

Arduino Controlled Lithium-ion Battery Charger

Ning Tian

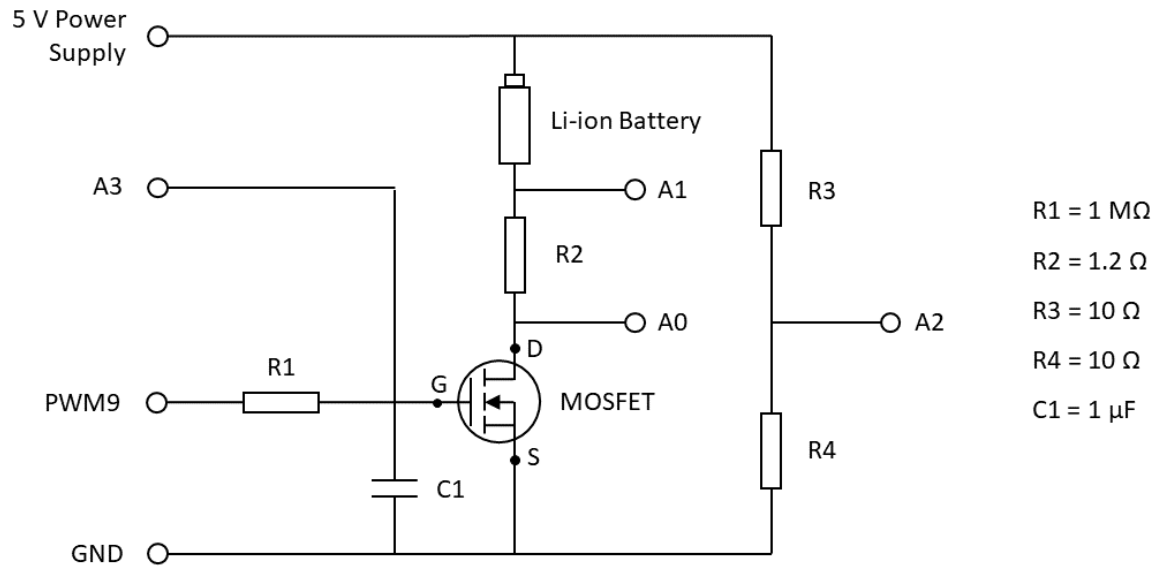


Figure 1. Charging circuit.

We start from the left-bottom part of Figure 1. The resistor R1 and capacitor C1 together work as a low-pass filter to smooth PWM signal produced by PWM9 (for more information about how a low-pass filter works on PWM signal, please see [1]). The MOSFET here works as an amplifier (here we use IRF510 MOSFET). If the voltage between Gate (G) and Source (S) (denoted as G-S voltage) increases, the current through Drain (D) and S will increase. Note that this happens only when the G-S voltage is above a cut-off value decided by the MOSFET (interested readers can refer to n-channel eMOSFET in [2]). From above, we can find that the current through the Li-ion battery can be tuned by changing the G-S voltage, which can be done by changing the duty cycle of the PWM signal by PWM9 [1]. To be brief, we can control the charging current by changing the duty cycle (for the information about how Arduino changes the duty cycle, please see [3]).

The above explains how we can control charging current. In order to do constant-current (CC) charging, we can simply formulate it as a feedback control problem as follows:

duty cycle = duty cycle + current difference,

where current difference = target current – measured current.

Similarly, constant-voltage (CV) charging can be formulated as

duty cycle = duty cycle + voltage difference,

where voltage difference = upper-limit voltage – measured voltage.

To do CCCV charging, we can combine above CC and CV together. If the measured voltage is below the upper-limit voltage, we apply CC, otherwise CV is imposed. The schematic diagram of the battery charger and the Arduino codes are attached below in the Appendix.

Acknowledgement

This charger is developed on the basis of the project in [4].

Appendix

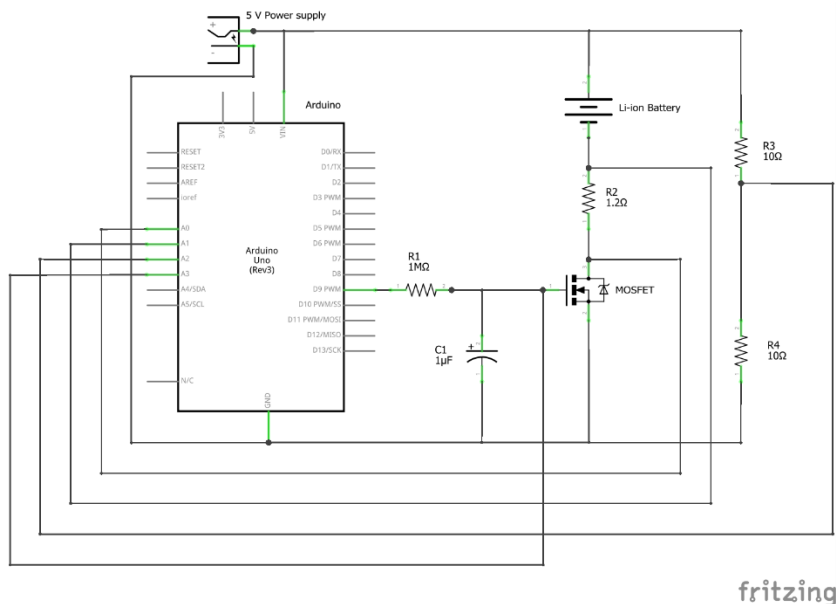
A1. Some further explanation about the circuit design

In the circuit, R2 is to help measure the current through the battery, which can be obtained from the voltage difference by A0 and A1. Despite of that, R2 introduces resistance to the circuit, so we had better choose an R2 with small resistance (like $1\ \Omega$), otherwise the charging current can be too small (e.g., less than 100 mA). Based on the above circuit, we set the CC to be 300 mA.

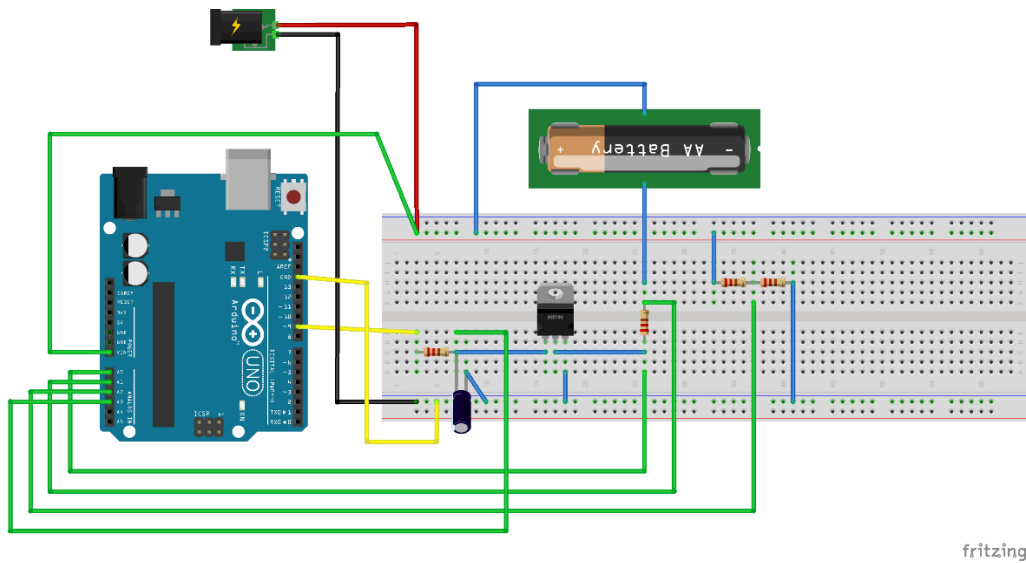
R3 and R4 work as a voltage divider, and we can thus interpolate the voltage of the power supply from the voltage across R4. In this circuit, we use a power supply of 5 V. But during charging, you will find the voltage delivered by the supply is not constant at 5 V and sometime can even go beyond it, e.g., 5.1 V. Therefore, it is necessary to measure the voltage of the power supply. As analog input channels like A0-A5 cannot read voltage over 5 V, we introduce the voltage divider and read the voltage across A3 to help us infer the supply's total voltage. Here, for R3 and R4, their values need to be carefully selected. If they are too large like $1\ \text{M}\Omega$, the voltage divider can not work properly [5]. If they are too small like $1\ \Omega$, the current through them can be very large and gets the resistors burned.

The analog input A3 reads the voltage across C1, which is optional here. The reason why it is included is to help us better understand the circuit. By looking at the voltage read by A3, we can find that, with the voltage increasing, the current passing through the battery will also increase, which is consistent with our previous analysis.

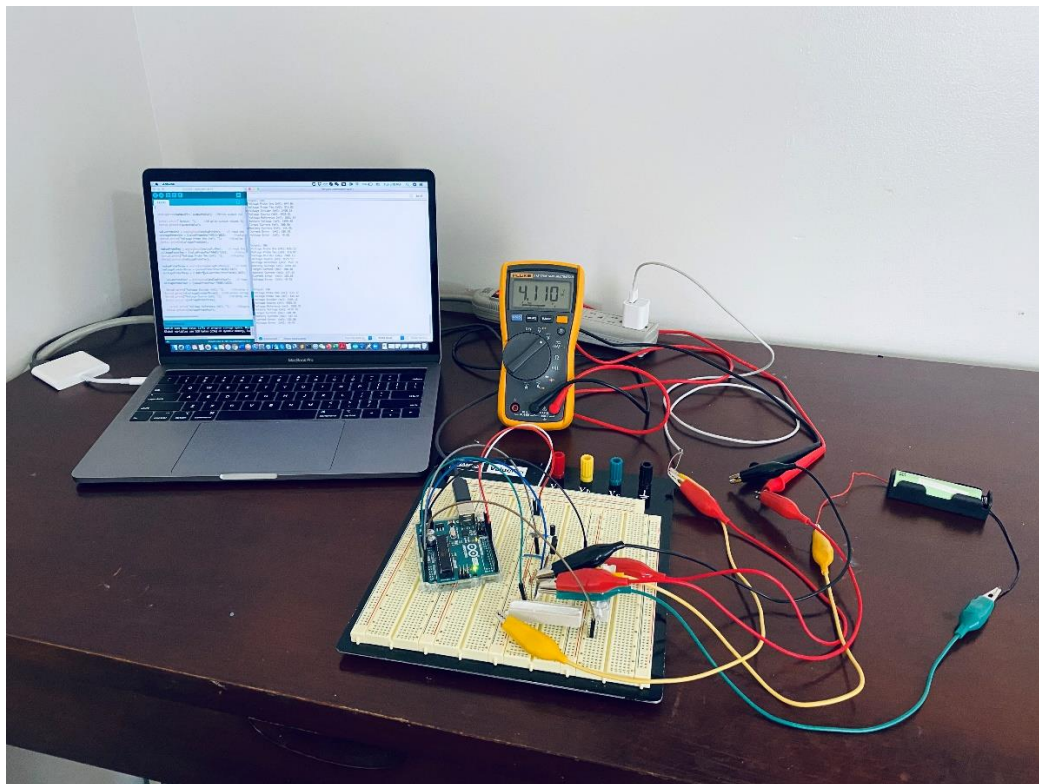
A2. Schematic diagram of the battery charger



A3. Virtual connection of the battery charger



A4. The finished battery charger (Panasonic NCR18650B Li-ion battery is used here)



A5. Codes to run in Arduino

// this code is to do CCCV charging for a lithium-ion battery

int batteryCapacity = 3000; //capacity rating of battery in mAh

float resistance = 1.2; //measured resistance of the resistor

```
int targetVoltage = 4100; //target battery voltage (in mV) in CV mode (usually targetVoltage is the same
with cutoffVoltage, for safety, we make them different here)
```

```
int cutoffVoltage = 4200; //maximum battery voltage (in mV) that should not be exceeded
```

```
int outputPin = 9; // Output signal wire connected to digital pin 9
```

```
int outputValue = 150; //value of PWM output signal
```

```
int analogPinOne = 0; //first voltage probe connected to analog pin A0
```

```
float valueProbeOne = 0; //variable to store the value of analogPinOne
```

```
float voltageProbeOne = 0; //calculated voltage at analogPinOne
```

```
int analogPinTwo = 1; //second voltage probe connected to analog pin A1
```

```
float valueProbeTwo = 0; //variable to store the value of analogPinTwo
```

```
float voltageProbeTwo = 0; //calculated voltage at analogPinTwo
```

```
int analogPinThree = 2; //third voltage probe connected to analog pin A2
```

```
float valueProbeThree = 0; //variable to store the value of analogPinThree
```

```
float voltageDividerThree = 0; //calculated voltage at analogPinThree, i.e., the voltage of R4
```

```
float voltageProbeThree = 0; //calculated voltage of the voltage supply
```

```
int analogPinFour = 3; //fourth voltage probe connected to analog pin A3
```

```
float valueProbeFour = 0; //variable to store the value of analogPinFour
```

```
float voltageProbeFour = 0; //calculated voltage of capacitor C1
```

```
float voltageDifference = 0; //difference in voltage between analogPinOne and analogPinTwo, to
calaculte current
```

```
float batteryVoltage = 0; //calculated voltage of battery
```

```
float current = 0; //calculated current through the load (in mA)
```

```
float targetCurrent = batteryCapacity / 10;  //target output current (in mA) set at C/10 or 1/10 of the
battery capacity per hour
```

```
float currentError = 0;  //difference between target current and actual current (in mA)
```

```
float voltageError = 0;  //difference between target voltage and actual voltage (in mA)
```

```
void setup()
```

```
{
  Serial.begin(9600);  // setup serial
  pinMode(outputPin, OUTPUT);  // sets the pin as output
}
```

```
void loop()
```

```
{
  analogWrite(outputPin, outputValue);  //Write output value to output pin
```

```
  Serial.print("Output: ");  //display output values for monitoring with a computer
```

```
  Serial.println(outputValue);
```

```
  valueProbeOne = analogRead(analogPinOne);  // read the input value at probe one
```

```
  voltageProbeOne = (valueProbeOne*4991)/1023;  //calculate voltage at probe one in milliVolts
(usually it is 5000 instead of 4991, but by calibration we find 4991 is more accurate, 4969 and 4890 are
obtained for the same reason)
```

```
  Serial.print("Voltage Probe One (mV): ");  //display voltage at probe one
```

```
  Serial.println(voltageProbeOne);
```

```
  valueProbeTwo = analogRead(analogPinTwo);  // read the input value at probe two
```

```
  voltageProbeTwo = (valueProbeTwo*4969)/1023;  //calculate voltage at probe two in milliVolts
```

```
  Serial.print("Voltage Probe Two (mV): ");  //display voltage at probe two
```

```
  Serial.println(voltageProbeTwo);
```

```
valueProbeThree = analogRead(analogPinThree); // read the input value at probe three  
voltageDividerThree = (valueProbeThree*4890)/1023; // calculate voltage of R4 in milliVolts  
voltageProbeThree = 2.0445*(valueProbeThree*4890)/1023; // calculate voltage of voltage supply in  
milliVolts
```

```
valueProbeFour = analogRead(analogPinFour); // read the input value at probe four  
voltageProbeFour = (valueProbeFour*4890)/1023; // calculate voltage of C1 in milliVolts
```

```
Serial.print("Voltage Divider (mV): "); //display voltage of R4  
Serial.println(voltageDividerThree);  
Serial.print("Voltage Source (mV): "); //display voltage of voltage supply  
Serial.println(voltageProbeThree);
```

```
Serial.print("Voltage Reference (mV): "); //display voltage of C1, by obsersing this value and the  
current, you can find current increases with this value increasing  
Serial.println(voltageProbeFour);
```

```
batteryVoltage = voltageProbeThree - voltageProbeTwo; //calculate battery voltage  
Serial.print("Battery Voltage (mV): "); //display battery voltage  
Serial.println(batteryVoltage);
```

```
current = (voltageProbeTwo - voltageProbeOne) / resistance; //calculate charge current  
Serial.print("Target Current (mA): "); //display target current  
Serial.println(targetCurrent);  
Serial.print("Battery Current (mA): "); //display actual current  
Serial.println(current);
```

```
currentError = targetCurrent - current; //difference between target current and measured current  
Serial.print("Current Error (mA): "); //display current error
```

```
Serial.println(currentError);
```

```
    voltageError = targetVoltage - batteryVoltage;    //difference between target current and measured current
```

```
    Serial.print("Voltage Error (mV): ");    //display current error
```

```
    Serial.println(voltageError);
```

```
Serial.println();    //extra spaces to make debugging data easier to read
```

```
Serial.println();
```

```
if(batteryVoltage <= targetVoltage)
```

```
{
```

```
    if(abs(currentError) > 10)    //if output error is large enough, adjust output
```

```
    {
```

```
        outputValue = outputValue + currentError / 30;
```

```
    if(outputValue < 1)    //output can never go below 0
```

```
    {
```

```
        outputValue = 0;
```

```
    }
```

```
    if(outputValue > 254)    //output can never go above 255
```

```
    {
```

```
        outputValue = 255;
```

```
    }
```

```
    analogWrite(outputPin, outputValue);    //write the new output value
```

```
}
```

```
}
```

```
if(batteryVoltage > targetVoltage)
```

```
{
```

```
    if(abs(voltageError) > 10)    //if output error is large enough, adjust output
```

```
    {
```

```
        outputValue = outputValue + voltageError / 30;
```

```
    if(outputValue < 1)    //output can never go below 0
```

```
    {
```

```
        outputValue = 0;
```

```
    }
```

```
    if(outputValue > 254)    //output can never go above 255
```

```
    {
```

```
        outputValue = 255;
```

```
    }
```

```
    analogWrite(outputPin, outputValue);    //write the new output value
```

```
}
```

```
}
```

```
if(batteryVoltage > cutoffVoltage)    //stop charging if the battery voltage exceeds the safety threshold
```

```
{
```

```
    outputValue = 165;
```

```
    Serial.print("Max Voltage Exceeded");
```

```
}
```



```
    delay(10000);    //delay 10 seconds before next iteration, in the circuit the time constant for the RC
circuit is 1 s, and 10 s is thus reasonable
}
```

Reference

- [1] <https://www.allaboutcircuits.com/technical-articles/low-pass-filter-a-pwm-signal-into-an-analog-voltage/>
- [2] <https://www.electronics-tutorials.ws/amplifier/mosfet-amplifier.html>
- [3] <https://www.arduino.cc/en/tutorial/PWM>
- [4] <https://www.allaboutcircuits.com/projects/create-an-arduino-controlled-battery-charger/>
- [5] <https://www.khanacademy.org/science/electrical-engineering/ee-circuit-analysis-topic/ee-resistor-circuits/a/ee-voltage-divider>