Smart Solar Lamp

Final Project Report

Faculty Mentor: **Prof Joseph John**

Group D-03

Ram Prakash Sri Sai Seeram, 170070047
Aaron John Sabu, 170070050
Koustav Jana, 170070051

Abstract

Energy crisis demands an efficient use of solar energy, with the use of solar-powered street lamps being one such step in that direction. Although the use of this technology is reported in various places across the country, the reliability and durability of these street lamps is still under scrutiny. Inefficient battery management, having an adverse effect on the battery life and thus the longevity of the device, remains one of the major concerns. Our project aims at coming up with a standalone solution for the Smart Solar Lamp while addressing the above mentioned problem. It will charge the battery using a solar panel, and will power an LED lamp when the light sensor suggests a poor ambient light intensity. A judicious use of the battery will be ensured by controlling the LED light intensity in feedback to the sensed battery voltage. Circuits will be implemented for both overcharge-protection and undercharge-protection.

1 Motivation and Project Application

The general implementations of the solar lamp are to turn on the lamp when needed and turn off when not. The problem with such an implementation is that it disregards the battery percentage available to it. If there is some irregularity in the sunlight received by the lamp due to factors such as rain, clouds etc., the battery will not have been charged enough and uncontrolled operation would drain the battery to undercharge. This might affect the battery health and the lamp would not be able to light up in the night due to the lack of power

To counter this, the intensity can be reduced when there is not a need for brighter light or there is not enough battery charge to support a brighter light. This allows the lamp to operate for more time reliably, until at least it can be recharged by sunlight

2 Project Objectives

- Build a stand-alone self-powered solar lamp that lights up when needed (dark conditions) using the power stored in the battery during the day
- Control the intensity of the lamp so that it can light up without affecting the health of the battery in spite of some irregularities in the sunlight available
- If time permits, integrate the lamp with IoT to enable monitoring of the device with the help of an android app. Monitored parameters include charge percent, status of charging, status of lamp

3 Specifications

- **Solar Panel** (Loom Solar @ $1000W/m^2$)
  - $V_{OC} = 22.5V$
  - $I_{SC} = 0.55A$
  - *At Maximum Power Point*: P = 10W, V = 19.25V, I = 0.52mA
- **Battery** (Panasonic UP-PW1245P1)
  - $V_{typ} = 12V$, $C = 7Ah$
  - We consider that at a slow discharge rate, as in our case, V_{min} of battery = 10.8V and V_{max} is 13.6V
- **LED Lamp**
  - $P = 5W$

For summer, winter and monsoon, we have a current of 0.5A from the solar panel (max irradiation):

- Summer : 12 hours daylight
  - On an average, we can assume 10 hours of full brightness
  - Charge available from battery = $0.5 \cdot 10 = 5Ah \approx 0.71C_{battery}$ (assume no charging loss)
  - The battery can support the LED up to (Charge in Ah)$(V_{lamp})/P_{Lamp} = 5 \cdot 12/5 = 12hr$ (assume loss from other circuits to be 0W and the lamp to consume 5W)
  - The charge stored during day can support the circuit for 12 hours of night if P_{loss} = (Charge [Ah])$(V_{Circuit})/12 = 5W$
- Winter : 10 hours daylight
  - On an average, we can assume 8 hours of full brightness
  - Charge available from battery = $0.5 \cdot 8 = 4Ah \approx 0.57C_{battery}$
  - The battery can support the LED up to (Charge in Ah)$(V_{lamp})/P_{Lamp} = 4 \cdot 12/5 = 9.6hr$
  - The charge stored during day can support the circuit for 14 hours of night if P_{loss} = $(Charge[Ah])(V_{Circuit})/14 = 3.43W$
- Monsoon : 10 hours daylight
  - On an average, we can assume 6 hours of full brightness
  - Charge available from battery = $0.5 \cdot 6 = 3Ah \approx 0.44C_{battery}$
  - The battery can support the LED up to $(Charge[Ah])(V_{lamp})/P_{Lamp} = 3 \cdot 12/5 = 7.2hr$
  - The charge stored during day can support the circuit for 14 hours of night if P_{loss} = $(Charge[Ah])(V_{Circuit})/14 = 2.57W$

4 System Design

The basic idea of intensity control is to place a series resistance with the lamp (Majority of the power is lost through the lamp) to control the power consumed by it. To control the series resistance value which is connected, an Arduino microcontroller module is used. The series resistance is chosen based on the above seasonal calculations so that, on an average, in a given season, the calculated rating of that season is met.

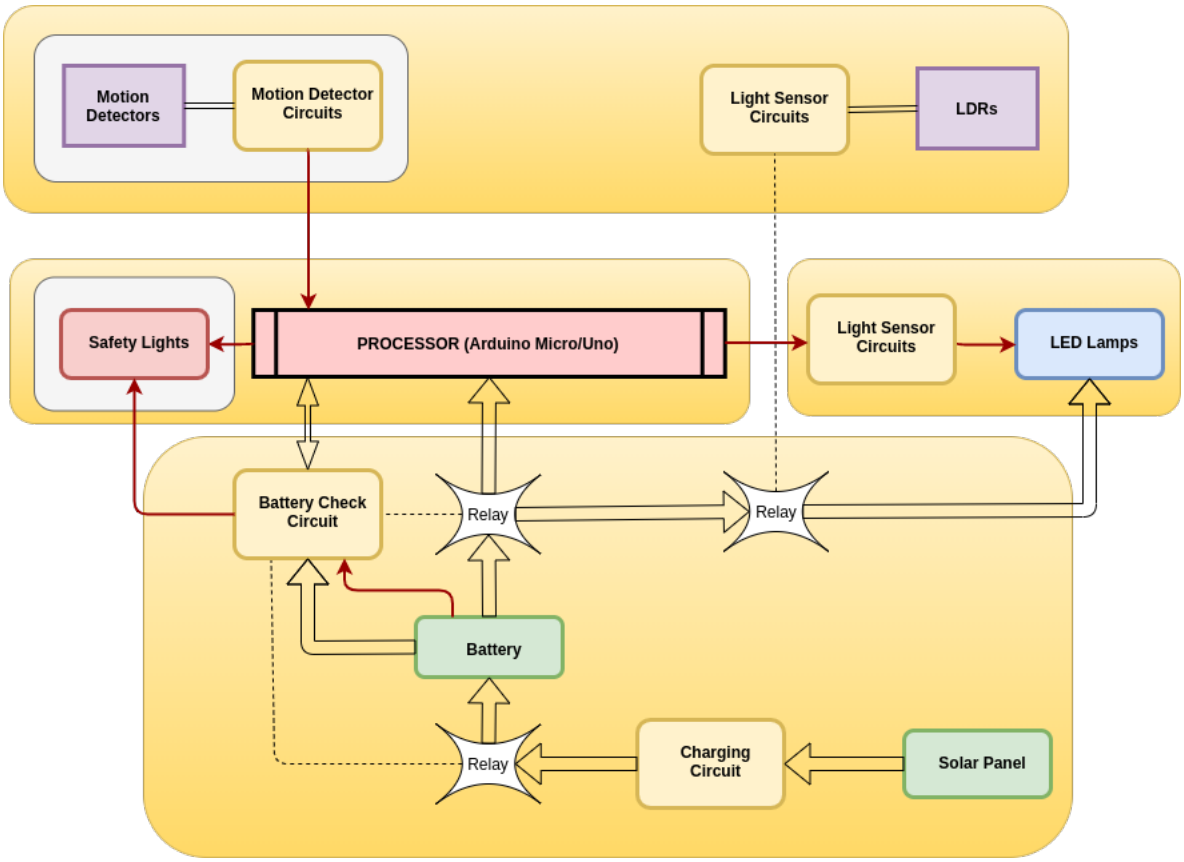


Figure 1: Block Diagram

4.1 Power Module

4.1.1 Charging the Battery(i)

- Maximum charge current rating of the battery = 0.1CA = 0.7A
- The solar panel gives a maximum of 0.5A under short circuit conditions ($I_{SC} = 0.55A$), so the battery can be charged directly with the solar panel without the need of a resistor
- A diode of maximum current rating 1A: 1N4007 is added in between to prevent the backward flow of current from the battery to the solar panel

$$P_{loss,diode} = V_{forward} * I_{diode} = 0.5W$$

4.1.2 Overcharge Protection Circuit(ii)

- The battery needs to stop charging on reaching 100% charge level (13.6V) to prevent damage
- To account for momentary fluctuations in the voltage and to prevent an immediate charge-discharge cycle, a Schmitt trigger $V_{TH} - V_{TL} = 0.5V$ is used. Since we have a single supply from the battery, LM324 is used for all operational amplifier circuits
- The circuit used is given in (ii) of Figure(2), where the threshold values are given as:

$$V_{TH} = V_Z \left(1 + \frac{R_1}{R_g} + \frac{R_1}{R_g}\right)$$

$$V_{TL} = V_Z \left(1 + \frac{R_2}{R_1 + R_2}\frac{R_1}{R_g}\right)$$

- Choosing $V_Z = 10V$, $R_1 = 1k\Omega$, $R_2 = 20k\Omega$, $R_g = 3.3k\Omega \implies V_{TH} = 13.53V$, $V_{TL} = 12.89V$

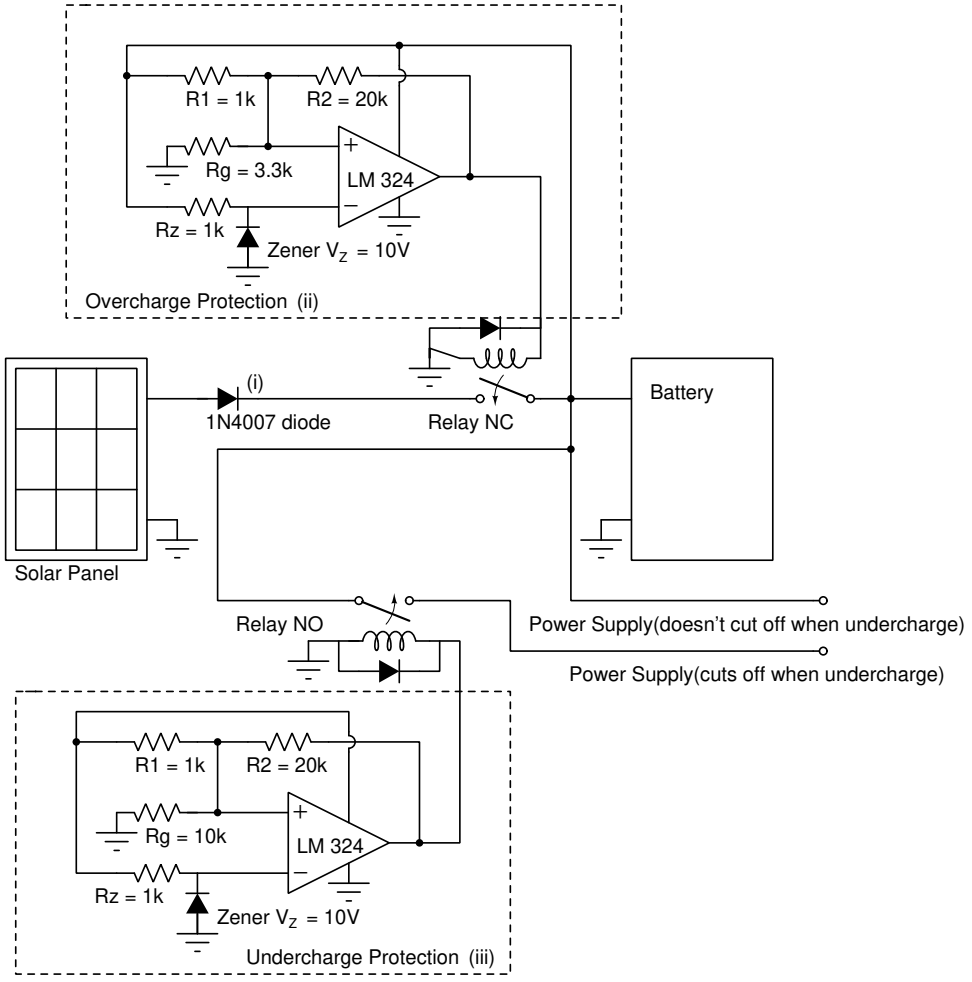


Figure 2: Power Module

- The output of the circuit is connected to a normally closed relay directly(since the relay has a coil resistance of 960Ω to limit the current), which opens the charging circuit when Schmitt trigger outputs $+V_{sat}$ and closes else
- The difference $0.64V \approx 23\%$ of battery is easily drained during the night, hence the circuit will not be stuck in overcharge for more than a day, allowing recharge the next day

4.1.3 Undercharge Protection Circuit

- We need to cutoff discharge of the battery when the battery reaches close to 0% (10.8V) to prevent it's damage
- This is similar to the previous circuit. We use a normally open relay which closes the circuit when output of the Schmitt trigger is $+V_{sat}$ and opens else. Refer (iii) of Figure(2)
- Choose, $V_Z = 10\text{V}$, $R_1 = 1\text{k}\Omega$, $R_2 = 20\text{k}\Omega$, $R_g = 10\text{k}\Omega \implies V_{TH} = 11.5\text{V}$, $V_{TL} = 10.95\text{V}$
- The difference $0.55\text{V} \approx 19.6\%$ of battery is easily recharged during the day, hence the circuit will not be stuck in undercharge for more than a day if properly charged, allowing LED to glow the next day

Two supplies generated from the power module, one which is regulated by the undercharge protection circuit and one otherwise. The unregulated supply will be used to power the Arduino and undercharge circuit, and all other functionalities are provided power from the regulated supply.

4.2 Sensor Module

4.2.1 Voltage Sensing

- Arduino accepts input of 0-5V and maps it to 0-1023. To map the voltage range 10.8-13.6V we use an op-amp based difference calculator and subtract 10V from battery voltage
- Refer Figure(6).The output of the circuit is,

$$V_{out} = V_{in}\left(1 + \frac{R_2}{R_1}\right)\left(\frac{R_4}{R_3 + R_4}\right) - \frac{R_2}{R_1}V_z$$

- Choose all the resistors to be of value 1kΩ and $V_Z = 10\text{V} \implies V_{out} = V_{in} - 10$

4.2.2 LDR-based LED switching circuit

- This circuit ensures the on-off switching of the LEDs based on the ambient light intensity sensed using a light-dependent resistor (resistance variation based on the intensity of light)
- The change in resistance is captured using a voltage divider given as input to a Schmitt trigger designed using a single-supply op-amp (LM324) and compared to a set reference voltage.
- The use of Schmitt trigger ensures the output to be robust, making it insensitive to unexpected and slight fluctuations in the ambient light intensity (hence the LDR resistance). An inverting Schmitt trigger circuit is used as shown in Figure 3

- The threshold values of LDR resistance at which switching happens, R_H and R_L are given by (assuming no loading from R_3 and R_4):

$$\frac{R_H}{R_s + R_H} = \frac{1}{(\alpha+1)(\beta+1)} + \frac{\alpha}{\alpha+1} \tag{1}$$

$$\frac{R_L}{R_L + R_s} = \frac{1}{(\alpha+1)(\beta+1)} \tag{2}$$

  where $\alpha = \frac{R_1}{R_2}$ and $\beta = \frac{R_3}{R_4}$.

- Based on experimental observations, we have chosen $R_H = 600k\Omega$ and $R_L = 160k\Omega$
  Thus choosing $R_1 = 100K\Omega$, $R_2 = 100K\Omega$, $R_3 = 620\Omega$, $R_4 = 1k\Omega$ and $R_s = 180K\Omega$ ensures a working circuit as per requirements and, since R_3 and R_4 are much less than R_1 and R_4, they do not result in loading the rest of the circuit

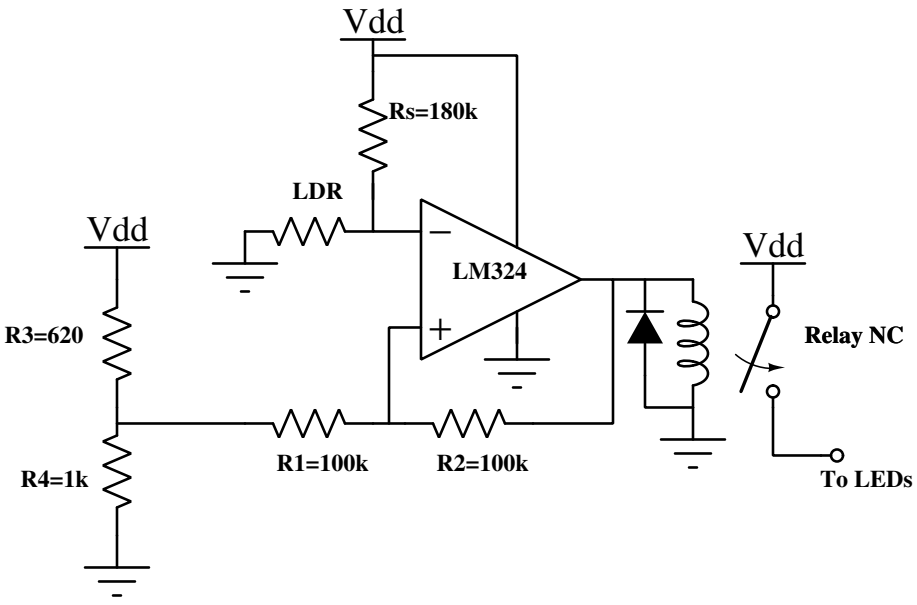


Figure 3: LDR-based LED switching circuit

4.3 Lamp Module

- The decoder outputs 0V to the pin which is selected and rest are set to be 5V. An open collector buffer(7407) was used to drive the relay circuit using the decoder output

- Ref Figure(4). We employed 4 levels of intensity, three of which correspond to the series resistance to meet the power requirements of the three seasons

- The last level is a low power mode which takes care of any irregularity that occurred

- If P is the power that can be consumed by the circuit during a season for the lamp to operate for 14hours, and the lamp is modelled as a voltage source(V_{on}) + a resistance(R_d), assuming power loss in rest of the circuit is $P_{loss,c}$

$$\frac{P - P_{loss,c}}{V_{dd}} = \frac{V_{dd} - V_{on}}{R_d + R_s}$$

$$\text{Series resistance to be connected} = \frac{12(12 - V_{on})}{P - P_{loss,c}} - R_d$$

4.4 Processor Module

- The Arduino takes input the current voltage of the battery and the state of lamp(on or off, detected from the relay of the LDR circuit)
- Whenever the LDR circuit turns the lamp on, Arduino based on the current battery percentage chooses an appropriate intensity level so the light will be able to operate for 14 hours(max night time) or selects none if condition is not met
- Arduino outputs the chosen resistance as a two bit binary number to the input of a decoder. It selects none by giving a low input to the enable of the decoder. Refer Figure(6)

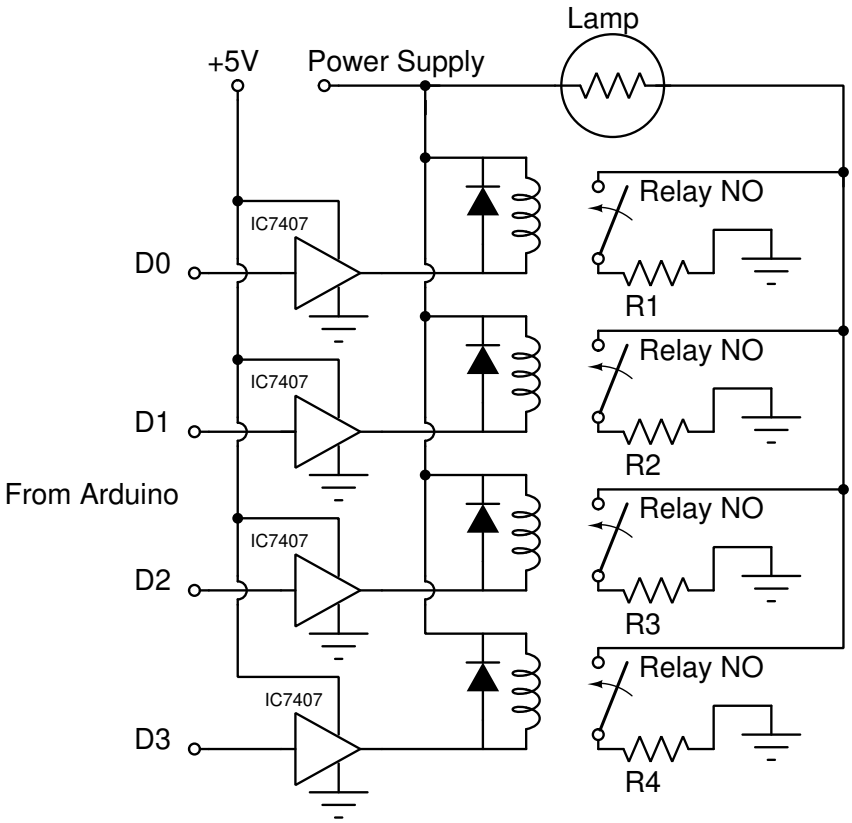


Figure 4: Intensity Control

- Directly quoting a website which mentions about the power consumption of the Arduino, an Arduino Uno runs less than one day on a 9 V battery because it uses about 45 mA current. As a result, it may be a better option to use an Arduino Mini, an Arduino Micro or an Arduino Nano since they consume phenomenally less quantities of power when compared to the Arduino Uno (which consumes about 0.5 W).
  (eg: Using an Arduino Pro Mini in power-down sleep, the power consumption can go as low as $23\mu A$ at 5.0V)

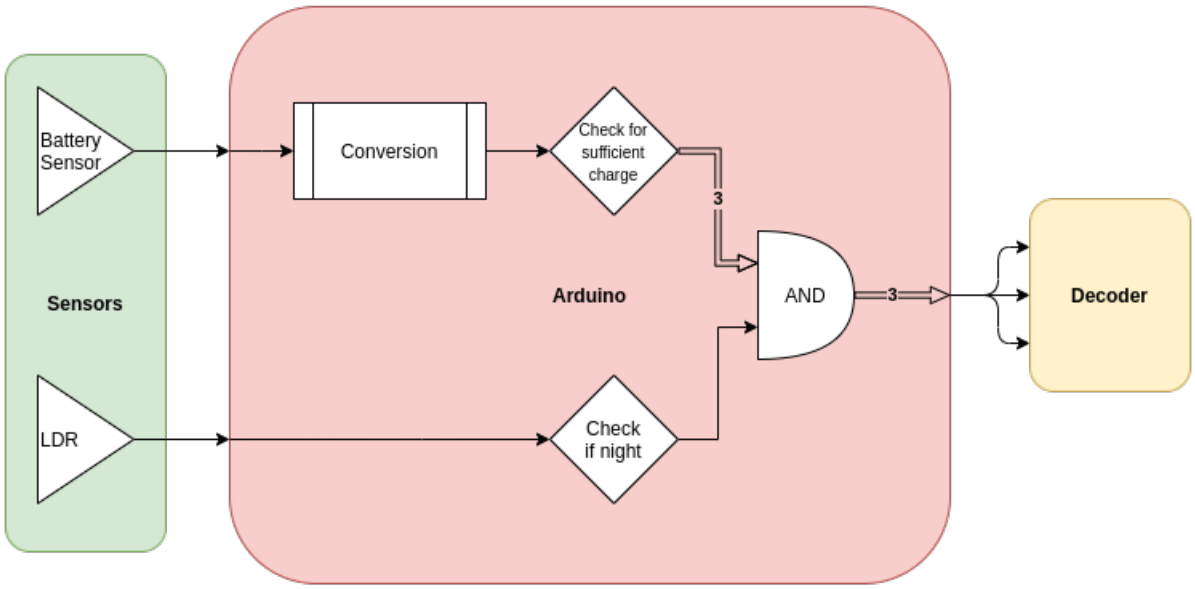


Figure 5: Arduino Code Flowchart

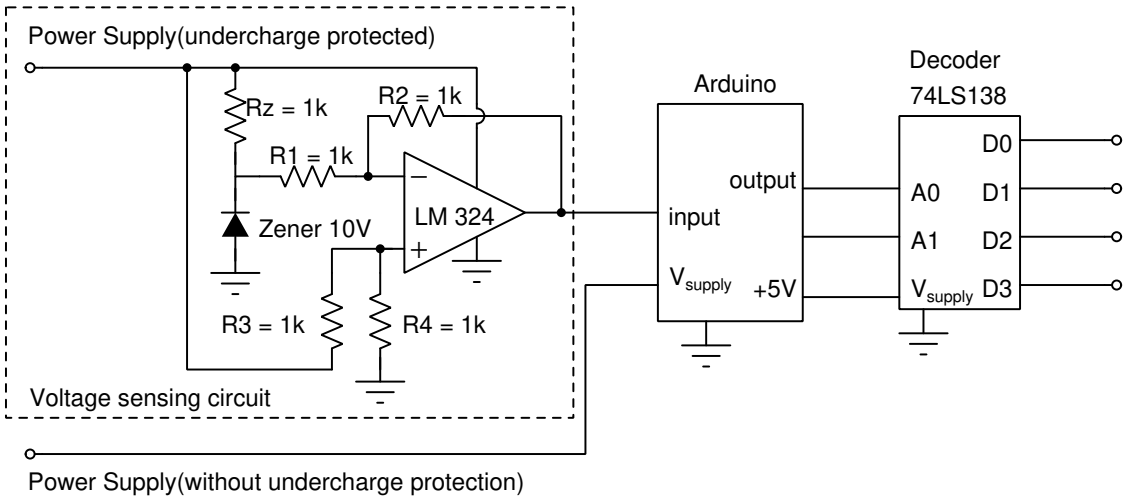


Figure 6: Processor Module

5 Preliminary Work

5.1 Overcharge Protection Circuit

The overcharge protection circuit has been wired based on the design proposed. However there were issues regarding the V_{TH} and V_{TL} values obtained from the circuit; i.e. the two values had an offset of +0.3V in the developed circuit. This was due to error in the Zener voltage of the diode.

5.2 Plan for Arduino

The design can be modified such that the overcharge and undercharge protection circuits are incorporated as logical instructions in the Arduino. This can be done with higher precision. The input to the Arduino will be the battery voltage level after compensation for the known 10.0V level (using Zener diode or a similar mechanism) and amplification to the range of 0V to 5V; i.e. the range (10.8V, 13.6V) will be mapped to (0.0V, 5.0V) which the Arduino will internally map to (0, 1023).

6 Further Completed Work

6.1 Charging the Battery

The battery was charged using the solar panel as part of which readings were taken with different values of resistances connected in series to the battery charging circuit. The following readings were obtained from the experiment:

R_S (Ω)	V_{Solar} (V)	$V_{Battery}$ (V)	I_{charge} (mA)
100	18.20	12.37	43.4
50	16.50	12.39	68.2
10	14.00	12.43	83.2
5	13.40	12.36	112.5

On analyzing this data, we were able to conclude that using a 5Ω resistor is apt for the circuit. There is no chance of battery damage due to high current flow in normal conditions since the battery charging current limit is $700mA$(as mentioned in datasheet 10% CA = 10% 7Ahr).

6.2 Undercharge and Overcharge Protection using Arduino

As mentioned above, there were some issues faced by us while trying a pure analog implementation of the overcharge and undercharge protection circuits, the most critical ones being:

- The $+V_{sat}$ value of the single-supply op-amp LM324 was seen to be dependent on the voltage source, thus undermining our efforts of sensing the voltage level of the battery using an op-amp based circuit
- To get rid of the above problem, Zener diodes were introduced, however the slight dependence of the Zener reverse bias voltage on the current led to an error of up to 0.3V in the V_{TH} and

V_{TL}, which could have been quite detrimental to the performance of our design, given the safe operating range(voltage level) of the battery is not much (about 2V).

As a result of these reasons, a complete shift to a digital Arduino-based implementation was finalized, and hence the hysteretic switching for undercharge, normal and overcharge conditions was done on software, thus enabling dynamic configurability of the thresholds from our side.
For passing the voltage reading to the Arduino, it had to be reduced to a range which was acceptable for the inbuilt ADC of the Arduino(0-5V). This was achieved using a resistor-based voltage divider as shown in figure below rather than the op-amp based solution that was initially proposed (A resistor divider was sufficient since we want to just downscale the voltage). The resistors have been chosen so keeping in mind that they are not too low which causes unnecessary draining of power and also not too high to avoid unnecessary loading.

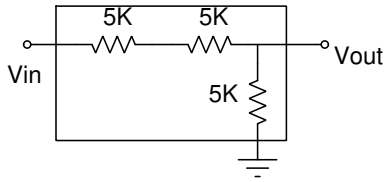


Figure 7: Voltage Divider Circuit

The scaled down voltage input was then given to the Arduino using its Analog Input pin, which further mapped it to the digital range 0-1023. The decision-making process regarding the battery being in undercharge, normal or overcharge mode is done on the basis of this input.
Input voltage vs output voltage of the resistor divider circuit was measured as follows:

V_{in}(V)	V_{out}(V)	V_{in}(V)	V_{out}(V)
10.51	3.48	12.52	4.15
10.81	3.58	12.88	4.26
11.35	3.75	13.03	4.32
12.07	4.00	13.60	4.50

Table 1: Voltage Divider Readings

The data measured was used to fit to the equation:

$$V_{scaled} = m \times V_{battery} + c \tag{3}$$

$$m = 0.3316 \;\; c = -0.0055$$

to calibrate the value read by the Arduino. The above equation was then used to get back the actual battery voltage from the scaled down input and this eased our task of setting precise threshold values for not only the overcharge and undercharge protection module but also the intensity control module as described below.Arduino can now measure a voltage fluctuation of:

$$\Delta V = \frac{1}{m} \cdot \frac{V_{max,Arduino}}{num.\ divisions\ in\ Arduino}$$

$$\implies \Delta V = \frac{5}{0.3316 * 1024} = 0.0147 \approx 0.53\%\ Full\ Charge$$

The threshold values used are is as follows:

- Overcharge
  - $V_{TH} = 13.89\ V$ 100% Full Charge
  - $V_{TL} = 13.53\ V$ 75% Full Charge
- Undercharge
  - $V_{TH} = 11.08\ V$ 10% Full Charge
  - $V_{TL} = 11.50\ V$ 25% Full Charge

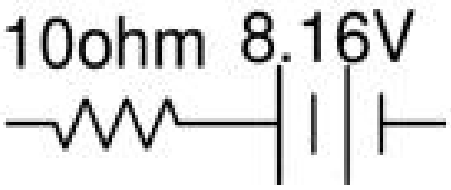
6.3 Construction of a prototype LED lamp

13 LED strips of three LEDs in series were used in parallel to develop a prototypical lamp,soldered on a perforated board for testing.The lamp IV characteristics are shown in table 2

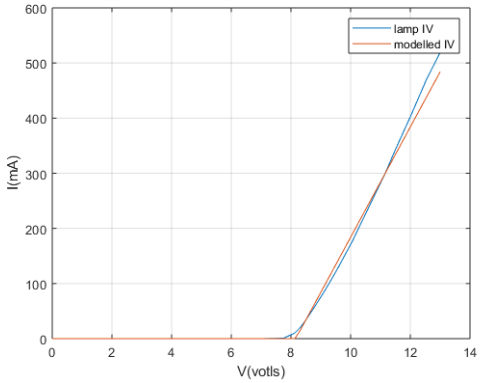
V_{Lamp}(V)	I_{Lamp}(mA)	V_{Lamp}(V)	I_{Lamp}(mA)
6.00	0.00	9.60	130.7
7.00	0.00	10.06	177.0
7.76	0.90	10.47	223.0
8.13	10.3	11.16	300.0
8.31	20.0	11.51	344.0
8.48	32.4	12.03	406.0
8.76	55.0	12.54	470.0
9.10	84.0	13.00	519.0
9.38	109.6		

Table 2: Lamp IV

The above data is used to approximate the LED as a battery + resistor model, which can be further used in the calculations of resistor values of the lamp control circuit.
Lamp in turn on : $I = \frac{V - V_{on}}{R_{on}}$ $V_{on} = 8.16V$ $R_{on} = 10\Omega$
Lamp in cutoff : $I = 0$



(a) Lamp model (turn-on mode)



(b) Lamp I-V Characteristics

Figure 8

6.4 Intensity Control Module using Arduino

The Intensity Control Module tunes the intensity of the solar lamp based on the present voltage level of the battery by choosing the appropriate resistor from the resistor bank. This practice ensures the efficient use of the battery. A basic overview of the plan in order to achieve this has been described above in section 4.4 and the adjoining figures.
The circuit was successfully developed and tested on a breadboard using multiple resistances as a prototype. Here, the resistances were chosen based on intuition for the purpose of testing although the actual circuit would have consisted of more carefully calculated resistances. This testing was done on smaller LEDs. The final circuit will interface with the lamp which was constructed as part of Section 6.3.
We have presently divided the safe operating range of the battery i.e 11.3V to 13.1V into 4 sub-ranges, each sub-range corresponding to a particular resistor. Based on the current battery voltage level, the Arduino checks for the correct sub-range and chooses the corresponding resistor.
However the appropriate resistance calculation based on efficient power consumption and optimum intensity required is yet to be done. This could not be completed because we need to account for the power consumed by the entire circuit apart from the LED for calculating the resistances.

The LDR values at required turn-on and turn-off times were found out to be $R_H = 600k\Omega - 700k\Omega$, $R_L = 80k\Omega - 110k\Omega$. Using a potentiometer, these resistances were tuned as the turn-on and turn-off values of LDR circuit respectively.

7 Completion of Proposed Milestones

Complete designing and implementing the automated turn on/off capability of the solar lamp	☑
Add intensity control and battery management (overcharge + undercharge protection)	☑
Add additional features such as PIR sensors, RTC and IoT	☐

References

[1] Panasonic VRLA Technical Handbook: Industrial Batteries for Professionals

[2] Panasonic UP-PW1245P Battery Datasheet

[3] SN74LS07 Hex Buffer

[4] LM324N Quad Operational Amplifier

[5] 1N4007 General-Purpose Rectifier

[6] 74LS138 3-line to 8-line decoder

A Arduino Code

Given here is the code developed for the Arduino to this point:

```
/// GLOBAL VARIABLES ///

int   battPin     = A0;
int   LDRPin      = A1;
int   OCPin       = 8;
int   UCPin       = 9;
int   L0Pin       = 5;
int   L1Pin       = 6;
int   L2Pin       = 7;

float mslope      = 0.3316;
float cintercept  = -0.0055;

float sensorVal   = 0.00;
float voltageVal  = 0.00;
float sunSense    = 0.00;

bool  OC          = false;
bool  UC          = false;
float buf_OH      = 13.53;
float buf_OL      = 12.89;
float buf_UH      = 11.50;
float buf_UL      = 11.08;
float buf_LT      =  2.50;

bool  LED_0       = false;
bool  LED_1       = false;
bool  LED_2       = false;
float buf_00      = 11.50;
float buf_01      = 12.25;
float buf_10      = 13.00;

///// DELAY_SECOND /////

void delay_sec(int seconds)
{
  for (int i = 0; i < seconds; i++)
  {
    delay(1000);
  }
}
```

```
//////// SETUP /////////

void setup()
{
  Serial.begin(9600);
  pinMode(OCPin,    OUTPUT);
  pinMode(UCPin,    OUTPUT);
  pinMode(battPin,  INPUT);
  pinMode(LDRPin,   INPUT);
  pinMode(L0Pin,    OUTPUT);
  pinMode(L1Pin,    OUTPUT);
  pinMode(L2Pin,    OUTPUT);
}

/////// PROCESS ////////

void loop()
{
  ///// DATA COLLECTION /////

  sensorVal  = analogRead(battPin);
  sunSense   = analogRead(LDRPin);
  //voltageVal = ((sensorVal*5.00/1024) + 9.66) / 0.977;
  voltageVal=(((sensorVal*5.00)/1024.0)-cintercept)/mslope;

  //// HEALTH MONITORING ////

  if (voltageVal < buf_UL)
  {
    UC = true;
  }
  else if (voltageVal > buf_UH)
  {
    UC = false;
  }

  if (voltageVal < buf_OL)
  {
    OC = false;
  }
  else if (voltageVal > buf_OH)
  {
    OC = true;
  }

  digitalWrite(OCPin, OC);
  digitalWrite(UCPin, UC);
```

```
  //// INTENSITY CONTROL ////

//  if ((sunSense < buf_LT) && !UC)
  if (!UC)            // Temporary statement
  {
    LED_0 = true; // LED_0 -> 7407 -> RELAY
    if (voltageVal < buf_00)
    {
      LED_1 = true; // LED_1 -> DECODER_1 -> RELAY
      LED_2 = true; // LED_2 -> DECODER_0 -> RELAY
    }
    else if (voltageVal < buf_01)
    {
      LED_1 = true;
      LED_2 = false;
    }
    else if (voltageVal < buf_10)
    {
      LED_1 = false;
      LED_2 = true;
    }
    else
    {
      LED_1 = false;
      LED_2 = false;
    }
  }
  else
  {
    LED_0 = false;
    LED_1 = false;
    LED_2 = false;
  }
  digitalWrite(L0Pin, LED_0); digitalWrite(L1Pin, LED_1); digitalWrite(L2Pin, LED_2);

  ////// USER FEEDBACK //////

  Serial.print("Voltage    = "); Serial.println(voltageVal);
  Serial.print("UC State   = "); Serial.println(UC);
  Serial.print("OC State   = "); Serial.println(OC);
  Serial.print("Lamp State = "); Serial.println((sunSense < buf_LT) && !UC);
  Serial.print("L0 = "); Serial.print(LED_0);
  Serial.print(", L1 = "); Serial.print(LED_1);
  Serial.print(", L2 = "); Serial.print(LED_2); Serial.println(""); Serial.println("");
  delay(500);
}
```