DAYANANDA SAGAR COLLEGE OF ENGINEERING

(An Autonomous Institute affiliated to VTU, Belagavi - 590018)
Accredited by NBA, National Assessment & Accreditation Council (NAAC) with 'A' grade

Project Report on

PREDICTIVE MANAGEMENT OF PV AND DOMESTIC DC LOAD

Submitted in partial fulfillment for the award of degree of

BACHELOR OF ENGINEERING
IN
ELECTRICAL AND ELECTRONICS ENGINEERING

Submitted by

Aayush Basavesh	**1DS16EE001**
Adarsh D Shetty	**1DS16EE005**
Divya D A	**1DS16EE032**
Rachana A Bujurke	**1DS16EE072**

Under the Guidance of
Mr. Satish B A
Asst. Professor
Dept. of E&E Engg.
DSCE, Bengaluru

VISVESVARAYA TECHNOLOGICAL UNIVERSITY
JNANASANGAMA, BELAGAVI-590018
2019-2020

DAYANANDA SAGAR COLLEGE OF ENGINEERING

(An Autonomous Institute affiliated to VTU, Belagavi – 590018, Approved by AICTE & ISO 9001:2008 Certified)
Accredited by NBA, National Assessment & Accreditation Council (NAAC) with 'A' grade
Shavige Malleshwara Hills, Kumaraswamy Layout
Bengaluru-560078
2019-2020

DEPARTMENT OF ELECTRICAL & ELECTRONICS ENGINEERING

CERTIFICATE

Certified that the Project report entitled **" PREDICTIVE MANAGEMENT OF PV AND DOMESTIC DC LOAD**" carried out by **Aayush Basavesh** bearing a **USN: 1DS16EE001, Adarsh D Shetty** bearing a **USN: 1DS16EE005, Divya D A** bearing a **USN: 1DS16EE032** and **Rachana A Bujurke** bearing a **USN: 1DS16EE072,** bonafide students of **DAYANANDA SAGAR COLLEGE OF ENGINEERING**, an autonomous institution affiliated to VTU, Belagavi in partial fulfilment for the award of Degree of **Bachelor of Engineering in Electrical and Electronics Engineering** during the year **2019-2020**. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the report deposited in the departmental library. The Project report has been approved as it satisfies the academic requirements in respect of work prescribed for the Bachelor of Engineering Degree.

Signature of the Guide
Name
Designation
Dept. of E&E Engg.
DSCE, Bengaluru

Signature of the HOD
Dr. P. Usha
Professor & HOD
Dept. of E&E Engg.
DSCE, Bengaluru

ABSTRACT

The reliability and stability of a power system declines with the inclusion of a large proportion of renewable energy. As the need to predict solar PV output is essential, it is also important to solve the problem of peak demand. The aim of this project is to benchmark machine learning approaches in PV power prediction and load forecasting. Data is collected from source and load profiles with other meteorological parameters as inputs. Two machine learning techniques i.e. Regression and K-nearest Neighbours are assessed. The prediction results are compared in terms of training and test accuracy. Results obtained from these methods can significantly contribute to PV power management and domestic DC load usage.

Keywords: Renewable energy, PV power, Machine learning, forecasting, Regression, K-nearest Neighbours.

ACKNOWLEDGEMENT

We would like to take this opportunity to express our sincere gratitude to everyone who helped and guided us in completing this report.

We would like to express our immense gratitude to our **Professor & HOD**

Dr. P USHA, department of Electrical and Electronics Engineering, Dayananda Sagar College of Engineering for her constant support, motivation, and encouragement to come up with this work.

We express our warm thanks to our guide **Asst Professor Mr. SATISH B A,** department of Electrical and Electronics Engineering, Dayananda Sagar College of Engineering for his skillful guidance, constant supervision, timely suggestions and constructive criticism in successful completion of my seminar on time.

Finally, we also take the opportunity to thank all our staff members of the department of electrical and electronics engineering who have rendered their whole-hearted support at all time for successful completion of the seminar

INDEX

LIST OF FIGURES

CHAPTER 1

PREDICTIVE MANAGEMENT OF PV AND DOMESTIC DC LOAD

1.1 Introduction

Solar power is a clean and inexpensive energy source which can be harnessed easily from anywhere in the world due to its abundance. Over the past few years solar systems are being extensively used to power small loads typically installed on roof tops of buildings. On site solar power for businesses and non-profits with small scale utilities have a large number of customers in the power market. Residential owners can install solar systems and design their premises to take full advantage of solar technology. Subsidies provided by the government for the installation of solar panels also add to their favour. Several factors have to be considered before employing a PV system like location, weather conditions, capital cost and maintenance. However these measures do not always yield state of the art performance in recent times.

In the present scenario solar energy is directly fed to the grid without checking the status of the grid. Power generated from the distributed solar panels doesn't exceed the limit of penetration of the grid. However, in the future almost every house will be using DC loads. But the grid is not designed such as to integrate large amounts of varying solar power into it. So in order to face this issue and to control the excess power, utilities should have an estimate of how much amount of solar power will be added to the grid after domestic usage. This will assist the utility to make use of these resources to its maximum and maintain the stability of the grid. Predicting the power output of a solar panel also helps in optimal load scheduling by individuals, bidding in electricity markets etc. But the power output from a solar panel can change rapidly due to unexpected weather conditions. This makes forecasting the power output of a solar panel even more challenging.

Many maximum power point tracking (MPPT) algorithms have been introduced in recent years to maximize the generated PV power. These algorithms are not sufficiently robust because of fast-changing environmental conditions. In the soft computing-based MPPT category, the most popular techniques are Fuzzy Logic Control (FLC) [1], artificial neural network (ANN) [2], and other Computational Intelligence (CI) [3] methods.

The prediction of PV power includes certain modelling methods based on historical data [4]. In recent years. The prediction of PV power is similar to the load forecasting, and the research on the evaluation was put forward based on statistical analysis. The necessity of the evaluation of the load regularity has been expounded in [5].

PV power has randomness and volatility due to the light intensity, humidity, temperature, and so on. This uncertainty of PV output causes some adverse effects on the grid when it is connected on a large scale. Therefore, accurate PV power prediction is of great significance to the safe and economical operation of power systems. The PV output forecasting can be of different forecast horizons such as very short term, short term, medium term and long term PV output forecast.

Long term forecast (1 yr. to 10 yrs. ahead).

Medium term forecast (1 month to 1 yr. ahead).

Short term forecast (1 hr or several hrs ahead to 1 day or 1 week ahead).

Very short term forecast (1 min. to several min. ahead)

Short term PV output forecast can be utilized for automatic generation control (AGC), better unit dispatching, load scheduling and power plant operational management. It helps the grid to reduce the ancillary costs associated with weather dependency and ensure quality of energy. Reduction of the power system operational costs is a main factor to develop PV forecast models. PV output power forecast may also be part of smart grid (future generation power grids) energy management system along with load forecast.

Forecast model inputs also play a vital role in improving the prediction accuracy and model performance in terms of computational complexity and cost. Prediction error of forecast model is increased due to improper selection of forecast model inputs. Therefore, PV output forecast models are unable to correctly map all input variables as a forecast output due to poor selection of essential influential variables.

Many PV output forecast models used the historical PV output data as forecast model input. The historical PV output data may contain various spikes and non-stationary components due to uncertain meteorological conditions. As a result, these glitches in data will lead to higher forecast error due to improper training. The historical PV output data can be screened for smoothing. In addition, missing input data points in historical data will also play a role to increase the forecast error. Therefore, the forecast accuracy of the model can be considerably improved by input pre-processing.

1.2 Classification of forecast techniques

The figure 1, shows a broad way of classifying forecasting techniques used widely.

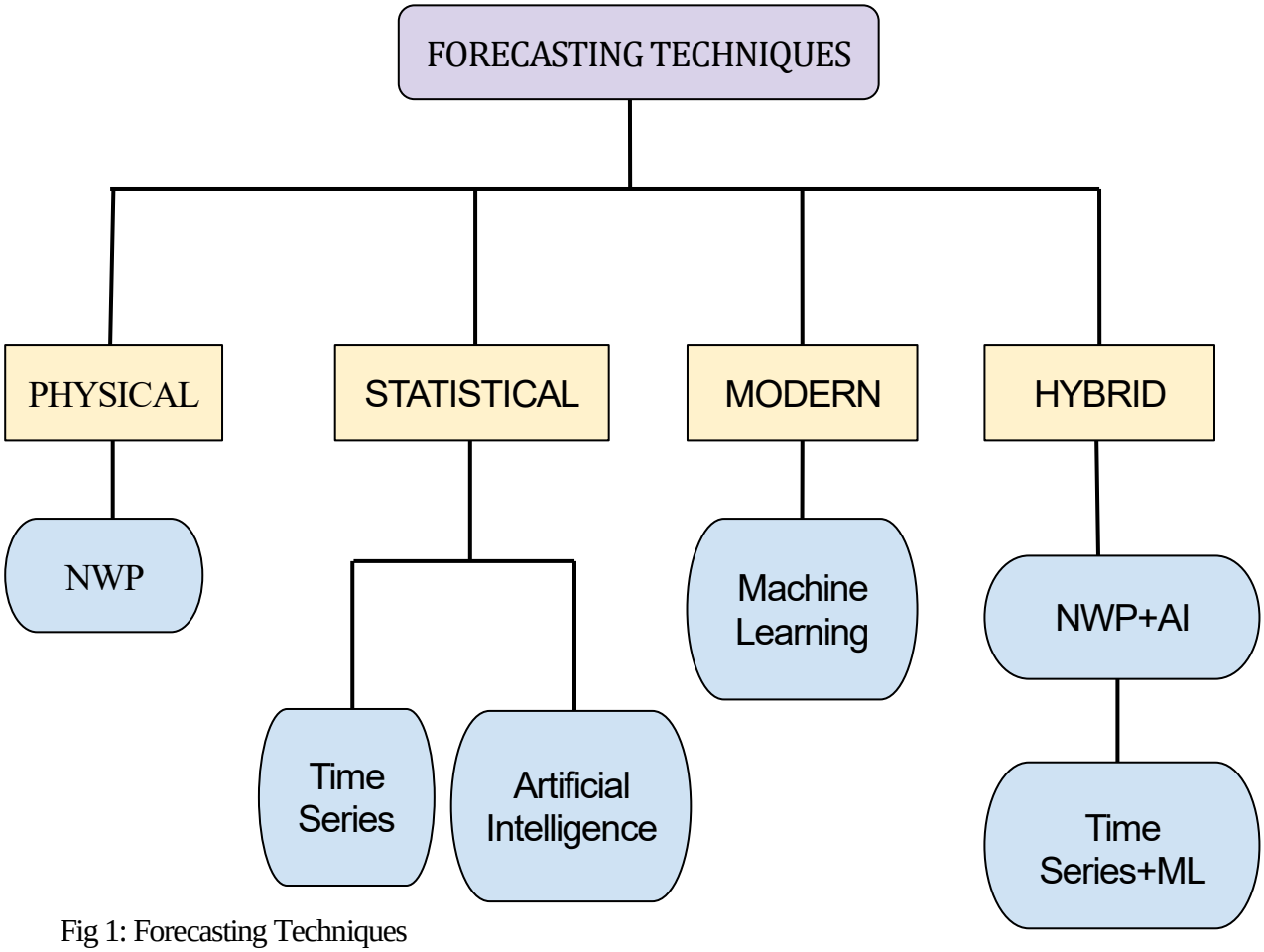


Fig 1: Forecasting Techniques

1.2.1 Physical models

A well-known physical model is numeric weather predictor (NWP). This predictor is based on a set of mathematical equations, which governs the physical state and dynamic motion of the atmosphere. These models are based on the PV plant characteristic such as location, orientation historical data and meteorological variables. It also depends on global horizontal irradiance (GHI), relative humidity, wind speed and direction along with PV models [6]. The forecast performance of a physical model is higher, when the weather conditions are stable [7]. However, the forecast accuracy is largely affected due to huge changes in meteorological variables.

1.2.2 Statistical techniques

Statistical techniques are dependent on the learning process of the forecast model with historical influential variables. The forecast model tries to minimize the network learning error by using the difference between network predicted PV output power and actual measured values. Forecast accuracy of statistical models relies on the length and quality of historical input data. Statistical

techniques have two sub groups named as time series and artificial intelligence (AI) based forecast models.

1.2.2.1 Time series models

In practice a suitable amount of data is fitted to a given time series and the corresponding parameters are predicted using the known data values. It comprises methods that attempt to study the nature of the series. In the time series forecasting method a suitable mathematical model is developed using past data, which holds the underlying data generating process for the series. This approach is particularly useful when there is no previous knowledge about the statistical pattern followed by the successive observations.

1.2.2.2 Artificial intelligence (AI)

Among the AI techniques, neural network (NN) is more popular and used as a power computational tool with higher degree of success since 1980. Generally, NN models produce higher forecast accuracy. It has the ability to capture sharp changes in the output with help of the intelligent training process of neural networks. An adaptive and robust NN training method can further improve the capability of the network to learn the complex relationship between input and output variables. Artificial neural network(ANN) based models map the input to model output without formulating the complex relationship between them. They attempt to model the simplest form of nonlinear behaviour of neurons. ANN model can map the input output relationship by using a feedback error system.

1.3 Machine Learning : A Closer Look

Machine Learning (ML) methods are being currently used as alternatives to statistical ones for time series forecasting. The objective of ML methods and statistical ones are the same. They both aim at increasing forecasting accuracy by minimizing some loss function, usually the sum of squared errors. Their difference lies in how such a minimization is done with ML methods utilizing nonlinear algorithms to do so while statistical ones use linear processes. ML methods are computationally more demanding than statistical ones, requiring greater dependence on computer science to be implemented, placing them at the intersection of statistics and computer science.

CHAPTER 2

LITERATURE SURVEY

In literature, different techniques have been used to predict photovoltaic (PV) power output. These approaches can be classified as direct and indirect techniques. Direct methods are based on the historical PV outputs and weather data as inputs . In indirect methods, solar radiation is predicted and these values are used to predict PV power output. Hiyama and Kitabayashi [8] developed an artificial neural network to predict maximum power generation from a PV system. They used solar irradiation, air temperature and wind velocity as input parameters. From the results, it was proved that the proposed method was able to provide better prediction than direct methods.

Yang et al. [9] proposed a weather-based hybrid model for a day-ahead hourly prediction of PV power. The authors used self-organising map (SOM) and learning vector quantization (LVQ) networks for arranging the historical PV power output data. Support vector regressions (SVR) was then employed to train the given dataset, and a fuzzy inference method was used to select an appropriate training model. It was found that the proposed method had better performance than the simple SVR and ANN.

It is evident from the literature that out of different methodologies applied to predict photovoltaic power output, the Artificial neural network is one of the most widely used methods. However, ANN requires the user to specify exact parameters of the model (e.g. no. of hidden layers, neurons, number of neurons in hidden layers, no. of training cells, etc). Also, ANN has some drawbacks while dealing with unexpected data. On the other hand, fuzzy logic cannot learn directly from historical data and may not be accurate as it is based on human knowledge [10]. Generating high end rules for fuzzy logic systems is a challenging task, and therefore best practices or human knowledge is often used to develop initial rules. A study on probabilistic forecasting techniques for PV power generation can be found in van der Meer et al. [11].

Successful implementation of regression analysis to generate probabilistic forecasts of solar energy has recently drawn a lot of attention. The same has been demonstrated by many authors in their work. Also load forecasting problems are dealt in similar ways as power forecasting.

Short Term Load Forecasting Methods

The system load to be forecasted is a random nonstationary process consisting of several of individual components. Therefore, the list of possible approaches to the load forecasting is wide. Some common methods are discussed below.

1. *Time-of-day method:* In the simplest form, a time-of day method takes the previous week's actual load pattern as a model to predict the load of the present week. Alternatively, a set of

load patterns is stored for typical weeks with different weather conditions. These are then heuristically combined to create the forecast.

2. *Stochastic time-series method:* This method has many attractive features. First, the theory of the models is well known hence it is easy to understand how the forecast is composed. The parameters of the model are easy to calculate; the estimate for the variance of the white noise component allows the confidence intervals for the forecasts to be created. The model identification is also relatively easy. Established methods for diagnostic checks are available. Moreover, the implementation of the model is not difficult. The weakness in the stochastic models is in their adaptability. In reality, load behaviour can change quite quickly in certain parts of the year. Another problem is the handling of the anomalous load conditions. If the load behaviours is abnormal on a certain day, this deviation from the normal conditions will be reflected in forecasts into the future.
3. *State-Space method:* There exist a number of variations of the state-space model. Some examples can be found in Camp and Ruiz[12]. According to Gross and Galiana, a potential advantage over Integrated Autoregressive Moving Average(ARIMA) models is the possibility to use prior information in parameter estimation via Bayesian techniques[13]. It is also pointed out that the advantages are not very clear and more experimental comparisons are needed.
4. *Expert systems:* Expert systems are heuristic models, which are usually able to take both quantitative and qualitative factors into account. A typical approach is to try to imitate the reasoning of a human operator. The method for a human expert to create the forecast is to search in the history database for a day that corresponds to the target day with regard to the day type, social factors, and weather factors. Then the load values of this similar day are taken as the basis for the forecast. An expert system can thereby be an automated version of this kind of a search process[14]
5. *Regression:* Regression models, normally, assume that the load can be divided into a standard load component and a component linearly dependent on some other variables. The most typical explanatory variables are weather factors. A typical regression model has been used by Ra¨sa¨nen and Ruusunen[15]. The load is divided into a repeated component and a temperature dependent component. The repeated component corresponds to the load of a certain hour in the average temperature of the modelling period. More complicated model variations have also been proposed. Some models use earlier load values as explanatory variables in addition to external variables. Regression methods are among the oldest methods suggested for load forecasting. They are quite immune to occasional disturbances in the measurements. The easy implementation is the strength of this method.

Chapter 3

COMPONENTS

3.1 ACS712 Current Sensor

Various devices need a different amount of current based on their functionality.

Every device has its own maximum peak current and voltage values. If the current of more than maximum peak value is provided, devices may burn and get damaged. Hence to avoid such conditions and to monitor the amount of current required current sensors are used. It is also used to measure the amount of current drawn by equipment connected to it.

There are two ways of sensing current :

Direct sensing:

In this type of sensing, Ohm's law is used to measure the voltage on the conductor whenever there is a flow of current. Some of the other examples are thermometers and barometers.

Indirect sensing:

When current flows through the conductor, a magnetic field is produced. The current is measured by calculating this magnetic field. Hall effect sensors, fiber optic current sensors are used to sense the magnetic field. ACS712 Current Sensor uses the Indirect Sensing method to calculate the current.

Working Principle:

ACS712 current sensor works on the principle of Hall effect which was discovered by Dr. Edwin Hall. According to this principle, when a current-carrying conductor is placed in the magnetic field, a voltage is generated perpendicular to the directions of both the current and the magnetic field.

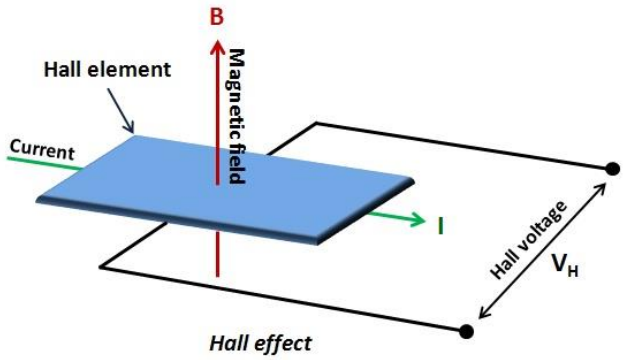


Fig 3.1Working of hall effect Sensor

The conductor is carrying a current (I) and is placed into a magnetic field (B) which is perpendicular to the direction of flow of current, due to the presence of Lorentz force the path of current flow has deviated and it is no more uniform from its actual path due to which a potential difference is created.

This voltage is known as Hall voltage and its value is in the order of microvolts. The Hall voltage is directly proportional to the magnitudes of I and B. Hence current can be calculated from the measured Hall voltage.

The accuracy of the device is decided by the proximity of the magnetic signal to the Hall sensor. This sensor can measure both AC and DC currents. It can measure current ranging from +5A to -5A, +20A to -20A, and +30A to -30A.

Interfacing this module with any Microcontrollers is easy since it outputs analog voltage ranging from 0V-5V.

In our project, we have used the ACS712 current sensor to measure the current drawn by the solar panel and the loads. It is connected in series with the panel and the output of this sensor is connected to Arduino UNO which is further used to calculate Power.

Applications of ACS712 Current Sensor:

1. ACS712 Current Sensor is used in Peak detection circuits
2. Rectification circuits
3. Circuits to increase the gain
4. Overcurrent fault latch
5. Motor control circuits
6. Load detection and management

3.2 Relay

The relay is an electromechanical switch which is used to safeguard the devices and also used as a protecting and controlling device for various circuit and electrical networks. It uses electrical contacts to either complete or interrupt a circuit. These relays can work with both AC and DC supply.

The two common types of relays are electromechanical relays and solid-state relays. In electromechanical relays, the magnetic force is used to either open or close the contacts. But in solid-state relays, there are no contacts and switching operation is totally electronic. Heavy-duty equipment requires the switching abilities of electromechanical relays. Solid State Relays uses electronic devices such as silicon controlled rectifiers.

An electromechanical relay consists of three terminals

- Common (COM)
- Normally closed (NC) contacts
- Normally opened (NO) contacts.

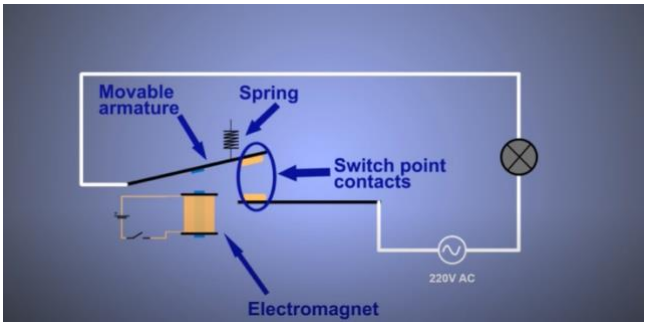


Fig. 3.2 Working of Electro-Mechanical Relay

There are two circuits the control circuit and load circuit.

Normally opened (NO) contacts:

The current starts flowing when the control switch is ON, it generates a magnetic field that attracts the armature and the load circuit is closed.

Normally opened contacts are generally used when a device must function in response to a specific signal.

Normally closed (NC) contacts:

It operates when the relay is not energized. When there is no voltage, it cannot attract the armature, thus the initial position is the armature connected in a normally closed position.

Normally closed contacts are generally used when a device must stop functioning in response to a specific signal.

Parts of the relay:

1. **Contacts:** These are the conducting parts that help the flow of current. They either open or close the circuit.
2. **Armature:** The armature opens and closes the contacts.
3. **Coil:** coil is wound around the core. When current flows through the coil, an electromagnetic field is generated.
4. **Frame:** Frame contains and supports the parts of the relay.

Applications of relays:

1. Control panels
2. Automation system
3. Commercial and Domestic buildings
4. Overcurrent detector
5. Overload detector

3.3 Voltage Divider

A voltage divider is a simple circuit that can produce a portion of its input voltage as an output. This circuit consists of a series of resistors or any passive elements and a voltage source, a voltage source is provided across these series resistors. This circuit is also called a potential divider. The source voltage gets divided at each resistor but the current value remains the same in series connection.

R_1 is the resistance close to supply voltage, R_2 is resistance near to the ground. V_{OUT} is the voltage drop across the resistor, R_2. It is the divider voltage which is the output.

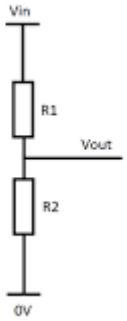


Fig.3.3 Voltage Divider

3.4 Arduino Uno

The Arduino Uno is an open-source microcontroller board based on the Microchip ATmega328P microcontroller and developed by Arduino.cc.

It is the periphery of ATmega328P microcontroller equipped with sets of digital and analog input/output pins, serial communication pins, power pins, PWM pins which can be used to interface various sensors, drivers, actuators with the microcontroller. In our project, we have used an Arduino UNO to interface Current sensor to measure the input current from the solar panel and relay to protect and control the circuit.

Pin description :

Pins have a standard rating ranging between 20mA to 40mA.

To limit the current within the safe limit internal pullup resistors are used but if a high amount of current flows to the Arduino then even these resistors cannot protect the board and it leads to damage of the board.

LED: Arduino has a built-in LED at pin 13.It Provides HIGH value to the pin will turn it ON and LOW will turn it OFF.

Vin: Vin is used to providing supply voltage. It provides an input voltage to the Arduino board

5V: 5V pin is used to provide an output voltage. Arduino can be powered up using either USB, Vin pin of the board or DC power jack. USB supports voltage of 5V, Vin and Power Jack supplies a voltage ranging between 7V to 20V.

GND: These are ground pins. There are 3 GND pins in Arduino UNO

Reset: This button is used to restart the program from the beginning. Arduino has the ability to reset using programming in IDE.

P*WM:* Pins 3,5,6,9,10, 11 are marked as PWM. These pins are configured to provide an 8-bit output PWM.

Serial Communication: Serial Communication is provided by 2 pins Rx and Tx ie Pin 0 and Pin1.

External Interrupts: Pin 2 and 3 are used for providing external interrupts.

Features of Arduino UNO:

- Arduino Uno consists of USB interface i.e. USB port is added to the Arduino board to interface Arduino with a computer to set up the serial communication
- 14 digital I/O pins
- 6 analog pins marked as A0 to A5 has a resolution of 10bits
- Atmega328 microcontroller.
- It also facilitates serial communication using Tx and Rx pins

- The ATmega328P microcontroller has a number of features like timer, PWM modulation, counters, interrupts, and based on a 16MHz clock
- The Reset pin helps to clear the memory of the Arduino and it takes back the running program in the initial stage. This pin is useful when the board hangs up and by pressing this button it starts the program right from the beginning.
- 13KB of flash memory is used to store the code
- A voltage source of 5V is required to turn on the board which can be provided using a USB port or external adapter
- It also consists of a voltage regulator which regulates the incoming voltage and keeps the board within safe permissible limit when it is connected to external devices or circuits

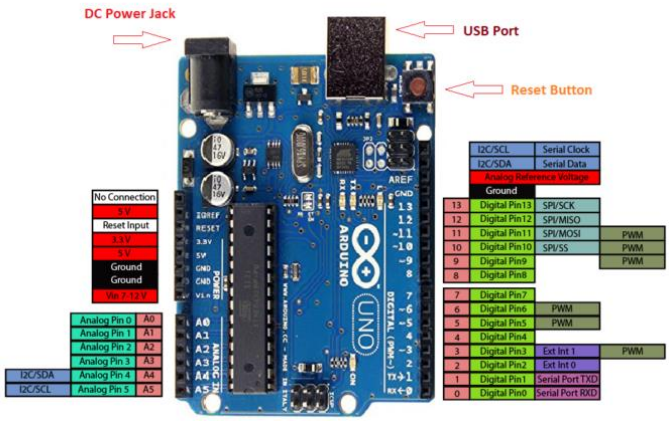


Fig. 3.4 Arduino Uno pin map

3.5 Raspberry Pi

The Raspberry Pi is a series of small single-board computers developed in the United Kingdom by the Raspberry Pi Foundation. The Raspberry Pi is a computer that runs Linux, it provides a set of GPIO (general purpose input/output) pins that allow various devices like sensors, electronic components and allows to explore the Internet of Things (IoT).

There are three generations of Raspberry Pi: Pi 1, Pi 2, and Pi 3, and there are a Model A and a Model B for these generations.

The SD card inserted in the slot on the Raspberry Pi is used as the hard drive or external storage. The monitor can be interfaced using HDMI port.

- It consists of ARM CPU/GPU it is a Broadcom BCM2835 System on a Chip (SoC) that's made up of an ARM central processing unit (CPU) and a Video core 4 graphics processing unit (GPU). GPU is used to handles graphics output and The CPU handles all the computations

- GPIO is the exposed general-purpose input/output pins which are used to interface external devices.
- An RCA jack is used to connect TV and other output devices.
- Audio out is a 3.55-millimeter jack used for connecting audio output devices like headphones or speakers etc.
- A USB port is the common port for peripheral devices like a keyboard or mouse to connect with Pi
- HDMI cable is used to interface high-definition TV or any other devices which use an HDMI port
- Raspberry Pi can be powered through Micro USB port
- For wired network access, Ethernet connectors are used. This facility is available only with Model B
- The Broadcom chip has 256 MB of RAM, which runs at 700 MHz and includes a 1080p-capable GPU.

3.6 Solar Panel

Solar panels are used to convert light energy into electricity that can be used to power electrical loads. Solar panels are composed of several solar cells which are again composed of layers of silicon.

A Solar panel is an assembly of photovoltaic cells mounted in a frame. Photo-voltaic cells use sunlight as a source of energy and generate direct current electricity. The resulting energy generated from photons striking the surface of the solar panel allows electrons to knock out of their atomic orbits and they are released into the electric field generated by the solar cells. This process is known as the Photovoltaic Effect. Each panel produces a very small amount of energy but can be connected together with other panels to produce higher amounts of energy such a system is known as a solar array. The solar panel(or array) produces energy in DC. Most of the electronic devices use DC current but the electrical utility grid provides (and requires) alternating current. Hence an inverter is used to convert the DC to AC.

A photovoltaic (PV) system is composed of one or more solar panels combined with an inverter and other electrical and mechanical hardware.

PV systems can vary in size from small portable systems to large utility-scale generation plants.

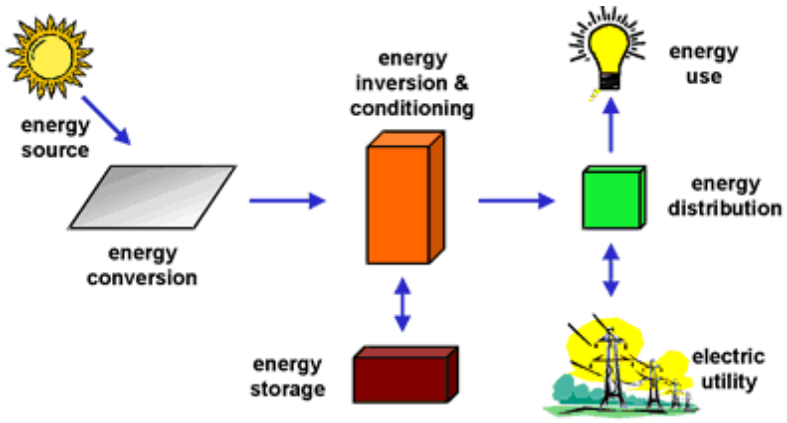


Fig. 3.5 Solar power Utilisation

Components of PV system:

1. Solar panel
2. Inverter
3. Battery or energy storage system
4. Converter
5. Load

PV array: PV array is the combinations of multiple solar panels which are used to convert solar energy to electrical energy

Battery: They are used to store the electrical energy produced by PV array during sunlight hours and supply it to electrical loads whenever needed(night or cloudy days). Batteries are used in a solar tracking system to track MPPT to operate the PV system at MPP in order to provide electrical loads with stable and maximum voltages. Most commonly used battery in PV systems are deep-cycle lead-acid batteries

Charge controller: batteries cause losses in the PV system due to insufficient battery maintenance and rapid recharging time. Hence a charge controller is used to prevent the battery from overcharging and over-discharging.

Inverter: PV array produces DC output. Most of the electronic devices operate with DC supply like Mobile, TV, etc but the electrical utility grid operates with AC. Hence an inverter is used to convert DC to AC.

There are two types of PV system:

1. *Grid-connected photovoltaic systems :*

Grid-connected photovoltaic systems consist of PV arrays that are connected to the grid through a power conditioning unit (PCU) and are designed to operate in parallel with the electric utility grid. PCU consists of the MPPT, the inverter, the grid interface, and BMS needed for efficient system performance.

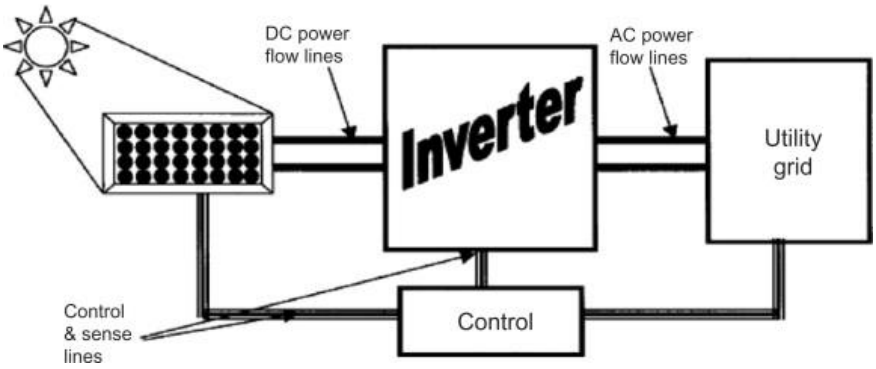


Fig. 3.6 Schematic of Grid-connected photovoltaic systems

2. *Standalone PV system:*

Stand-alone photovoltaic systems are designed to operate independent of the electric utility grid and are available in different sizes to supply certain DC and/or AC electrical loads.

- *Direct-coupled system:* Standalone PV system designed without battery storage they are known as a direct-coupled system, and DC output of a photovoltaic module or array is directly connected to a DC load. There is no energy storage battery system in direct-coupled systems hence load operates only during sunlight hours. They are suitable in applications such as water pumps, circulation pumps, and ventilation fans

Fig. 3.7 Schematic of Direct-coupled system

- *Standalone system with battery storage:* In many stand-alone PV systems, batteries are used for energy storage. They can power both DC and AC loads. They generally include a charge controller, batteries, inverters, and PV arrays.

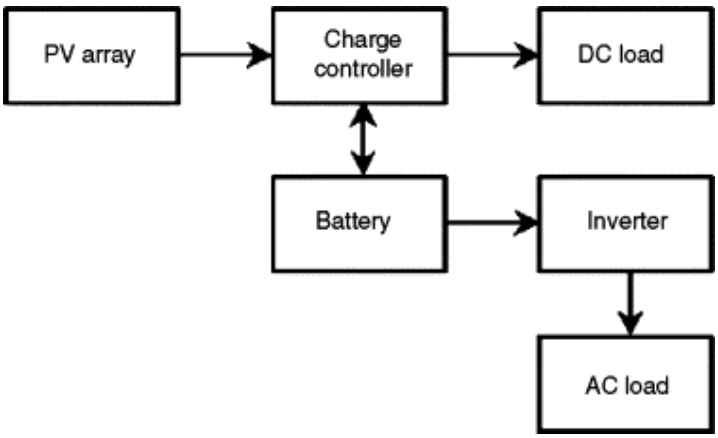


Fig. 3.8 Schematic Standalone system with battery storage

Fig. 3.9 Panel and Specifications of solar panel used

3.7 *Buck Converter*

A Buck converter is a DC to DC electronic converter. The output voltage is transformed to a voltage level less than the input voltage. It is also called a step-down converter. Since in step down transformer the output voltage is step down to voltage level than the input voltage, buck converter analogous to step down transformer hence it is called step down converter.

This converter is used to step down the voltage from the solar panel so that the devices like Arduino will be safeguarded.

According to the law of conservation of energy, the input power has to be equal to output power assuming no losses in the circuit.

$$\text{Input Power } (P_{in}) = \text{Output Power } (P_{out})$$

In a buck converter $V_{in} > V_{out}$, hence the output current will be greater than the input current. Therefore in the buck converter $V_{in} > V_{out}$ and $I_{in} < I_{out}$

Working Principle of Buck converter:

The inductor in the input circuit resists change in input current. The inductor stores energy in the form of magnetic energy when the switch is ON and when the switch is closed it discharges. The value capacitor in the output circuit is very large such that the time constant of the RC circuit in the output stage is high. When the time constant is large compared to the switching period a constant output voltage Vo(t) = Vo(constant) is ensured.

Modes of operation of Buck converter

There are two modes operation of Buck converter

1. **Continuous conduction mode**

In this mode, the current through induction never becomes zero. Before the start of the switching cycle, the inductor partially discharges.

2. **Discontinuous conduction mode**

In this mode, the current through induction becomes zero. At the end of the switching cycle, the inductor is completely discharged.

In our project, we have designed our own buck converter using Autodesk Eagle software, the schematic and board design of which are shown below.

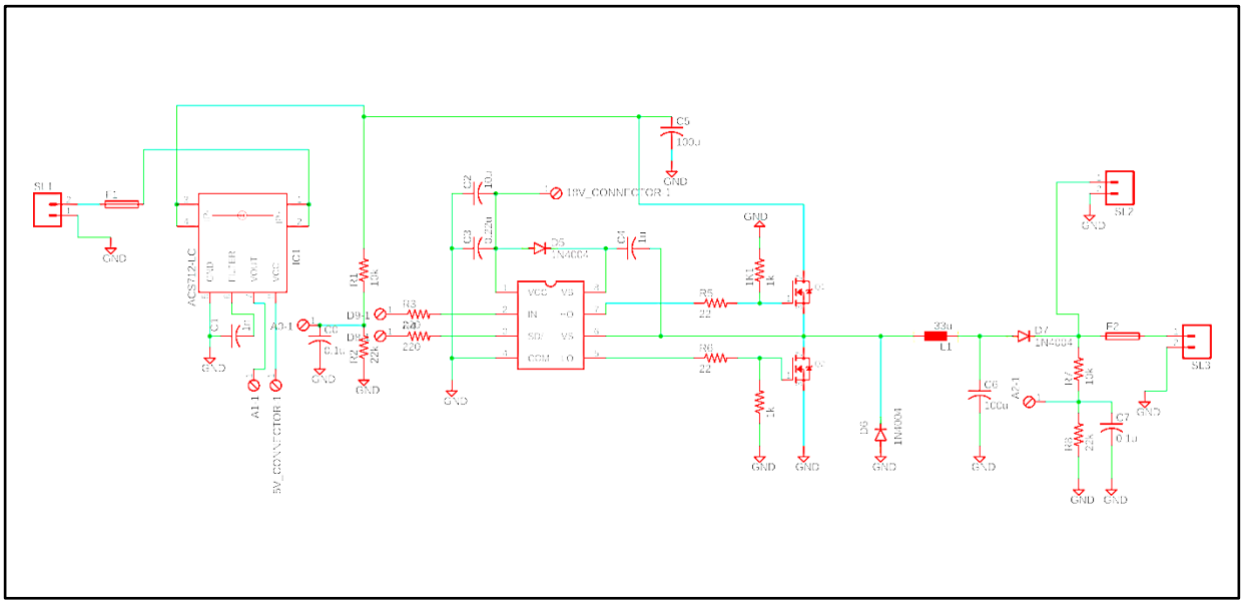


Fig. 3.10 Schematic of buck-converter designed in Eagle CAD

Before manufacturing the PCB, the schematics were tested on a breadboard and the output was observed on an oscilloscope.

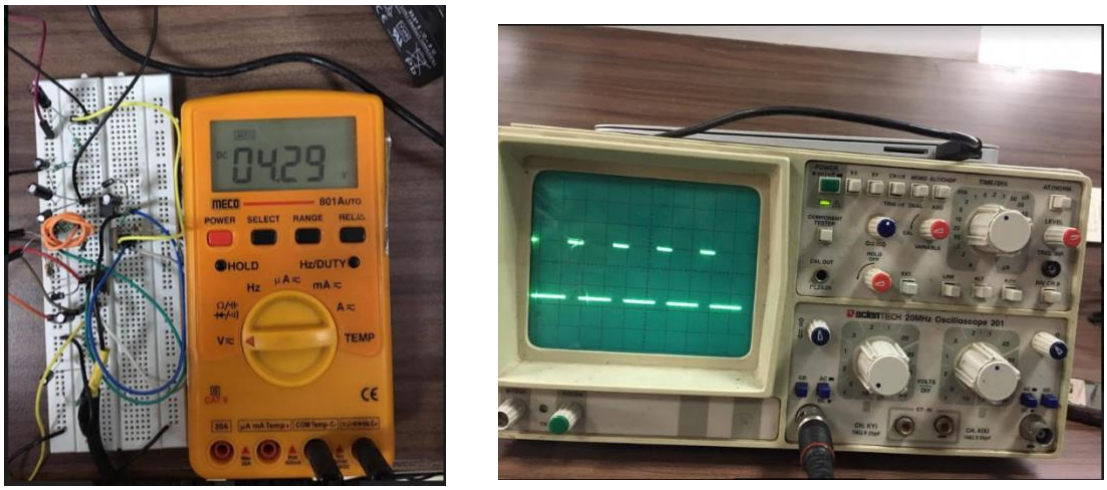


Fig. 3.11 Testing of buck converter before manufacturing

Chapter 4

HARDWARE SETUP

We have two different hardware setups to collect the solar panel power generation data and the domestic load usage data.

4.1 Solar Panel Data Collector

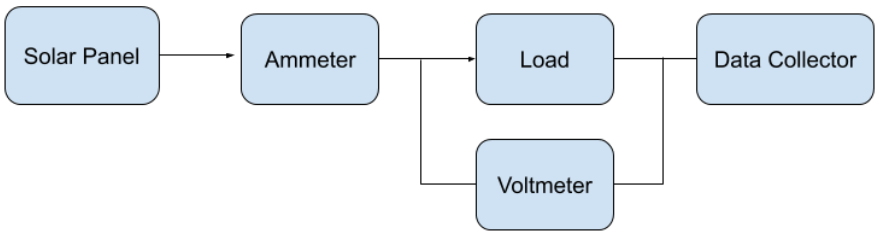


Fig. 4.1 Block diagram of Solar Panel Data Collector

To have a good accuracy while predicting future power generating capabilities, it is important to have good data over a prolonged time period. Collecting data manually over a course of 1 year tends to become tedious and cannot be done for fixed interval of time

Hence, we developed a circuit to help us collect data continuously over a period of 1 year with minimal errors and deviations.

A timer circuit (RTC) was initially used to collect data every 3 minutes and store in to an SD card in comma separated value (.csv) format. Due to an issue of redundancy errors, the RTC module was eliminated and a Raspberry Pi 3 B+ was introduced into the circuit to monitor and collect the data.

The solar panel was connected to a load through a relay circuit. A digital ammeter (ACS712) was connected in series and a voltage divider circuit was connected in parallel to the load. The relay helped in toggling between open and closed circuits for voltage and current value collection respectively.

The Arduino was connected to the Raspberry Pi serially, and all the data collected by the Arduino was sent immediately to the Raspberry Pi.

A python script was run continuously on the Raspberry Pi to read, format and store the data in a comma Separated value (.csv) file format. The stored data then periodically got uploaded to a git server which allowed monitoring of the data remotely.

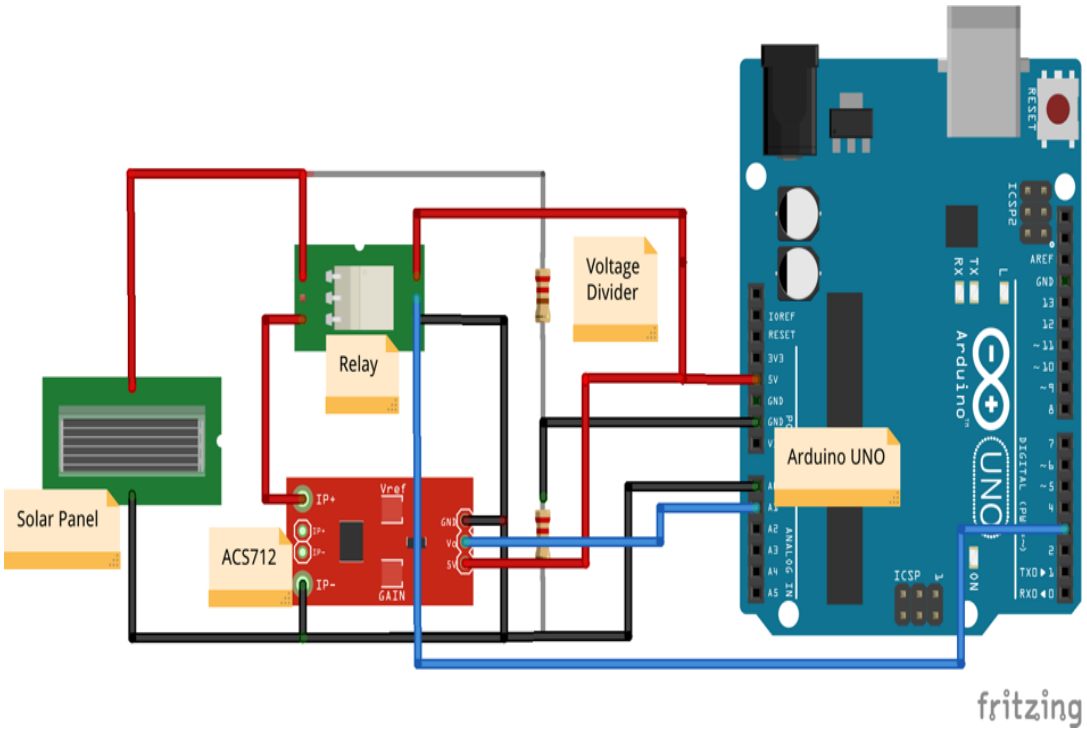


Fig. 4.2 Circuit Diagram of Solar Panel data Collector

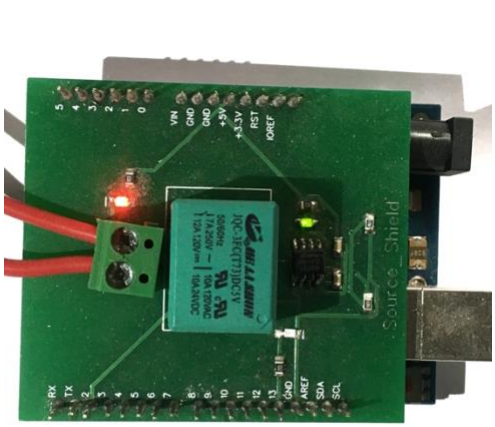


Fig. 4.3 Solar Panel Data Collector Setup

4.2 Domestic Load Usage Replicator

To be able to predict the domestic load usage, like the solar panel data, it is necessary collect the domestic load usage data as well. To collect the usage data from an actual house, it is quite tedious and gruesome to attach a power meter to each appliance drawing power.

To ease the process of collecting the data, a Load usage replicator was developed with the help of

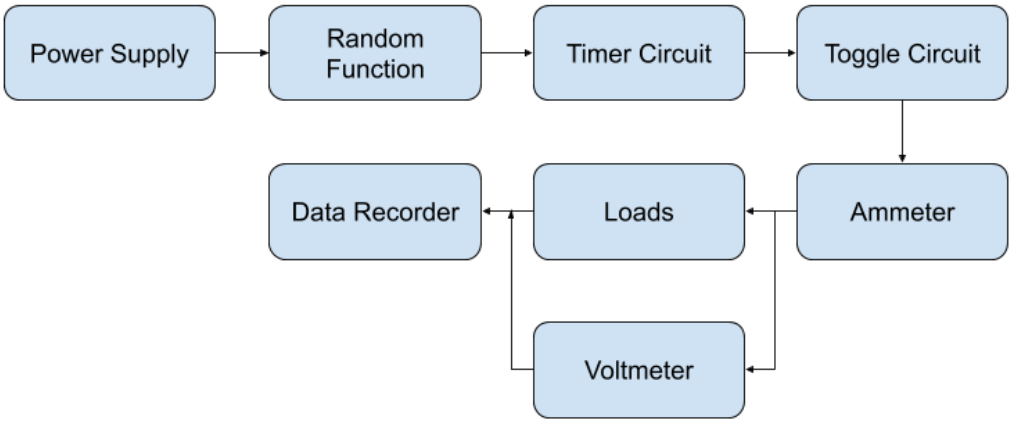


Fig. 4.4 Block diagram of Domestic Load usage replicator

high wattage bulbs and regular lighting lamps. To dissipate the heat generated by the high wattage bulbs, a heat sink was placed in contact with the bulbs. A 12V SMPS was deployed to power the active loads.

A random function was created to replicate the load usage through the day. The function was given different parameters throughout the week to simulate the actual presence of people and their plausible power consumption. The data was collected every minute, and was stored on an SD card in a comma separated value (.csv) file format. Periodically, the SD card was removed and the data stored in the same was uploaded to a git server to be access remotely.

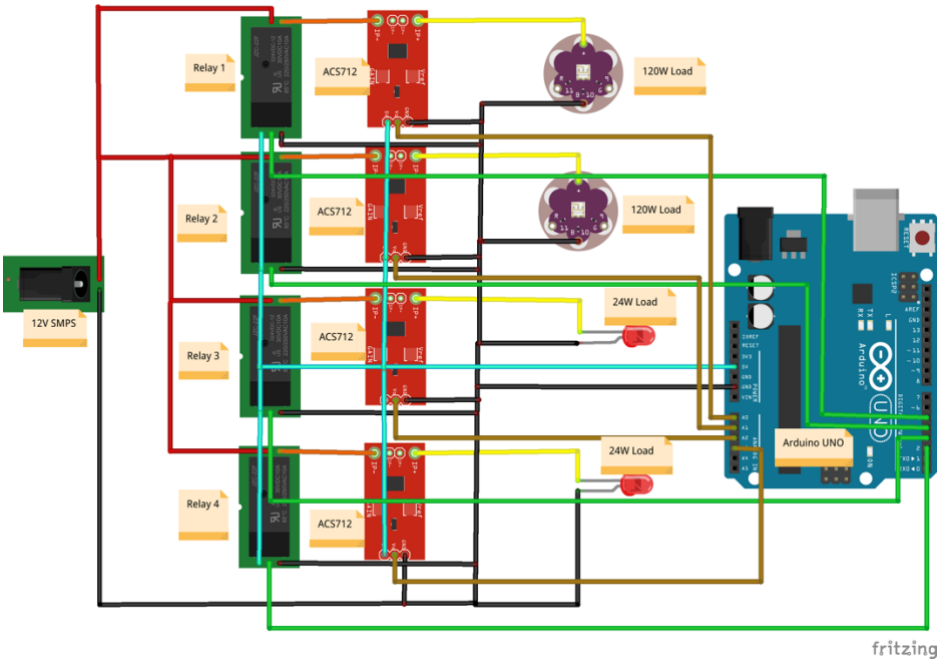


Fig. 4.5 Circuit diagram of Domestic Load usage replicator

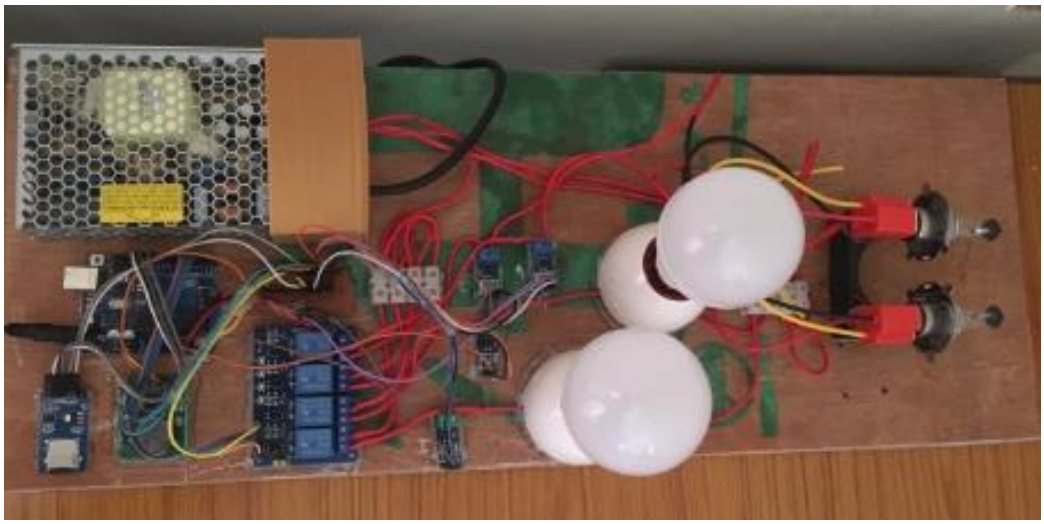


Fig. 4.6 Domestic Load Usage Replicator Setup

Chapter 5

MACHINE LEARNING TECHNIQUES USED

5.*1 Introduction to Machine Learning*

Machine learning systems learn how to combine various inputs to produce useful predictions data which has never been seen before. It is the study of algorithms which improve automatically through experience. It builds a mathematical model based on sample data, or otherwise known as training data, in order to make predictions or decisions without being exclusively programmed to[16]. These machine learning algorithms are used in a wide variety of applications such as email spam detecting, computer vision and other areas where it is tough to develop conventional algorithms to perform the needed task. Here in our project, we are employing different techniques or algorithms of machine learning to predict the output power of the solar panels.

The approaches based on the algorithms depending on the nature of signal or feedback to the learning system were classified into supervised learning, unsupervised learning and reinforcement learning[16]. The very objective of machine learning is to generalize from its experience. Generalization in this context is the ability of the machine learning algorithm to perform accurately on new and unseen examples after having learning through previously available data.

The computational analysis of machine learning algorithms and their performance is a branch of theoretical computer science known as computational learning theory[17]. Because training sets are finite and the future is uncertain, learning theory usually does not yield guarantees of the performance of algorithms. Instead, probabilistic bounds on the performance are quite common. The bias–variance decomposition is one way to quantify generalization error.

For the best performance in the context of generalization, the complexity of the hypothesis should match the complexity of the function underlying the data. If the hypothesis is less complex than the function, then the model has under fitted the data[17]. If the complexity of the model is increased in response, then the training error decreases. But if the hypothesis is too complex, then the model is subject to over fitting and generalization will be poorer.

In addition to performance bounds, learning theorists study the time complexity and feasibility of learning. In computational learning theory, a computation is considered feasible if it can be done in polynomial time. There are two kinds of time complexity results[18]. Positive results show that a certain class of functions can be learned in polynomial time. Negative results show that certain classes cannot be learned in polynomial time.

5.2 Key Machine Learning Terminology

LABELS:

Labels are something we are predicting. In this case we are predicting the output of the solar array. It is usually the y variable in simple linear regression. The label can usually be anything for which we want a prediction of its future. A few examples could be prices of things like rice, or to predict the type of flower shown in the picture.

FEATURES:

An input variable is known as a feature. It is the x variable in a simple regression. There might be one or more features depending upon the complexity of the machine learning project. Here in our project we have multiple parameters or features on which the power output depends. The features are represented by x1,x2…xN.

The features in a spam detector could be along the lines of the following:

1. Number of words in the email
2. The sender's mail ID
3. The part of the day when the mail was sent
4. Emails containing phrases like "Lose fat in 7 days"

EXAMPLES:

It is a particular occurrence of data. There are two types of Examples:

1. Labelled examples
2. Unlabeled examples

An example which includes both feature(s) and the label is known as labelled example. These types of examples are used to train the model.

An example which contains just features but not the label is known as unlabeled example.

We train our model with labelled examples and then we make use of that model to predict the label on unlabeled examples. In our project, the unlabeled examples are the days for which we have not predicted the power output yet and labelled examples will be the output we have already measured.

MODELS:

It defines the relationship between features and label. The two phases of a model's life are as follows:

1. Training: This means creating or learning the model. In this phase you introduce the model to labelled examples and the model picks up the relationships between the various features and the label.
2. Inference: This refers to applying the trained to unlabeled examples. You use the trained model to make predictions.

REGRESSION:

Continuous values are predicted with the help of regression models. It can be used to predict things such as power output, price of a house and so on.

CLASSIFICATION:

A classification model is used to predict values which are discrete. It can be used to make predictions such as if the image is that of a dog's or a cat's and so on.

TRAINING SET AND TEST SET:

The training set of data is a part of your data on which the model gets trained. The model will learn to predict the dependent variable with the variables which are independent. The test set is used to evaluate the model to check whether the model manages to predict the output correctly. The test set is usually the complimentary subset from the training set.

SUPERVISED LEARNING:

Here, in our project we use supervised learning. In this, we are given a data set and we know the relationship between the input and output and therefore we know how our output should look. These problems are of types: Regression and Classification problems. These algorithms build a model of a set of data that contains both the input and the outputs desired. The data is known as training data. Each training example has one or more inputs and the desired output, also known as a supervisory signal. In the mathematical model, each training example is represented by an array or vector, sometimes called a feature vector, and the training data is represented by a matrix. Through iterative optimization of an objective function, supervised learning algorithms learn a function that can be used to predict the output associated with new inputs. An optimal function will allow the algorithm to correctly determine the output for inputs that were not a part of the training data. An algorithm that improves the accuracy of its outputs or predictions over time is said to have learned to perform that task.

UNSUPERVISED LEARNING:

In this type of learning, we have no idea how the results should look. We don't know the effect of each variables in the data. An example of unsupervised learning would be to take a collection of a million samples of genes and divide them into groups and then predict what decides factors such as lifespan, diseases and so on. Cluster analysis is the assignment of a set of observations into subsets (called clusters) so that observations within the same cluster are similar according to one or more predesignated criteria, while observations drawn from different clusters are dissimilar[19]. Different clustering techniques make different assumptions on the structure of the data, often defined by some similarity metric and evaluated, for example, by internal compactness, or the similarity between

members of the same cluster, and separation, the difference between clusters. Other methods are based on estimated density and graph connectivity.

REINFORCEMENT LEARNING:

In this type of learning the computer programs needs to interact with a dynamic environment where it is required to perform a certain goal[19]. This goal may vary according to the model. Few such goals maybe driving a car or playing a game online. As it navigates the problem space, the feedback is provided to the program which is analogous to rewards which it tries to maximise.

SEMI-SUPERVISED LEARNING:

This category falls between supervised learning which has completely labelled training data and unsupervised learning which has no labelled training data[20]. Some of the training examples sometimes are missing training labels but still many researchers claim that unlabeled data used with a small amount of labelled data does produce a considerable improvement in the accuracy at which the machine learning model learns.

5.3 K-Nearest Neighbors Algorithm

The KNN algorithm or the k-nearest neighbours algorithm is a method very often used to solve problems on classification and regression. In both classification and regression cases, the input for the model consists of the k closest training examples which are in the feature space. Depending on whether k-NN is used for classification or regression, the output changes. In k-NN the output is the property value for the object whose value is the average values of k nearest neighbours.

KNN is a type of lazy learning or instance based learning in which the function is only approximated locally[21]. Weights are assigned to the contribution of neighbours in both classification and regression, such that the nearer neighbours contribute more to the average than the ones which are distant. A common weighing scheme is where the neighbours are given a weight of 1/d where d is the distance to the neighbour.

5.3.1 ALGORITHM

The training examples used in KNN are vectors in a multidimensional feature space, each with a class label. The training phase of the KNN algorithm consists of storing the feature vectors and also the class labels of the training samples. K is a user defined constant which is an unbalanced vector in the classification phase. It is classified by assigning the label which occurs the most among the k training samples nearest to that test point[22]. One of the commonly used distance metric for continuous variables is known as Euclidean distance. For problems such as text classification, a metric known as overlap metric can be used. The classification accuracy of KNN can be often improved significantly with the usage of specialized algorithms.

5.3.2 PARAMETER SELECTION

The data decides the best value of k that can be used. Larger values of k reduces the effect of noise, but the boundaries between classes becomes less distinct. Various heuristic methods can be used to select a good value of k[21]. The nearest neighbour algorithm is a case where the class is predicted to be the class of the closest training sample, that is, when the value of k is one. The presence of noisy features or irrelevant features can severely degrade the accuracy of the KNN algorithm. The accuracy can be affected if the feature scales are not consistent with their importance too. A lot of effort has been put into improving classification by scaling or selecting features. Evolutionary algorithms is a popular approach to optimize feature scaling.

In binary classification problems, the value of k is usually set as an odd number so as to avoid tied votes. Bootstrap method is one way which is often used to choose the empirically optimal k in this setting.

5.3.3 PROPERTIES

The original version of the KNN algorithm can be easily implemented by computing the distances from the test example to all the stored examples. But this gets tedious for larger training sets. As an alternative, using an approximate nearest neighbour search algorithm makes KNN easier to be used for large data sets.

Many nearest neighbour search algorithms have been tried over the years usually trying to reduce the number of distance evaluations that have to be performed. KNN produces strong consistency results. Various improvements to the speed of KNN can be made by using proximity graphs.

5.3.4 METRIC LEARNING

The performance of KNN classification can be significantly improved through supervised metric learning. Few of the popular algorithms are neighbourhood components analysis and large margin nearest neighbour. These supervised metric learning algorithms make use of label information to learn a new metric also known as pseudo metric.

5.3.5 FEATURE EXTRACTION

When the input data to the algorithm is too big to be processed and redundant, then the input data can be transformed into a reduced representation set of features vector[23]. This transforming of input data into a set of features is called feature extraction. If these features are carefully chosen then it is expected that it will extract the relevant information from the original input data in order to perform the desired task at hand using this reduced representation instead of the original full sized data. This feature extraction is performed on the original data also known as raw data before applying the KNN algorithm on the transformed data in feature space.

5.3.6 DIMENSION REDUCTION

When the number of dimensions are higher than 10, dimension reduction is usually performed before applying the KNN algorithm. This is done in order to avoid the effects of the curse of dimensionality which means that the Euclidean distance is not helpful in high dimensions because all vectors are almost equidistant to the search query vector[24]. Feature extraction and dimension reduction is usually combined in one step using the principal component analysis (PCA), LDA or CCA as a pre-processing step which is followed by clustering by KNN on feature vectors in reduced dimension space. This process is known as low dimensional embedding in machine learning.

5.3.7 DATA REDUCTION

This is one of the most important problems when there is a huge data set. More often than not, only some of the data points are required for accurate classification. Such data are called the prototypes and can be found as following:

1. Select the class outliers , that is, the training data that are not classified correctly by KNN for a given K value.

2. Separate the rest of the data into two sets namely

i) The prototypes which are used for classification decisions

ii) Absorbed points that can be classified correctly by KNN using prototypes

5.4 Linear Regression

The approach to modelling the relationship between a dependent variable and one or more independent variables is known as linear regression. When there is only one independent variable or explanatory variable, then it is known as simple linear regression. When there are multiple explanatory variables, then it is known as multiple linear regression[25]. The relationships are modelled using linear predictor functions where the model parameters are unknown and are estimated from the data. These type of models are called linear models. Linear regression, like all other forms of regression analysis, focuses on the conditional probability distribution of the response given the values of the predictors rather than on the joint probability distribution of all the variables. Linear regression was one of the first type of regression analysis to be studied intensely and be used in practical applications.

The practical used of linear regression broadly can be classified into two categories:

1. When the goal is forecasting, reduction of error or prediction, linear regression is usually used to fit a predictive model to a data set of values of dependent and independent variables. In our project too, since we are focusing on prediction of power, we use linear regression[25]. If additional values of the independent variables are collected after developing such a model without an accompanying response value, the already fitted model is used to make a prediction of the response.

2. When the goal is to determine the variation or change in the dependent variable that can be attributed to the change in the independent variable, linear regression is used. In solar prediction, the change in the power output can be explained with respect to the changes in parameters like temperature, time of the day and so on. In this type, the strength of relationship between different independent variables and dependent variables can be figured out using linear regression.

Linear regression models are fitted using an approach known as least squares but other ways can also be used. The least squares approach can also be used to fit nonlinear models too.

ASSUMPTIONS MADE IN LINEAR REGRESSION:

Linear regression models which use standard estimation techniques make a few assumptions about the dependent and independent variables and the relationship between the two[26]. A large number of extensions have been developed that allow each of these assumptions to be reduced and in rare cases they are even eliminated. Usually these extensions used make the process of estimation a lot more complex and consumes a lot of time. These may also require additional data to produce a precise model.

The few of the numerous assumptions made by standard linear regression models are as follows:

1. Weak exogeneity: This considers the predictor variables x as fixed values rather than random variables. This essentially means that the predictor variables are error free, that is, they are not contaminated with measurement errors. Even though this is not a realistic assumption, dropping this assumption leads to significantly more difficult errors-in-variables models[27].

2. Linearity: This assumption states that the mean of the independent variable is a linear combination of the parameters and the predictor values. This assumption is a lot more restrictive than it seems at first. Because the predictor variables are treated as fixed values, linearity is just a restriction on the various parameters. The predictor variables can be arbitrarily transformed themselves, and also multiple copies of similar underlying predictor variable can be added with each one of the transformed differently.

This technique is usually used in polynomial regression, which makes use of linear regression to fit the response variable as an arbitrary polynomial function of a predictor variable. This is one thing that makes linear regression a very powerful inference method. Polynomial regression models are often too powerful, which means they overfit the data quite too often which makes the model undesirable. Due to this some kind of regularization must usually be used to avoid unreasonable solutions coming out of the estimation process.

3. Constant variance/ Homoscedasticity: This assumption means that different values of the response variable have the same variance in their errors regardless of the predictor variables. This assumption is invalid if the response variable varies over a wide scale. In order to check for error variance of

heterogenous or when a pattern of residuals violated model assumptions of homoscedasticity, it is good practice to look for a fanning effect between predicted values and residual error. This basically means that there will be a systematic change in the squared residuals or in the absolute when plotted against the predictive values. Errors will not be evenly distributed across the regression line due to which heteroscedasticity will result in the averaging over of unique variances around the points to get a single variance that is inaccurately representing all the variances of the regression line. Due to this, residuals appear clustered and spread apart on their predicted plots for larger for smaller and larger values for points along the linear regression line and the mean squared error for the model will be faulty.

4. Independence: This assumes that the errors of the response variables are not correlated with each other. Some methods such as generalized least squares are capable of handling correlated errors, although they typically require significantly larger data unless regularization is used to bias the model towards assuming errors which are not correlated.

5. Lack of perfect multicollinearity: For standard least squares estimation, X(the design matrix) must have full column rank p otherwise we have a condition known as perfect multicollinearity in the predictor variables. This can be triggered when there is two or more perfectly correlated predicted variables. It can also happen when there is very little data available compared to the number of parameters, for example, when the data points are fewer than regression coefficients. In the case of perfect multicollinearity, the vector beta will have no unique solution. Methods for fitting linear models with multicollinearity have been developed where some of them require additional assumptions such as effect sparsity.

6. Lack of or no auto-correlation: Autocorrelation occurs when the residual errors are dependent on each other.

5.5 Time Series Analysis

A time series is a series of data points in time order. Usually a time series is a sequence taken at successive equally spaced points in time. Therefore it is a sequence of discrete time data. Time series analysis comprises of methods for analysing time series data with a purpose to obtain meaningful statistics and other characteristics of the data whereas time series forecasting is the use of a model to predict future values based on previously observed values[28]. In our project too we make use of time series forecasting in order to predict future power output from the solar array with the help of already obtained values. While regression analysis is often employed in such a way as to test theories that the current values of one or more independent time series affect the current value of another time series, this type of analysis of time series is not called "time series analysis", which focuses on comparing values of a single time series or multiple dependent time series at different points in time. Interrupted time series analysis is the analysis of interventions on a single time series.

Time series data have a natural temporal ordering which makes time series analysis distinct form cross sectional studies in which there is no natural ordering of the observations. It is also distinct from spatial data analysis where the observations usually relate to geographical locations. Time series analysis can be applied to real-valued, continuous data, discrete numeric data, or discrete symbolic data.

5.5.1 CURVE FITTING

Curve fitting is the process of constructing a curve such that it firs the series of data points. It can involve either interpolation or smoothing[28]. Interpolation is where an exact fir to the data is required whereas smoothing is where an approximate fit is enough. Fitted curves can be used as an aid for data visualization, to infer values of a function where no data are available, and to summarize the relationships among two or more variables Extrapolation refers to the use of a fitted curve beyond the range of the observed data,] and is subject to a degree of uncertainty since it may reflect the method used to construct the curve as much as it reflects the observed data.

The construction of economic time series involves the estimation of some components for some dates by interpolation between values ("benchmarks") for earlier and later dates. Interpolation is estimation of an unknown quantity between two known quantities (historical data), or drawing conclusions about missing information from the available information ("reading between the lines")[29]. Interpolation is useful where the data surrounding the missing data is available and its trend, seasonality, and longer-term cycles are known. This is often done by using a related series known for all relevant dates. Alternatively polynomial interpolation or spline interpolation is used where piecewise polynomial functions are fit into time intervals such that they fit smoothly together. A different problem which is closely related to interpolation is the approximation of a complicated function by a simple function (also called regression).The main difference between regression and interpolation is that polynomial regression gives a single polynomial that models the entire data set. Spline interpolation, however, yield a piecewise continuous function composed of many polynomials to model the data set.

Extrapolation is the process of estimating, beyond the original observation range, the value of a variable on the basis of its relationship with another variable. It is similar to interpolation, which produces estimates between known observations, but extrapolation is subject to greater uncertainty and a higher risk of producing meaningless results.

5.5.2 PREDICTION AND FORECASTING

Forecasting is a part of statistical inference and one particular approach such inference is known as predictive inference but the prediction can be done within any of the various approaches to statistical inference. One description of statistics is that it provides a way to transfer knowledge about a sample of a population to the whole population. When information is transferred across time, that process is

known as forecasting. Fully formed statistical models for stochastic simulation purposes, so as to generate alternative versions of the time series, representing what might happen over non-specific time-periods in the future. Simple or fully formed statistical models to describe the likely outcome of the time series in the immediate future, given knowledge of the most recent outcomes (forecasting). Forecasting on time series is usually done using automated statistical software packages and programming languages, such as python, R, Apache and many others. Forecasting on large scale data is done using Spark which has spark-ts as a third party package.

Chapter 6

RESULT

6.1 PV Output Power Predictions

	A	B	C	D	E	F	G
1	datetime	Voltage	Current	Temperature	Degree	Power	Lintensity
2	25/10/19 17:22	18.68	0.22	22	256.79	4.1096	115.848571
3	25/10/19 17:23	18.68	0.22	22	256.79	4.1096	115.848571
4	25/10/19 17:23	18.7	0.23	22	256.79	4.301	116.021429
5	25/10/19 17:23	18.67	0.22	22	256.79	4.1074	115.787143
6	25/10/19 17:35	18.21	0.16	22	257.49	2.9136	112.661429
7	25/10/19 17:36	18.16	0.16	22	257.49	2.9056	112.354286
8	25/10/19 17:36	18.09	0.14	22	257.49	2.5326	111.824286
9	25/10/19 17:36	18.03	0.14	22	257.49	2.5242	111.455714
10	25/10/19 17:37	17.97	0.13	22	257.49	2.3361	111.037143
11	25/10/19 17:37	17.91	0.13	22	257.49	2.3283	110.668571
12	25/10/19 17:37	17.88	0.13	22	257.49	2.3244	110.484286
13	25/10/19 17:37	17.81	0.12	22	257.49	2.1372	110.004286
14	25/10/19 17:38	17.76	0.13	22	257.49	2.3088	109.747143
15	25/10/19 17:38	17.72	0.11	22	257.66	1.9492	109.401429
16	25/10/19 17:38	17.66	0.11	22	257.66	1.9426	109.032857
17	25/10/19 17:39	17.62	0.1	22	257.66	1.762	108.737143
18	25/10/19 17:39	17.54	0.1	22	257.66	1.754	108.245714
19	25/10/19 17:39	17.48	0.11	22	257.66	1.9228	107.927143
20	25/10/19 17:40	17.42	0.1	22	257.66	1.742	107.508571
21	25/10/19 17:40	17.35	0.09	22	257.66	1.5615	107.028571
22	25/10/19 17:40	17.31	0.09	22	257.66	1.5579	106.782857
23	25/10/19 17:37	17.07	0.08	22	257.55	1.3656	105.258571
24	25/10/19 17:42	17.02	0.07	22	257.61	1.1914	104.901429
25	25/10/19 17:42	16.99	0.07	22	257.61	1.1893	104.717143
26	25/10/19 17:42	16.94	0.07	22	257.61	1.1858	104.41
27	25/10/19 17:43	16.91	0.07	22	257.66	1.1837	104.225714
28	25/10/19 17:43	16.88	0.07	22	257.66	1.1816	104.041429
29	25/10/19 17:43	16.84	0.07	22	257.66	1.1788	103.795714
30	25/10/19 17:44	16.81	0.08	22	257.72	1.3448	103.661429
31	25/10/19 17:44	16.74	0.06	22	257.72	1.0044	103.131429
32	25/10/19 17:44	16.67	0.06	22	257.72	1.0002	102.701429
33	25/10/19 17:44	16.59	0.06	22	257.78	0.9954	102.21

Fig. 6.1 Snapshot of Data Collected

The output of the developed algorithms are listed below in the Table. Their performance is evaluated in terms of percentage accuracy in each category.

Metric	**LR**	**K-NN**
Training error	13.67%	1.370%
Training accuracy	86.33%	98.62%
Test error	7.88%	1.60%
Test accuracy	92.11%	98.39%

We can observe that the difference between LR and KNN values are distinct. One can note the significant increase in error percentage for LR method. The training accuracy is better in K-NN because of following reasons:

- K-NN stores the entire training data set and does not assume any model, whereas LR assumes a linear relationship between the variables.
- K-NN can produce nonlinear solutions.

- K-NN makes real-time predictions by calculating the similarity between an input sample and each training instance.

The testing accuracy can be improved if the following processes are carried out before applying a model:

- Hypothesis generation: It is always important to explore the data before collection and understand the relationships between the variables. Hence hypothesis generation gives an insight into the relativity of a data set.
- Data sufficiency: Good amount of data always results in accurate results. We can always increase the size of training data and allow the data to speak for itself.
- Algorithm Tuning: Parameters decide the outcome of the learning process. Parameter tuning can increase the accuracy of a model. For ex in K-NN choosing the best value of K after several trials.

6.2 Power Consumption Prediction

The figure below shows a plot of predicted power and actual power. We can observe that the values tend to converge with the minimum deviation. The LSTM algorithm proves to be a good fit for the load variation forecast. An accuracy of 64.33% was achieved.

The accuracy is quite low due to the following reasons:

1. Inadequate data
2. Randomization of collected data

	A	B	C	D	E	F
1	Date	Time	Load 1(A)	Load 2(A)	Load 3(A)	Load 4(A)
2	13/10/16	16:42:30	-0.34	-0.59	0.05	-1.42
3	17/02/20	8:40:13	0.03	1.11	1.76	0
4	17/02/20	8:40:22	-0.08	0.26	5.37	6.64
5	17/02/20	8:40:34	0.05	0.45	6.35	11.67
6	17/02/20	8:40:48	0.05	0.29	5.47	11.67
7	17/02/20	8:41:00	-0.13	0.37	6.49	12.4
8	17/02/20	8:41:06	0.05	0.37	12.21	11.72
9	17/02/20	8:41:16	0.05	0.34	5.91	11.96
10	17/02/20	8:41:30	0	0.4	6.35	12.6
11	17/02/20	8:47:04	0.03	0.32	5.96	12.99
12	17/02/20	8:49:36	-0.03	0.34	7.62	12.89
13	17/02/20	8:52:36	0	0.82	5.27	8.15
14	17/02/20	8:54:19	0.05	0.34	6.15	8.3
15	17/02/20	8:54:53	0.05	0.34	8.06	5.81
16	17/02/20	9:02:08	0.05	0.29	4.35	11.67
17	17/02/20	9:05:07	0.03	0.32	6.3	0.1
18	17/02/20	9:08:07	-0.08	0.37	4.93	0.1
19	17/02/20	9:11:07	0	0.37	6.59	0.05
20	17/02/20	9:14:07	0.03	0.34	3.86	-0.05
21	17/02/20	9:17:07	0.53	0.34	4.83	0
22	17/02/20	9:20:07	-0.05	0.34	5.32	-0.05
23	17/02/20	9:23:07	-0.03	0.34	5.47	0.05
24	17/02/20	9:26:07	-0.05	0.4	5.96	0.05
25	17/02/20	9:29:07	0.55	0.26	4.35	0
26	17/02/20	9:32:07	0.61	0.34	5.71	0.05
27	17/02/20	9:35:07	-0.03	0.34	4.74	0
28	17/02/20	9:38:06	-0.03	0.29	6.88	0.05
29	17/02/20	9:41:06	0.08	0.37	6.88	0.05

Fig. 6.2 Snapshot of data collected from load

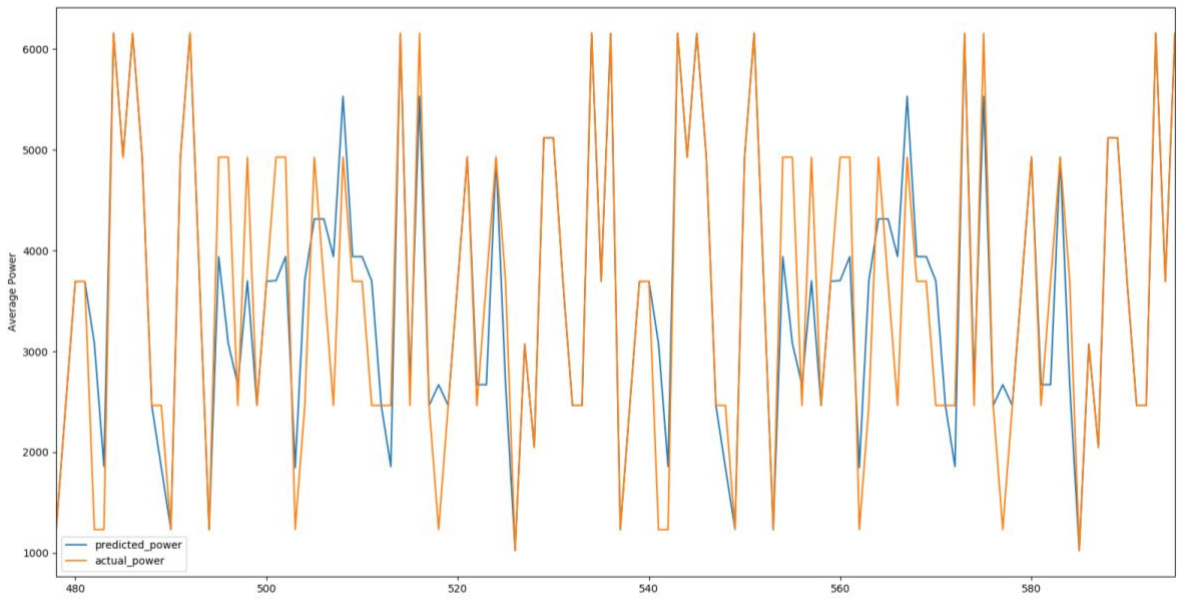


Fig. 6.3 Forecast Plot

Chapter 7

CONCLUSION

The purpose of PV power prediction is emphasized for safe and economical operation of power systems. A forecasting model is proposed by analysing the historical data of source and load . Machine learning techniques are demonstrated to understand the nature of prediction. To validate the use of methods proposed algorithms were tested under strict meteorological and technical conditions. The K-NN and LSTM methods are found to respond quickly and can be adopted for real time predictions.

The future scope and application of this project can achieve source management to satisfy the variable demand by parallel operation of sources, Load automation using IOT and minimizing power outages in sudden grid failure, smart and direct real time load control and load selection.

Chapter 8

REFERENCES

[1]. K. Punitha, D. Devaraj, and S. Sakthivel, "Development and analysis of adaptive fuzzy controllers for photovoltaic systems under varying atmospheric and partial shading conditions" Applied Soft Computing Journal, vol. 13, no. 11, pp. 4320–4332, 2013.

[2]. K. Punitha, D. Devaraj, and S. Sakthivel, "Artificial neural network based modified incremental conductance algorithm for maximum power point tracking in photovoltaic systems under partial shading conditions," Energy, vol. 62, pp. 330–340, 2013.

[3]. K. Ishaque and Z. Salam, "An improved modeling method to determine the model parameters of photovoltaic (PV) modules using differential evolution (DE)," Solar Energy, vol. 85, no. 9, pp. 2349–2359, 2011.

[4]. Mao Yang and Xin Huang, "An Evaluation Method of the Photovoltaic Power Prediction Quality", mathematical Problems in Engineering, Volume 2018, Article ID 9049215

[5]. G. Mu, K. Hou, and Y. Yang, "Research on the evaluation method of load regularity," CSEE, vol. 10, pp. 96–101, 2001.

[6] S. Pelland, G. Galanis, G. Kallos, "Solar and photovoltaic forecasting through post-processing of the Global Environmental Multiscale numerical weather prediction model", Progress in Photovoltaics: Research and Applications, 21 (2013) 284-296.

[7] S.S. Soman, H. Zareipour, O. Malik, P. Mandal, "A review of wind power and wind speed forecasting methods with different time horizons", in: North American Power Symposium (NAPS), 2010, 2010, pp. 1-8.

[8] H.T. Yang, C.M. Huang, Y.C. Huang, Y.S. Pai, "A weather-based hybrid method for 1-day ahead hourly forecasting of PV power output" IEEE Trans Sustain Energy, 5 (3) (2014), pp. 917-926

[9] T. Hiyama, K. Kitabayashi, "Neural network based estimation of maximum power generation from PV module using environmental information" IEEE Trans Energy Convers, 12 (3) (1997), pp. 241-247

[10] M.W. Ahmad, M. Mourshed, B. Yuce, Y. Rezgui, "Computational intelligence techniques for HVAC systems": a review Build Simulat, 9 (4) (2016), pp. 359-398

[11] D. van der Meer, J. WidÃl'n, J. Munkhammar, "Review on probabilistic forecasting of photovoltaic power production and electricity consumption" Renew Sustain Energy Rev, 81 (2018), pp. 1484-1512

[12] R. Campo and P. Ruiz, "Adaptive Weather-Sensitive Short-Term Load Forecast", IEEE Transactions On Power Systems., Vol. PWRS-2, No. 3, Aug. 1987, at 592-600.

[13] G. Gross and F.D. Galiana, ShortTerm Forecasting, Proceedings of IEEE,75(12), 1987, at 1558–73.

[14] K Jabbour, J.F.V. Riveros, D.Landsbergen and W. Meyer, ALFA, "Automated Load Forecasting Assistant", IEEE Transactions. On Power Systems.,Vol. 2, No. 3, Aug. 1988, at 908-914.

[15]. M Ra¨sa¨nen and J Ruusunen,Verkoston Tilan Seurantamittauksillaja Kuormitusmalleilla, Research Report B17, Systems Analysis Laboratory, Helsinki Univ. of Technology (in Finnish).

[16] Friedman, Jerome H. (1998). "Data Mining and Statistics: What's the connection?". Computing Science and Statistics. 29 (1): 3–9.

[17] Alex Ratner; Stephen Bach; Paroma Varma; Chris. "Weak Supervision: The New Programming Paradigm for Machine Learning". hazyresearch.github.io. referencing work by many other members of Hazy Research. Retrieved 2019-06-06.

[18] Y. Bengio; A. Courville; P. Vincent (2013). "Representation Learning: A Review and New Perspectives". IEEE Transactions on Pattern Analysis and Machine Intelligence. 35 (8): 1798–1828.

[19] Hernandez, Da"IBM Has a Watson Dilemma". Wall Street Journal. ISSN 0099-9660niela; Greenwald, Ted (2018-08-11).

[20] Julia Angwin; Jeff Larson; Lauren Kirchner; Surya Mattu (2016-05-23). "Machine Bias". ProPublica. Retrieved 2018-08-20.

[21] Altman, Naomi S. (1992). "An introduction to kernel and nearest-neighbor nonparametric regression" (PDF). The American Statistician. 46 (3): 175–185.

[22] Jaskowiak, Pablo A.; Campello, Ricardo J. G. B. "Comparing Correlation Coefficients as Dissimilarity Measures for Cancer Classification in Gene Expression Data". Brazilian Symposium on Bioinformatics (BSB 2011): CiteSeerX 10.1.1.208.993

[23] Everitt, Brian S.; Landau, Sabine; Leese, Morven; and Stahl, Daniel (2011) "Miscellaneous Clustering Methods", in Cluster Analysis, 5th Edition, John Wiley & Sons, Ltd., Chichester, UK

[24] Toussaint, Godfried T. (April 2005). "Geometric proximity graphs for improving nearest neighbor methods in instance-based learning and data mining". International Journal of Computational Geometry and Applications. 15 (2): 101–150.

[25] David A. Freedman (2009). Statistical Models: Theory and Practice. Cambridge University Press. p. 26. A simple regression equation has on the right hand side an intercept and an explanatory variable with a slope coefficient. A multiple regression equation has two or more explanatory variables on the right hand side, each with its own slope coefficient

[26] Rencher, Alvin C.; Christensen, William F. (2012), "Chapter 10, Multivariate regression – Section 10.1, Introduction", Methods of Multivariate Analysis, Wiley Series in Probability and Statistics, 709 (3rd ed.), John Wiley & Sons, p. 19, ISBN 9781118391679.

[27] Swindel, Benee F. (1981). "Geometry of Ridge Regression Illustrated". The American Statistician**.** 35 (1): 12–15. doi:10.2307/2683577. JSTOR 2683577

[28] Liao, T. Warren (2005). "Clustering of time series data - a survey". Pattern Recognition. Elsevier. 38 (11): 1857–1874. doi:10.1016/j.patcog.2005.01.025

[29] Friedman, Milton. "The interpolation of time series by related series." Journal of the American Statistical Association 57.300 (1962): 729–757.

APPENDIX - A

CODE

6.1 Solar Panel Data Collector : Arduino

```
#include <Wire.h>
#include <TimeLib.h>

//**********************VALUES FOR millis**********************//
unsigned long previousMillis1 = 0;
unsigned long previousMillis2 = 0;
const long interval1 = 180000;
const long interval2 = 1000;

//**********************INITIALLZING PINS**********************//
int relay_Out = 2;
int SC_Current = A0 ;
//int load_Current = A2 ;
int OC_Voltage = A1 ;

//**********************VALUES FOR ACS712**********************//
int sensitivity = 100; //100 for 20A Module and 66 for 30A Module
int volt_read= 0;
int ACSoffset = 2500;
double Voltage = 0;
double Amps = 0;

//**********************VALUES FOR VOLTMETER**********************//
float vout = 0.0;
float vin = 0.0;
float R1 = 100000;
float R2 = 22000;
int value = 0;
```

```
//******************FUNCTION FOR READING CURRENT******************//
float readCurrent(float raw_Current)
{
  float AcsValue=0.0,Samples=0.0,AvgAcs=0.0,AcsValueF=0.0;
  for (int x = 0; x < 150; x++){ //Get 150 samples
  AcsValue = analogRead(raw_Current);     //Read current sensor values
  Samples = Samples + AcsValue;  //Add samples together
  delay (3); // let ADC settle before next sample 3ms
}
AvgAcs=Samples/150.0; //Taking Average of Samples

//((AvgAcs * (5.0 / 1024.0)) is converitng the read voltage in 0-5 volts
//2.5 is offset(I assumed that arduino is working on 5v so the viout at no current comes
//out to be 2.5 which is out offset. If your arduino is working on different voltage than
//you must change the offset according to the input voltage)
//0.100v(100mV) is rise in output voltage when 1A current flows at input
AcsValueF = (2.5 - (AvgAcs * (5.0 / 1024.0)) )/0.100;
return AcsValueF; //Print the read current on Serial monitor
}
//******************FUNCTION FOR READING VOLTAGE******************//
float readVoltage(float raw_Voltage)
{
   double VolValue=0.0,Samples=0.0,AvgVol=0.0;
   for (int x = 0; x < 500; x++)
   { //Get 150 samples
   VolValue = analogRead(raw_Voltage);     //Read current sensor values
   Samples = Samples + VolValue;  //Add samples together
   delay (3); // let ADC settle before next sample 3ms
   }
   AvgVol = Samples/500.0;
   AvgVol = ((AvgVol*5.0)/1023);
   double vin = (AvgVol * 5.545454 );
   return vin;
}
```

```
void setup()
{
  Serial.begin(9600);
  pinMode(relay_Out, OUTPUT);
  pinMode(SC_Current, INPUT);
  pinMode(OC_Voltage, INPUT);
  while (!Serial) ; //Wait for serial monitor
}
void loop()
{
  unsigned long currentMillis1 = millis();
  //unsigned long currentMillis2 = millis();
  if (currentMillis1 - previousMillis1 >= interval1)
  {
    // save the last time you transmitted data
    previousMillis1 = currentMillis1;
    //Send Load Current
    /*Serial.print(readCurrent(load_Current));
    Serial.print(",");*/
    //Send Open Circuit Voltage
    digitalWrite(relay_Out, LOW);
    unsigned long currentMillis2 = millis();
    previousMillis2=currentMillis2;
    while(currentMillis2-previousMillis2<=interval2)
    {
      currentMillis2 = millis();
    }
    Serial.print(readVoltage(OC_Voltage));
    Serial.print(",");
    currentMillis2 = millis();
    previousMillis2=currentMillis2;
    while(currentMillis2-previousMillis2<=interval2)
    {
      currentMillis2 = millis();
    }
```

```
   digitalWrite(relay_Out, HIGH);
   currentMillis2 = millis();
   previousMillis2=currentMillis2;
   while(currentMillis2-previousMillis2<=interval2)
   {
    currentMillis2 = millis();
   }
   // Send Short Circuit Current
   Serial.println(abs(readCurrent(SC_Current)-0.27));
  }
}
```

6.2 Solar Panel Data Collector : Raspberry Pi

```
#!/usr/bin/python
################
## Script listens to serial port and writes contents into a file
################
## requires pySerial to be installed
import serial
import time
import sys
import csv
timestamp = time.strftime("%Y-%m-%d-%H-%M-%S")
ser = '/dev/ttyACM0'
baud_rate = 9600; #In arduino, Serial.begin(baud_rate)
filepath = "/home/pi/Documents/final_project/source_data/V1.2/" + timestamp + ".csv";
input_stream = serial.Serial(ser, baud_rate,)
fields = ['Date&Time','Voltage','SC_Current']
print("Writing output to file: " + filepath + "\n")
csv.register_dialect('mydialect', delimiter = ',', skipinitialspace = True, lineterminator = '\r\n', quoting
= csv.QUOTE_ALL)
with open(filepath, 'a+') as f:
        writer = csv.writer(f, dialect =  'mydialect')
        writer.writerow(fields)
f.close()
while 1:
```

```
line = input_stream.readline();
data_list = line.split(",")
timestamp = time.strftime("%Y/%m/%d-%H:%M:%S")
str(timestamp)
data_list.insert(0,timestamp)
print(data_list);
with open(filepath, 'a+') as f:
    for item in data_list:
                str(item)
                f.write(item)
                f.write(",")
        f.write('\n')
        #writer = csv.writer(f, dialect = 'mydialect')
    f.close()
```

6.3 Domestic Load Usage Replicator : Arduino

```
#include "Wire.h"
#include "SD.h"
#include"SPI.h"
#define DS3231_I2C_ADDRESS 0x68
#define relay1 2
#define relay2 3
#define relay3 4
#define relay4 5
int relay[]={relay1,relay2,relay3,relay4};
const int CSpin = 10;
String dataString ="";
File loadData;
long randNumber;
long randNumber1;
long randNumber2;
long randNumber3;
float load1;
float load2;
float load3;
float load4;
```

```
unsigned long previousMillis = 0;
bool flag = 1;
const int currentPin1 = A0;
const int currentPin2 = A1;
const int currentPin3 = A2;
const int currentPin4 = A3;
int sensitivity = 0;
int adcValue= 0;
float offsetVoltage = 2500;
double adcVoltage = 0;
double currentValue = 0;
// Convert normal decimal numbers to binary coded decimal
byte decToBcd(byte val)
{
  return( (val/10*16) + (val%10) );
}
// Convert binary coded decimal to normal decimal numbers
byte bcdToDec(byte val){
  return( (val/16*10) + (val%16) );
}
void setup(){
  Wire.begin();
  Serial.begin(9600);
  // set the initial time here:
  // DS3231 seconds, minutes, hours, day, date, month, year
  //setDS3231time(00,41,16,5,13,02,20);
    randomSeed(analogRead(A0));
    pinMode(relay1, OUTPUT);
    pinMode(relay2, OUTPUT);
    pinMode(relay3, OUTPUT);
    pinMode(relay4, OUTPUT);
    pinMode(CSpin, OUTPUT);
    if (!SD.begin(CSpin))
    {
      Serial.println("Card failed, or not present");
```

```
      // don't do anything more:
      return;
    }
    Serial.println("card initialized.");
}
void saveData()
{
   if(SD.exists("data.csv"))
   {
     // check the card is still there
     // now append new data file
     loadData = SD.open("data.csv", FILE_WRITE);
     if (loadData)
     {
       loadData.println(dataString);
       loadData.close(); // close the file
       Serial.println("Written to file !");
     }
   }
   else
   {
     Serial.println("Error writing to file !");
   }
}
float current(int currentData, int sens)
{
 sensitivity = sens ;
 adcValue = currentData;
 adcVoltage = (adcValue / 1024.0) * 5000;
 currentValue = ((adcVoltage - offsetVoltage) / sensitivity);
 return currentValue;
}
void setDS3231time(byte second, byte minute, byte hour, byte dayOfWeek, byte dayOfMonth, byte
month, byte year)
{
```

```
 // sets time and date data to DS3231
 Wire.beginTransmission(DS3231_I2C_ADDRESS);
 Wire.write(0); // set next input to start at the seconds register
 Wire.write(decToBcd(second)); // set seconds
 Wire.write(decToBcd(minute)); // set minutes
 Wire.write(decToBcd(hour)); // set hours
 Wire.write(decToBcd(dayOfWeek)); // set day of week (1=Sunday, 7=Saturday)
 Wire.write(decToBcd(dayOfMonth)); // set date (1 to 31)
 Wire.write(decToBcd(month)); // set month
 Wire.write(decToBcd(year)); // set year (0 to 99)
 Wire.endTransmission();
}
void readDS3231time(byte *second, byte *minute, byte *hour, byte *dayOfWeek, byte *dayOfMonth, byte *month, byte *year)
{
 Wire.beginTransmission(DS3231_I2C_ADDRESS);
 Wire.write(0); // set DS3231 register pointer to 00h
 Wire.endTransmission();
 Wire.requestFrom(DS3231_I2C_ADDRESS, 7);
 // request seven bytes of data from DS3231 starting from register 00h
 *second = bcdToDec(Wire.read() & 0x7f);
 *minute = bcdToDec(Wire.read());
 *hour = bcdToDec(Wire.read() & 0x3f);
 *dayOfWeek = bcdToDec(Wire.read());
 *dayOfMonth = bcdToDec(Wire.read());
 *month = bcdToDec(Wire.read());
 *year = bcdToDec(Wire.read());
}
void loop()
{
 digitalWrite(relay[0], HIGH);
 digitalWrite(relay[1], HIGH);
 digitalWrite(relay[2], HIGH);
 digitalWrite(relay[3], HIGH);
 int condition;
```

```
byte second, minute, hour, dayOfWeek, dayOfMonth, month, year;
// retrieve data from DS3231
readDS3231time(&second, &minute, &hour, &dayOfWeek, &dayOfMonth, &month, &year);
// Toggle load during weekdays
if(dayOfWeek>=2 && dayOfWeek<=6)
{
 if(hour>= 7 && hour<9)
 {  unsigned long interval;
   unsigned long currentMillis = millis();
   if (flag == 1)
   {
     do
     {
      randNumber1 = random(0,4);
      randNumber2 = random(0,4);
     } while(randNumber1==randNumber2);
     interval = random(30e+4,66e+4);
     flag = 0;
   }
   digitalWrite(relay[randNumber1], LOW);
   digitalWrite(relay[randNumber2], LOW);
   if (currentMillis - previousMillis >= interval)
   {
     previousMillis = currentMillis;
     digitalWrite(relay[randNumber1], HIGH);
     digitalWrite(relay[randNumber2], HIGH);
     flag = 1;
   }
 }
 if(hour>=9 && hour<18)
 {
   unsigned long interval;
   unsigned long currentMillis = millis();
   if (flag == 1)
   {
```

```
    randNumber = random(0,4);
    interval = random(30e+4,66e+4);
    flag = 0;
   }
   digitalWrite(relay[randNumber], LOW);
   if (currentMillis - previousMillis >= interval)
   {
    previousMillis = currentMillis;
    digitalWrite(relay[randNumber], HIGH);
    flag = 1;
   }
 }
 if(hour>= 18 && hour<22)
 {
   unsigned long interval;
   unsigned long currentMillis = millis();
   if (flag == 1)
   {
     do
     {
      randNumber1 = random(0,4);
      randNumber2 = random(0,4);
     } while(randNumber1==randNumber2);
     interval = random(30e+4,66e+4);
     flag = 0;
   }
   digitalWrite(relay[randNumber1], LOW);
   digitalWrite(relay[randNumber2], LOW);
   if (currentMillis - previousMillis >= interval)
   {
     previousMillis = currentMillis;
     digitalWrite(relay[randNumber1], HIGH);
     digitalWrite(relay[randNumber2], HIGH);
     flag = 1;
   }
```

```
  }
  if(hour>= 22 && hour<7)                //CHECK
  {
    unsigned long interval;
    unsigned long currentMillis = millis();
    if (flag == 1)
    {
      do
      {
       randNumber1 = random(0,4);
       randNumber2 = random(0,4);
      } while(randNumber1==randNumber2);
      interval = random(30e+4,66e+4);
      flag = 0;
    }
    digitalWrite(relay[randNumber1], LOW);
    digitalWrite(relay[randNumber2], LOW);
    if (currentMillis - previousMillis >= interval)
    {
      previousMillis = currentMillis;
      digitalWrite(relay[randNumber1], HIGH);
      digitalWrite(relay[randNumber2], HIGH);
      flag = 1;
    }
  }
}
if(dayOfWeek == 1 || dayOfWeek==7)
{
  if(hour>=7 && hour<18)
  {
    unsigned long interval;
    unsigned long currentMillis = millis();
    if (flag == 1)
    {
      do
```

```
      {
       randNumber1 = random(0,4);
       randNumber2 = random(0,4);
      } while(randNumber1==randNumber2);
      interval = random(30e+4,66e+4);
      flag = 0;
    }
    digitalWrite(relay[randNumber1], LOW);
    digitalWrite(relay[randNumber2], LOW);
    if (currentMillis - previousMillis >= interval)
    {
      previousMillis = currentMillis;
      digitalWrite(relay[randNumber1], HIGH);
      digitalWrite(relay[randNumber2], HIGH);
      flag = 1;
    }
  }
  if(hour>=18 && hour<22)
  {
    unsigned long interval;
    unsigned long currentMillis = millis();
    if (flag == 1)
    {
      do
      {
       randNumber1 = random(0,4);
       randNumber2 = random(0,4);
       randNumber3 = random(0,4);
      }
while((randNumber1==randNumber2)||(randNumber2==randNumber3)||(randNumber3==randNu
mber1));
      interval = random(30e+4,66e+4);
      flag = 0;
    }
    digitalWrite(relay[randNumber1], LOW);
```

```
    digitalWrite(relay[randNumber2], LOW);
    digitalWrite(relay[randNumber3], LOW);
    if (currentMillis - previousMillis >= interval)
    {
      previousMillis = currentMillis;
      digitalWrite(relay[randNumber1], HIGH);
      digitalWrite(relay[randNumber2], HIGH);
      digitalWrite(relay[randNumber3], HIGH);
      flag = 1;
    }
  }
 if(hour>= 22 && hour<7)
 {
    unsigned long interval;
    unsigned long currentMillis = millis();
    if (flag == 1)
    {
      do
      {
       randNumber1 = random(0,4);
       randNumber2 = random(0,4);
      } while(randNumber1==randNumber2);
      interval = random(30e+4,66e+4);
      flag = 0;
    }
    digitalWrite(relay[randNumber1], LOW);
    digitalWrite(relay[randNumber2], LOW);
    Serial.println("Condition7");
    if (currentMillis - previousMillis >= interval)
    {
      previousMillis = currentMillis;
      digitalWrite(relay[randNumber1], HIGH);
      digitalWrite(relay[randNumber2], HIGH);
      flag = 1;
    }
```

```
  }
 }
load1 = abs(current(analogRead(currentPin1),185));
load2 = abs(current(analogRead(currentPin2),185));
load3 = abs(current(analogRead(currentPin3),100));
load4 = abs(current(analogRead(currentPin4),100));
dataString = String(dayOfMonth) + "/" + String(month) + "/" + String(year) + "," + String(hour) + ":" + String(minute) + ":" + String(second) + "," + String(load1) + "," + String(load2) + "," + String(load3) + "," + String(load4);
Serial.println(dataString);
saveData();
delay(180000);
}
```

6.4 Solar Panel Power Generation Prediction : Python

```
import pandas as pd
import numpy as np
from pandas import read_csv
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.linear_model import LogisticRegression
data=
pd.read_csv('/home/aayush/Documents/FinalYearProject/SolarPanel_PredictiveAnalysis/FinalDataSet.csv')
data = data[data['Power']!=0]
y=data["Power"]
y=np.array(y)
y= y.reshape(-1,1)
y.shape
x=data.drop(columns=["Power","datetime","Voltage","Current"])
x.head()
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.25, random_state=1)
```

```
regressor = LinearRegression()
regressor.fit(X_train, y_train)
y_pred = regressor.predict(X_train)
from sklearn.metrics import r2_score
r2_score(y_train,y_pred)
import math
error =[np.power((b-a),2) for (a,b) in zip(y_pred,y_train)]
error0 = np.sum(error)
error = math.sqrt(error0)
error = (error/len(y_test))*100
print("training error % = {}".format(error))
accuracy = 100 - error
print("training Accuracy % = {}".format(accuracy))
y_test_pred = regressor.predict(X_test)
r2_score(y_test,y_test_pred)
error =[np.power((b-a),2) for (a,b) in zip(y_test_pred,y_test)]
error0 = np.sum(error)
error = math.sqrt(error0)
error = (error/len(y_test))*100
print("Test error % = {}".format(error))
accuracy = 100 - error
print("Test Accuracy % = {}".format(accuracy))
from sklearn.neighbors import KNeighborsRegressor
errort=[]
for i in range(1,100):
  knn=KNeighborsRegressor(algorithm='brute',n_neighbors=i)
  knn.fit(X_train,y_train)
  pred_i=knn.predict(X_test)
  error =[np.power((b-a),2) for (a,b) in zip(y_test_pred,y_test)]
  error0 = np.sum(error)
  error = math.sqrt(error0)
  error = (error/len(y_test))*100
  accuracy = 100 - error
  errort.append(accuracy)
import matplotlib.pyplot as plt
```

```
plt.figure(figsize=(12,6))
plt.plot(range(1,100),errort,color='red',linestyle='dashed',marker='o',markerfacecolor='blue',markersi
ze=10)
plt.title("Error vs K")
plt.xlabel('K val')
plt.ylabel('Mean error')
#print(error.index(min(error))+1)
knn=KNeighborsRegressor(algorithm='brute',n_neighbors=4)
knn.fit(X_train,y_train)
y_pred=knn.predict(X_train)
y_pred.shape,y_train.shape
from sklearn.metrics import r2_score
r2_score(y_train,y_pred)
import math
error =[np.power((b-a),2) for (a,b) in zip(y_pred,y_train)]
error0 = np.sum(error)
error = math.sqrt(error0)
error = (error/len(y_test))*100
print("training error % = {}".format(error))
accuracy = 100 - error
print("training Accuracy % = {}".format(accuracy))
y_test_pred = knn.predict(X_test)
r2_score(y_test,y_test_pred)
error =[np.power((b-a),2) for (a,b) in zip(y_test_pred,y_test)]
error0 = np.sum(error)
error = math.sqrt(error0)
error = (error/len(y_test))*100
print("Test error % = {}".format(error))
accuracy = 100 - error
print("Test Accuracy % = {}".format(accuracy))
```

6.5 Domestic Load Usage Predictor : Python

```
import pandas as pd
import numpy as np
# matplotlib and seaborn are used for plotting graphs
import matplotlib.pyplot as plt
```

```
import seaborn
from google.colab import drive
path='/home/aayush/Documents/FinalYearProject/SolarPanel_PredictiveAnalysis/FinalDataSet.cs'
Df= pd.read_csv(path)
Df.head()
Df.shape
Df['S_3'] = Df['Average Power'].shift(1).rolling(window=3).mean()
Df['S_9']= Df['Average Power'].shift(1).rolling(window=9).mean()
Df.head()
Df =Df.dropna()
X = Df[['S_3','S_9']]
X.head()
y = Df['Average Power']
y.head()
X.shape, y.shape
t=.8
t =(t*len(Df))
# Train dataset
t=int(t)
t
X_train = X[:t]
y_train = y[:t]
# Test dataset
X_test = X[t:]
y_test = y[t:]
X_train.shape, y_train.shape, X_test.shape, y_test.shape
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(X_train, y_train)
# Import the model we are using
from sklearn.ensemble import RandomForestRegressor
# Instantiate model with 1000 decision trees
model= RandomForestRegressor(n_estimators = 1000, random_state = 42,verbose=2)
# Train the model on training data
model.fit(X_train,y_train);
```

```
model.score(X_train, y_train)
pred_power = model.predict(X_test)
import math
erro_r = [np.power((b-a),2) for (a, b) in zip(pred_price, y_test)] #mean absolute percentage error
error0 = np.sum(erro_r)
error = math.sqrt(error0)
error=(error/len(y))*100
print("test_error % = {}".format(error))
accuracy= 100 - error
print("test_accuracy %={}".format(accuracy))
predicted_power = pd.DataFrame(pred_power,index=y_test.index,columns = ['Average Power'])
predicted_power.plot(figsize=(100,50))
y_test.plot()
plt.legend(['predicted_power','actual_power'])
plt.ylabel("Average Power")
plt.show()
from sklearn.metrics import r2_score
r2_score(y_test, pred_price)
```

SONGLE RELAY

松乐继电器 ® SONGLE RELAY	**RELAY ISO9002**	SRD

1. MAIN FEATURES

☐ Switching capacity available by 10A in spite of small size design for highdensity P.C. board mounting technique.

☐ UL,CUL,TUV recognized.

☐ Selection of plastic material for high temperature and better chemical solution performance.

☐ Sealed types available.

☐ Simple relay magnetic circuit to meet low cost of mass production.

2. APPLICATIONS

☐ Domestic appliance, office machine, audio, equipment, automobile, etc.

(Remote control TV receiver, monitor display, audio equipment high rushing current use application.)

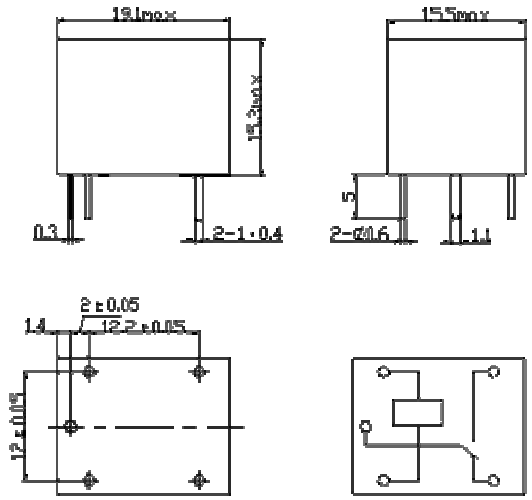
3. ORDERING INFORMATION

SRD	XX VDC	S	L	C
Model of relay	Nominal coil voltage	Structure	Coil	Contact form
SRD	03|05|06|09|12|24|48VDC	S:Sealed type	L:0.36W	A:1 form A
				B:1 form B
		F:Flux free type	D:0.45W	C:1 form C

4. RATING

CCC	FILE NUMBER:CQC03001003729	7A/240VDC
CCC	FILE NUMBER:CQC03001003731	10A/250VDC
UL /CUL	FILE NUMBER: E167996	10A/125VAC 28VDC
TUV	FILE NUMBER: R50056114	10A/250VAC 30VDC

5. DIMENSION(unit:mm) DRILLING(unit:mm) WIRING DIAGRAM



6. COIL DATA CHART (AT20 ö C)

Coil Sensitivity	Coil Voltage Code	Nominal Voltage (VDC)	Nominal Current (mA)	Coil Resistance (Ω) ☐ 10%	Power Consumption (W)	Pull-In Voltage (VDC)	Drop-Out Voltage (VDC)	Max-Allowable Voltage (VDC)
SRD (High Sensitivity)	03	03	120	25	abt. 0.36W	75%Max.	10% Min.	120%
	05	05	71.4	70				
	06	06	60	100				
	09	09	40	225				
	12	12	30	400				
	24	24	15	1600				
	48	48	7.5	6400				
SRD (Standard)	03	03	150	20	abt. 0.45W	75% Max.	10% Min.	110%
	05	05	89.3	55				
	06	06	75	80				
	09	09	50	180				
	12	12	37.5	320				
	24	24	18.7	1280				
	48	48	10	4500	abt. 0.51W			

7. CONTACT RATING

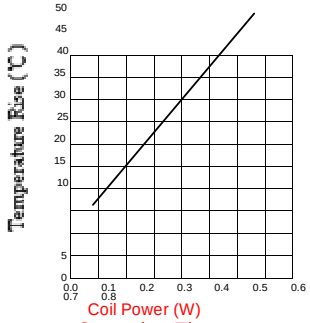
Item \ Type	SRD FORM C	SRD FORM A
Contact Capacity	7A 30VDC	10A 30VDC
Resistive Load (cosΦ=1)	10A 125VAC	10A 240VAC
Inductive Load (cosΦ=0.4 L/R=7msec)	10A 250VAC	5A 120VAC 5A 28VDC
	3A 120VAC 3A 28VDC	
Max. Allowable Voltage	250VAC/110VDC	250VAC/110VDC
Max. Allowable Power Force	800VAC/240W	1200VA/300W
Contact Material	AgCdO	AgCdO

8. PERFORMANCE (at initial value)

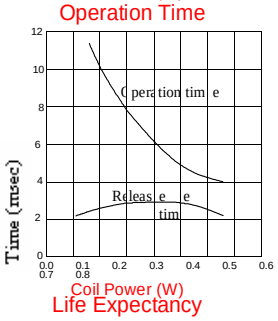
Item \ Type	SRD
Contact Resistance	100mΩ Max.
Operation Time	10msec Max.
Release Time	5msec Max.
Dielectric Strength	
Between coil & contact	1500VAC 50/60HZ (1 minute)
Between contacts	1000VAC 50/60HZ (1 minute)
Insulation Resistance	100 MΩ Min. (500VDC)
Max. ON/OFF Switching	
Mechanically	300 operation/min
Electrically	30 operation/min
Ambient Temperature	-25 ☐C to +70 C
Operating Humidity	45 to 85% RH
Vibration	
Endurance	10 to 55Hz Double Amplitude 1.5mm
Error Operation	10 to 55Hz Double Amplitude 1.5mm
Shock	
Endurance	100G Min.
Error Operation	10G Min.
Life Expectancy	
Mechanically	10^7 operations. Min. (no load)
Electrically	10^5 operations. Min. (at rated coil voltage)
Weight	abt. 10grs.

9.REFERENCE DATA

Coil Temperature Rise

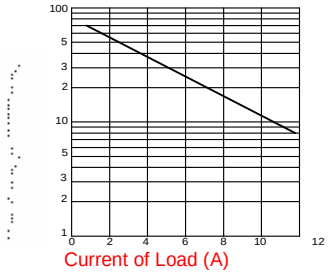


Operation Time

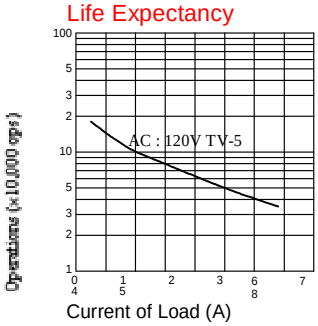


Life Expectancy

AC120V/DC24V cosΦ=1



Life Expectancy



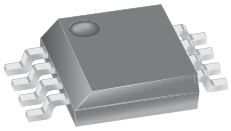
Fully Integrated, Hall Effect-Based Linear Current Sensor with 2.1 kVRMS Voltage Isolation and a Low-Resistance Current Conductor

Features and Benefits

- Low-noise analog signal path
- Device bandwidth is set via the new FILTER pin
- 5 µs output rise time in response to step input current
- 80 kHz bandwidth
- Total output error 1.5% at T_A = 25°C
- Small footprint, low-profile SOIC8 package
- 1.2 mΩ internal conductor resistance
- 2.1 kV_{RMS} minimum isolation voltage from pins 1-4 to pins 5-8
- 5.0 V, single supply operation
- 66 to 185 mV/A output sensitivity
- Output voltage proportional to AC or DC currents
- Factory-trimmed for accuracy
- Extremely stable output offset voltage
- Nearly zero magnetic hysteresis
- Ratiometric output from supply voltage

TÜV America
Certificate Number:
U8V 06 05 54214 010

Package: 8 Lead SOIC (suffix LC)



Approximate Scale 1:1

Description

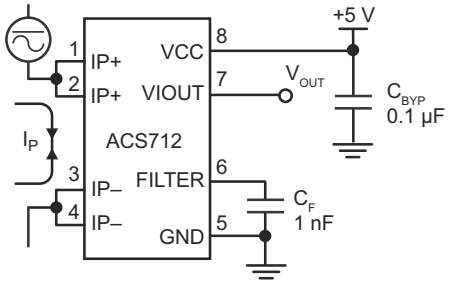
The Allegro® ACS712 provides economical and precise solutions for AC or DC current sensing in industrial, commercial, and communications systems. The device package allows for easy implementation by the customer. Typical applications include motor control, load detection and management, switched-mode power supplies, and overcurrent fault protection.

The device consists of a precise, low-offset, linear Hall sensor circuit with a copper conduction path located near the surface of the die. Applied current flowing through this copper conduction path generates a magnetic field which is sensed by the integrated Hall IC and converted into a proportional voltage. Device accuracy is optimized through the close proximity of the magnetic signal to the Hall transducer. A precise, proportional voltage is provided by the low-offset, chopper-stabilized BiCMOS Hall IC, which is programmed for accuracy after packaging.

The output of the device has a positive slope (>$V_{IOUT(Q)}$) when an increasing current flows through the primary copper conduction path (from pins 1 and 2, to pins 3 and 4), which is the path used for current sensing. The internal resistance of this conductive path is 1.2 mΩ typical, providing low power

Continued on the next page…

Typical Application



Application 1. The ACS712 outputs an analog signal, V_{OUT}. that varies linearly with the uni- or bi-directional AC or DC primary sensed current, I_P, within the range specified. C_F is recommended for noise management, with values that depend on the application.

Description (continued)

loss. The thickness of the copper conductor allows survival of the device at up to 5× overcurrent conditions. The terminals of the conductive path are electrically isolated from the sensor leads (pins 5 through 8). This allows the ACS712 current sensor to be used in applications requiring electrical isolation without the use of opto-isolators or other costly isolation techniques.

The ACS712 is provided in a small, surface mount SOIC8 package. The leadframe is plated with 100% matte tin, which is compatible with standard lead (Pb) free printed circuit board assembly processes. Internally, the device is Pb-free, except for flip-chip high-temperature Pb-based solder balls, currently exempt from RoHS. The device is fully calibrated prior to shipment from the factory.

Selection Guide

Part Number	Packing*	T_A (°C)	Optimized Range, I_P (A)	Sensitivity, Sens (Typ) (mV/A)
ACS712ELCTR-05B-T	Tape and reel, 3000 pieces/reel	–40 to 85	±5	185
ACS712ELCTR-20A-T	Tape and reel, 3000 pieces/reel	–40 to 85	±20	100
ACS712ELCTR-30A-T	Tape and reel, 3000 pieces/reel	–40 to 85	±30	66

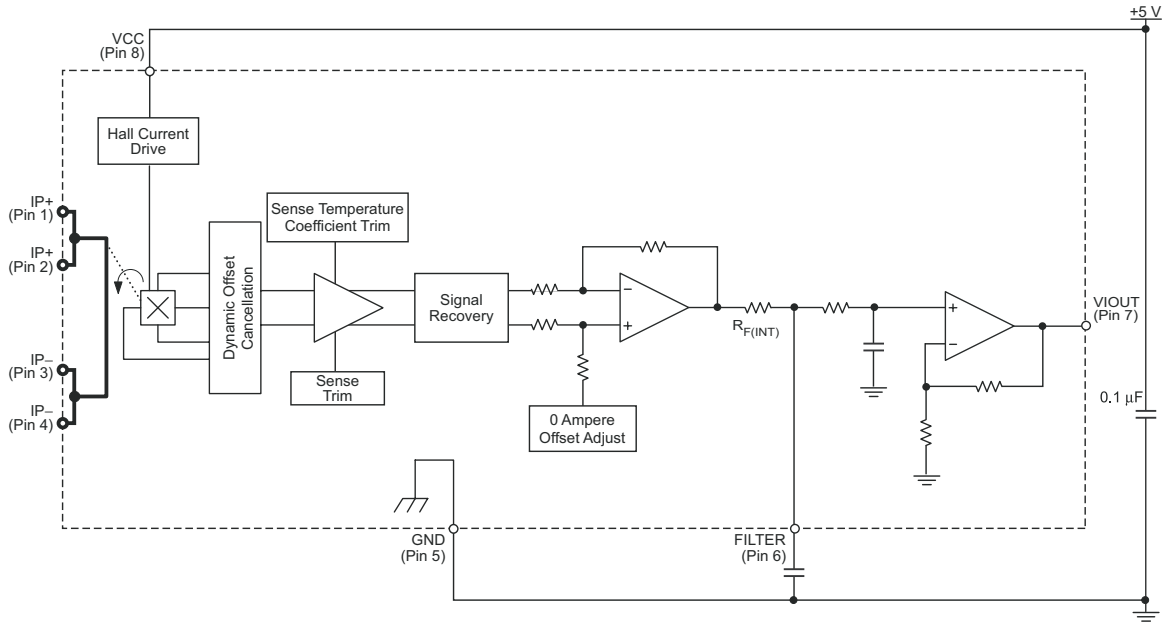
*Contact Allegro for additional packing options.

Absolute Maximum Ratings

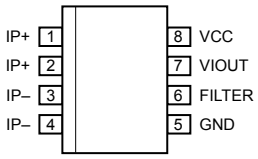
Characteristic	Symbol	Notes	Rating	Units
Supply Voltage	V_{CC}		8	V
Reverse Supply Voltage	V_{RCC}		–0.1	V
Output Voltage	V_{IOUT}		8	V
Reverse Output Voltage	V_{RIOUT}		–0.1	V
Reinforced Isolation Voltage	V_{ISO}	Pins 1-4 and 5-8; 60 Hz, 1 minute, T_A=25°C	2100	V
		Voltage applied to leadframe (Ip+ pins), based on IEC 60950	184	V_{peak}
Basic Isolation Voltage	$V_{ISO(bsc)}$	Pins 1-4 and 5-8; 60 Hz, 1 minute, T_A=25°C	1500	V
		Voltage applied to leadframe (Ip+ pins), based on IEC 60950	354	V_{peak}
Output Current Source	$I_{IOUT(Source)}$		3	mA
Output Current Sink	$I_{IOUT(Sink)}$		10	mA
Overcurrent Transient Tolerance	I_P	1 pulse, 100 ms	100	A
Nominal Operating Ambient Temperature	T_A	Range E	–40 to 85	ºC
Maximum Junction Temperature	T_J(max)		165	ºC
Storage Temperature	T_{stg}		–65 to 170	ºC

Parameter	Specification
Fire and Electric Shock	CAN/CSA-C22.2 No. 60950-1-03 UL 60950-1:2003 EN 60950-1:2001

Functional Block Diagram



Pin-out Diagram



Terminal List Table

Number	Name	Description
1 and 2	IP+	Terminals for current being sensed; fused internally
3 and 4	IP–	Terminals for current being sensed; fused internally
5	GND	Signal ground terminal
6	FILTER	Terminal for external capacitor that sets bandwidth
7	VIOUT	Analog output signal
8	VCC	Device power supply terminal

COMMON OPERATING CHARACTERISTICS[1] over full range of T_A, C_F = 1 nF, and V_{CC} = 5 V, unless otherwise specified

Characteristic	Symbol	Test Conditions	Min.	Typ.	Max.	Units
ELECTRICAL CHARACTERISTICS						
Supply Voltage	V_{CC}		4.5	5.0	5.5	V
Supply Current	I_{CC}	V_{CC} = 5.0 V, output open	–	10	13	mA
Output Capacitance Load	C_{LOAD}	VIOUT to GND	–	–	10	nF
Output Resistive Load	R_{LOAD}	VIOUT to GND	4.7	–	–	kΩ
Primary Conductor Resistance	$R_{PRIMARY}$	T_A = 25°C	–	1.2	–	mΩ
Rise Time	t_r	I_P = I_P(max), T_A = 25°C, C_{OUT} = open	–	5	–	µs
Frequency Bandwidth	f	–3 dB, T_A = 25°C; I_P is 10 A peak-to-peak	–	80	–	kHz
Nonlinearity	E_{LIN}	Over full range of I_P	–	1.5	–	%
Symmetry	E_{SYM}	Over full range of I_P	98	100	102	%
Zero Current Output Voltage	$V_{IOUT(Q)}$	Bidirectional; I_P = 0 A, T_A = 25°C	–	V_{CC} × 0.5	–	V
Power-On Time	t_{PO}	Output reaches 90% of steady-state level, T_J = 25°C, 20 A present on leadframe	–	35	–	µs
Magnetic Coupling[2]			–	12	–	G/A
Internal Filter Resistance[3]	$R_{F(INT)}$			1.7		kΩ

[1]Device may be operated at higher primary current levels, I_P, and ambient, T_A, and internal leadframe temperatures, T_A, provided that the Maximum Junction Temperature, T_J(max), is not exceeded.

[2]1G = 0.1 mT.

[3]$R_{F(INT)}$ forms an RC circuit via the FILTER pin.

COMMON THERMAL CHARACTERISTICS[1]

			Min.	Typ.	Max.	Units
Operating Internal Leadframe Temperature	T_A	E range	–40	–	85	°C
					Value	Units
Junction-to-Lead Thermal Resistance[2]	$R_{\theta JL}$	Mounted on the Allegro ASEK 712 evaluation board			5	°C/W
Junction-to-Ambient Thermal Resistance	$R_{\theta JA}$	Mounted on the Allegro 85-0322 evaluation board, includes the power consumed by the board			23	°C/W

[1]Additional thermal information is available on the Allegro website.

[2]The Allegro evaluation board has 1500 mm^2 of 2 oz. copper on each side, connected to pins 1 and 2, and to pins 3 and 4, with thermal vias connecting the layers. Performance values include the power consumed by the PCB. Further details on the board are available from the Frequently Asked Questions document on our website. Further information about board design and thermal performance also can be found in the Applications Information section of this datasheet.

x05B PERFORMANCE CHARACTERISTICS T_A = –40°C to 85°C[1], C_F = 1 nF, and V_{CC} = 5 V, unless otherwise specified

Characteristic	Symbol	Test Conditions	Min.	Typ.	Max.	Units
Optimized Accuracy Range	I_P		–5	–	5	A
Sensitivity	Sens	Over full range of I_P, T_A = 25°C	180	185	190	mV/A
Noise	$V_{NOISE(PP)}$	Peak-to-peak, T_A = 25°C, 185 mV/A programmed Sensitivity, C_F = 47 nF, C_{OUT} = open, 2 kHz bandwidth	–	21	–	mV
Zero Current Output Slope	$\Delta I_{OUT(Q)}$	T_A = –40°C to 25°C	–	–0.26	–	mV/°C
		T_A = 25°C to 150°C	–	–0.08	–	mV/°C
Sensitivity Slope	ΔSens	T_A = –40°C to 25°C	–	0.054	–	mV/A/°C
		T_A = 25°C to 150°C	–	–0.008	–	mV/A/°C
Total Output Error[2]	E_{TOT}	I_P =±5 A, T_A = 25°C	–	±1.5	–	%

[1]Device may be operated at higher primary current levels, I_P, and ambient temperatures, T_A, provided that the Maximum Junction Temperature, $T_{J(max)}$, is not exceeded.

[2]Percentage of I_P, with I_P = 5 A. Output filtered.

x20A PERFORMANCE CHARACTERISTICS T_A = –40°C to 85°C[1], C_F = 1 nF, and V_{CC} = 5 V, unless otherwise specified

Characteristic	Symbol	Test Conditions	Min.	Typ.	Max.	Units
Optimized Accuracy Range	I_P		–20	–	20	A
Sensitivity	Sens	Over full range of I_P, T_A = 25°C	96	100	104	mV/A
Noise	$V_{NOISE(PP)}$	Peak-to-peak, T_A = 25°C, 100 mV/A programmed Sensitivity, C_F = 47 nF, C_{OUT} = open, 2 kHz bandwidth	–	11	–	mV
Zero Current Output Slope	$\Delta I_{OUT(Q)}$	T_A = –40°C to 25°C	–	–0.34	–	mV/°C
		T_A = 25°C to 150°C	–	–0.07	–	mV/°C
Sensitivity Slope	ΔSens	T_A = –40°C to 25°C	–	0.017	–	mV/A/°C
		T_A = 25°C to 150°C	–	–0.004	–	mV/A/°C
Total Output Error[2]	E_{TOT}	I_P =±20 A, T_A = 25°C	–	±1.5	–	%

[1]Device may be operated at higher primary current levels, I_P, and ambient temperatures, T_A, provided that the Maximum Junction Temperature, T_J(max), is not exceeded.

[2]Percentage of I_P, with I_P = 20 A. Output filtered.

x30A PERFORMANCE CHARACTERISTICS T_A = –40°C to 85°C[1], C_F = 1 nF, and V_{CC} = 5 V, unless otherwise specified

Characteristic	Symbol	Test Conditions	Min.	Typ.	Max.	Units
Optimized Accuracy Range	I_P		–30	–	30	A
Sensitivity	Sens	Over full range of I_P, T_A = 25°C	64	66	68	mV/A
Noise	$V_{NOISE(PP)}$	Peak-to-peak, T_A = 25°C, 66 mV/A programmed Sensitivity, C_F = 47 nF, C_{OUT} = open, 2 kHz bandwidth	–	7	–	mV
Zero Current Output Slope	$\Delta I_{OUT(Q)}$	T_A = –40°C to 25°C	–	–0.35	–	mV/°C
		T_A = 25°C to 150°C	–	–0.08	–	mV/°C
Sensitivity Slope	ΔSens	T_A = –40°C to 25°C	–	0.007	–	mV/A/°C
		T_A = 25°C to 150°C	–	–0.002	–	mV/A/°C
Total Output Error[2]	E_{TOT}	I_P = ±30 A, T_A = 25°C	–	±1.5	–	%

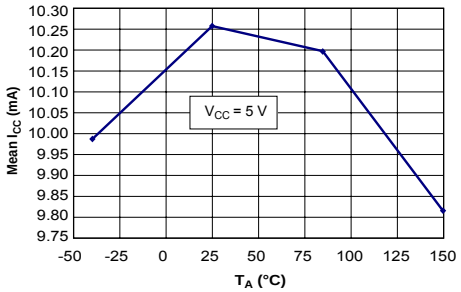
[1]Device may be operated at higher primary current levels, I_P, and ambient temperatures, T_A, provided that the Maximum Junction Temperature, T_J(max), is not exceeded.

[2]Percentage of I_P, with I_P = 30 A. Output filtered.

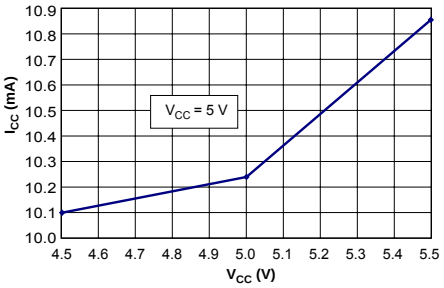
Characteristic Performance

I_P = 5 A, unless otherwise specified

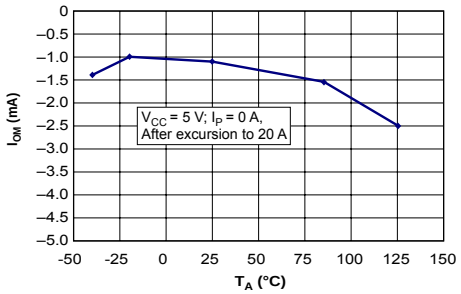
Mean Supply Current versus Ambient Temperature



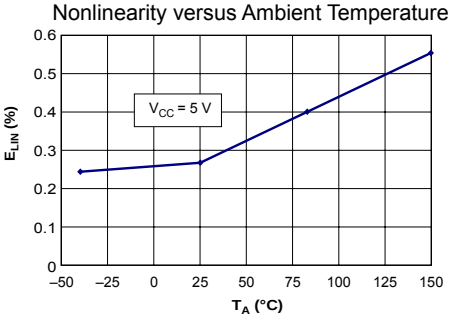
Supply Current versus Supply Voltage



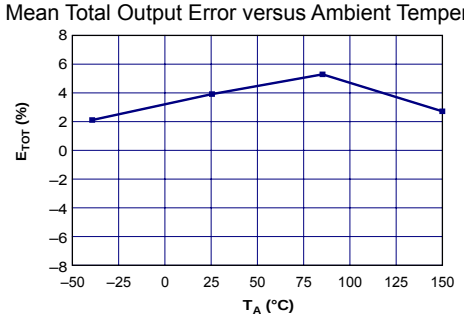
Magnetic Offset versus Ambient Temperature



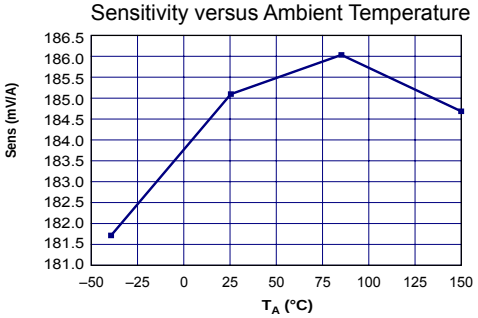
Nonlinearity versus Ambient Temperature



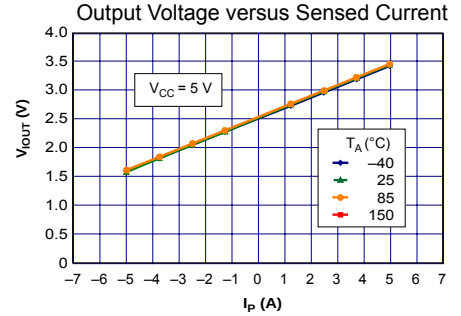
Mean Total Output Error versus Ambient Temperature



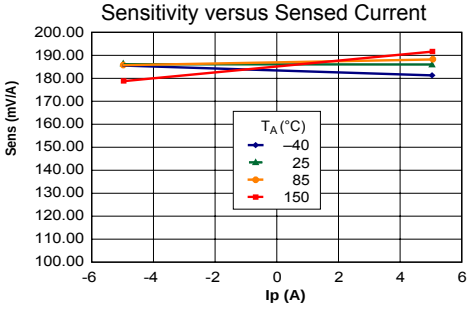
Sensitivity versus Ambient Temperature



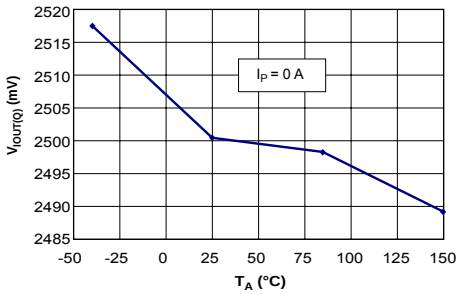
Output Voltage versus Sensed Current



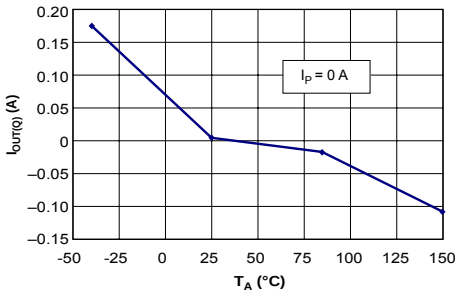
Sensitivity versus Sensed Current



0 A Output Voltage versus Ambient Temperature



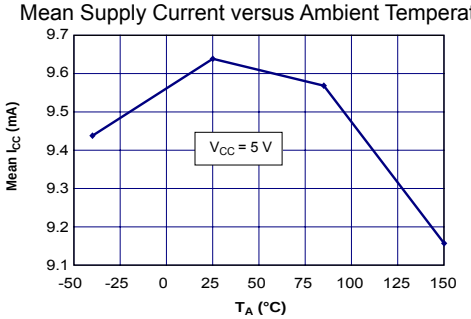
0 A Output Voltage Current versus Ambient Temperature



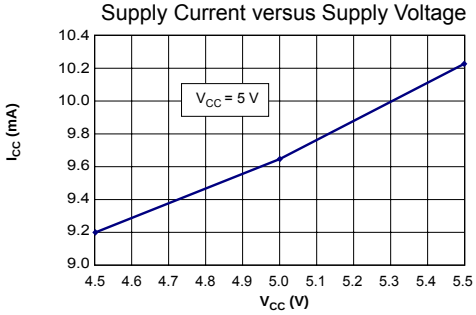
Characteristic Performance

I_P = 20 A, unless otherwise specified

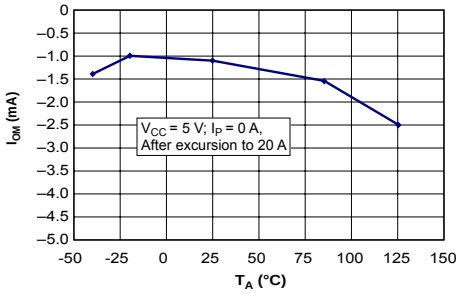
Mean Supply Current versus Ambient Temperature



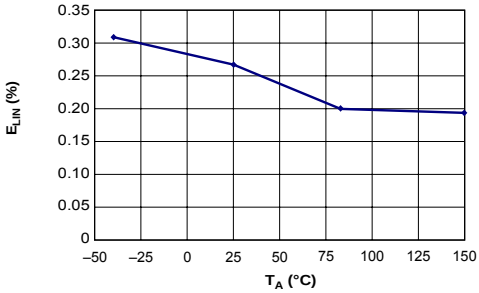
Supply Current versus Supply Voltage



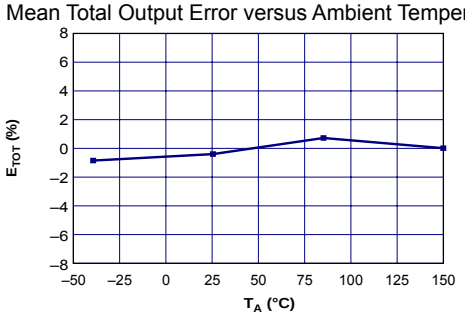
Magnetic Offset versus Ambient Temperature



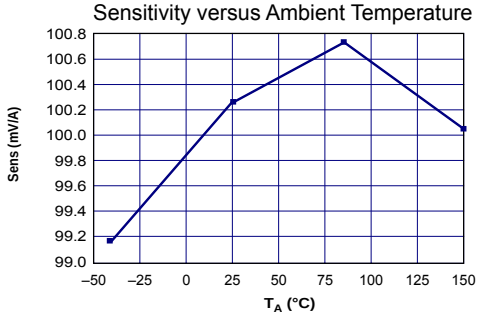
Nonlinearity versus Ambient Temperature



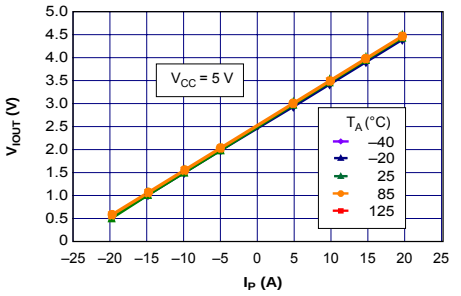
Mean Total Output Error versus Ambient Temperature



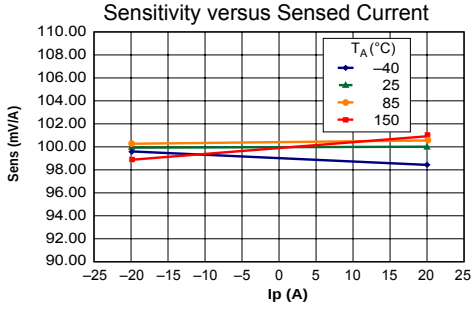
Sensitivity versus Ambient Temperature



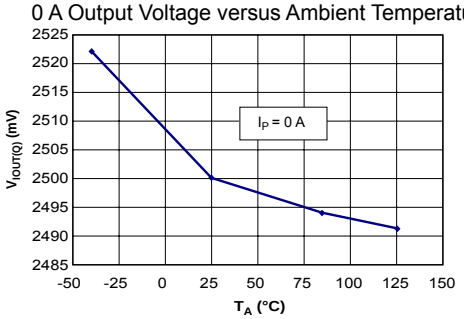
Output Voltage versus Sensed Current



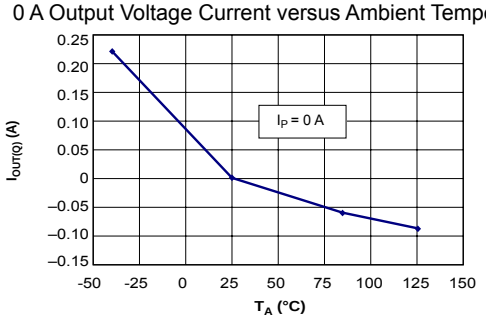
Sensitivity versus Sensed Current



0 A Output Voltage versus Ambient Temperature



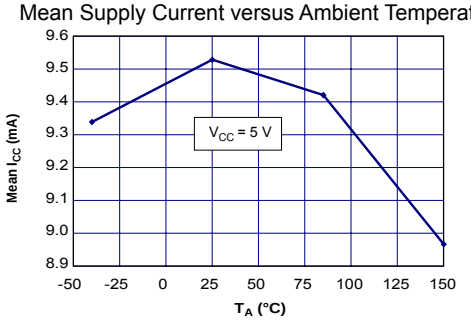
0 A Output Voltage Current versus Ambient Temperature



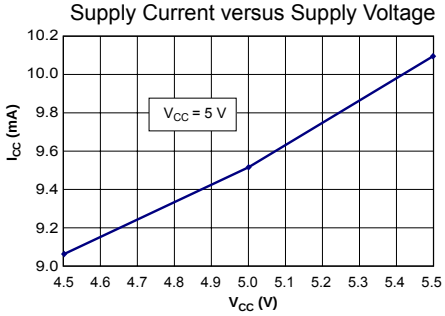
Characteristic Performance

I_P = 30 A, unless otherwise specified

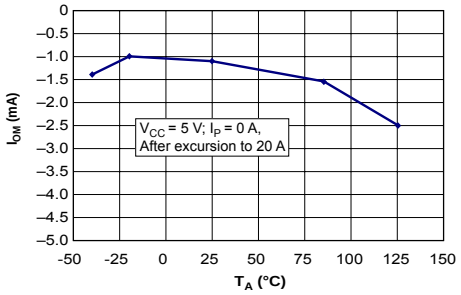
Mean Supply Current versus Ambient Temperature



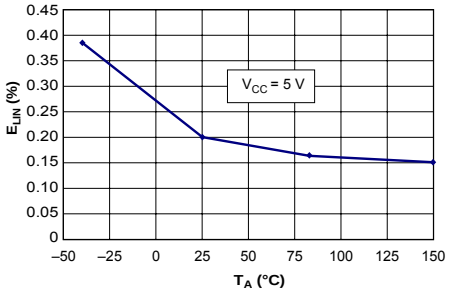
Supply Current versus Supply Voltage



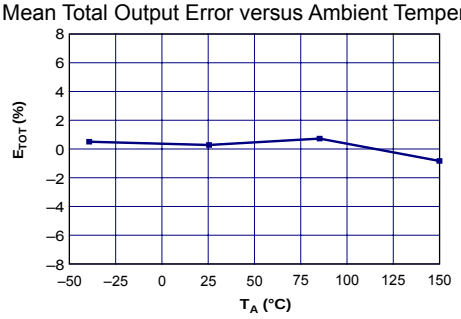
Magnetic Offset versus Ambient Temperature



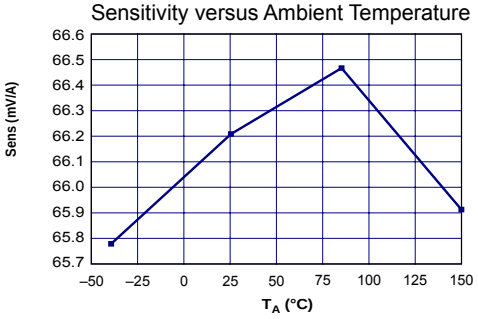
Nonlinearity versus Ambient Temperature



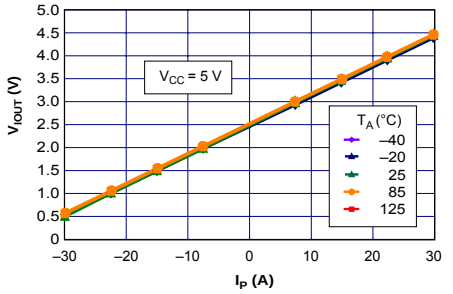
Mean Total Output Error versus Ambient Temperature



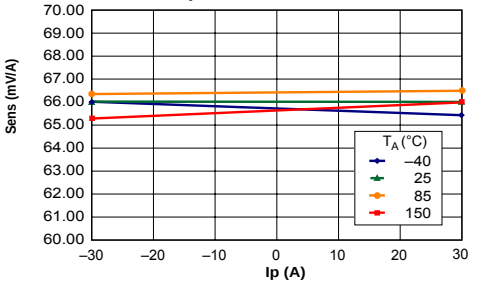
Sensitivity versus Ambient Temperature



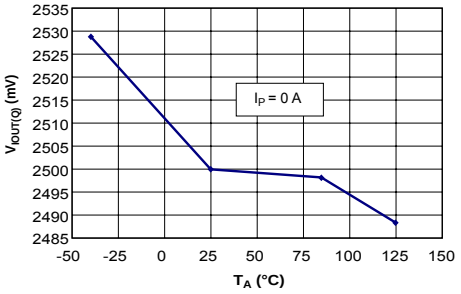
Output Voltage versus Sensed Current



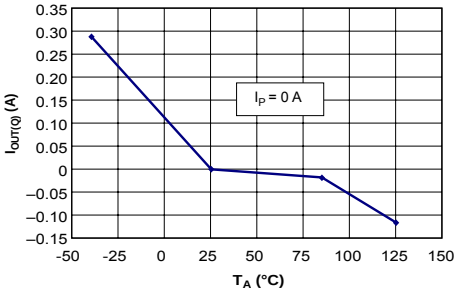
Sensitivity versus Sensed Current



0 A Output Voltage versus Ambient Temperature



0 A Output Voltage Current versus Ambient Temperature



Definitions of Accuracy Characteristics

Sensitivity (Sens). The change in sensor output in response to a 1 A change through the primary conductor. The sensitivity is the product of the magnetic circuit sensitivity (G/A) and the linear IC amplifier gain (mV/G). The linear IC amplifier gain is programmed at the factory to optimize the sensitivity (mV/A) for the full-scale current of the device.

Noise (V_{NOISE}). The product of the linear IC amplifier gain (mV/G) and the noise floor for the Allegro Hall effect linear IC (≈1 G). The noise floor is derived from the thermal and shot noise observed in Hall elements. Dividing the noise (mV) by the sensitivity (mV/A) provides the smallest current that the device is able to resolve.

Linearity (E_{LIN}). The degree to which the voltage output from the sensor varies in direct proportion to the primary current through its full-scale amplitude. Nonlinearity in the output can be attributed to the saturation of the flux concentrator approaching the full-scale current. The following equation is used to derive the linearity:

$$100\left\{1-\left[\frac{\Delta\text{ gain}\times\%\text{ sat}\,(V_{\text{IOUT\_full-scale amperes}}-V_{\text{IOUT(Q)}})}{2\,(V_{\text{IOUT\_half-scale amperes}}-V_{\text{IOUT(Q)}})}\right]\right\}$$

where $V_{\text{IOUT\_full-scale amperes}}$ = the output voltage (V) when the sensed current approximates full-scale $\pm I_P$.

Symmetry (E_{SYM}). The degree to which the absolute voltage output from the sensor varies in proportion to either a positive or negative full-scale primary current. The following formula is used to derive symmetry:

$$100\left(\frac{V_{\text{IOUT\_+ full-scale amperes}}-V_{\text{IOUT(Q)}}}{V_{\text{IOUT(Q)}}-V_{\text{IOUT\_–full-scale amperes}}}\right)$$

Quiescent output voltage ($V_{IOUT(Q)}$). The output of the sensor when the primary current is zero. For a unipolar supply voltage, it nominally remains at $V_{CC}/2$. Thus, V_{CC} = 5 V translates into $V_{IOUT(Q)}$ = 2.5 V. Variation in $V_{IOUT(Q)}$ can be attributed to the resolution of the Allegro linear IC quiescent voltage trim and thermal drift.

Electrical offset voltage (V_{OE}). The deviation of the device output from its ideal quiescent value of $V_{CC}/2$ due to nonmagnetic causes. To convert this voltage to amperes, divide by the device sensitivity, Sens.

Accuracy (E_{TOT}). The accuracy represents the maximum deviation of the actual output from its ideal value. This is also known as the total ouput error. The accuracy is illustrated graphically in the output voltage versus current chart at right.

Accuracy is divided into four areas:

- **0 A at 25°C.** Accuracy of sensing zero current flow at 25°C, without the effects of temperature.
- **0 A over Δ temperature.** Accuracy of sensing zero current flow including temperature effects.
- **Full-scale current at 25°C.** Accuracy of sensing the full-scale current at 25°C, without the effects of temperature.
- **Full-scale current over Δ temperature.** Accuracy of sensing full-scale current flow including temperature effects.

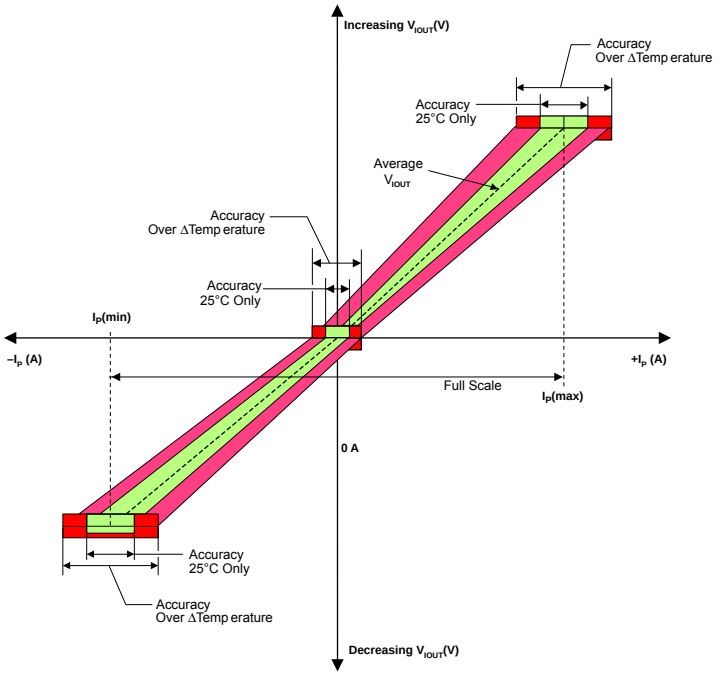
Ratiometry. The ratiometric feature means that its 0 A output, $V_{IOUT(Q)}$, (nominally equal to $V_{CC}/2$) and sensitivity, Sens, are proportional to its supply voltage, V_{CC}. The following formula is used to derive the ratiometric change in 0 A output voltage, $\Delta V_{IOUT(Q)RAT}$ (%).

$$100\left(\frac{V_{\text{IOUT(Q)VCC}}/V_{\text{IOUT(Q)5V}}}{V_{CC}/5\text{ V}}\right)$$

The ratiometric change in sensitivity, $\Delta Sens_{RAT}$ (%), is defined as:

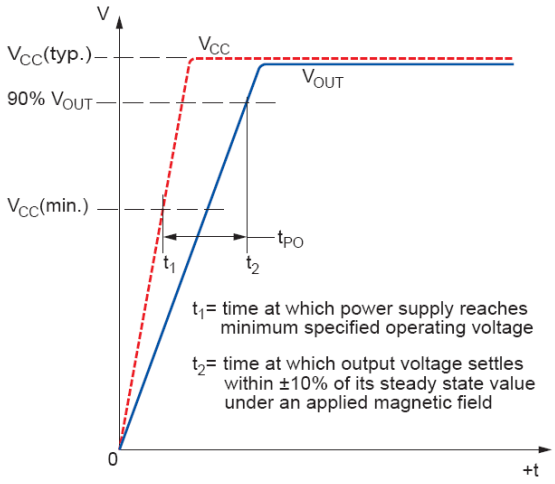
$$100\left(\frac{Sens_{VCC}/Sens_{5V}}{V_{CC}/5\text{ V}}\right)$$

Output Voltage versus Sensed Current
Accuracy at 0 A and at Full-Scale Current

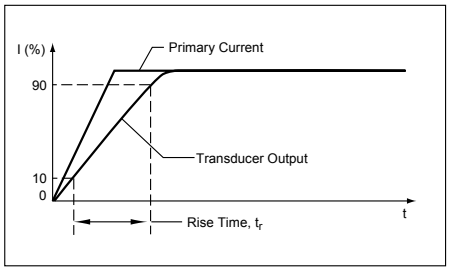


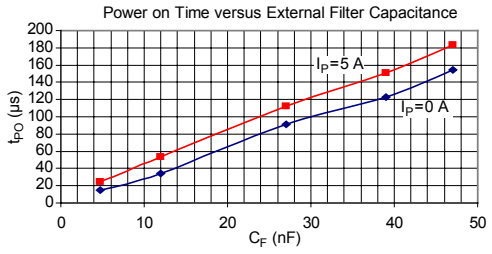
Definitions of Dynamic Response Characteristics

Power-On Time (t_{PO}). When the supply is ramped to its operating voltage, the device requires a finite time to power its internal components before responding to an input magnetic field. Power-On Time, t_{PO} , is defined as the time it takes for the output voltage to settle within ±10% of its steady state value under an applied magnetic field, after the power supply has reached its minimum specified operating voltage, V_{CC}(min), as shown in the chart at right.

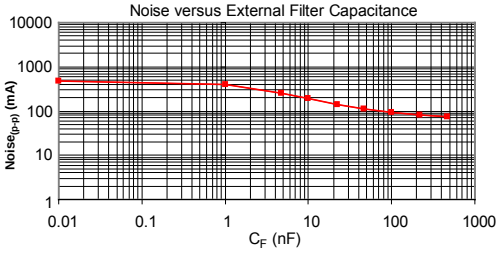


Rise time (t_r). The time interval between a) when the sensor reaches 10% of its full scale value, and b) when it reaches 90% of its full scale value. The rise time to a step response is used to derive the bandwidth of the current sensor, in which $f(-3\text{ dB}) = 0.35/t_r$. Both t_r and $t_{RESPONSE}$ are detrimentally affected by eddy current losses observed in the conductive IC ground plane.



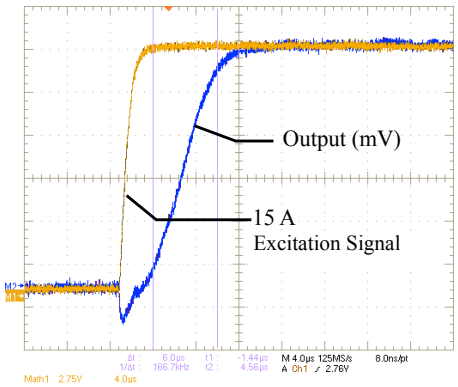


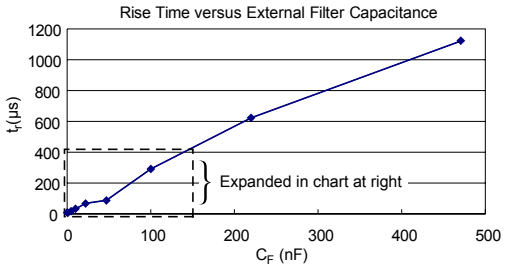
Noise vs. Filter Cap



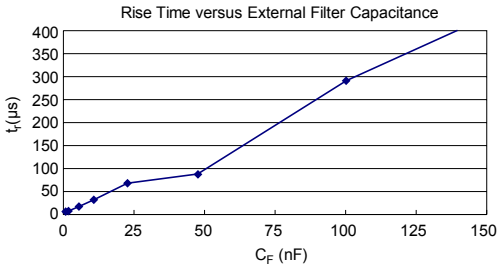
Step Response

T_A=25°C





C_F (nF)	t_r (µs)
0	6.6
1	7.7
4.7	17.4
10	32.1
22	68.2
47	88.2
100	291.3
220	623.0
470	1120.0

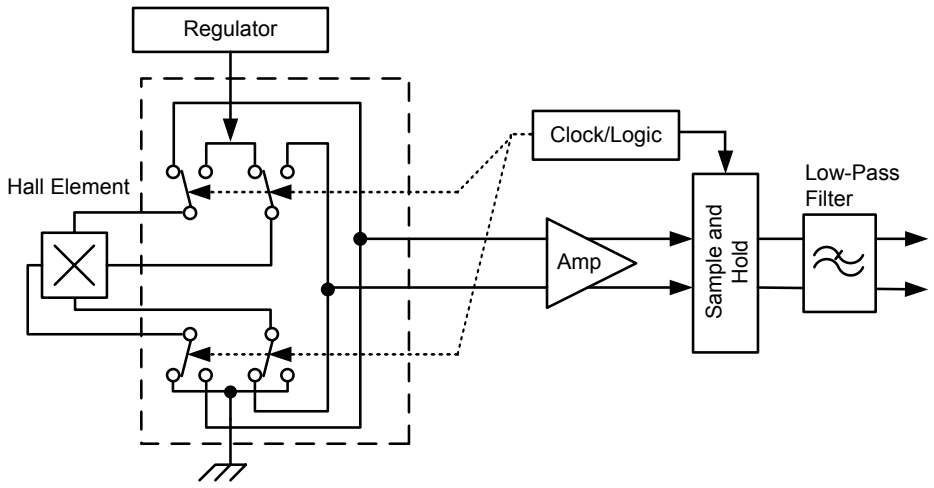


Chopper Stabilization Technique

Chopper Stabilization is an innovative circuit technique that is used to minimize the offset voltage of a Hall element and an associated on-chip amplifier. Allegro patented a Chopper Stabilization technique that nearly eliminates Hall IC output drift induced by temperature or package stress effects. This offset reduction technique is based on a signal modulation-demodulation process. Modulation is used to separate the undesired dc offset signal from the magnetically induced signal in the frequency domain. Then, using a low-pass filter, the modulated dc offset is suppressed while the magnetically induced signal passes through the filter.

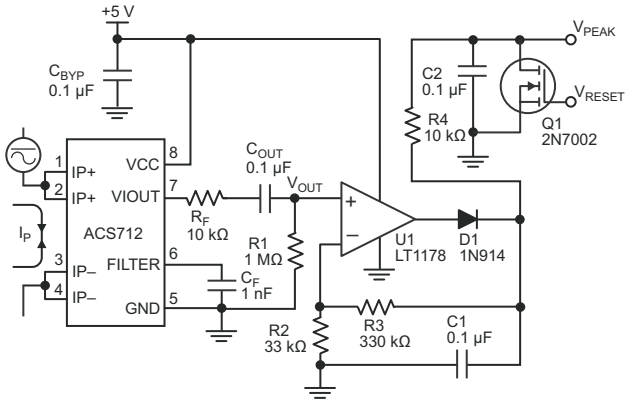
As a result of this chopper stabilization approach, the output voltage from the Hall IC is desensitized to the effects of temperature and mechanical stress. This technique produces devices that have an extremely stable Electrical Offset Voltage, are immune to thermal stress, and have precise recoverability after temperature cycling.

This technique is made possible through the use of a BiCMOS process that allows the use of low-offset and low-noise amplifiers in combination with high-density logic integration and sample and hold circuits.

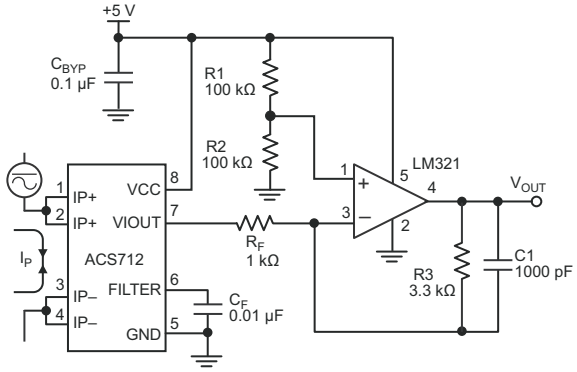


Concept of Chopper Stabilization Technique

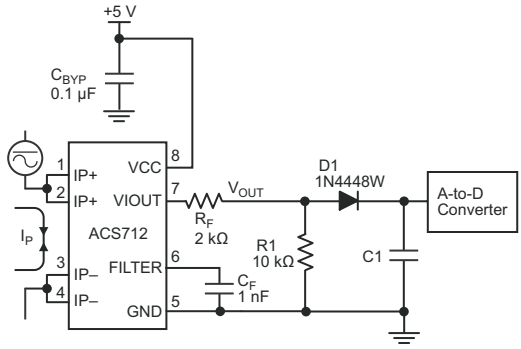
Typical Applications



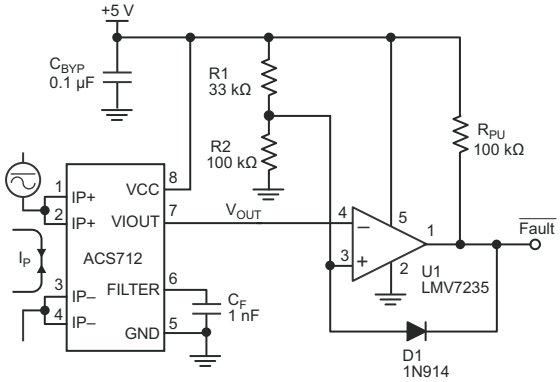
Application 2. Peak Detecting Circuit



Application 3. This configuration increases gain to 610 mV/A (tested using the ACS712ELC-05A).



Application 4. Rectified Output. 3.3 V scaling and rectification application for A-to-D converters. Replaces current transformer solutions with simpler ACS circuit. C1 is a function of the load resistance and filtering desired. R1 can be omitted if the full range is desired.



Application 5. 10 A Overcurrent Fault Latch. Fault threshold set by R1 and R2. This circuit latches an overcurrent fault and holds it until the 5 V rail is powered down.

Improving Sensing System Accuracy Using the FILTER Pin

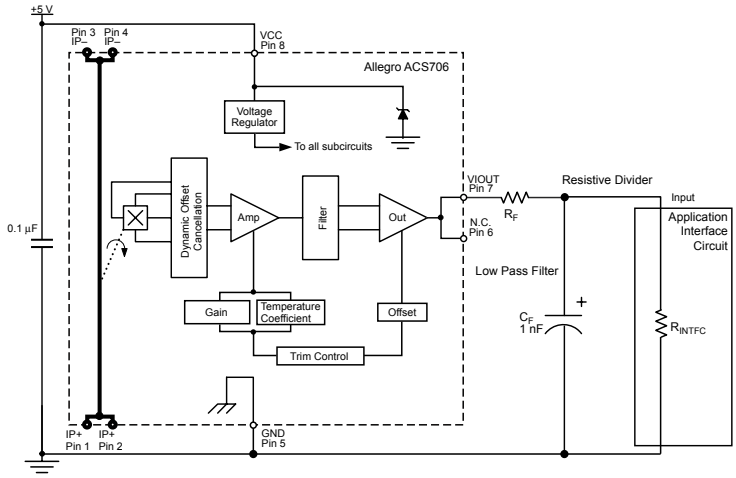
In low-frequency sensing applications, it is often advantageous to add a simple RC filter to the output of the sensor. Such a low-pass filter improves the signal-to-noise ratio, and therefore the resolution, of the sensor output signal. However, the addition of an RC filter to the output of a sensor IC can result in undesirable sensor output attenuation — even for dc signals.

Signal attenuation, ΔV_{ATT}, is a result of the resistive divider effect between the resistance of the external filter, R_F (see Application 6), and the input impedance and resistance of the customer interface circuit, R_{INTFC}. The transfer function of this resistive divider is given by:

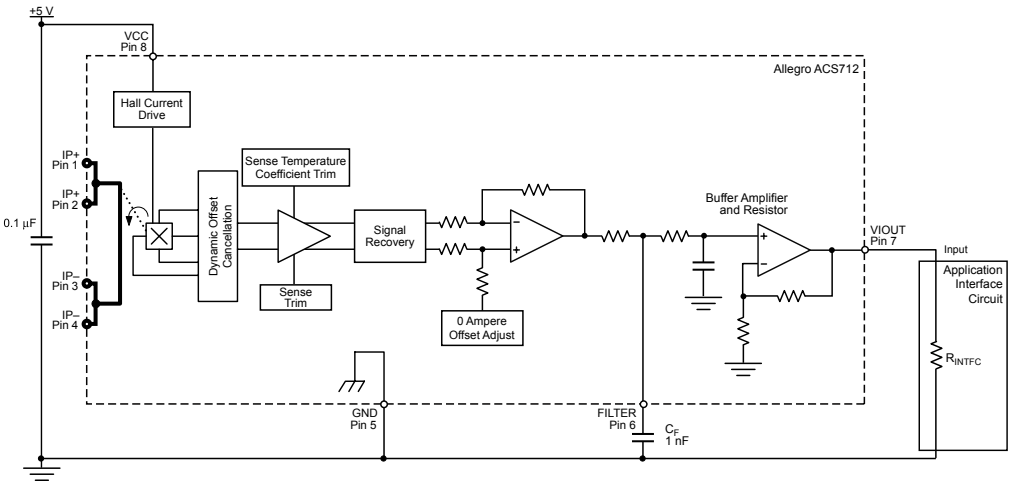
$$\Delta V_{ATT} = V_{IOUT}\left(\frac{R_{INTFC}}{R_F + R_{INTFC}}\right) .$$

Even if R_F and R_{INTFC} are designed to match, the two individual resistance values will most likely drift by different amounts over temperature. Therefore, signal attenuation will vary as a function of temperature. Note that, in many cases, the input impedance, R_{INTFC}, of a typical analog-to-digital converter (ADC) can be as low as 10 kΩ.

The ACS712 contains an internal resistor, a FILTER pin connection to the printed circuit board, and an internal buffer amplifier. With this circuit architecture, users can implement a simple RC filter via the addition of a capacitor, C_F (see Application 7) from the FILTER pin to ground. The buffer amplifier inside of the ACS712 (located after the internal resistor and FILTER pin connection) eliminates the attenuation caused by the resistive divider effect described in the equation for ΔV_{ATT}. Therefore, the ACS712 device is ideal for use in high-accuracy applications that cannot afford the signal attenuation associated with the use of an external RC low-pass filter.

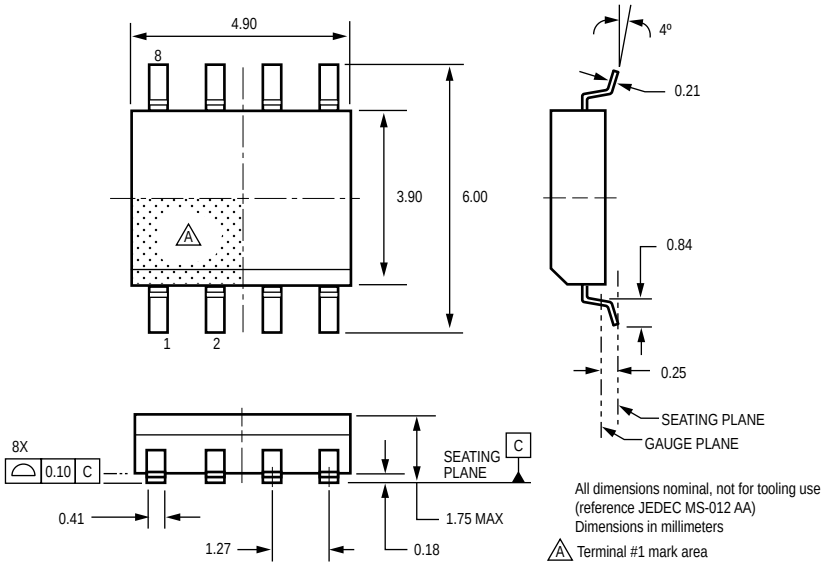


Application 6. When a low pass filter is constructed externally to a standard Hall effect device, a resistive divider may exist between the filter resistor, R_F, and the resistance of the customer interface circuit, R_{INTFC}. This resistive divider will cause excessive attenuation, as given by the transfer function for ΔV_{ATT}.



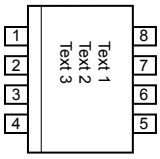
Application 7. Using the FILTER pin provided on the ACS712 eliminates the attenuation effects of the resistor divider between R_F and R_{INTFC}, shown in Application 6.

Package LC, 8-pin SOIC



Package Branding

Two alternative patterns are used



ACS712T *RLCPPP* *YYWWA*	ACS	Allegro Current Sensor
	712	Device family number
	T	Indicator of 100% matte tin leadframe plating
	R	Operating ambient temperature range code
	LC	Package type designator
	PPP	Primary sensed current
	YY	Date code: Calendar year (last two digits)
	WW	Date code: Calendar week
	A	Date code: Shift code

ACS712T *RLCPPP* *L...L* *YYWW*	ACS	Allegro Current Sensor
	712	Device family number
	T	Indicator of 100% matte tin leadframe plating
	R	Operating ambient temperature range code
	LC	Package type designator
	PPP	Primary sensed current
	L...L	Lot code
	YY	Date code: Calendar year (last two digits)
	WW	Date code: Calendar week

For the latest version of this document, visit our website:

www.allegromicro.com