ARDUINO SOLAR CHARGE CONTROLLER (PWM)

Build instructions:
The whole circuit shown above is divided into 6 parts

1. Voltage sensing
2. PWM signal generation
3. MOSFET switching and driver
4. Filter and protection
5. Display and Indication
6. LOAD On/OFF
Voltage sensing

As the ARDUINO analog pin input voltage is restricted to 5V, I designed the voltage divider in such a way that the output voltage from it should be less than 5V. I used a 5W (Voc=10V) solar panel and a 6v and 5.5Ah SLA battery for storing the power. So I have to step down both the voltage to lower than 5V. I used R1=10k and R2=4.7K in sensing both the voltages (solar panel voltage and battery voltage). The value of R1 and R2 can be lower one but the problem is that when resistance is low higher current flow through it as a result large amount of power \( P = I^2R \) dissipated in the form of heat. So different resistance value can be chosen but care should be taken to minimize the power loss across the resistance.

I have designed this charge controller for my requirement (6V battery and 5w, 6V solar panel), for higher voltage you have to change the divider resistors value. For choosing the right resistors you can also use an [online calculator](#).

In code I have named the variable "solar_volt" for voltage from the solar panel and "bat_volt" for battery voltage.

\[
V_{out} = \frac{R2}{(R1+R2)}V
\]

Let panel voltage = 9V during bright sunlight

R1=10k and R2=4.7 k

solar_volt = \frac{4.7}{(10+4.7)}*9.0 = 2.877v

Let the battery voltage is 7V

bat_volt = \frac{4.7}{(10+4.7)}*7.0 = 2.238v
Both the voltages from voltage dividers are lower than 5v and suitable for ARDUINO analog pin

**ADC Calibration:**

Let's take an example:

Actual volt/divider output = 3.127
2.43 V is eqv to 520 ADC
1 is eqv to 0.004673V

Use this method to calibrate the sensor.

**ARDUINO CODE:**

```c
for(int i=0; i<150; i++)
{
    sample1+=analogRead(A0); // read the input voltage from solar panel
    sample2+=analogRead(A1); // read the battery voltage
    delay(2);
}

sample1=sample1/150;
sample2=sample2/150;
solar_volt=(sample1*4.673*3.127)/1000;
battery_volt=(sample2*4.673*3.127)/1000;
```

.`
PWM Signal generation

**Pulse Width Modulation**

- 0% Duty Cycle – `analogWrite(0)`
- 25% Duty Cycle – `analogWrite(64)`
- 50% Duty Cycle – `analogWrite(127)`
- 75% Duty Cycle – `analogWrite(191)`
- 100% Duty Cycle – `analogWrite(255)`

![Image of PWM Signal generation](image_url)
PWM (Pulse Width Modulation) is a technique by which we simply control the
digital output signal by switching it on and off very quickly, by varying the width of
the on/off duration, it will give the effect of varying the output voltage.

\[ V_{out} = \frac{T_{on}}{T_{on} + T_{off}} \times V_{in} \]

Time period \( (T) = T_{on} + T_{off} \)

So \( V_{out} = \frac{T_{on}}{T} \times V_{in} \)

Where \( \frac{T_{on}}{T_{on} + T_{off}} \times 100 \) or \( \frac{T_{on}}{T} \times 100 \) is called Duty Cycle

See the above examples graph (I have taken from Arduino website) for better
understanding.

PWM enables a digital output to provide a range of different power levels, similar
to that of an analog output.

The best example is fading a LED with different light intensity. This can be done
by arduino, using the analogWrite() function.

As Analog output pin is 8bit we can get maximum \( 2^8 = 256 \) or a range of values
between 0 to 255.

Sending the value 255, to the LED input produces 100% duty-cycle, which
results in full power on a PWM pin.

Sending the minimum value 0, to the LED input produces 0% duty-cycle, which
results in no power on a PWM pin.

In Arduino there are 6 PWM pins (3, 5, 6, 9, 10 and 11)

I used pin 6 (timer 0) for generating the PWM signal. Arduino Clock has
maximum frequency of 16MHz, this can be reduce by using the prescaler. By
default Arduino PWM pin have prescaler = 64.

So in pin -6 , I will get a PWM of frequency 976.5625 Hz by default. But this
frequency is quite high for battery charging. So I decided to reduce this frequency
to a lower one( 61.035Hz ) by choosing the prescaler value 1024.

Note : I used this frequency because I found that maximum PWM charge
controller available in the market have frequency 25/50/100Hz. I don't have much
expertise in this field. If anyone have good expertise in battery charging, please
comments about the right frequency selection for PWM . So that I will modify my
code.
**How to change the Arduino PWM frequency:**

You can adjust the PWM frequency by changing a register value. The register associated with this is TCCR0B.

By selecting three Clock Select Bits of TCCR0B register we can set the proper prescaler.

Are you thinking it is difficult? No its very very simple just place the single line code given bellow in your program's void setup() part. Everything is done. I check this by testing the fade example code given in Arduino IDE, it will generate 61Hz.

\[ \text{TCCR0B} = \text{TCCR0B} \& 0b11111000 | 0x05 ; \text{ // prescaling 1024} \]

For more details to set the PWM frequency click [here](#)

---

**How to Choose MOSFETs**

In my charge controller I used two MOSFETs one is for controlling the power flow from solar panel to battery and other is to drive the load. When I started working with MOSFET, little bit confused how to select a right one. After reading so many forum I found that it is so easy. I think anyone can choose the MOSFET by using few guidelines. These are the few important parameters which you will need during choosing a right MOSFET.

**Note:**

I designed the charge controller as per my requirement, you have to choose the MOSFETs according to your system requirement. The charge controller rating is mostly depends on MOSFET rating. So choose carefully.

1. **N or p channel:**

   When a MOSFET is connected to ground and the load is connected to supply voltage, it is considered to be a low-side switch. In a low-side switch, an n-channel device is used. In my charge controller I used IRF 540 for driving the load.

   A high-side switch is used when the MOSFET is connected to the supply voltage and the load is connected to ground. A p-channel MOSFET is usually used in this topology. I used IRF 9530 as the main switching MOSFET for PWM.
See the schematics how load is connected in both the cases.

2. Drain Source Voltage Vds :

When the MOSFET is turned off, the whole supply voltage will be measurable across it, so this rating should be larger than your supply voltage to provide sufficient protection so that the MOSFET does not fail. The maximum voltage a MOSFET can handle varies with temperature.

3. Continuous Drain Current Ids :

This is the amount of current the MOSFET can handle. You need to simply select the device that can handle the maximum amount of current including surges or "peaks". Current rating also decreases with rise in temperature.

So keep sufficient margin in Ids. It is better to take current rating @ 125deg Cel.

4. RDS(on) :

When a MOSFET is "on," it acts as a variable resistor determined by the RDS(on) that changes with temperature with a power dissipation calculated by Iload^2 x RDS(on). So choose a MOSFET with smaller value of RDS(on).

5. Thermal Loss:

It can be found from datasheets: the maximum junction temperature and the thermal resistance from junction-to-ambient for the package. The device's junction temperature is equal to the maximum ambient temperature plus the product of the thermal resistance and the power dissipation (Junction Temperature = Maximum Ambient Temperature + (Thermal Resistance x Power Dissipation) ).

6. Gate Threshold Voltage VGS(th) :

This is the minimum voltage required between the gate and source terminals to turn the MOSFET on. It will need more than this to turn it fully on.

7. Switching Loss:

Charging and discharging the gate capacitance(Cgs) contributes to the switching losses. This loss also depends on the switching frequency. Losses increased with higher switching frequency and gate to source capacitance.

Apart from this, there are several other parameters you have to consider for a good design. For us I think it is sufficient.

For more details on how to choose a right MOSFET clicks [here](here)
Power MOSFET has limitations operation in terms of voltage, current and power dissipation. The power MOSFET current rating is related with the heat dissipated in the devices. This rating will be take in consideration for designing appropriate circuit to protect power MOSFET against high voltage and current, thus cause heat generation.

I used a p-channel MOSFET (IRF9530) for switching between solar panel and battery and a n-channel MOSFET (IRF540) for load.

For higher rated system you have to choose the MOSFET accordingly.

What is a MOSFET Driver?

A gate driver is a power amplifier that accepts a low-power input from a micro controller and produces a high-current drive input for the gate of a high-power MOSFET.
**Why you need a MOSFET driver:**

MOSFETs have a large stray capacitance between the gate and the other terminals, which must be charged or discharged each time the MOSFET is switched on or off. As a transistor requires a particular gate voltage in order to switch on, the gate capacitor must be charged to at least the required gate voltage for the transistor to be switched on. Similarly, to switch the transistor off, this charge must be dissipated, i.e. the gate capacitor must be discharged.

When a transistor is switched on or off, it does not immediately switch from a non-conducting to a conducting state; and may transiently support both a high voltage and conduct a high current. Consequently, when gate current is applied to a transistor to cause it to switch, a certain amount of heat is generated which can, in some cases, be enough to destroy the transistor. Therefore, it is necessary to keep the switching time as short as possible, so as to minimize switching loss.

Therefore the circuit that drives the gate terminal should be capable of supplying a reasonable current so the stray capacitance can be charged up as quickly as possible. The best way to do this is to use a dedicated MOSFET driver.

There are dedicated chip available in the market for MOSFET driving but it is costly. I used a simple MOSFET driver by using a NPN general purpose transistor. I used 2N3904 but you can use any general purpose transistors like 2N2222 or BC547 etc.
Filter and Protection:

Filter:
The capacitor (C1) used after the solar panel at the input side is used as filter which removes any unwanted ripple/noise signal.

I used a 100uF, 35V.

Optional:
You can also put a capacitor in the load side also. For a better voltage sensor you can use a 0.1uF ceramic capacitor across the R2 and R6.

Protection:
Protections are provided in the charge controller to protect against extreme and abnormal operating condition.

In commercial charge controller there are so many protections. But in my charge controller all are not included.

Following protections are used:

Protection to over current:
Two fuses F1 and F2 are used for over current protection. One is at the incoming side i.e after the solar panel and other is before the load. I forgot to show in the first schematics.

Over Voltage Protection:
During lightening and thundering over voltage occurs in the system. To protect the system a zener diode is used.
I used a 11V zener diode as my solar panel open circuit voltage is 10v. When the voltage is more than 11v, it will provide the ground path to the surges.

**PV panel reverse current:**

During night battery voltage is higher in compare to solar panel voltage (0 V). So power can be flow in the reverse direction. This avoided by using a diode (D1) after the solar panel.

**Auto Load Disconnection:**

To avoid deep discharging of battery, auto load disconnection scheme is implemented through the software. When the battery voltage is falls below a certain voltage (6.2V) load is automatically disconnected.

**Overcharge protection:**

Over charge protection is given to improve the battery life and prevent from overheating. When the battery voltage is greater than the fully charged voltage (7.2V), it is disconnected from the solar panel to avoid further charging. This is also implemented through the software.

**Display and Indication:**

**System status display on LCD:**

A 16X2 char LCD is used for monitoring solar panel voltage and battery voltage. It also show the % of charge.

When the battery condition is dead it will display: “BATTERY IS DEAD!!” and when battery is discharged it will display “BAT DISCHARGED…”

**LED indication on system condition:**

2 LEDS (RED, GREEN) are used for indicating the system condition. RED LED will glow when battery is discharged.
(< 6.2 V) GREEN LED will glow when fully charged (> =7.2 V) and will blink during charging.

When battery voltage is less than 6.2V but greater than 6V RED LED will glow indicating battery is discharged and GREEN LED will blink indicating battery is charging.

**LCD -> Arduino**

1. VSS -> Arduino GND
2. VDD -> Arduino +5v
3. VO -> Arduino GND pin + Resistor or Potentiometer
4. RS -> Arduino pin 12
5. RW -> Arduino pin 11
6. E -> Arduino pin 10
7. D0 -> Arduino - Not Connected
8. D1 -> Arduino - Not Connected
9. D2 -> Arduino - Not Connected
10. D3 -> Arduino - Not Connected
11. D4 -> Arduino pin 5
12. D5 -> Arduino pin 4
13. D6 -> Arduino pin 3
14. D7 -> Arduino pin 2
15. A -> Arduino Pin 13 + Resistor (Backlight power)
16. K -> Arduino GND (Backlight ground)

If you need more information about LCD interfacing click [here](#)
How the charge controller works:
Power is coming from the solar panel through the diode (D1). A zener diode (D2) is placed at the input terminal to suppress the over voltage. Capacitor C1 is used to remove any unwanted noise/spikes. Then the voltage divider (R1 and R2) is used to sense the solar panel voltage. The output from the voltage divider goes to Arduino analog pin A0.

The power coming from the solar panel can't go directly to battery until the Mosfet(Q1) is On. The switching of the mosfet is done by a PWM signal from Arduino pin-6. Transistor T1 and associated resistance R4 is used for driving the Mosfet(Q1). The resistor R3 is used as a pull up resistor for gate. When the Mosfet is On power goes to battery and charging process starts.

The second voltage divider circuit (R5 and R6) is used for sensing the battery voltage. The output of voltage divider goes to Arduino analog pin A1.

The second Mosfet Q2 is used to drive the load and second transistor T2 is used for driving the Mosfet. During night load will automatically turned on by turning the MosfetQ2 on and will removed when battery voltage is low or day time.

Fuses F1 and F2 are used for over current protection.

LED1 (RED) and LED2(GREEN) are hooked to digital pin 7 and 8 of the Arduino for indication. The resistor R7 and R8 are used for limiting the current goes to LEDs.
If you want to use a relay instead of MOSFET Q2 you can use. The schematics -2 is given for relay connection.

**How Software Works:**

At first the charge controller will check the solar panel voltage and compare it with battery voltage. If it is greater than the Arduino will starts sending pulse width modulation (PWM) signals to the mosfet(Q1) in order to charge the battery. When the solar panel voltage was below the battery voltage, this pwm signals will not send by Arduino.

Then next the micro controller will check the battery voltage, if the battery voltage was below 6.96 v volts then the battery will be charged in boost mode, that mean the battery will be charged with maximum amperage, this boost mode of charging will be done by sending pulse width modulation signals with 95% duty cycle.

When the battery voltage reaches above 6.96V the charging mode will turn as absorption mode from boost mode, this was done by changing the duty cycle from 95% to 10%. This absorption mode will keep the battery fully charged.

A pulse will be send to the load mosfet(Q2) to enable the load during night, if the battery was too low and reaches to 6.2v volts then in order to prevent deep discharge of the battery, the load will disconnected.
Solder the circuit

After testing the circuit on a bread board, solder everything on a perforated board.

* clean the copper part of board
* place the components suitably
* Hold it by using a vice
* solder one by one as per schematics
* trim the long legs of all components
* solder 6 wires for terminal connections.(2 for solar panel input,2 for battery and 2 for load)
Make the final product (CHARGE CONTROLLER)
HOLES FOR VENTILATION

FUSE

6 TERMINAL CONNECTOR
1. Mark the size of LCD by a marker or pencil

2. Cut it by a hobby knife. I used my Leatherman ES4 which was given by Instructables.

3. Make 2 holes by a drill just below the LCD for inserting 2 LEDs. I used my DREMEL 3000 for drilling.

4. Hold the 6 pin connector and mark the screw holes, drill on the marked portion.

5. Drill a big hole for mounting the fuse. Solder two wires for connection with the controller.

6. Drill few holes on the side wall of the project box for ventilation.

7. Place the LCD, LEDs and connector in their place. Then glue it by a hot glue gun. I used my DREMEL glue gun.

8. Put the ARDUINO and Controller inside the box side by side.

9. Put a 9V battery just side to the arduino for powering i.e. used SCOTCH mounting squares for holding.

10. Connect the jumper wires appropriately and test it

Note: put heat shrink or any other insulation tap at all the bare conductors.