**Lithium-ion Battery Charger**

In this report we are going to discuss the lithium-ion battery charger that we designed to charge a 115V-150AH Lithium-ion battery (typical home inverter-battery).

**Charging strategy for Lithium-ion battery**

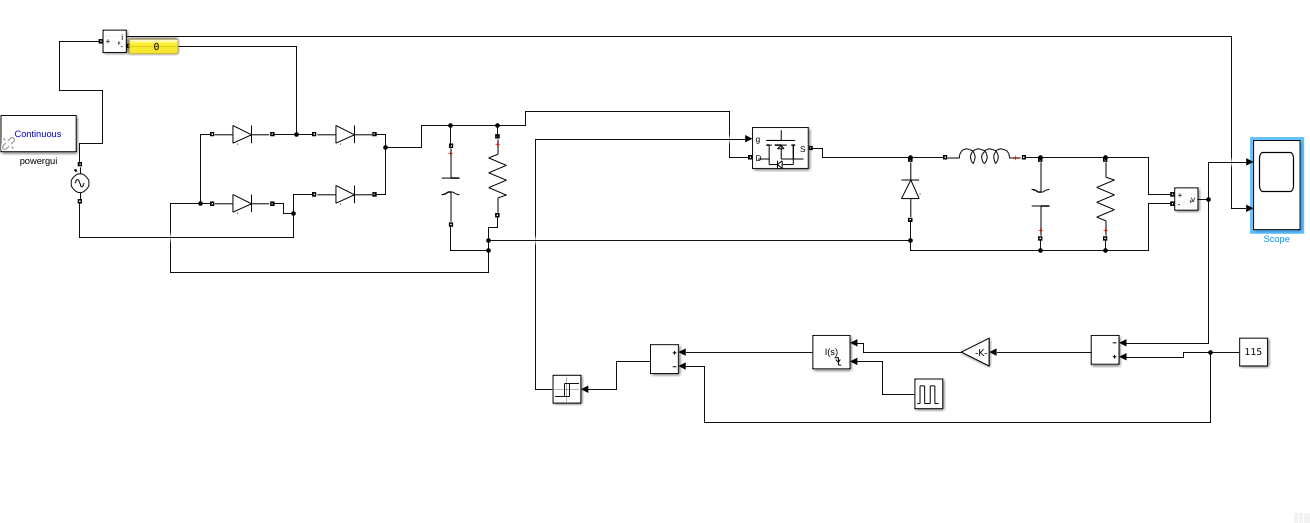
Lithium-ion battery has a high charge density around 100Wh/Kg-265Wh/Kg which is 5-6 times of a lead-acid battery. Therefore, while charging they have high power demand. While charging these batteries, voltage per cell must be kept in mind because these batteries are much sensitive and specific to voltages. Typical LIBs have a safe voltage per cell range of 3.3V to 4.2V and even few millivolt difference can have a great impact on battery life. Apart from battery voltage there is also limit of current passing through these batteries. While charging current passing through these batteries should be below 0.7C (1C is the rated capacity of battery) and while discharging it should not cross 2C.

Keeping safety in mind, our charger will be providing a voltage range of 89.1V to 115V and maximum charging current of 30A.

**AC to DC converter**

We need to convert 220v (rms) ac voltage to 115v dc voltage.

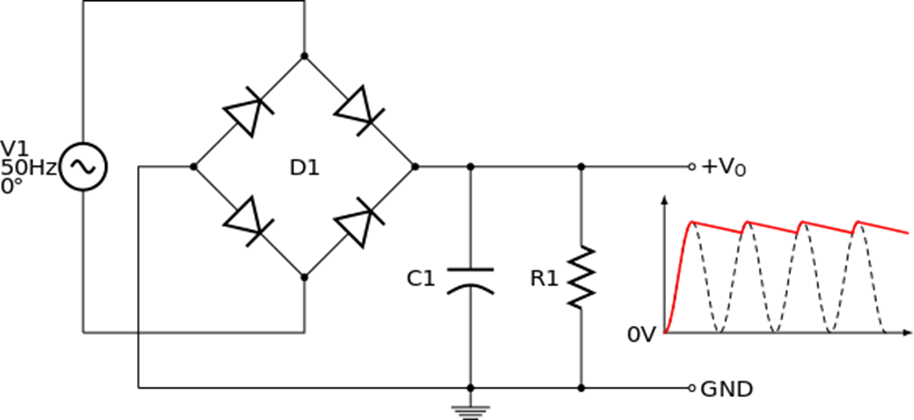
* In order to achieve that we need below mentioned circuits.
* Converting 230V AC into DC using Rectifier circuit.
* The above circuits output is given to Buck converter circuit in order to step down the voltage to 115V and also to decrease the ripple to get perfect DC.
* To maintain stable dc voltage at the output independent of load a closed loop buck converter has to be designed

This is our final circuit

FINAL CIRCUIT

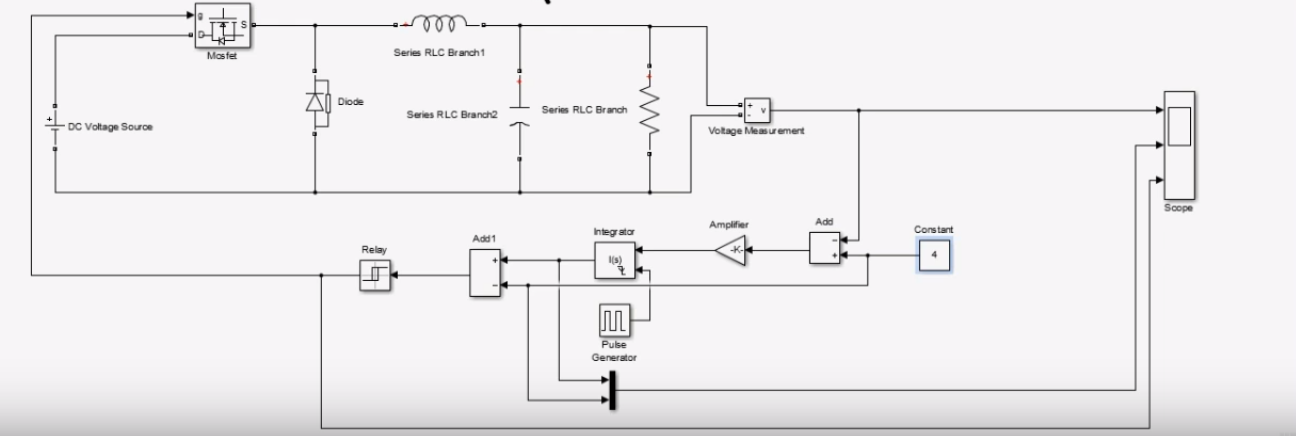
**Rectifier circuit**

* This is AC to DC convertor circuit containing diodes and capacitors.
* There are 4 diodes arranged so that these allow current in both directions through them but the output current is only in 1 direction.
* This is called full wave rectification.
* This out voltage is given to capacitor as input and the capacitor constantly charges and discharges.
* The voltage across the capacitor is dc but there are ripples present in it.
* We can calculate the value of capacitor so that ripples can be minimized.
* This output across the capacitor is given to the Buck convertor in order to step down the voltage and produce perfect DC.



**Buck converter circuit with closed loop**

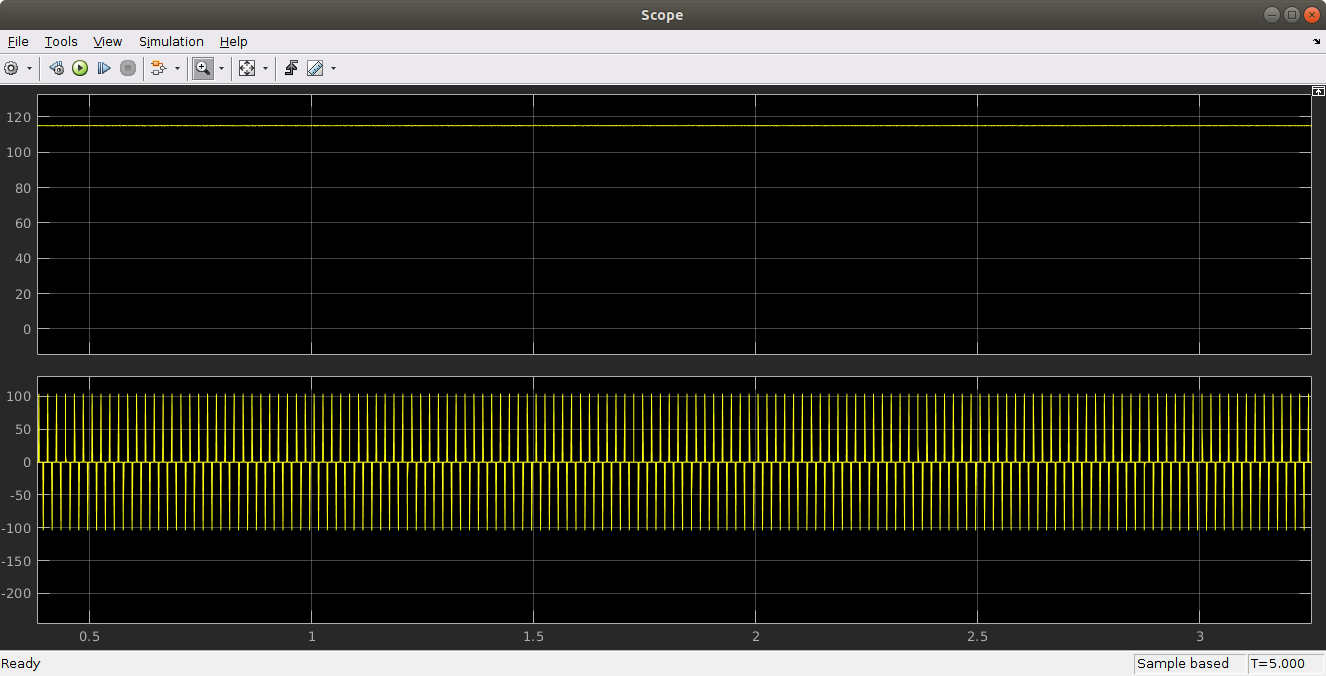
* The Buck converter circuit is used for step down the voltage without any power loss.
* We should reduce the ripple voltage in order to produce constant dc voltage
* We can control the ripple voltage at the output by changing the respective values accordingly.
* Ripple voltage = Vin\*D\*(1-D)/8\*f^2\*LC
* Here D and f are duty cycle and frequency of pulse which controls switch
* We are using closed loop buck converter to produce Stable dc voltage



* Rippled dc voltage given to buck converter as an input to step down to 115v and make it stable dc voltage.
* In buck converter we need a switch which keeps on and off repeatedly here MOSFET is used for that purpose
* When switch is on, the storing devices like inductor and capacitor stores the energy. When the switch is off, connection with the source is lost and the energy stored in the devices discharges across the load.
* Due to this constant charge and discharge there will be ripples at the load voltage. Which is to be minimized to least by setting up the circuit accordingly.
* In the feedback part,
* The output voltage is compared with reference 115v and the error is amplified through an amplifier
* This amplified error is sent to the (PID controller used as integrator) integrator which is tuned 4 times the frequency of pulse given to it (i.e, Ki = 4 \*f; Kp=0; Kd=0; in the PID controller, f is frequency of pulse).
* The frequency of the pulse given to the PID controller also control the ripples at the output, As frequency increases ripples decreases.
* This integrator gives output a pulse which controls the relay, the relay is connected to MOSFET, which is now used as switch in the Buck converter. Due to the small pulse of voltage given to the gate of the MOSFET it works as the switch.

PRACTICAL ISSUES

* The load at the output of buck converter should be high in order decrease the dissipation of power
* The power =V\*2/R, as the voltage across the load is constant, power is inversely proportional to resistance.
* The inductor, capacitor and resistor of buck converter has to be selected so that we can reduce ripples to 0.5v.
* While charging the actual circuit we have to change the output voltage of this circuit according to the SOC of the battery. In order to achieve that we should change the constant voltage present in the closed loop according to the SOC (Since output voltage will be equal to the constant voltage present in the closed loop). We can achieve this by using any microcontroller which control the constant voltage accordingly.
* Output voltage and input current are shown in the below graph.

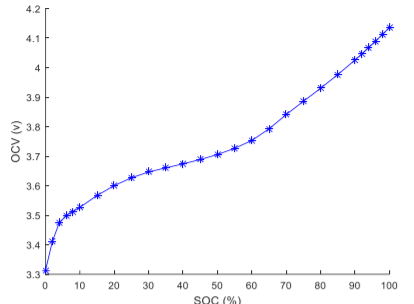


**Estimating State of Charge of Battery**

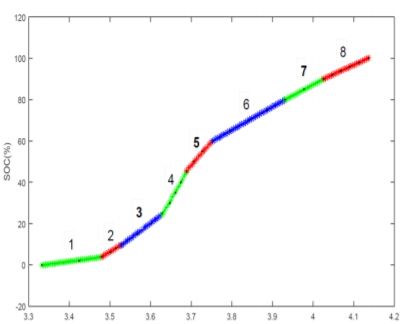
To estimate State of Charge of battery, we will be using following methods

* Coulomb Counting method
* SOC-OCV characteristic Graph

Coulomb counting method is straightforward. We will calculate the amount of current passing through the battery and integrate it with respect to time. This method is very efficient to get the difference in SOC at different time but has a drawback that it doesn’t actually estimates the charge present in the battery rather how much charge has flown into or out of battery. So to determine the present state of charge of battery we will use SOC-OCV characteristic graph. This graph actually is an experimental plot of SOC over different Open Circuit Voltage(OCV).



This graph cannot tell the values at all the point as it has been sampled with discrete point so we have to approximate it with linear curve fitting technique.

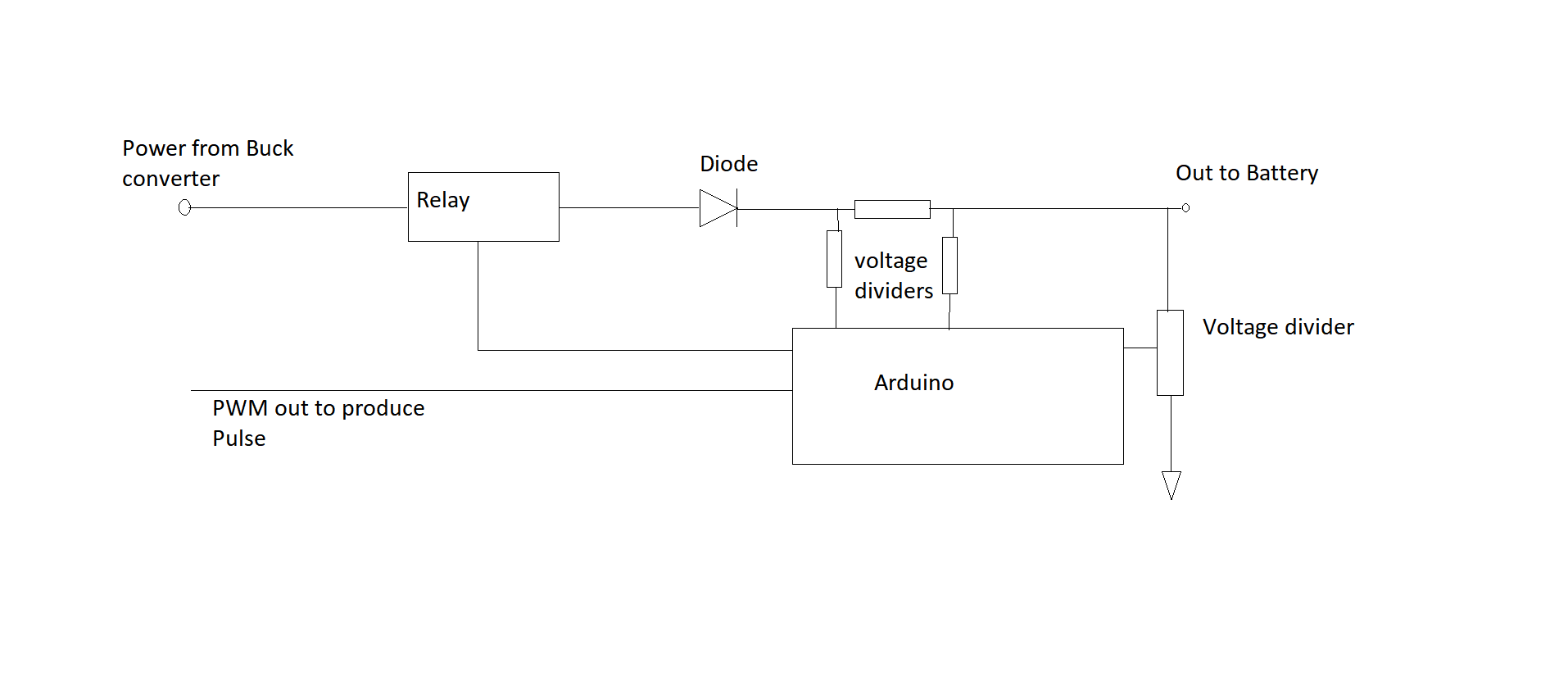
We will divide this graph into eight different pieces and linearize them so that we can implement it in future will writing code for it. Here is the graph of linearized version.

Using this SOC-OCV curve we will first determine the present SOC though this technique is not that accurate and have some error associated with it. But we can be accurate after this has been done by using coulomb counting method which will be quiet accurate and thus estimating State of Charge.

**Circuit and model for Charging**

To charge the battery we will keep voltage at constant value and that is 115 volts os that our battery can take the requeired current to charge at safe volatge but if the current level rises more than a danger level i.e. 50A circuit will disconnect the power supply using an electrical relay.

Circuit contains three potential dividers to map the voltage reading from range 0-120V to range 0-5V so that we can feed it safely to our arduino micro-controller. It also contain a current sensing resistor which will is there to convert current to voltage so that we can measure it.



**Arduino code**

This code include chraging techniques and SOC estimation technique. This charging circiut includes safty features like over charging, detects over discharging, over current through battery.

Below is the arduino code.

// declaring ports

int pulse = 3;

int ocv = 0; //Analog pin 0 to read battery voltage

int voltage1 = 1; // Analog pin 1 to read +resistor voltage

int voltage2 = 2; // Analog pin 1 to read -resistor voltage

int relay = 4; // relay switch to cut suply

float inisoc; //initial curr\_soc

float curr\_soc; //current curr\_soc

float soc = 0; //

boolean v = true;

boolean c = true;

const float battery\_capacity = 150;

const int delay\_value = 3000;

const float resistance = 100; // we can change it as per requirment

const float no\_cell = 12/4;

int counter = 0;

float adc\_factor = 0.004702;

void setup() {

Serial.begin(9600);

// TCCR2B = TCCR2B & B11111000 | B00000001; // for PWM frequency of 31372.55 Hz

pinMode(ocv, INPUT);

pinMode(voltage1, INPUT);

pinMode(voltage2, INPUT);

pinMode(relay, OUTPUT);

pinMode(pulse, OUTPUT);

}

void loop() {

// analogWrite(pulse, 77);

//----------------------------------------------------------------------------

int vo\_dif\_dumm = (analogRead(voltage1) - analogRead(voltage2));

float vo\_dif = vo\_dif\_dumm\*adc\_factor; //adc

//----------------------------------------------------------------------------

int x = analogRead(ocv);

float voltage = x\*adc\_factor\*3.0; // multiplied by potential divider factor adc

//----------------------------------------------------------------------------

float voltage\_per\_cell = voltage/no\_cell;

float current = vo\_dif/resistance;

//----------------------------------------------------------------------------

Serial.print("voltage difference ");

Serial.println(vo\_dif);

Serial.print("voltage difference per cell ");

Serial.println(voltage\_per\_cell);

Serial.print("batteryVolatge ");

Serial.println(voltage);

Serial.print("current ");

Serial.println(current);

Serial.print("curr\_soc ");

Serial.println(int(curr\_soc));

//----------------------------------------------------------------------------

if (counter < 1){

inisoc = table\_soc(voltage\_per\_cell);

Serial.print("initial curr\_soc ");

Serial.println(inisoc);

soc = inisoc;

}

//----------------------------------------------------------------------------

curr\_soc = coulomb\_count(current);

c = over\_curr\_det(current);

if (counter == 60){ // at every 3 minute it will determine OCV to avoid over chargning

v = over\_vol\_det(voltage\_per\_cell);

counter = 1;

}

//----------------------------------------------------------------------------

boolean Switch = c&&v;

relay\_controller(Switch);

delay(delay\_value);

counter++;

//----------------------------------------------------------------------------

}

//----------------------------------------------------------------------------

// formula for calculating curr\_soc

float curr\_soc\_calculator(float VPC, float a, float b){

float soc\_cal = VPC\*a - b;

return soc\_cal;

}

//----------------------------------------------------------------------------

// table for calculating curr\_soc

float table\_soc(float voltage\_per\_cell){

if((3.452>voltage\_per\_cell)&&(voltage\_per\_cell>=3.3)){

inisoc = curr\_soc\_calculator(voltage\_per\_cell, 26.55, 88.6);

}

else if((3.508>voltage\_per\_cell)&&(voltage\_per\_cell>=3.452)){

inisoc = curr\_soc\_calculator(voltage\_per\_cell, 125, 431.1);

}

else if((3.595>voltage\_per\_cell)&&(voltage\_per\_cell>=3.508)){

inisoc = curr\_soc\_calculator(voltage\_per\_cell, 149, 516.1);

}

else if((3.676>voltage\_per\_cell)&&(voltage\_per\_cell>=3.595)){

inisoc = curr\_soc\_calculator(voltage\_per\_cell, 344, 1225);

;}

else if((3.739>voltage\_per\_cell)&&(voltage\_per\_cell>=3.676)){

inisoc = curr\_soc\_calculator(voltage\_per\_cell, 229.5, 800.9);

}

else if((3.967>voltage\_per\_cell)&&(voltage\_per\_cell>=3.739)){

inisoc = curr\_soc\_calculator(voltage\_per\_cell, 111.9, 359.9);

}

else if((4.039>voltage\_per\_cell)&&(voltage\_per\_cell>=3.967)){

inisoc = curr\_soc\_calculator(voltage\_per\_cell, 104.8, 332);

}

else if((4.09>voltage\_per\_cell)&&(voltage\_per\_cell>=4.039)){

inisoc = curr\_soc\_calculator(voltage\_per\_cell, 91.61, 274.7);

}

else if (voltage\_per\_cell<3.3){

inisoc = 0;

}

else{

inisoc = 100;

}

return inisoc; // check it

}

//----------------------------------------------------------------------------

// implementing coulomb counting method

float coulomb\_count(float curr){

soc = soc + (curr\*delay\_value)\*(0.001/battery\_capacity);

return soc;

}

//----------------------------------------------------------------------------

// controls relay

int relay\_controller(boolean Switch){

digitalWrite(relay, Switch);

}

//----------------------------------------------------------------------------

// protection form over current

boolean over\_curr\_det(int current){

boolean current\_a ;

if (current > 20)

current\_a = false;

else

current\_a = true;

return current\_a;

}

//----------------------------------------------------------------------------

//protection from over voltage and over discharging

boolean over\_vol\_det(float voltage\_per\_cell){

boolean voltage\_a;

relay\_controller(0);

delay(1000);

float voltage = analogRead(ocv)\*adc\_factor\*3; //adc

float voltage\_per\_cell\_0 = voltage/no\_cell;

float check\_soc = table\_soc(voltage\_per\_cell);

Serial.print("paused \n voltage\_per\_cell\_0");

Serial.println(voltage\_per\_cell\_0);

Serial.print("soc-ocv graph SOC ");

Serial.println(int(check\_soc));

if ((3.5 < voltage\_per\_cell\_0) && (voltage\_per\_cell\_0 < 4.1)){

voltage\_a = true;

return voltage\_a;}

else{

voltage\_a = false;

return voltage\_a;

}

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_