Bangladesh University of Engineering and Technology

Department of Electrical and Electronic Engineering

EEE 460(June 2023)

Optoelectronics Laboratory

**Final Project Report**

**Section: G2 Group: 01**

Experimental Setup to Measure and Analysis the I-V characteristics of a Solar Panel

**Course Instructors:**

**Dr. Md Zunaid Baten**

**Redwanul Mahbbub Talukder**

**Signature of Instructor: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Academic Honesty Statement:**

**IMPORTANT! Please carefully read and sign the Academic Honesty Statement, below. Type the student ID and name, and put your signature. *You will not receive credit for this project experiment unless this statement is signed in the presence of your lab instructor.***

*“In signing this statement, We hereby certify that the work on this project is our own and that we have not copied the work of any other students (past or present), and cited all relevant sources while completing this project. We understand that if we fail to honor this agreement, We will each receive a score of ZERO for this project and be subject to failure of this course.”*

|  |  |  |
| --- | --- | --- |
| **Signature: \_\_\_\_\_\_Tasfiq\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  **Full Name: Md. Tasfiq Rahman**  **Student ID: 1806116** | **Signature: \_\_Samiul\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  **Full Name: Samiul Islam Shadin**  **Student ID:1806151** |  |
| **Signature: \_\_\_Siyam\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  **Full Name: Omar Faruk Siyam**  **Student ID: 1806159** | **Signature: \_\_\_\_\_\_\_\_Meherab\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  **Full Name: Md. Meherab Hossain**  **Student ID: 1806182** |  |

# Abstract

|  |
| --- |
| Energy harvesting through solar technology is one of the promising sources to meet the requirement of empowering renewable energy source and to reduce carbon emission. Utilization of photovoltaic solar cell ranges from daily household apparatus to energizing the component used for space mission. Before operating PV cell, it is crucial to understand its current voltage relationship, its maximum efficiency and other physical parameter to determine the optimum region to be used for. However, in this characterization procedure, manual contact can create noise which affects in actual data calculation. To overcome this problem, a microcontroller-based circuit has been designed to collect data by varying resistance. For this purpose, decoder and metal oxide semiconductor transistor has been used. For displaying plot, library from Python has been used. |

# Introduction

Current voltage (I-V) characteristics of a solar cell contains a number of hidden variables that is required to understand the performance of a solar cell. A common approach to measure this characteristic is to connect a PV solar cell with a variable resistance and read the output voltage using voltmeter and to read current using potentiometer. The final task is to plot current vs voltage. Though this approach is widely used, it is required to be in physical contact wit solar connection. If the situation is like this where a device is being operated remotely powered by solar panel (eg using in agricultural land), measuring I-V characteristic without physically contact with solar cell would be a smart choice. The device we are going to implement would pave the way for this.

Base on previous work, related to automated I-V curve tracer, we have tried to use the number of microcontroller pin more efficiently by introducing decoder circuit. We have also tried to measure solar irradiance in a cost-effective way. To interpret the censored data into I-V characteristics, we have designed an algorithm using least square regression. From the I-V characteristics, efficiency, fill factor, series and shunt resistance of the solar panel has been measured. In addition, we have formulated a way to design the operating range of our circuit which can also be used as user manual.

# Design

### Identification of Scope

I-V characteristics cannot be measured without measuring intensity of solar radiation. The device used for this purpose in known as irradiance meter which is the costliest tools used in I-V characterization for small solar panel. Another cheap component sensing light intensity is light dependent resistor whose resistance decreases with the increase of light intensity. So, instead of using irradiance meter directly, mapping resistance of LDR with the reading of irradiance meter would be a new approach.

Instead of connecting Arduino pin with MOSFET, using active high decoder IC would be enable us controlling more transistor switching using a few numbers of Arduino pins.

Every device has some limitation. If a user in informed about it, he would be able to made the best use of the device. Based on the value of our used resistance, we have formulated a way to determine the range of open circuit voltage and corresponding short circuit current. If a PV cell is lies within the range, our device can be used to measure its I-V characteristics.

### Literature Review

PV cell was characterized by varying resistance with the switching of transistor. Data was displayed through M2M communication protocol [1]. Based on this work, an approach was design based on voltage, and current sensor to measure data quickly [2]. In addition, images of photoluminescence spectra have been used to draw IV characteristics based on neural network [3].

### Formulation of Problem

The first task we emphasized was to implement a system where resistance will be varied automatically. Next mission we undertook is to confine our resistance variation process within the range the solar panel requires.

### Analysis

We first observed the variation MOSFET drain to source resistance. We have found that it is varied by the duty cycle of PWM signal for analogWrite. Next We proceed to use digital potentiometer. But its minimum resolution we found was 60 ohm. But the operating resistance for our first attempted solar cell requires 1 ohm to 200 ohm. Finally we took Z44 MOSFET for final operation.

Equation to find out the resistance of LDR:

Equation for curve fitting by least square method:

The best fit values are then,

Where B=b and A=exp(a)

Current for solar cell I=-Iph+-Io

## Design Method

## Circuit Diagram

## 

## Figure: Block Diagram of the circuit

## 

## 

## Figure : Different Components of the circuit

## 

## 

## Figure: Components and overall circuit

## Hardware Design

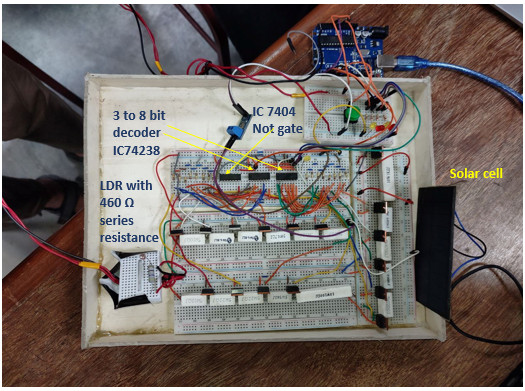


Figure: Final implementation of the circuit

## Full Source Code of Firmware

## Table 1: Code for Arduino Uno

## 

|  |
| --- |
| //#include "ACS712.h"  //ACS712 sensor(ACS712\_05B, A3);  // Define pins for output  const int pinA = 2; // LSB -> A0  const int pinB = 3;  const int pinC = 4;  const int pinD = 5; // MSB -> A3  // Define analog input pins for adjusting clock period  const int analogInput1 = A0; // Connect input 1 to A0  const int analogInput2 = A5; // Connect input 2 to A5  const int analogInput3 = A3; // Connect input 3 to A3  // Define pin for the push button  const int buttonPin = 6; // Connect the push button to pin 6  float V = 0;  // Variables to store current count and time  int count = 0;  unsigned long previousMillis = 0;  const int numReadings = 10; // Number of analog readings per clock cycle  float readingsInput1[numReadings]; // Array to store analog readings for input 1  float readingsInput2[numReadings]; // Array to store analog readings for input 2  float readingsInput3[numReadings]; // Array to store analog readings for input 2  int readIndex = 0; // Index of the current reading  float input1Value = 0;  float input2Value = 0;  float input3Value = 0;  bool buttonState = false; // Current state of the button  bool lastButtonState = false; // Previous state of the button  unsigned long lastDebounceTime = 0; // Time at which the button last changed state  unsigned long debounceDelay = 50; // Delay in milliseconds for debouncing the button  bool countingStarted = false; // Flag to indicate if counting has started  bool countingCompleted = false; // Flag to indicate if counting has completed  int adc = 0;  float voltage\_acs =0;  float current = 0;  void setup() {  // Initialize Serial communication  Serial.begin(9600); // Set baud rate to 9600 (or any other baud rate you prefer)  // Initialize digital pins as outputs  pinMode(pinA, OUTPUT);  pinMode(pinB, OUTPUT);  pinMode(pinC, OUTPUT);  pinMode(pinD, OUTPUT);  // Initialize analog input pins  pinMode(analogInput1, INPUT);  pinMode(analogInput2, INPUT);  // Initialize the arrays of readings to 0  for (int i = 0; i < numReadings; i++) {  readingsInput1[i] = 0;  readingsInput2[i] = 0;  readingsInput3[i] = 0;  }  // Initialize button pin  pinMode(buttonPin, INPUT\_PULLUP); // Use internal pull-up resistor  // sensor.calibrate();  }  void loop() {  // Read the state of the button  int reading = digitalRead(buttonPin);  // Check if the button state has changed  if (reading != lastButtonState) {  lastDebounceTime = millis(); // Reset debounce timer  }  // Check if debounce delay has passed  if ((millis() - lastDebounceTime) > debounceDelay) {  // Update button state if it has been stable for a certain time  if (reading != buttonState) {  buttonState = reading;  // If the button is pressed (LOW), start the operation  if (buttonState == LOW && !countingStarted) {  countingStarted = true; // Set the flag to indicate counting has started  // Get current time  unsigned long currentMillis = millis();  // Calculate clock period based on analog input values  unsigned long period = 7000; // Map input2 to adjust the range furtherr  // Reset count and previousMillis  count = 0;  previousMillis = currentMillis;  // Perform counting and output operations  while (millis() - previousMillis < period \* 16) { // Count for 16 cycles  if (millis() - previousMillis >= period) {  // Increment count and reset if it reaches 16 (4-bit overflow)  count++;  if (count == 16) {  count = 0;  countingCompleted = true; // Set flag to indicate counting has completed  break; // Exit the counting loop once completed  }  // Read analog input 1 multiple times per clock cycle  for (int i = 0; i < numReadings; i++) {  readingsInput1[i] = analogRead(analogInput1);  delay(100); // Delay between readings, adjust as needed  // Serial.println(readingsInput1[i]);  }  // Read analog input 2 multiple times per clock cycle  for (int i = 0; i < numReadings; i++) {  readingsInput2[i] = analogRead(analogInput2);  delay(100); // Delay between readings, adjust as needed  }  // Read analog input 2 multiple times per clock cycle  for (int i = 0; i < numReadings; i++) {  readingsInput3[i] = analogRead(analogInput3);  delay(100); // Delay between readings, adjust as needed  }  // Calculate average of analog input 1 readings  input1Value = 0;  for (int i = 0; i < numReadings; i++) {  input1Value += readingsInput1[i];  }  input1Value /= numReadings;  input1Value = input1Value\*(25.0/1024.0);  // Calculate average of analog input 2 readings  input2Value = 0;  for (int i = 0; i < numReadings; i++) {  input2Value += readingsInput2[i];  }  input2Value /= numReadings;  // input2Value = (460\*input2Value)/(5-input2Value);  // Calculate average of analog input 2 readings  input3Value = 0;  for (int i = 0; i < numReadings; i++) {  input3Value += readingsInput3[i];  }  input3Value /= numReadings;  voltage\_acs = input3Value\*5/1023.0;  input3Value = (voltage\_acs-2.5)/0.185;  // Serial.print("Current : ");  // Initialize the arrays of readings to 0  for (int i = 0; i < numReadings; i++) {  readingsInput1[i] = 0;  readingsInput2[i] = 0;  readingsInput3[i] = 0;  }  // Print input values to serial monitor  // Serial.print("Input 1: ");  Serial.print(input1Value);  // Serial.print("\t Input 2: ");  Serial.print(" ");  Serial.println(input2Value);  // Serial.print(" ");  // Serial.println(input3Value);  delay(1000);  // Output binary count to the pins  digitalWrite(pinA, (count >> 0) & 0x01);  digitalWrite(pinB, (count >> 1) & 0x01);  digitalWrite(pinC, (count >> 2) & 0x01);  digitalWrite(pinD, (count >> 3) & 0x01);  delay(1000);  // Update previousMillis  previousMillis += period;  }  }  // Read analog input 1 multiple times per clock cycle  for (int i = 0; i < numReadings; i++) {  readingsInput1[i] = analogRead(analogInput1);  delay(100); // Delay between readings, adjust as needed  // Serial.println(readingsInput1[i]);  }  // Read analog input 2 multiple times per clock cycle  for (int i = 0; i < numReadings; i++) {  readingsInput2[i] = analogRead(analogInput2);  delay(100); // Delay between readings, adjust as needed  }  // Read analog input 2 multiple times per clock cycle  for (int i = 0; i < numReadings; i++) {  readingsInput3[i] = analogRead(analogInput3);  delay(100); // Delay between readings, adjust as needed  }  // Calculate average of analog input 1 readings  input1Value = 0;  for (int i = 0; i < numReadings; i++) {  input1Value += readingsInput1[i];  }  input1Value /= numReadings;  input1Value = input1Value\*(25.0/1024.0);  // Calculate average of analog input 2 readings  input2Value = 0;  for (int i = 0; i < numReadings; i++) {  input2Value += readingsInput2[i];  }  input2Value /= numReadings;  // input2Value = (460\*input2Value)/(5-input2Value);  // Calculate average of analog input 2 readings  input3Value = 0;  for (int i = 0; i < numReadings; i++) {  input3Value += readingsInput3[i];  }  input3Value /= numReadings;  voltage\_acs = input3Value\*5/1023.0;  input3Value = (voltage\_acs-2.5)/0.185;  // Serial.print("Current : ");  // Initialize the arrays of readings to 0  for (int i = 0; i < numReadings; i++) {  readingsInput1[i] = 0;  readingsInput2[i] = 0;  readingsInput3[i] = 0;  }  // Print input values to serial monitor  // Serial.print("Input 1: ");  Serial.print(input1Value);  // Serial.print("\t Input 2: ");  Serial.print(" ");  Serial.println(input2Value);  // Serial.print(" ");  // Serial.println(input3Value);  // Set all outputs low after one cycle  digitalWrite(pinA, LOW);  digitalWrite(pinB, LOW);  digitalWrite(pinC, LOW);  digitalWrite(pinD, LOW);  }  }  }  // If button is released and counting has started and completed, reset the flags  if (buttonState == HIGH && countingStarted && countingCompleted) {  countingStarted = false;  countingCompleted = false;  }  // Update lastButtonState  lastButtonState = reading; } |

Table 2: Code for Python

|  |
| --- |
| #importing libraries  import pandas as pd  import numpy as np  import matplotlib.pyplot as plt  from scipy.signal import savgol\_filter  from scipy.optimize import curve\_fit  import math  import scipy.ndimage as ndimage  import serial  from openpyxl import Workbook  # Initialize serial port  ser = serial.Serial('COM4', 9600) # Example: COM4 is the port, 9600 is the baud rate  num\_samples = 16  time\_master = []  aceel\_x\_master = []  aceel\_y\_master = []  data\_final = []  for \_ in range(num\_samples):  data = ser.readline().decode('latin-1').strip().split() # Assuming data is space-separated  if len(data) >= 3: # Assuming you have at least three values in data  time\_master.append(float(data[0]))  aceel\_x\_master.append(float(data[1]))  aceel\_y\_master.append(float(data[2]))  data\_final.append(data)  print(data)  # Close serial port when done  ser.close()  wb = Workbook()  ws = wb.active  # Writing header row  header = ["V", "Irr", "I"]  ws.append(header)  # Writing data rows  for row in data\_final:  ws.append(row)  wb.save("data.xlsx")  # Read Resistance from Excel file  R = pd.read\_excel("R.xlsx")  R = R["R"][:]  #reading the data file  data = pd.read\_excel("data.xlsx")  # Extract resistance and voltage data  V = data["V"][:]  Irr\_R\_arr = data["Irr"][:]  I\_sensor=data["I"][:]  # Calculate current  I = V / R  #calculating irradiaance  Irr\_R=np.mean(Irr\_R\_arr)  Irr\_R=(5/1024)\*Irr\_R  Irr\_R=(460\*Irr\_R)/(5-Irr\_R)  c=3.3337\*10\*\*3  m=-0.4791  Irr=(Irr\_R/c)\*\*(1/m)  print("irradiance = ",Irr);  #plot the experimental I\_V  #plt.figure(1)  #plt.plot(V,I)  #smoothing data  #window\_size =2 # Adjust for desired smoothing level  #I\_smooth = savgol\_filter(I.values, window\_size, 1)  #gaussian smoothing  sigma = .7  I\_smooth =ndimage.gaussian\_filter(I, sigma)  Iph=np.max(I\_smooth)  #plotting the experimental and smoothened data  #plt.scatter(V,I)  #plt.plot(V,I\_smooth)  I\_exp=np.insert(I\_smooth,0,Iph)  V\_exp=np.insert(V,0,0)  y=Iph-I\_exp  x=V\_exp  #plot data after reversing  #plt.figure(2)  #plt.plot(x,y)  #finding the tolerance for fitting the curve  tol=8e-3  prev\_err=1e20;  prev\_tol=1e20;  for i in range(1,20):  for j in range(9,0,-1):  tol=j\*1\*10\*\*(-i)  y\_log=np.log((y+tol))  A=np.ones(len(x))  A=A.reshape(-1,1)  x\_reshaped=x.reshape(-1,1)  A=np.hstack((A,x\_reshaped))  An=np.dot(A.T,A)  Bn=np.dot(A.T,y\_log)  xn = np.linalg.lstsq(An, Bn, rcond=None)[0]      exp\_term1=np.exp(xn[0])  exp\_term2=np.exp(xn[1]\*x)  y\_fitpoints=exp\_term1\*exp\_term2  err\_arr=np.abs(y\_fitpoints-y)  err=np.sum(err\_arr)  if (err>prev\_err):  break  prev\_tol=tol  prev\_err=err  if (err>prev\_err):  break  #fitting the data by matrix  tol=prev\_tol;  #tol=.00001  y\_log=np.log(y+tol)  A=np.ones(len(x))  A=A.reshape(-1,1)  x\_reshaped=x.reshape(-1,1)  A=np.hstack((A,x\_reshaped))  An=np.dot(A.T,A)  Bn=np.dot(A.T,y\_log)  xn = np.linalg.lstsq(An, Bn, rcond=None)[0]  xx=np.linspace(0,20,1000)  exp\_term1=np.exp(xn[0])  exp\_term2=np.exp(xn[1]\*xx)  y1=exp\_term1\*exp\_term2  #plotting the fitted data I\_V  #plt.plot(xx,y1)  fig, ax1 = plt.subplots()  ax1.plot(xx, y1, color='b', label='current')  ax1.set\_xlabel('Voltage(V)')  ax1.set\_ylabel('current(A)')  ax1.tick\_params(axis='y', labelcolor='b')  ax1.set\_axis\_on()  ax1.set\_xlim(xmin=0)  ax1.set\_ylim(ymin=0)  ax1.grid(True,which='both')  ax1.set\_title("I\_V curve")  ax1.scatter(x,y)  # Add a legend  #ax1.legend(loc='upper left')  plt.show()  #reversing the data  I\_process\_ini=-y1+Iph  I\_processed=I\_process\_ini[I\_process\_ini>=0]  V\_processed=xx[I\_process\_ini>=0]  P=I\_processed\*V\_processed  P\_max=np.max(P)  idx=np.argmax(P)  V\_max=V\_processed[idx]  I\_max=I\_processed[idx]  fig, ax1 = plt.subplots()  ax1.plot(V\_processed, I\_processed, color='b', label='current')  ax1.set\_xlabel('Volatage(V)')  ax1.set\_ylabel('current(A)', color='b')  ax1.tick\_params(axis='y', labelcolor='b')  ax2 = ax1.twinx()  ax2.plot(V\_processed, P, color='r', label='power',linestyle='dashed')  ax2.set\_ylabel('power(watt)', color='r')  ax2.tick\_params(axis='y', labelcolor='r')  # Add a legend  ax1.legend(loc='upper left')  ax2.legend(loc='upper right')  ax1.grid(True,which='both')  ax1.scatter(V, I,color='b' )  plt.title("solar panel characteristics")  plt.show()  #finding the series and parallel resistance  Isc=Iph  Voc=np.max(V\_processed)  FF=P\_max/(Voc\*Isc)  h=V\_processed[1]-V\_processed[0]  conductance=(I\_processed[1]-I\_processed[0])/h  resistance=1/conductance  Rp=-resistance  #finding the efficiency  Area=0.31\*0.34  P\_in=Area\*Irr  efficiency=P\_max/P\_in  #printing the necessary data  print("\n \n \n")  print("efficiency, =",efficiency)  print("fill factor, =",FF)  print("open circuit voltage, Voc =",Voc)  print("short circuit current, Isc =",Voc)  print("maximum power, Pmax =",P\_max)  print("voltage at max power point, Vmax =",V\_max)  print("current at max power point, Imax =",I\_max)  print("parallel resistance, Rp =" ,Rp)  print("\n \n \n") |

# Design Analysis and Evaluation

## Novelty

## Use of decoder IC and calibration of intensity with LDR and developing the way for selecting operating range are new addition to our project.

## Design Considerations (PO(c))

## We have worked in around room temperature. Since, we calibrated during early February and demonstrated the project in early March, there was some difference towards the angle of sunlight. There was also slightly difference in air mass.

### Design of Experiment

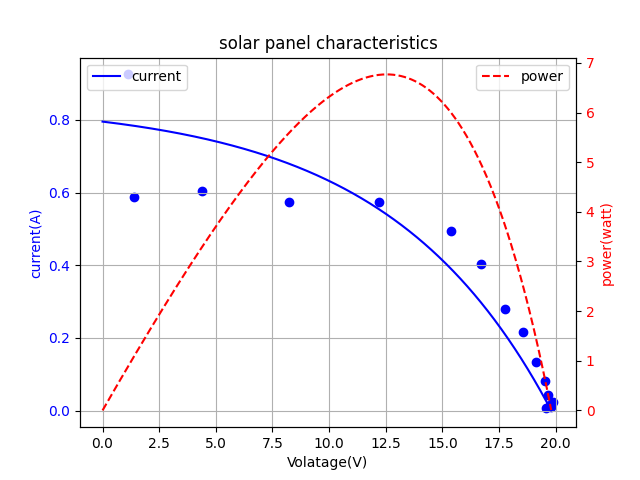
Take voltage data using Arduino->Press switch-> Wait for 16 data-> send to python-> Display graph and required data

### Data Collection

Table 3: Points of wrong operation

|  |  |  |
| --- | --- | --- |
| Voc | Maximum acceptable value of Isc | Minimum acceptable value of Isc |
| 0.5V | 0.12 | 1.20E-04 |
| 5 | 1.74 | 1.20E-03 |
| 9 | 3.75 | 2.80E-03 |
| 16 | 6.97 | 4.60E-03 |
| 25 | 13.6 | 1.00E-02 |

### Results and Analysis



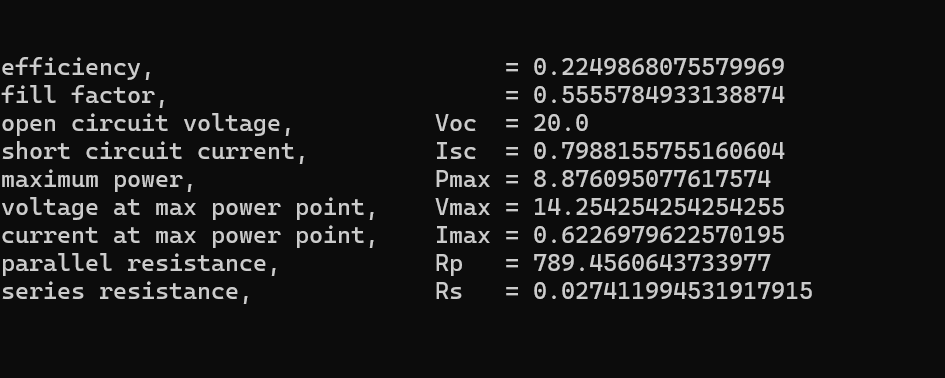
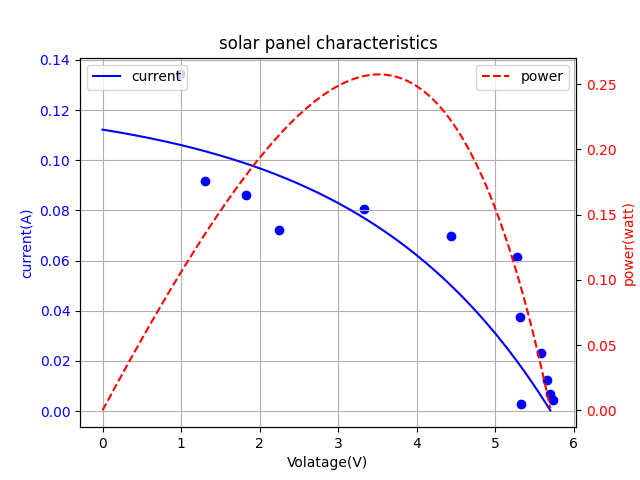


Figure: Output for large solar panel



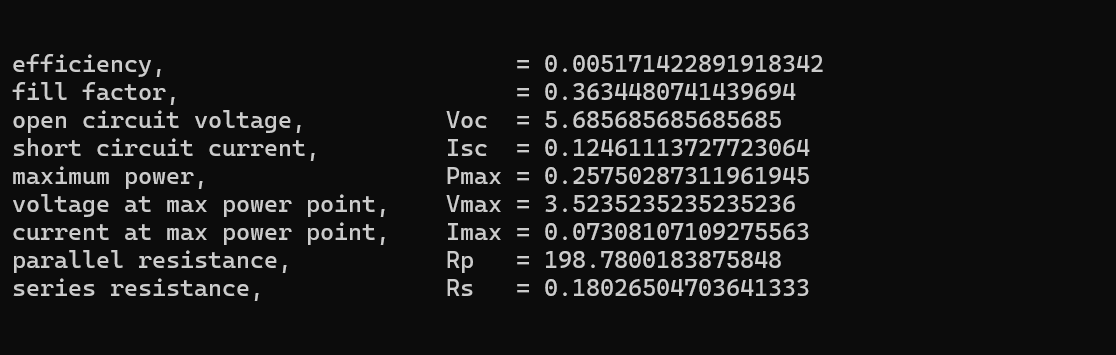
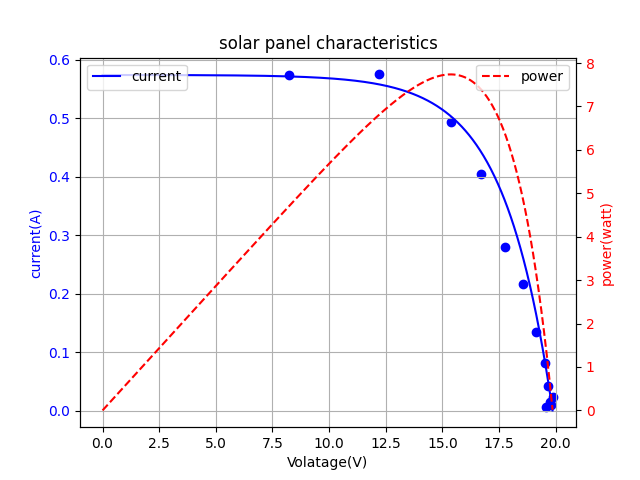


Figure: Output for mini solar panel



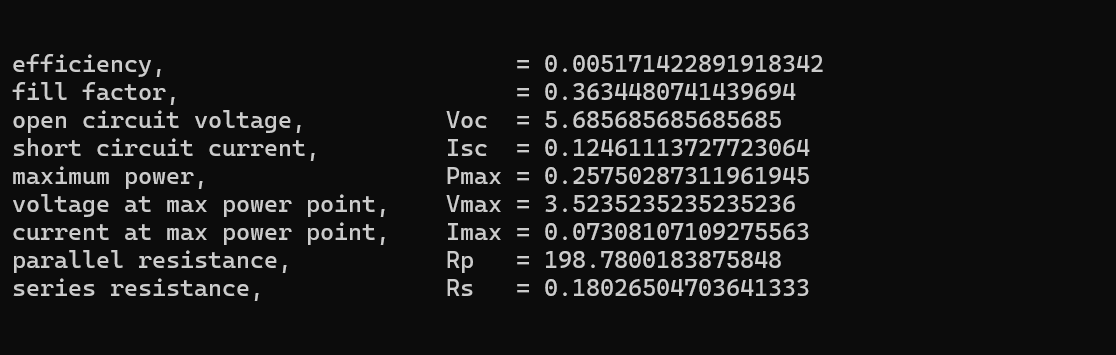


Figure: Post processing data for mini solar panel

### Interpretation and Conclusions on Data

To understand whether, our decoder IC work perfectly, we have added LED light

Table 4: Interpretation of decoder code

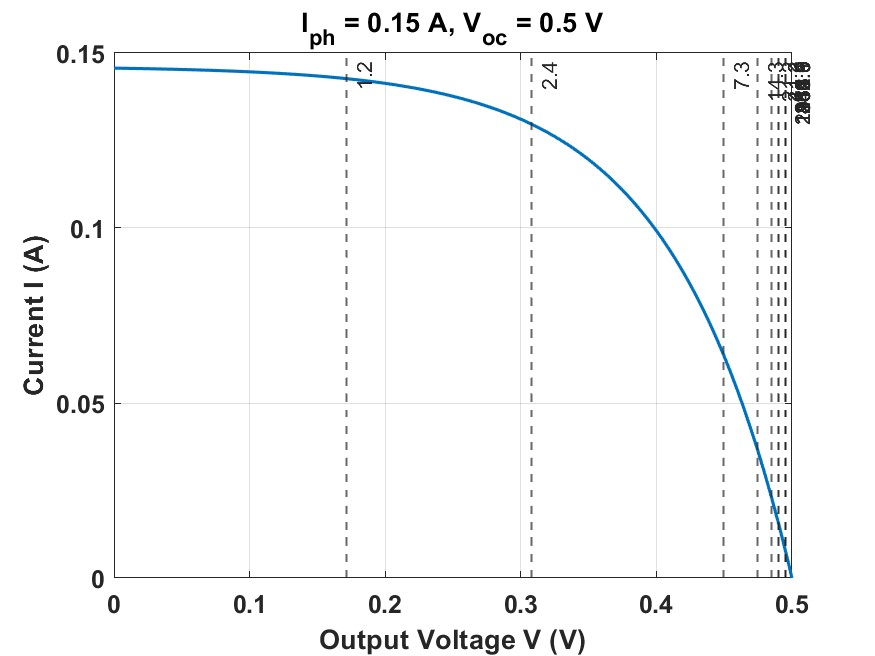
|  |  |
| --- | --- |
| Code | Active resistance (Ω) |
| 0000 | 2964.9 |
| 0001 | 1981.9 |
| 0010 | 1312.9 |
| 0011 | 841.9 |
| 0100 | 460.9 |
| 0101 | 239.9 |
| 0110 | 142.1 |
| 0111 | 85.6 |
| 1000 | 63.5 |
| 1001 | 41.3 |
| 1010 | 31.2 |
| 1011 | 21.2 |
| 1100 | 14.3 |
| 1101 | 7.3 |
| 1110 | 2.4 |
| 1111 | 1.2 |

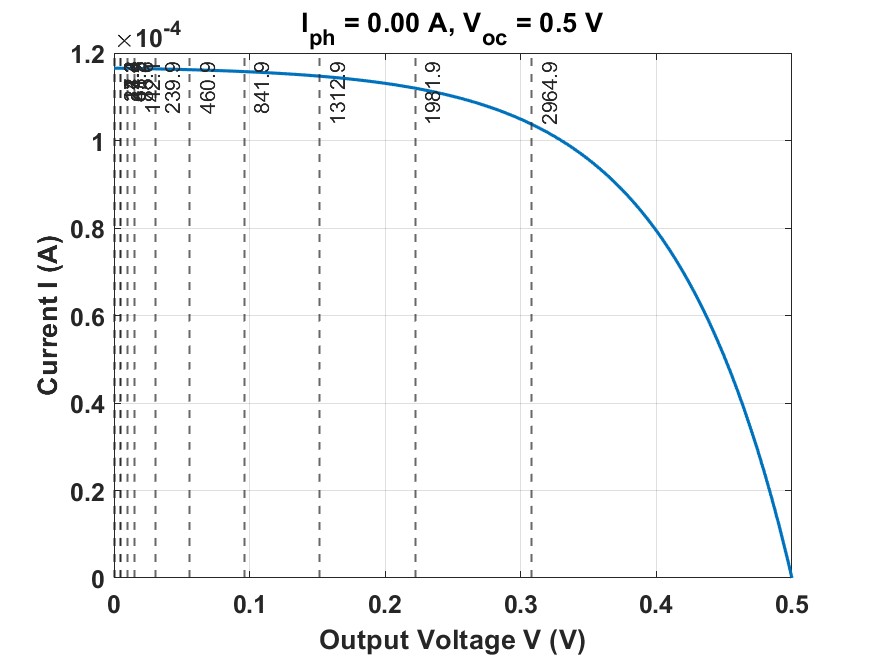
## Limitations of Tools

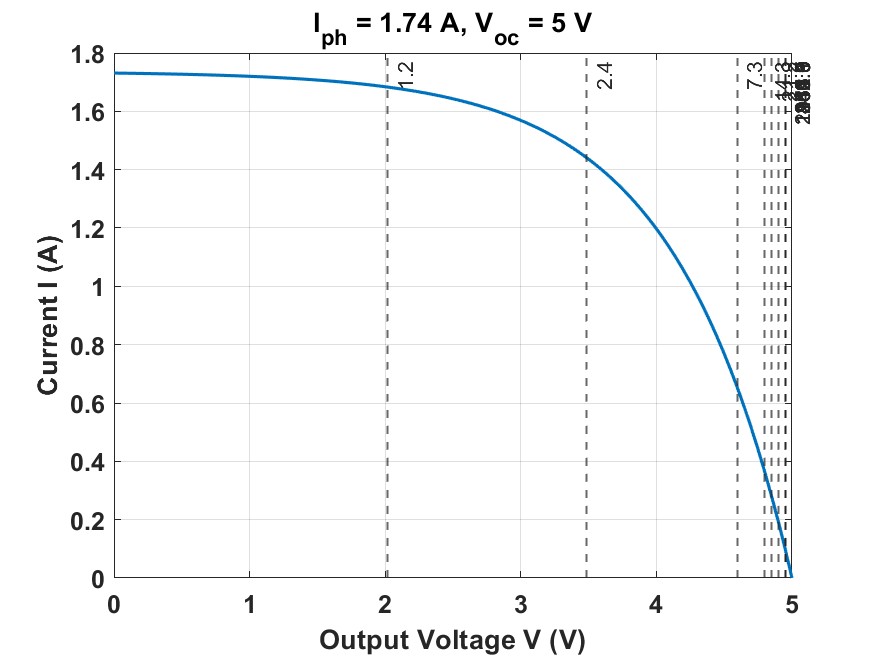
Sometimes performance of LDR is changed with time and temperature. Hence, exact irradiance cannot be measured sometimes. Moreover, with our setup we cannot measure I-V characteristics for all types of solar cell. With our used resistance, we have tune the value of Io and n to check the perfect region of operation.

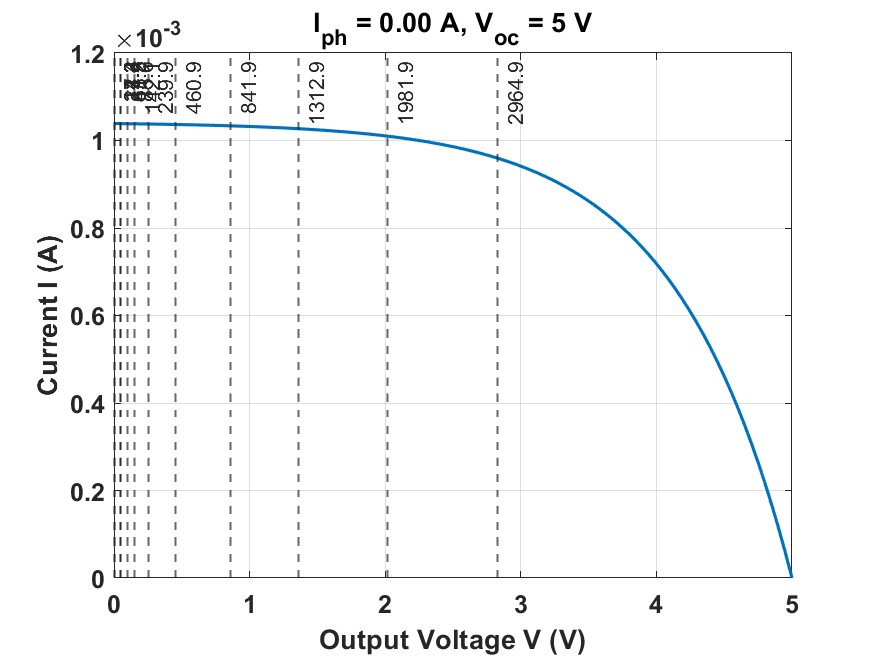
Table 5: Matlab code for determining effective range

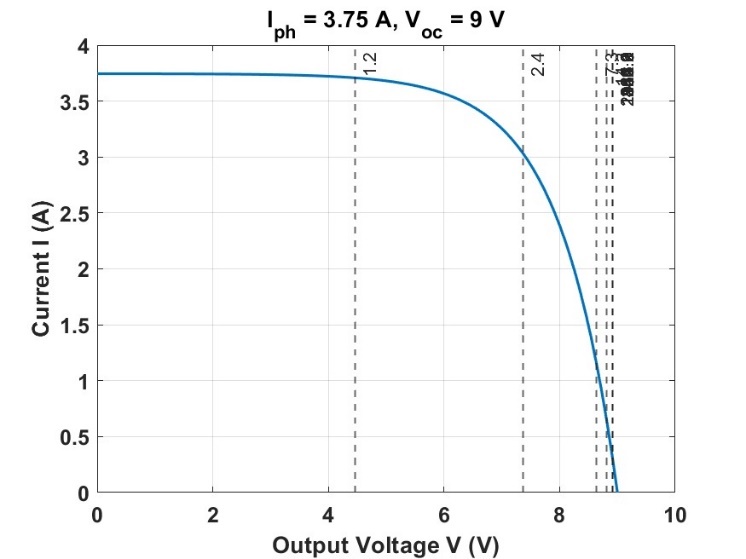
|  |
| --- |
| clc  clear all  close all  %%  load("R.mat")  V = linspace(0,25,100); % for different Voc, we need to select different value  Vt = 0.0259;  n =75; % FOR Voc = 5 V, n=3.4 for 5V n=33 for 5, 38 for 9 n=50 for 16 n=75 for 25V  % n = 80 ; % FOR Voc = 25 V  Io = 1.7e-8;  I\_ph = 0.8;  I = Io\*exp(V/(n\*Vt));  I\_ph = max(I)  I = -I + I\_ph;  figure(1)  plot(V,I,'LineWidth',1.5)  xlabel('Output Voltage V (V)')  ylabel('Current I (A)')  grid on  R\_eqn = V./I;  R\_near = [];  indx = [];  for i = 1:16  [temp,idx] = min(abs(R\_eqn-R(i)));  R\_near = [R\_near,temp];  indx = [indx idx];  end  % V(indx);  R\_cell = num2cell(R);  figure(1)  xline([V(indx)],'--',R\_cell,'linewidth',1)  str = ['I\_{ph} = ',num2str(I\_ph,'%0.2f'),' A, V\_{oc} = ',num2str(V(end)),' V']  title(str)  set(gca,'FontSize',12,'fontweight','b')  saveas(gcf,'f14.jpg','jpeg'); |
| **Code for determining final operating range** |
| clc;  clear;  close all;  Voc = [0.5, 5, 9, 16, 25];  Imax = [0.12, 1.74, 3.75, 6.97, 13.6];  Imin = [1.2e-4, 1.2e-3, 2.8e-3, 4.6e-3, 1e-2];  figure;  semilogy(Voc, Imax, '--o', 'LineWidth', 2); % Plotting maximum values  hold on;  semilogy(Voc, Imin, '--o', 'LineWidth', 2); % Plotting minimum values  % Fill the space between the two curves with gray color  patch([Voc fliplr(Voc)], [Imax fliplr(Imin)], 'k', 'FaceColor', [0.7, 0.7, 0.7], 'EdgeColor', 'none');  hold off;  xlabel('Open circuit Voltage (V)');  ylabel('Short Circuit Current (A)');  title('Range of Voc and Isc');  legend('Maximum Isc', 'Minimum Isc', 'Operating Region', 'Location', 'Best');  set(gca,'FontSize',12,'fontweight','b')  grid on; |











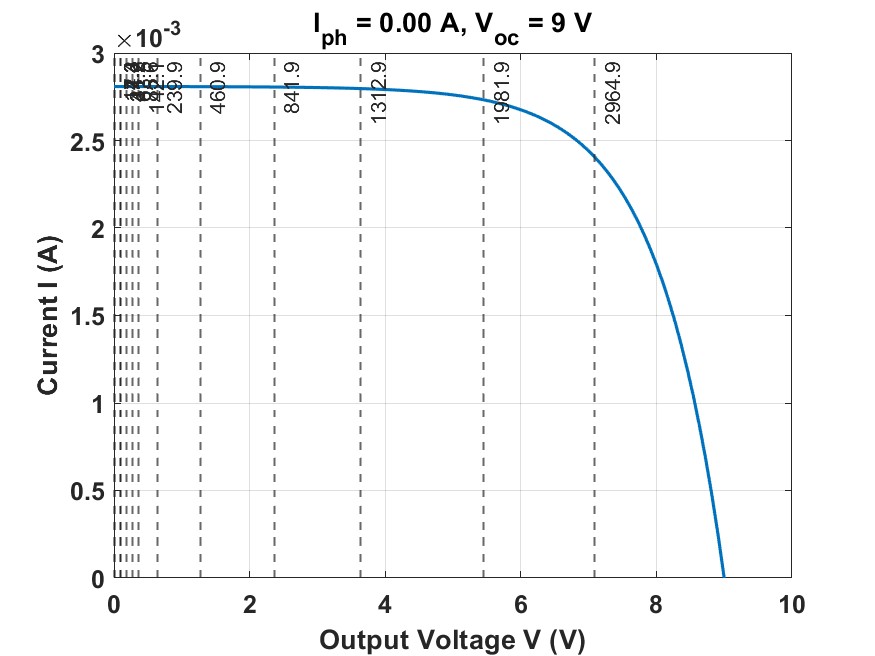


Figure: Determination of operating range by observing the position of our used resistance

## Impact Assessment

As a project prototype this device may not have huge impact. However, if this is implemented for mass production and better calibration, this type of device can be commercialized.

## Individual Contribution of Each Member

## ID:1806116 contributed to experimental setup

## ID:1806151 contributed to coding part

## ID: 1806159 contributed to analyse experimental result

## ID: 1806182 contributed to final circuit design

## Mode of Team Work

## Hardware related part has been done in offline. Discussion related to theoretical equation and algorithm for coding has sometimes occurred in online.

## Diversity Statement of Team

## In our team, both attached and residents students involved.

## Log Book of Project Implementation

|  |  |  |
| --- | --- | --- |
| **Date** | **Milestone achieved** | **Member Involved** |
| 16 January,2024 | Buying required components | Tasfiq |
| 17 January, 2024 | Approach for utilizing drain to source resistance | Meherab |
| 23 January,2024 | Calibrate solar intensity with 4 LDR and irradiance meter | Samiul, Tasfiq, Meherab |
| 5 February,2024 | Mapped resistance of LDR and light intensity in MATLAB | Siyam |
| 16 February,2024 | Measured drain to source resistance for different voltage bias | Meherab |
| 22 February,2024 | Use of digital potentiometer for automatically varying resistance | Tasfiq, Samiul |
| 25 February,2024 | Take the approach of switching MOSFET for varying resistance | Meherab |
| 27 February,2024 | Use of active low decoder for efficient switching | Siyam |
| 28 February,2024 | Use of active high decoder for switching | Meherab, Samiul |
| 29 February,2024 | First I-V characteristics plotting in MATLAB | Tasfiq, Meherab |
| 1 March,2024 | Converting MATTLAB code into Python code | Samiul |
| 2 March,2024 | Writing Report and Presentation | Tasfiq, Siyam |

# Communication to External Stakeholders

## Executive Summary

## Operation of this measurement is safe within about 25V. The circuit has been tested in temperature. Solar spectrum used for testing is around AM 1.5.

## User Manual

## 

## Figure: Final datasheet for user manual

To operate this device, the user needs to look the back side of the solar panel and read the value of open circuit voltage and short circuit current. If the reading data fall within the gray region, he an use our circuit to measure I-V characteristics of the solar cell. If there is no data in nameplate, and he measures it randomly, he can also measure the characteristic of unknown solar panel but if the actual rating of the solar cell is out of the above range , he cannot observe saturation and retardation in a single plot.

The first task is to connect the positive and negative terminals of the solar panel with the indicated place of the breadboard. The user needs to place the LDR near the solar panel. The top surface of LDR should be towards the direction of sunlight. Arduino is being previously coded. It requires to be connected with 5V power supply. The Arduino will read 16 data. When 16 data are read, a signal can be sent to Python or MATLAB if any of this software is installed in the system. Data will be processed and I-V,P-V characteristics, maximum power, efficiency, fill factor, series and shunt resistance.

## YouTube Link

## [(5) EEE 460 Solar Cell I-V characteristics - YouTube](https://www.youtube.com/playlist?list=PLVnVuTWKXyoEInrBFqizvrz9iPAPFkQ5I)

# Project Management and Cost Analysis

## Bill of Materials

|  |  |  |  |
| --- | --- | --- | --- |
| Component | Per unit cost (BDT) | Number of components | Total |
| Arduino Uno | 600 | 1 | 600 |
| IC 74238 decoder | 35 | 2 | 70 |
| IC 7404 | 35 | 1 | 35 |
| IRFZ44 MOSFET | 35 | 16 | 560 |
| 5W power resistor | 10 | 16 | 160 |
| LDR | 45 | 1 | 45 |
| Push Button | 20 | 1 | 20 |
| Total cost | | | 1490 BDT |

## Calculation of Per Unit Cost of Prototype

## Each unit of the circuit costs 1490 BDT.

## Calculation of Per Unit Cost of Mass-Produced Unit

## In case of mass production, PCB design would be best option. Besides designing IC with three decoder and 16 MOSFETs with optimum heat sink will significantly reduce the size and cost of circuit. Overall cost per unit will be less than 1000 BDT.

## Timeline of Project Implementation

## 

## Figure : First Trial with MOSFET

## 

## Figure: First Approach to measure I-V characteristics

## 

## Figure: Calibration using LDR

## 

## Figure: Choosing Resistance for solar panel

# Future Work

# Designing a single IC with proper heat sink to reduce the size of the device.

# Increase the tolerance limit of voltage sensor can increase the range of operation.

# Taking data for different air mass unit will make the device more realiable.

# References

# [1] Kunze, Philipp, et al. "Contactless Inline IV Measurement of Solar Cells Using an Empirical Model." *Solar RRL* 7.8 (2023): 2200599.

# [2] Amiry, H., et al. "Design and implementation of a photovoltaic IV curve tracer: Solar modules characterization under real operating conditions." *Energy Conversion and Management* 169 (2018): 206-216.

# [3] Papageorgas, P., et al. "A low-cost and fast PV IV curve tracer based on an open source platform with M2M communication capabilities for preventive monitoring." *Energy Procedia* 74 (2015): 423-438.