**Lithium-Ion Battery Charger:**

This is **Lithium-Ion battery charger** implemented on **Arduino**. Has more advanced features like:

* State of charge estimation.
* **EEPROM** logging.
* Command-Line interface.

It uses the **constant current constant voltage** **(CC-CV).**

The rationale behind this project was to upgrade the depleted battery pack and charger of an old cordless drill from **Nickel-Cadmium** **(Ni-Cd)** to **Lithium-Ion** **(Li-Ion)** technology.

**Warning: Lithium-ion batteries are dangerous devices. Overcharging, short circuiting, or misuse of lithium-ion batteries may result in fire and/or violent explosion. It is necessary to equip each lithium-ion battery with its own dedicated battery protection board (or battery management system also known as BMS).**

**1- Theory of Operation**

The following subsections cover the theoretical and mathematical aspects of charging a **Li-Ion battery**.

**1.1- CC-CV Charging**

**Li-Ion batteries** must be charged using the **Constant Current Constant Voltage** **(CC-CV)** charging method. This method consists of charging the battery at a constant current **Icharge** until a certain voltage threshold **Vmax=4.2Vcell** is reached, then gradually reducing the charging such that the constant cell voltage **Vmax** is not exceeded. Charging is terminated once the current reaches a certain minimum threshold **Ifull** of typically **50-150 mA**.

**Additional End of Charge** **(EOC)** standards have been implemented for safety reasons. These include time-based and capacity-based **EOC** detection. When the battery is connected, the charger measures the voltage at its terminals. The **SOC** value is used to calculate the remaining capacity **Cmax**and charging duration **Tmax**. Charging is terminated if any of these values ​​are reached.

**1.2- Control Loop**

The battery **“+”** terminal is connected to the positive power supply through a power **MOSFET** (field-effect transistor). The battery **“-”** terminal is connected to the power supply ground through a low-value **shunt resistor** **Rshunt**.

The charging current is regulated by **pulse width modulation** **(PWM)**, where the **MOSFET** is periodically turned on and off by the **Arduino** at a frequency of **31,250 kHz**. The charging current is controlled by gradually adjusting the **PWM** duty cycle which is the ratio between the **ON** and **OFF** duration of the **MOSFET**.

**V1** is the voltage measured at the **“+”** terminal of the battery and **V2** is the voltage measured at the **“-”** terminal of the battery. Both voltages are measured relative to the power supply ground and are used to calculate the **voltage** **V** across the battery pack and the charging **current I** as follows:

 $V\left(Volt\right)=V1-V2$

$I(Ampere)=\frac{V2}{R\_{shunt }}$

**Two** separate **ADC** channels on the **Arduino** are used for measuring the above voltages. The **Arduino** continuously monitors **V** and **I** and adjusts the **PWM** duty cycle in order to achieve the desired constant current or constant voltage regulation.

**1.3- State-of-Charge Estimation**

The **state of charge** **SOC** is estimated by reading the battery **voltage** **V** and comparing it to a series of values stored in a lookup table **L = (l0, l1, l2, l3, l4, l5, l6, l7, l8).** The threshold voltages are derived from the particular discharge curve shown below for the **LG 18650 HE4** cells used in this project. **(Source:**[**https://lygte-info.dk/review/batteries2012/LG%2018650%20HE4%202500mAh%20%28Yellow%29%20UK.html**](https://lygte-info.dk/review/batteries2012/LG%2018650%20HE4%202500mAh%20%28Yellow%29%20UK.html)**).**



**Figure 1: Statistic of Discharge, capacity (LG-18650-HE4-2500mAh)**

The red discharge curve corresponding to **0.2A** discharge current has been used, whereas the values of **L** were assigned such that:

* **Ɩ0= V~2.25Ah**
* **Ɩ1= V~2.00Ah**
* **Ɩ2= V~1.75Ah**

**.**

**.**

**.**

* **Ɩ8= V~0.25Ah**

**SOC** is calculated as follows:

* V< **Ɩ0: SOC= 0%**
* **Ɩ0<V< Ɩ1: SOC= 10%**
* **Ɩ1<V< Ɩ2: SOC= 20%**

**.**

**.**

**.**

* **Ɩ8<V: SOC= 90%**

The remaining capacity **Cmax** and charge duration **Tmax** are derived as follows:

$$C\_{max }(mAh)=C\_{full }×(100-SOC)×1.3$$

$T\_{max}=3600×\frac{C\_{full }}{I\_{charge }}$ **× (90 – SOC) + 45.6**

Where **C** full is the design capacity of the battery and **C** is the nominal charging current. Note that **Cmax** is increased by **30%** and **Tmax** is increased by **45 min** in order to account for resistance losses and inaccuracy of **SOC** estimation.

**1.4- Safety**

The charger implements several safety features. These include:

* **Undervoltage.**
* **Overvoltage.**
* **Short circuit.**
* **Open circuit detection**.

The typical voltage range where a **Li-Ion battery** can safely operate is between$V\_{min }= SI\left(2.5\right)V\_{cell }$ and $V\_{max }= SI\left(4.2\right)V\_{cell }$. Operating outside this range is likely to cause permanent damage to the **Li-Ion** cells and may even result in a catastrophic failure such as an explosion or fire.

In addition, the battery pack is protected by a battery protection board (or battery management system also known as **BMS**). The **BMS** measures the voltages of individual battery cells as well as the charging and discharging current flowing through the battery. The **BMS** uses a **solid-state switch** to disconnect the battery once the voltage or current values ​​become outside the specified limits.

For the most part, the **BMS** is completely transparent and does not interfere with the charging process, except for the case where the **BMS** disconnects the depleted battery in order to prevent **over-discharge**. In this case, the voltage of the depleted battery is still present across the **BMS** terminals through a high value resistor placed in series with the battery. This high value resistor causes a much lower voltage value to be measured at the charger terminals. Consequently, the charger must ignore the **Vmin** lower limit and start charging at a much lower value of as low as$V\_{start }= SI\left(0.5\right)V\_{cell }$**.**

When served with a depleted battery, the charger will start charging at a low safety current $I\_{safe }=\frac{I\_{charge }}{10}$ until the battery voltage reaches $V\_{safe }= SI\left(2.8\right)V\_{cell }$, after which full charging current **Icharge** will be applied. Once the voltage reaches this threshold, it is no longer allowed to drop below **Vmin**. A voltage lower than **Vmin** may cause an **“under voltage fault”** which may be caused by either a short circuit or open circuit of the battery.

Open circuit is also detected if the charging current stays equal to zero while the **PWM** duty cycle increases beyond a specific threshold. This condition would raise an **“open circuit error”***.*

Overvoltage is detected whenever the battery pack voltage momentarily exceeds$V\_{surge }= SI\left(4.25\right)V\_{cell }$. Exceeding this value would raise an **“overvolt error”.**

**1.5- Trickle Charging**

Once the **end-of-charge (EOC)** criteria has been met, the charger would **cut-off** the charging current and switch to an idle mode where it will continuously monitor the battery voltage. Once the voltage drops below a specific threshold of $V\_{trickel/start} = SI\left(4.10\right)V\_{cell }$, a new charging cycle will be initiated using the following parameters:

$$V\_{max }(V\_{cell })=V\_{trickel\\_max}=(4.15)V\_{cell }$$

$$C\_{max}(mAh) =C\_{full}×0.3+C$$

$$T\_{max}(s)=20×60+T$$

Where **Cfull** is the battery design capacity. **C** and **T** are the accumulated charge capacity and charge time since the battery has been connected, including the initial charge and all of the subsequent trickle charge cycles.

Given the above formulas, the trickle charge cycle uses a reduced **Vmax** and allows for charging up to a maximum of **3%** of the battery design capacity during a maximum duration of **20 minutes**.

**2- Hardware**

The following **sub-sections** describe the hardware design aspects of the **Li-Ion charger.**

**2.1- Mechanical Design**

We used **four LG 18650 HE4 Li-Ion cells** and a battery protection board (or **battery management system** also known as **BMS**). Modern **lithium-ion cells** use much less space.

**2.2- Battery Protection Board**

It is necessary to use a dedicated battery protection plate for each battery pack. This provides an extra layer of protection to prevent **over charging** or **over discharging** due to software or hardware malfunction.

in **figure 2** below, it shows the **4S/30A** **(4S means 4 cells in series)** battery protection board (or **BMS**).



**Figure 2: 4S Battery Protection Board**

In **figure 2** can see the wiring diagram for connection the **4 Li-Ion cells** with the **BMS**.

This particular **BMS** includes the cell balancer feature. If the voltage of one or more cells becomes higher than the rest of the pack, the **BMS** would actively discharge those cells to ensure that all the cells of the battery pack share the exact same voltage.

**3- Circuit Diagram**

In **figure 3** below, it shows the **Li-Ion charger circuit diagram**.



**Figure 3: Li-Ion charger circuit diagram**

In **figure 3** above schematic, the **19.5 V** of the power supply are stepped-down to **5 V** by the **7805-voltage regulator U1**. The **5 V** is used for powering the **Arduino board**.

The **Arduino Pro Mini** compatible board **U2** hosts an **AT-mega 328P** microcontroller running at **16 MHz** clock frequency and is used as the main processing unit for the device.

The **Lithium-Ion battery** is connected across the **B+** and **B-** terminals. The battery charging current is regulated by switching **P-Channel MOSFET (field-effect transistor) Q1** via **pulse-width modulation (PWM).**

The **PWM-enabled digital output pin 9** on the **Arduino** generates a **PWM** signal which drives the gate of the **MOSFET Q1** through the **NPN transistor Q2**. The **voltage divider** formed by **R1**and **R2** ensures that the **gate-source voltage** of the **MOSFET** stays within the specified limits.

A **current-sensing** shunt resistor connects the **B-** terminal with **ground**. It consists of **two 1 Ω / 3 W resistors R8** and **R9** connected in **parallel**. This results in a **total resistance** of **0.5 Ω**. At a **charging current** of$ I\_{charge}=2A$, the voltage across the shunt will be exactly **1 V**; which is slightly below the **1.1 V** internal voltage reference of the **Arduino** thus corresponds to the full range of the **Arduino**’s **analog-to-digital converter (ADC).**

The **analog pin A0** on the **Arduino** is used for measuring the **voltage V1**between **B+** and **0 V**. and the **Analog pin A1** is used for measuring **V2** between **B-** and **0V**.

**B+** is connected to **pin A0** through a **voltage divider** consisting of **R4** and **R7**, the ratio has been chosen such that the **maximum battery** pack voltage of **16.8 V** would result in slightly less than the **Arduino**’s internal reference voltage of **1.1 V** at **A0**. Please note that the value of **R4** needs to be adapted to the number of cells in use. For example, using a **1 cell** setup would require reducing the value of **R4** to **39 KΩ**.

**B-** is connected to **A1** through a **current-limiting resistor R5**; A **voltage divider** is not required for measuring **V2** as its value stays below the **Arduino**’s **ADC** internal reference voltage.

**Two 100 nF** **capacitors C4** and **C5** are used for blocking the **high-frequency** noise caused by the **PWM** from reaching the **analog inputs**, an essential measure for smooth ADC readings.

The **Diode D1** protects the **7805** regulator from a reverse power supply polarity. The **diode D2** protects the battery from a reverse polarity. it also prevents the battery from feeding power back into the **Arduino** in case the main power supply has been disconnected.

A **LED** **indicator D3** and its dropper **resistor R6** are connected to **Arduino**’s **digital pin13.**

**Important: The battery terminals in the circuit diagram are labeled as B+ and B-. It is important to connect these terminals to the P+ and P- terminals of the Battery Management System (BMS) depicted in the figure 3. The BMS has its own set of B+ and B- terminals that must be connected directly to the battery terminals. It is crucial to avoid connecting the charger’s B+ and B- terminals to the B+ and B- terminals of the BMS, as this would bypass the BMS and prevent it from safeguarding the battery against overcharging.**

**3.1- Different Number of Cells**

The following values for **R2**, **R4** and the power supply voltage need to be chosen in order to charge different numbers of Cells:

|  |  |  |  |
| --- | --- | --- | --- |
| Ncells | Power Supply | R2 | R4 |
| 1 | 5V-6V | 220Ω | 39KΩ |
| 2 | 10V-15V | 100Ω | 82KΩ |
| 3 | 14V-20V | 220Ω | 120KΩ |
| 4 | 18.5V-20V | 220Ω | 180KΩ |

**Table 1: Ncell with Power Supply, R2, R4**

When charging **1 cell**, the following circuit modifications must be performed:

* Remove the **voltage regulator U1** and **capacitor C3** and power the **Arduino** directly from the output of **D1.**
* Replace **Q1** with a **IRLML2244 MOSFET**.
* Increase **R1** to **10 KΩ.**
* Remove **Q2** and **R3.**
* Connect **R2** directly to **Arduino digital pin 9**.
* Modify the code in **li-charger.ino** to invert the **PWM** signal by subtracting the **PWM** duty cycle from **255** within all instances of **analogWrite ()** using one of the following statements in the figure below:



 **Figure 4: code of analogWrite () in li-charger.ino**

**4- PCB Layout**

All components are of the **punch-hole** **type** and are mounted on a **PCB board**. The **Figure 4** shows the **PCB layout** of a **Li-Ion charger**.



**Figure 5:PCB layout of the Li-Ion charger**

The **MOSFET Q1 (TO-220 device in the top right corner)** and large **green-colored** **shunt resistors** will get pretty hot so adequate ventilation needs to be assured. The following measures has been taken to avoid overheating:

* The **shunt resistors** **R8** and **R9** are raised by around **5mm** from the **PCB** in order to assure adequate cooling.
* A series of holes has been drilled in the bottom of the enclosure in order to allow for a better air flow.
* The charging current **Icharge** has been limited to **1.5 A**.

The **electrolytic capacitor C1** towards the top center of the board is in a **sub-optimal** position due to its location between two hot components, the **7805 regulator** and the **MOSFET**. High temperatures reduce the lifespan of **electrolytic capacitors** thus the must be kept away from heat sources.

The **pin header** located at the top right corner is used for connecting all the external wires. Following is the pinout assuming that **pin 1** is at the top right corner and **pin 10** is towards the middle of the board.

|  |  |
| --- | --- |
| Pin | Purpose |
| 1 \* | **LED +** |
| 2 \* | **LED -** |
| 3,4 ⸸ | **Power supply +** |
| 5,6 ⸸ | **Battery +** |
| 7,8 ⸸ | **Power supply -** |
| 9,10 ⸸ | **Battery -** |

**Table 2: All pinhead work**

\* The LED dropper resistor is located on a separate PCB together with the LED itself.

⸸ Two pins are connected in parallel in order to increase their current capacity.

**5- User Interface**

The following sections describe the user interface of the **Lithium-Ion charger**. It consists of a **LED indicator** and **a Command-Line Interface (CLI).**

**5.1- LED Indicator**

The charger status is displayed by turning on or flashing a single **LED** as shown

in **Table 3.**

|  |  |
| --- | --- |
| Blinking Pattern | Meaning |
| On for half a second every 2 seconds | Ready, waiting for the battery to be connected |
| Solid on | Battery charging |
| On For 0.1 second every 2 seconds | Battery fully charged |
| Blinking fast (0.4 s period) | Error |
| Blinking very fast (0.2 s period) | Calibrationmode |

**Table 3: The meaning of what is displayed via the LED**

**5.2- Command-Line Interface**

This **Lithium-Ion battery charger** features a **Command-Line Interface (CLI)** that can be accessed via the **Arduino**’s **RS232 serial port**. The easiest way to connect to the **CLI** is to open the serial monitor of the **Arduino IDE** while connected to the charger using a **FTDI USB** to **Serial converter**. Please ensure that the **Baud rate** is set to **115200**.

Once up and running, the charger will display a welcome message on the serial monitor, show the current firmware version and present with the list of available commands shown in the following list.

Some of these **CLI commands** need to be provided with arguments. Thus, one needs to enter the command followed by **one** or **two** arguments separated by a **white space**. SSS

|  |  |
| --- | --- |
| Command | Description |
| h | **Help- show the list of available commands** |
| . | **Display the real-time parameters, including the charge duration T, charge capacity C, battery voltage V, charging current I, maximum charge duration Tmax, maximum charge capacity Cmax, maximum charging voltage Vmax, maximum charging current Imax, PWM duty cycle, voltages V1, V2 and their raw ADC values V1, raw and V2, raw** |
| r | **Show the list of calibration constants that are stored within EEPROM** |
| t | **Show the contents of the trace circular buffer** |
| ncells<integer> | **Set the total number of cells within the battery pack Ncells, the value provided as an argument will be validated and stored in EEPROM** |
| cfull<integer> | **Set the battery design capacity Cfull in mAh, the value provided as an argument will be validated and stored in EEPROM** |
| ichrg<integer> | **Set the battery charging current Icharge, the value provided as an argument will be validated and stored in EEPROM** |
| ifull<integer> | **Set the end-of-charge current Ifull in mA, the value provided as an argument will be validated and stored in EEPROM** |
| iut<index><voltage> | **Configure the state-of-charge lookup table (LUT). This command takes and index** $i=0,1,2,…,8$ **and the reference voltage li in mV as arguments. Each time this command is called, a new reference voltage value li is populated into the LUT and stored into EEPROM, more on this in the following section** |
| rshunt<integer> | **Set the shunt resistor value Rshunt in mΩ, the value provided as an argument will be validated and stored in EEPROM** |
| cal<start|stop|v1|v2><mv> | **The voltage calibration mode is entered by calling cal start and exited by calling cal stop.V1 is calibrated using cal v1 <mv>.V2 is calibrated using cal v2 <mv>.<mv> is the measured voltage level in millivolts. Please refer to the next section for more details about the calibration procedure.** |

**Table 4: command followed by one or two arguments separated**

**5.3- Calibration Procedure**

This section provides an example on how to perform the **first-time** calibration of the **Lithium-Ion battery charger** using the **CLI** over the serial monitor.

The calibration values are stored into the **Arduino**’s **electrically erasable programmable read-only memory (EEPROM).** A **cyclic redundancy check (CRC)** checksum is appended to the configuration parameters set and stored into **EEPROM** as well. All configuration parameters are validated and **out-of-range values** are automatically replaced with the corresponding failsafe values.

The current example assumes a system consisting of **Ncells= 4** connected in series having a design capacity of **Cmax= 2500mAh** charged using a current of **Icharge = 1500mA**.

**Important:**

* **Do not connect the battery** during the calibration procedure unless instructed otherwise.
* Ensure that the voltage calibration procedure has been properly executed and verified prior to attempting to connect a **Lithium-Ion battery**. It is mandatory to connect a good quality battery protection board between the charger and battery. Failing to observe these precautions may lead to permanent damage or even explosion of the **Lithium-Ion cells**.

**5.3.1- Initial ranking**

A first step, the initial configuration parameters need to be loaded into **EEPROM** by executing the command sequence as shown in **figure 6** below:

****

**Figure 6: command sequence**

A confirmation message will be printed on the serial monitor following each value entry.

**5.3.2- Voltage calibration**

 Having performed the above initial step, please proceed for calibrating the **ADC** readings for the voltages **V1, V2** as shown below:

1. Enter the **Cal start** command in the serial monitor, this will activate calibration mode. The **Start Calibration** message should appear on the serial monitor.
2. Connect a constant voltage source of approximately **750 mV** between the **B-** terminal and the power supply **ground**, and measure its exact value using a digital multimeter. Note that **750 mV** corresponds to **1.5 A** flowing through the **shunt resistors R8** and **R9**.
3. Enter the command **cal v2 <value>** into the serial monitor, where **<value>** is the value in **mV** of the voltage measured in the previous step. The value of the calibration constant **V2,cal**will be displayed upon the successful calibration of **V2**. If the calibration fails, the message Out of range will appear in the serial monitor.
4. Connect a **constant voltage source** of approximately **16800 mV (4200 mV per cell)** between the **B+** terminal and the **power supply ground**, and measure its exact voltage using a **digital multimeter**.
5. Enter the command **cal v1 <value>** into the serial monitor, where **<value>** is the value in **mV** of the voltage measured in the previous step. The value of the calibration constant **V1,cal** will be displayed upon the successful calibration of **V1**. If the calibration fails, the message Out of range will appear in the serial monitor.
6. Verify the voltage calibration by applying a known voltage to each of **B+** and **B- (relative to 0 V)**, then enter the **[.](dot)** command and check the displayed values for **V1** and **V2** which must match the measured voltages at **B+** and **B-** as closely as possible.
7. Repeat **steps 2, 3, 4, 5** **and 6** until the **voltage V** readings are correct.
8. Enter the command **cal stop** in order to exit the voltage **calibration mode**. The message **Calibration stop** should appear on the serial monitor.

**5.3.3- Current calibration**

Please proceed with calibrating the reading of the **current I** by following the steps below:

1. Connect a discharged **lithium-ion battery** in series using a **digital ampere** **meter (set to the 10 A range)** to terminals **B+** and **B-.**
2. The message Charging should appear in the serial monitor and the measured current value should start to gradually increase until it reaches a maximum of approximately

 **1.5 A.**

1. Enter the **[.] (dot)** command and check the displayed value for **I** which must match the measured current as closely as possible.
2. If the output of the **[.]** command is **higher** than the ampere meter reading: **Increase** the **Rshunt** value by **10 mΩ** by calling the **rshunt** command.
3. If the output of the **[.]** command is **lower** than the ampere meter reading: **decrease** the **Rshunt** value by **10 mΩ** by calling the **rshunt** command.
4. Repeat **steps 3, 4, and 5** until the **current I** readings are correct.

**5.4- Trace Buffer**

A **lithium-ion battery charger** records events that occur during the charging process in a circular buffer within the available **EEPROM** space. The contents of the trace buffer are dumped using the **t command**. Here is a sample trace log output for a complete shipping cycle as shown in **figure 7** below:

****

**Figure 7: t command**

Trace messages have the format **<timestamp>: <event> <value>.** While the timestamp counts the minutes that have passed since the beginning of the charging process. The following table shows the available events and their descriptions:

|  |  |
| --- | --- |
| Event | Description |
| \* | **Beginning of the charging cycle, indicates the maximum battery voltage Vmax in V** |
| % | **Initial charge state %** |
| T | **Maximum permissible charging time Tmax in minutes** |
| C | **Maximum permissible charging capacity Cmax in mAh** |
| S | **Safety charging in progress, Isafe is indicated in mA** |
| I | **Normal charge in progress, indicates Icharge in mA** |
| v | **Instantaneous battery voltage V=V1-V2 in mV** |
| i | **Instantaneous battery current I in mA** |
| F | **Battery full, indicates the end-of-charge condition (1 = Ifull reached, 2 = Cmax reached, 3 = Tmax reached)** |
| t | **Actual charging time T in minutes** |
| c | **Actual charged capacity C in mA** |
| E | **Error (1 = over-volt, 2 = under-volt, 3 = open-circuit, 99 = CRC fail)** |

**Table 5: Available events and descriptions**

**6- Download:**

Below you can find **GitHub download links** for the **Arduino** **firmware** **source** **code**, **Eagle** **schematic** **source** **files** and bill of material. All of the source code is distributed under the **GNU** **General** **Public** **License** **v3.0**.

Please note that the current implementation uses the watchdog timer functionality which requires the customized **Arduino** bootloader found under the **link** below. For more details, please follow the installation instructions found within the **README file** on **GitHub**.

 [Customized Arduino Bootloader](https://github.com/microfarad-de/bootloader)

[Lithium-Ion Charger Firmware](https://github.com/microfarad-de/li-charger)

[Eagle Schematic Source Files](https://github.com/microfarad-de/eagle/tree/master/projects/li-charger)

[Bill of Material](https://raw.githubusercontent.com/microfarad-de/li-charger/master/doc/li-charger-bom.txt)