Smart Solar Panel

Ramesh Kamath
Manav Jaiswal

A survey report

Contents

Table of Figures

1. Introduction

This project demonstrates a novel and inexpensive method of monitoring the essential values of a solar panel, to be precise estimated values of Surface temperature of solar panel, irradiance falling on the panel, Power generated by the panel and efficiency of the panel. Monitoring the light irradiance requires an equipment/sensor known as pyranometer which is very expensive and using it in academic/experimental use is not feasible. We found a hack in the Light sensor primarily used to give LUX readings, and found a way to calculate the irradiance falling on it. Use of two sensors give us two direct values of Temperature (from temperature sensor) and Irradiance (light sensor) as well as giving us two more indirect readings of Power generated (Related to the irradiance reading) and efficiency (Related to the temperature reading). The ESP8266 module is used as a microcontroller to control the sensors and since it has a built in WiFi chip, we connect it to a Web server to store, visualize and process the data from the sensors.

The idea of this project is to monitor the surface temperature and irradiance falling on the solar panel and depending on this we calculate the power and efficiency. If the irradiance of light falling on the panel is high, we get a good amount of power from the panel and if the temperature exceeds a certain constant value which can be found in the solar panels manufacturing sheet, the efficiency of the solar panel decreases. We use these 4 data and send it to a web server which is hosted on an amazon AWS EC2 instance with its backend a PHP mySQL database server. The ESP8266 Module communicates using HTTP protocol with the web server and has the capability of receiving messages from the web server. Initially a Threshold temperature value will be set according to the solar panel used from the web server into the ESP8266 Module. If the temperature exceeds the threshold provided, the Huzzah ESP8266 module must send a signal to a relay which is connected to a solenoid valve to switch it ON. The solenoid valve controls water flow from a mobile water source and pours it down the panel if switched ON to decrease the panel's temperature so that the efficiency does not degrade.

An Interactive Website is hosted on an amazon AWS EC2 instance which provides real time monitoring of the data using graphs and all the above said functionalities.

2. Literature Survey

Every moment the sun radiated a lot of energy. It radiates out more energy in one day than the energy used by the world in one year. About two out of a billion part of the radiated energy reaches the Earth. Out of this energy, thirty percent of the energy is reflected by the atmosphere and seventy percent is absorbed by the green-house gasses. This energy which reaches the ground in one day on Earth is enough to power United States for a year. The Sun is regarded as a renewable source of energy which radiates due to continuous nuclear fusion reaction happening at its core as well as on its surface. Global warming is not a myth anymore. We are observing the effects of pollution in many different ways as by increase in temperature of Earth, polluted river and sea water with harmful reactants and oil spills, extinction of many rare species of plants, birds, animals and fishes. The major contribution to pollution happens due to factories and industries and by power generation plants which uses non-renewable resources like fossil fuels. It is a high time that we start sufficing our power needs through renewable ways and make sure that we do not harm the nature any more.

Energy supply is a very significant factor for a country's growth. Countries having higher energy consumption tend to have higher HDI (Human Development Index) [3]. This fact doesn't mean that fossil fuels should be more exploited to derive more energy. The amount of fossil fuel that Earth have is limited. Also, the electricity generation through fossil fuels causes air and water pollution at large scales, directly affecting health of living beings. The air pollution is giving rise to global warming which is resulting in rise in Earth's temperature, extinction of many plant and animals species, melting of glaciers and rise in sea levels. This brings us to the solution, that is, to switch to renewable energy sources for a safe and clean energy. There are many solutions about how to satisfy our energy needs by renewable needs but Solar appliances are most popular than other types of renewable energy based products due presence of solar energy on any point of planet and reception of more than one type of energy like heat and light. The two main necessities for accessing solar energy are installation space and installation costs. IoT is an evolving field where the applications of this technologies are yet to be scaled. There are many multi-domain problem solving capability in this upcoming field.

Thus, IoT can be used efficiently in domain of power monitoring and simulation and then displaying the values on a web-based GUI for monitoring. By displaying real-time irradiance and efficiency, these device can used as a solar panel simulator where it can estimate the power that a panel with particular efficiency will generate and the data from this device can be visualized on the web. Based on the literature mentioned in this section, this product idea would be in great demand if in the market.

The following figure shows current vs. voltage at different temperature and a similar figure but in terms of irradiance. It shows the phenomena of change of efficiency of PV cell with change in temperature.

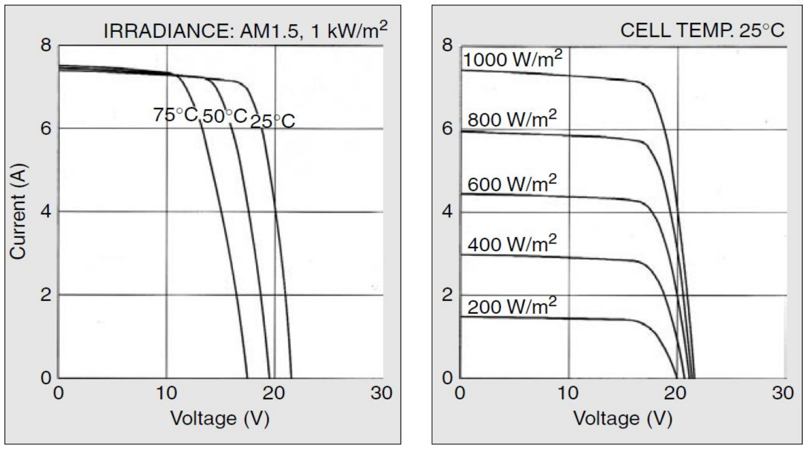


Figure 1: Variation of efficiency w.r.t. temperature

The value of total energy generated from a solar panel can be given by this equation:

$$E = e_m e_p AG\,(1 + (T_c - T_0)t_{cp})$$

Where,

e_m → intrinsic cell efficiency

e_p → plant efficiency

A → area of the panel
t_{cp} → temperature coefficient of the PV material
T_c → Current Temperature
T_0 → Ambient Temperature (25°C)

The factor of $\mathbf{e_m e_p AG}$ represents power generated given that the temperature always remain 25° C at all times. The second factor $\mathbf{e_m e_p AG(T_c - T_0)t_{cp}}$ represents the change in the power generated at 25° C with change in temperature. Similarly, the formula for overall efficiency is given by removing $\mathbf{e_p AG}$ from above equation. Thus, the resulting equation will give overall current efficiency of the PV panel as shown in below equation.

$$\mathbf{e = e_m + e_m(T_c - T_0)t_{cp}}$$

The ESP8266 lwIP or light weight Internet Protocol stack. The below figure represents the same. Apart from less complexity, the ESP8266 also features hard sleep and soft sleep modes (called differently in different references) which help in power efficient computing.

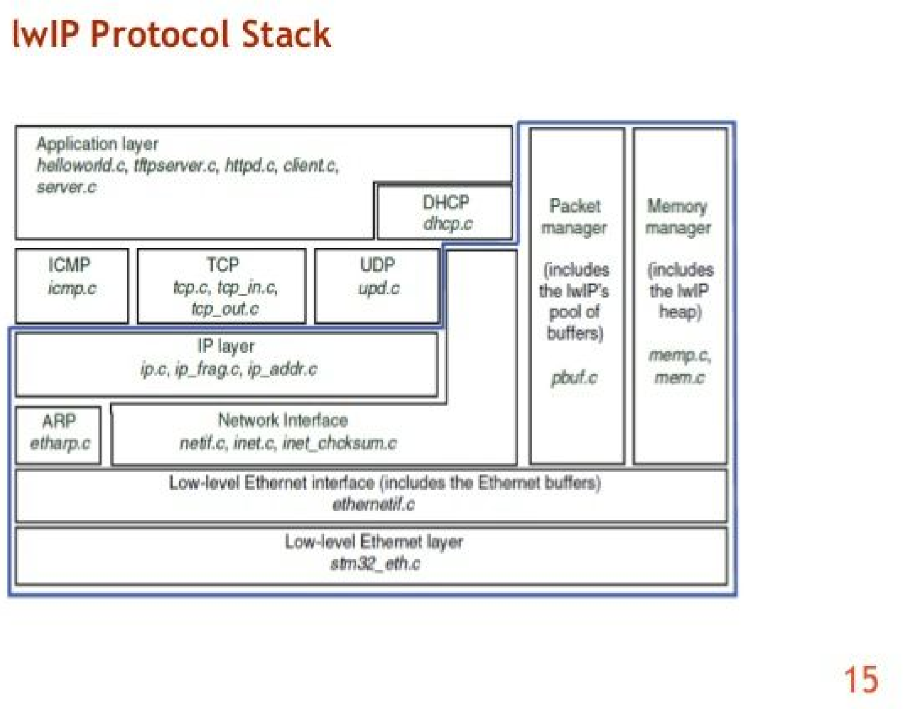
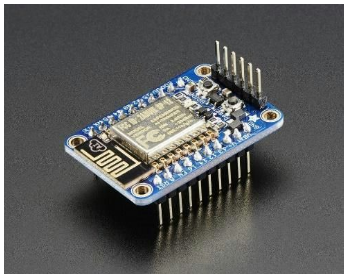

lwIP Protocol Stack
Application layer
helloworld.c, tftpserver.c, httpd.c, client.c, server.c
DHCP
dhcp.c
Packet manager
(includes the lwIP's pool of buffers)
pbuf.c
Memory manager
(includes the lwIP heap)
memp.c, mem.c
ICMP
icmp.c
TCP
tcp.c, tcp_in.c, tcp_out.c
UDP
upd.c
IP layer
ip.c, ip_frag.c, ip_addr.c
ARP
etharp.c
Network Interface
netif.c, inet.c, inet_chcksum.c
Low-level Ethernet interface (includes the Ethernet buffers)
ethernetif.c
Low-level Ethernet layer
stm32_eth.c
15

Figure 2: ESP8266 lwIP Protocol Stack

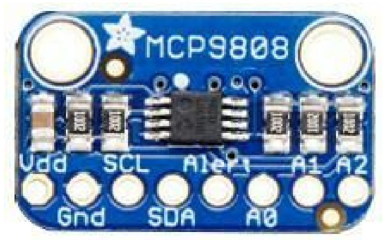
3. Hardware

COMPONENTS USED:

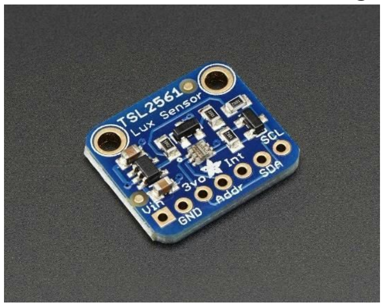
1 X Adafruit Huzzah 8266



1 X Adafruit MCP9808 I^2C Temperature Sensor



1 X Adafruit TSL2561 I^2C Light Sensor



3.1. Connection

The above figure gives us a clear idea on the pin diagram and connection between the components used in this project. Let us now talk about the different components used and why it was chosen over other options.

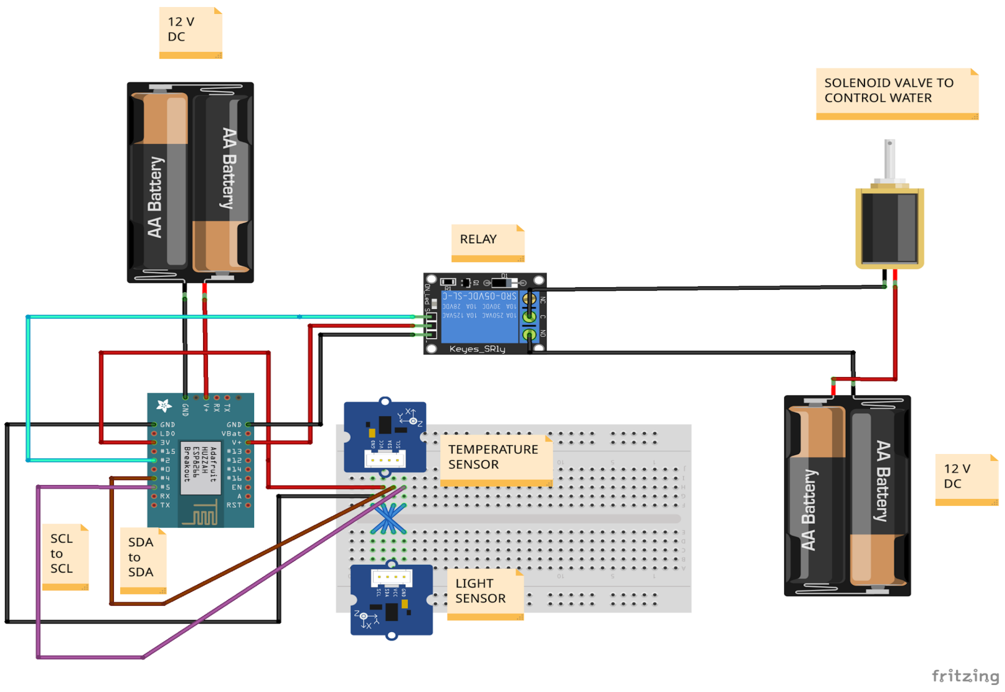


Figure 3: Connection Diagram

3.2. Adafruit Huzzah ESP8266 Module

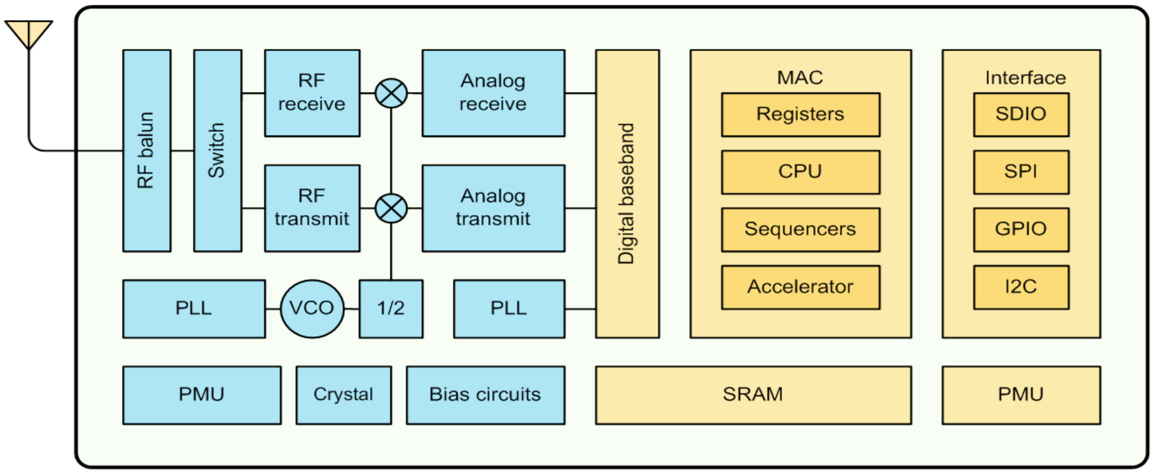


Figure 4: ESP8266 Block Diagram

As you can see from the internal block diagram of the ESP8266 module, A chip designed by Espressif Semiconductors, we can observe that it has a microcontroller. ESP8266EX offers a complete and self-contained WiFi networking solution; it can be used to host the application or to offload WiFi networking functions from another application processor. ESP8266EX also integrates an enhanced version of Tensilica's L106 Diamond series 32-bit processor, with on-chip SRAM, besides the WiFi functionalities. ESP8266EX is often integrated with external sensors and other application specific devices through its GPIOs and other bus interfaces such as I^{2}C, SPI, UART and SDIO.

This chip is developed into a module board by Adafruit for easy access of the pins and made the board Arduino compatible for ease in coding and flashing the code in the memory of microcontroller. The best part of this module is that it costs $9 which is very cheap if we want to use sensor nodes with processing capabilities in huge numbers.

3.3. Adafruit I^{2}C sensor modules

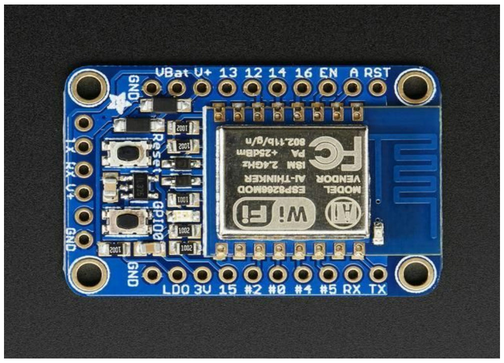


Figure 5: ESP8266 module

As seen in the pinout of the ESP8266 module, we have limited GPIO and interfaces, pin 4 and pin 5 from the I^{2}C bus of the module, and I^{2}C protocol works on master slave communication, and one bus can have maximum upto 8 slaves, Thus we decided to use the I^{2}C interface to interface the sensors. (Inter-Integrated Circuit) is a two-wire serial bus typically used inside computers for low-level communication between components, but it's also seen in robotics and hobbyist electronics for interfacing all manner of sensors, displays and actuators. I^{2}C connections are often readily available on microcontrollers and esoteric embedded systems

Both the modules have the 4 pins used in I^{2}C communication, SDA (Data pin), SCL (clock pin), VCC (+5V) and GND (ground). We first connect both the sensors in parallel i.e, SDA of temperature sensor connected to SDA of light sensor. SCL of temperature sensor connected to SCL of light sensor. Both the sensor's VCC and GND are connected in common. As shown in the diagram above, I^{2}C master differentiates from both the sensors by observing their I^{2}C address which will be different for different type of sensors.

3.4. CONNECTIONS from SENSOR INTERFACE to HUZZAH ESP8266

As we have done earlier, the SDA pins and SCL pins of both the sensors are connected together. We see what to do next with the connections from this interface to the ESP8266 module.

SDA → PIN 4 (ESP8266)
SCL → PIN 5 (ESP8266)
Vin → 3V (ESP8266)
GND → GND (ESP8266)

A relay is given signal from pin 2 of the ESP8266, This pin will switch the relay ON or OFF depending on the threshold value of temperature, even there is a button on the web site which can be used to toggle the pin 2 HIGH and LOW. In turn, the relay is connected to a water valve solenoid which is used to control the flow of the water falling on the solar panel.

4. Software

We use the arduino IDE for programming the adafruit huzzah ESP8266. Before starting the actual programming we need to download a few libraries for the support.

1. SparkfunTSL2561 (we are using an adafruit sensor still we use a Sparkfun library due to ease of functions)
2. ESP8266Wifi library (Our module is ESP8266)
3. Arduinojson (to parse the data sent from the web server)

We program the temperature sensor without the library using register level programming.

4.1. Main Code

```
#include <ArduinoJson.h>            // Library for parsing json object
#include <SparkFunTSL2561.h>    // Library for Light sensor
#include <Wire.h>                        // Library for arduino interfaces such as I2C and SPI.
#include<ESP8266WiFi.h>           // Library to include the ESP8266 module
#include <string.h>
#define TEMP_REG 0x05              // This is the register address from where we get the
                                              //ambient temperature
#define I2C_ADDR 0x18               // This is the address of the device temperature sensor

SFE_TSL2561 light;                    // We create an object of the light sensor



const char* ssid     = "**********";     // Used to configure the ssid of ESP8266 to your network
const char* password = "*********";   // Password of your network
const char* host = "*******.com";     // This is the name of your web server used to communicate with
                                                  //ESP8266


//INITIAL DECLARATIONS


int area = 4;                              //Area of your solar panel square meters
String Sensors;                       // String format for sending sensor Data using HTTP
unsigned int data0, data1;          // variables which will store the raw data coming from the Light
sensor
float effp = 0.12;                     // panel efficiency constant
float tcp = -0.0045;                   // Temperature co-efficient of solar panel
boolean gain;                              // Gain setting, 0 = X1, 1 = X16;
unsigned int ms;                         // Integration ("shutter") time in milliseconds
float temp, irr;                         // variable to store temperature and irradiance
float power, eff;                       // variables to store power and efficiency
String temp1,irr1, power1, eff1; // variables to convert the float variables to string
int flag = 0;
int valve = 0;                          // status of the solenoid valve valve = 0 means OFF state
int sys = 0;                            // status of the entire system, sys = 0 means OFF state

StaticJsonBuffer<200> jsonBuffer;   //Json Buffer to store the json object
String url;                                  // url to make the web address of the web server and add the data


//SETUP PROGRAM

void setup(void) {
 Serial.begin(9600);                         //9600 serial baud rate
 delay(1000);
 pinMode(BUILTIN_LED, OUTPUT);    // Make the built in LED of ESP8266 module as output
 digitalWrite(BUILTIN_LED,HIGH);
 pinMode(2,OUTPUT);                      // Make the pin 2 of ESP8266 module as output
 digitalWrite(2,LOW);
 Serial.println();
 Serial.print("Connecting to ");
 Serial.println(ssid);
 WiFi.begin(ssid, password);              // Function to connect to the SSID provided by us

// Loop till ESP8266 gets connected to the Network provided by us in SSID and password field

 while (WiFi.status() != WL_CONNECTED) {
  delay(500);
  Serial.print(".");
 }

 Serial.println("");
 Serial.println("WiFi connected");
 Serial.println("IP address: ");
 Serial.println(WiFi.localIP());
 Wire.begin();                     // Begin the Wire function for interfacing the temperature sensor

 light.begin();                          //begin function of the light sensor

 gain = 0;                               // If time = 0, integration will be 13.7ms
                                             // If time = 1, integration will be 101ms
                                             // If time = 2, integration will be 402ms


 unsigned char time = 1;             // Set time as 101ms
 light.setTiming(gain,time,ms);     // set the timing and gain of the sensor
 light.setPowerUp();                  // power up the light sensor

}


//LOOP CODE

// Loop till the ESP8266 module becomes a wifi client
void loop(void) {
 WiFiClient client;
 if(sys == 0)                                // check is system flag is set to 0
 {
digitalWrite(BUILTIN_LED, HIGH);
digitalWrite(2, LOW);

const int httpPort = 80;                   // set the http port to Port no 80
```

```
// check if the ESP8266 client gets connected to the host provided by us

 if (!client.connect(host, httpPort)) {
   Serial.println("connection failed");
   return;
 }

// url to send to the host but sending all NULL values as the system flag is off

   url = "/home/send_data/0/0/0/0";
 Serial.print("Requesting URL: ");
 Serial.println(url);

  // This will send the request to the server
 client.print(String("GET ") + url + " HTTP/1.1\r\n" +
             "Host: " + host + "\r\n" +
             "Connection: close\r\n\r\n");
             String line;

 delay(500);


// Check if there is any message from the web server to the ESP8266 client

 while(client.available()){
 line = client.readStringUntil('\n');           // Save the received message in a String variable
 Serial.println(line);
   }

// Convert String to const char array required for json object parsing
 char buf[line.length()+1];
 line.toCharArray(buf,line.length()+1);
 Serial.println("BUFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF");
 Serial.println(buf);
 StaticJsonBuffer<200> jsonBuffer;
 JsonObject& root = jsonBuffer.parseObject(buf);//(buf);


 }

// This is the condition if system flag is ON ( sys = 1)

else if(sys = 1)                          // This will send the request to the server
{
  if (light.getData(data0,data1))
 {
   // getData() returned true, communication was successful
   irr = data0*0.046;                                        //Convert the raw data into irradiance
 }

 uint16_t t;                                                // initialize a 16 bit variable for temp data
 Wire.beginTransmission(I2C_ADDR);                      // begin I2c transmission
 Wire.write(TEMP_REG);                                   // Write to the temperature register
 Wire.endTransmission();                                 // end transmission

 Wire.requestFrom(I2C_ADDR, 2);                         // We get 2 8 bit data from the sensor
 t = Wire.read();                                           // read the first byte
 t<<=8;                                                    // left shift it by 8 bits
 t=t | Wire.read();                                      // OR it with the second byte to get the 16 bit value

 temp = t&0x0FFF;                                       // Check the first 12 bits of the temp register
 temp=temp/16;                                           // divide it by 256 according to the data sheet
 if(t&0x1000)                                           // if the MSB is 1, negative temperature
 temp = temp-256;

 eff = effp+(tcp*effp*(temp-25));                     // Efficiency formula
 power = (effp*irr*area)-(area*irr*(effp-eff));      // Power formula
 Serial.print("Temperature = ");
 Serial.println(temp);
 Serial.print("Irradiance = ");
 Serial.println(irr);
 Serial.print("Efficiency = ");
 Serial.println(eff*100);
 Serial.print("Power = ");
 Serial.println(power);
 Serial.print("connecting to ");
 Serial.println(host);
 eff1 = String(eff*100);                            //multiply by 100 for percentage and convert to string
 power1 = String(power);                            //convert power to string
 temp1 = String(temp);                              // convert temperature to string
 irr1 = String(irr);                                // convert irradiance value to string


// Do the same thing of defining the port and connecting to the host
 const int httpPort = 80;
 if (!client.connect(host, httpPort)) {
   Serial.println("connection failed");
   return;
 }

// Send the sensor data and power and efficiency calculations by URL

 Sensors = temp1+"/"+ irr1 +"/"+ eff1 +"/"+ power1;
```

```
 url = "/home/send_data/";
 url = url+Sensors;
 Serial.print("Requesting URL: ");
 Serial.println(url);

 String line;
  // This will send the request to the server
 client.print(String("GET ") + url + " HTTP/1.1\r\n" +
             "Host: " + host + "\r\n" +
             "Connection: close\r\n\r\n");
 delay(500);
  while(client.available()){
 line = client.readStringUntil('\n');
 Serial.println(line);
   }
 char buf[line.length()+1];
 line.toCharArray(buf,line.length()+1);
 Serial.println("BUFFFFFFFFFFFFFFFFFFFFFFFFFFFFF");
 Serial.println(buf);
 StaticJsonBuffer<200> jsonBuffer;
 JsonObject& root = jsonBuffer.parseObject(buf);//(buf);
// if () {
 //  Serial.println("parseObject() failed");
  // return;
 //}
sys = root["system"];
thresh = root["threshold"];
valve = root["valve"];

Serial.println("SYYYYYYYYYYYYYYYYYYyyyystem");
Serial.println(sys);
Serial.println("Threshhhhhhhhhhhhhhh");
Serial.println(thresh);
Serial.println("VAAAAAAAAAALVE");
Serial.println(valve);

// Check if The valve value is 1 or greater than equal to the threshold temperature
// If yes then switch on the built in LED and toggle to pin 2 to HIGH

 if((valve == 1)||(temp>=thresh))
  {
   digitalWrite(BUILTIN_LED, LOW);
   digitalWrite(2, HIGH);
  }

// Check is the valve value is zero or the temperature value is less than threshold
// if yes, LED is switched OFF and toggle the pin 2 as LOW
  else if((valve == 0)||(temp<thresh))
  {
    digitalWrite(BUILTIN_LED, HIGH);
   digitalWrite(2, LOW);
    }
}
 delay(4000);
}
```

4.2. Adafruit TSL2561 Light to Digital Converter

Some of the features of TSL2561 from the datasheet are:

- Approximates Human Eye Response
- Programmable Interrupt Function with User-Defined Upper and Lower Threshold Settings
- 16-Bit Digital Output with SMBus (TSL2560) at 100 kHz or I2C (TSL2561) Fast-Mode at 400 kHz
- Programmable Analog Gain and Integration Time Supporting 1,000,000-to-1 Dynamic Range
- Automatically Rejects 50/60-Hz Lighting Ripple
- Low Active Power (0.75 mW Typical) with Power Down Mode
- RoHS Compliant

TSL2561 is an ambient light to digital converter with two 16 bit ADCs (Analog to Digital Converter) which converts the output of two photodiodes to digital values. One of the two photodiode sense full spectrum light while other responds only to IR content of light as input. The below figure shows functional block diagram of the sensor.

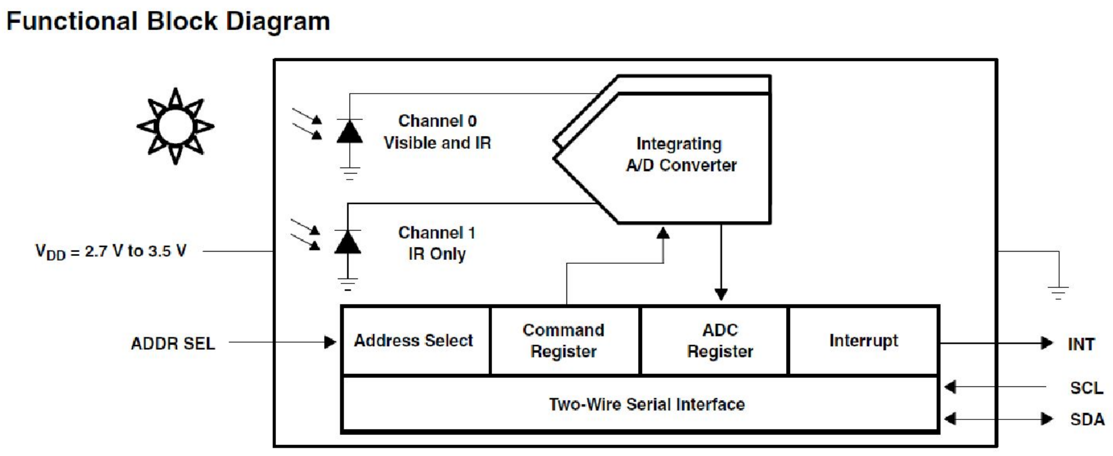


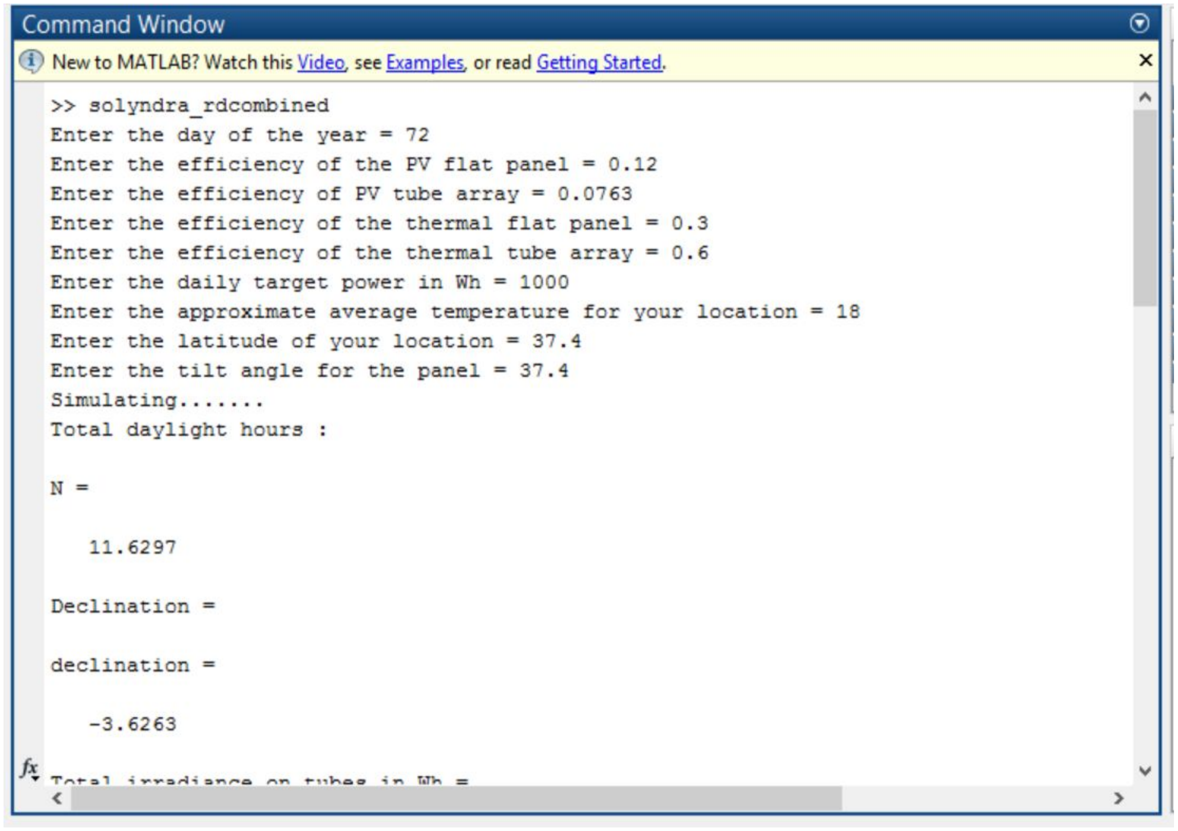
Figure 6: TSL2561 Register Blocks

The sensor works on I2C protocol with host microcontroller. The address register contains the I2C address of the device. The ADC samples the data at a very high rate and the output is the value corresponding to light which fell on the photodiodes for a very small span of time. These values are needed to be integrated before using in calculations. Thus along with ADC register and Interrupt register which are used for sampling and I2C operation respectively, the command register defines the period so the values falling in that span are integrated and passed to microcontroller through I2C. The sensor gives output in terms of counts depending on analog photodiode input.

The Adafruit library for the sensor was the first choice to use as the manufacturer of the model that we are using is Adafruit, but despite of who wrote the library, all of them are made to return values in lux that is lumens/m^2. Basically the libraries are made to calculate luminosity that is measurement of visible light. While the quantity needed for the project is irradiance which is amount of energy calculated in Watt/m^2. The Adafruit library was very hard to edit to make it fit for the project. The variables which receive the counts are all made private in class and then further the library is created on them. This makes it complex to access them. While the Sparkfun library have functions to directly return the count values. After having a hold of the counts, they can be used to compute irradiance using factors mentioned in the datasheet.

5. Experiments

We wrote a MATLAB program based on Solar geometry and equations by referring "Renewable and Efficient Power Systems" by Gilbert M. Masters, "Solar Engineering of Thermal Processes" by Duffie and Beckman and "Mini-Grids for Rural Electrification of Developing Countries" by Bhattacharya and Palit. The MATLAB program screen shot is shown below.

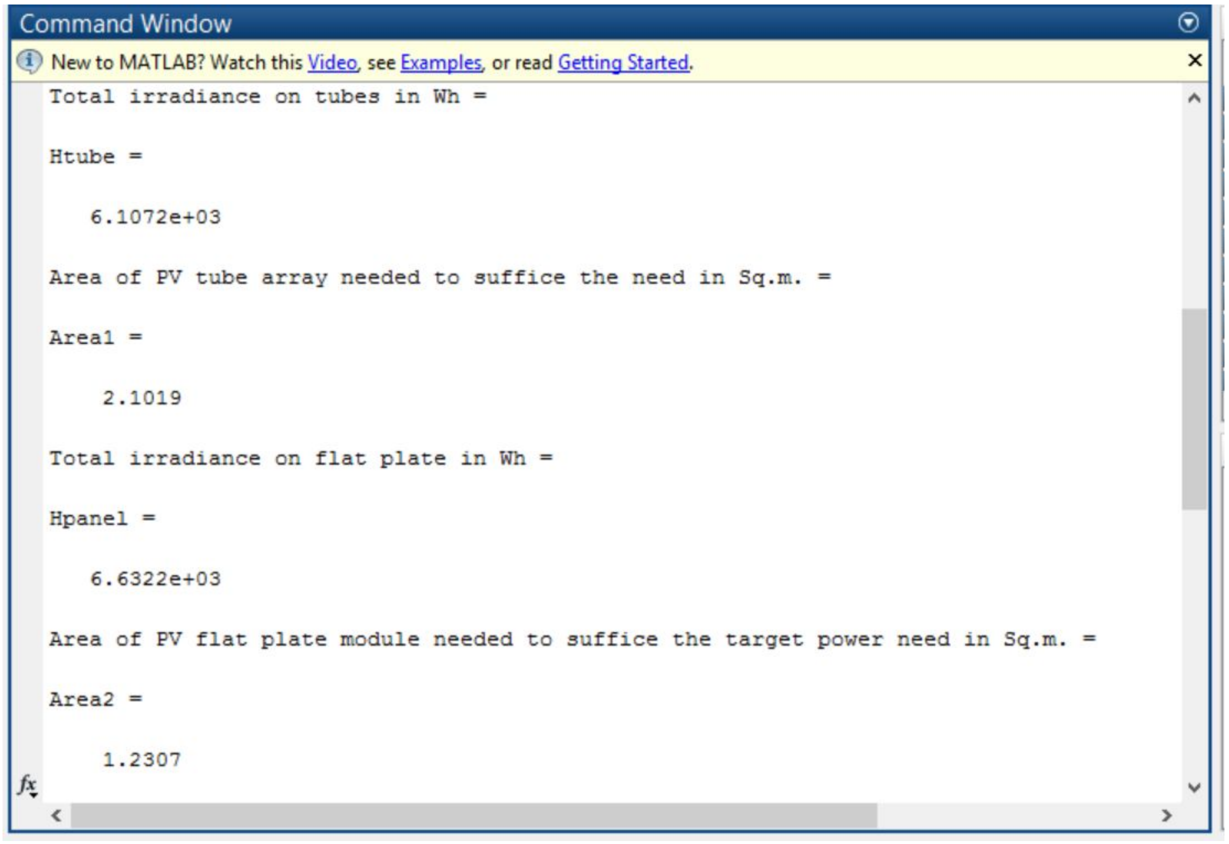


Command Window

New to MATLAB? Watch this Video, see Examples, or read Getting Started.

```
>> solyndra_rdcombined
Enter the day of the year = 72
Enter the efficiency of the PV flat panel = 0.12
Enter the efficiency of PV tube array = 0.0763
Enter the efficiency of the thermal flat panel = 0.3
Enter the efficiency of the thermal tube array = 0.6
Enter the daily target power in Wh = 1000
Enter the approximate average temperature for your location = 18
Enter the latitude of your location = 37.4
Enter the tilt angle for the panel = 37.4
Simulating.......
Total daylight hours :

N =

   11.6297

Declination =

declination =

   -3.6263

Total irradiance on tubes in Wh =
```

Figure 7: MATLAB Simulation Screenshot 1


Command Window
New to MATLAB? Watch this Video, see Examples, or read Getting Started.

```
Total irradiance on tubes in Wh =

Htube =

   6.1072e+03

Area of PV tube array needed to suffice the need in Sq.m. =

Area1 =

    2.1019

Total irradiance on flat plate in Wh =

Hpanel =

   6.6322e+03

Area of PV flat plate module needed to suffice the target power need in Sq.m. =

Area2 =

    1.2307
```

Figure 8: MATLAB Simulation Screenshot 2

The program asks for many data and returns the following:

- Total daylight hours on that day
- Solar Declination (Used to calculate the simulated irradiance)
- Returns irradiance on the flat panel
- Area of the flat panel needed to suffice the daily need of entered amount of energy
- It also returns many interesting plots and data but those are beyond the scope of this document.

One thing to notice that the simulation do consider airmass but do not consider weather and cloud conditions, and hence the simulated values and practical values may differ at any time. From the figures above we get to know that on day 72nd that is the day simulation is done the irradiance value (simulated) for a day is 6.32 kWh/m^2. Then the program also calculates the area needed to suffice a daily given target power which is 1kWh which is 1.2307 m^2. Now, we run the smart solar panel system for a minute on the same day to get the following readings as shown in the below figure.

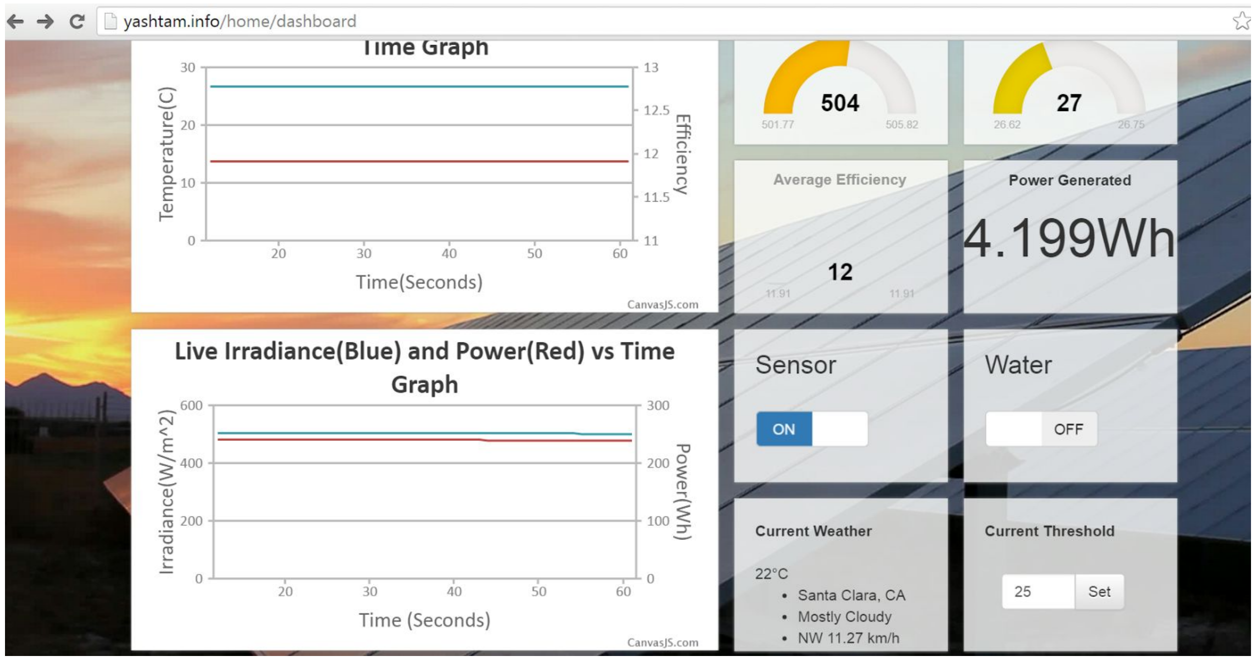


Figure 9: GUI of Smart Solar Panel

The code of the system is programed to calculate values considering a 4 m^2 solar flat panel. From the figure, we can observe that 4.2 Wh is generated for 4 m^2 in 60 seconds. For 1 m^2, the readings will be 1.05 Wh/m^2. Power generated in one day will be 1.05 x 60 x 12 = 756 Wh. Now considering the same target need of 1kWh as assumed in the simulation, the area needed to suffice that target power need will be 1000/756 = 1.32 m^2. From the simulation we get the value of 1.22 m^2 which is slightly lesser. This difference is caused due reduced practical irradiance because cloud and weather conditions cannot be simulated in program. Thus we can conclude that the proposed smart solar panel system is reliable in terms of technical correctness, though not very precise. Another section on the GUI that is current weather conditions is displayed so as to get the idea of the difference in simulated and practical values. The threshold text box allows us to set a temperature value so that above that temperature, the smart solar panel system sprays water on the panel.

6. Future Scope

This implementation can be refabricated in a box such that placement of that box over a particular location will give on a given day, the box will simulate that solar panel of a given efficiency and area and the system will give user an Idea how successful installing solar system on that location will be.

7. References

This document is prepared as a how-to for making a Smart Solar panel. All the materials are derived from respective hardware datasheets and manufacturer websites. Any other material, except code, idea, GUI and simulations which are personal original work, are mentioned there itself in reference to modules.