fault diagnosis of photovoltaic arrays using machine learning and the Internet of Things. *Renew Energy*. 2023;208:399-408.
- [45] Cardinale-Villalobos L, Murillo-Soto LD, Brenes R. Low-Cost IoT System Prototype to Detect Suboptimal Conditions in PV Arrays. In: *Smart Cities – 7th Ibero-American Congress (ICSC-CITIES 2024), Revised Selected Papers*. Cham: Springer; 2025. p.45-58.
- [46] Enphase Energy. Enphase Enlighten Monitoring Platform. Product documentation. Fremont (CA): Enphase Energy; 2024.
- [47] SolarEdge Technologies. mySolarEdge Monitoring Platform. Product documentation. Herzliya: SolarEdge Technologies; 2024.
- [48] Nedaei A, Eskandari A, Aghaei M. Photovoltaic fault detection and classification: Reconsideration of classic machine learning and dataset shrinkage techniques for simplification. *Results Eng*. 2025;25:106356.
- [49] Ratsheola K, Setlhaolo D, Rasool A, Ali A, Mabunda NE. A hybrid artificial intelligence for fault detection and diagnosis of photovoltaic systems using autoencoders and random forests classifiers. *Eng*. 2025;6(10):254.
- [50] Srisiri W, Le NT, Saleem MA, et al. Artificial intelligence-based fault classification on photovoltaic plants using a low-cost open-source IoT system. *Energy Rep*. 2024;11:233-245.
- [51] Benghanem M, Mellit A. Remote monitoring of photovoltaic systems using wireless sensor networks: Case study in Algeria. *Energy Sustain Dev*. 2020;54:11-21.
- [52] López-Vargas A, Fuentes M, Vivar M. Current challenges for the advanced mass scale monitoring of Solar Home Systems: A review. *Renew Energy*. 2021;163:2098-2114.
- [53] Taghezouit B, et al. Multi-sensor fault detection for PV systems under variable irradiance. *IEEE Trans Instrum Meas*. 2021;70:1-12.
- [54] Harrou F, et al. Anomaly detection in PV systems using machine learning. *IEEE Trans Sustain Energy*. 2021;12(3):1789-1799.