**Creating a USB charger with an RGB wire that displays charge levels based on color:**

**Method Used: -**

The Coulomb counting method is a mathematical approach to measure the charge level of a battery by tracking the amount of charge flowing in or out over time. This technique is based on the fundamental relationship between current and charge and is commonly used in applications where real-time tracking of battery charge is required.

### **1. Core Principle of Coulomb Counting**

The fundamental formula behind Coulomb counting is derived from the relationship:

*Q=∫I dt*

In this relationship:

* **Current** is defined as the flow of electric charge over time, so integrating the current over time yields the total charge that has passed.

### **2. Converting to Practical Units**

In practical applications, particularly with small electronic devices, we often express charge in milliampere-hours (mAh) instead of Coulombs. To convert from Coulombs to mAh, we use:

*1 mAh=3.6 Coulombs*  
Thus, when Coulomb counting is implemented in a microcontroller, we accumulate charge in mAh by measuring current over small-time intervals. The total charge *Q* in mAh is then given by:

*Q(t)=∑I(t)⋅Δt*

where:

* *I(t)*  is the instantaneous current measurement at each time interval,
* *Δt*  is the time interval in hours over which each current measurement is made.

### **3. Calculating Battery Capacity and Remaining Charge**

For a battery with a known capacity, we can track its **remaining charge** by keeping an updated value of the cumulative charge that has either entered or left the battery:

1. Initial State: If the battery starts fully charged, we can initialize the remaining capacity as the full capacity C in mAh.
2. Discharge (or Charge) Calculation: Each measured current over time reduces (or increases) the remaining charge:

*Remaining Capacity=C−∑I(t)⋅Δt*

In this case:

* Discharging (current leaving the battery) decreases the remaining ca-pacity.
* Charging (current entering the battery) increases the remaining capacity.

For a battery, charge flows both in (during charging) and out (during discharging). We use a sign convention where current is positive when the battery is charging and negative when discharging. The cumulative charge (and therefore the SoC) is updated as follows:

* During **discharge**, *Qused(t)*) is increased by *Ii⋅Δt*During **charge**, *Qused* is decreased.

This gives us the net charge drawn or replenished, allowing us to calculate the battery’s remaining capacity

### **4. Accuracy and Calibration**

The accuracy of Coulomb counting depends on several factors:

* Current measurement accuracy: Sensors like the INA219 are used to measure current precisely.
* Sampling frequency: Higher sampling rates improve the accuracy of *Q(t)* by reducing time gaps where unmeasured current flows might occur.
* Calibration: Periodic recalibration based on the battery’s fully charged or fully discharged state helps to correct errors over time.

Coulomb counting is sensitive to measurement drift and inaccuracies over long periods. To address this, recalibration is often employed using:

* **Full charge voltage threshold**: When the battery voltage hits a known full-charge voltage (e.g., 4.2V for lithium-ion), the system resets *Qused*  to zero and sets the SoC to 100%.
* **Full discharge threshold**: Similarly, if the battery voltage drops to a set low voltage (e.g., 3.0V), it resets the charge level and adjusts *Cnominal* if using a dynamic SoC model.

**Mathematical Equation for Calibration in Coulomb Counting**

Calibration in a Coulomb Counting system involves ensuring that the charge (capacity) measurements are accurate against the nominal and actual conditions of the battery. Here's a breakdown of the mathematical equations used for calibration:

**1. Nominal Capacity (C\_nominal):**

The nominal capacity of the battery is the manufacturer's rated value for the battery, given in milliampere-hours (mAh):

Cnominal=Inominal⋅t

Where:

* InominalI ​: Nominal discharge current (A or mA)
* t: Time to discharge the battery fully (hours)

**2. Measured Capacity (C\_measured):**

Measured capacity is calculated by integrating the current over time:

A math equation with white text

Description automatically generated with medium confidence

Where:

* Iload: Instantaneous current drawn by the load
* t0​: Start time of measurement
* tn: End time of measurement

Numerically, this is often calculated in a discrete form as:

A black background with white text

Description automatically generated

Where:

* Iload(tk: Measured current at the kkk-th interval
* Δt: Time step (seconds)

**3. Correction Factor (K):**

The correction factor compensates for inaccuracies due to system noise, temperature effects, or aging of the battery:

K=Cnominal/Cmeasured

**4. Calibrated Capacity:**

The calibrated capacity is updated using the correction factor:

Ccalibrated=Cmeasured⋅K

**5. State of Charge (SoC):**

The State of Charge is expressed as the ratio of remaining capacity to the nominal or calibrated capacity:

SoC=Cremaining/Ccalibrated×100%

Here:

* Cremaining ​: The accumulated charge during operation.

### **Final SoC Calculation with Calibration**

After recalibration, the SoC can be calculated as follows:

*SoC=(Cnominal−Q(t))/Cnominal×100%*

may be adjusted based on calibration cycles. This keeps the estimated capacity and charge levels aligned with the actual battery performance over time.

This method, while effective, depends on careful calibration and regular resets to minimize drift and

### **Practical Example Calculation**

Suppose:

* A battery has an initial capacity *C* of 1000 mAh.
* It discharges with a current of 200 mA.
* We measure the current every second (1/3600 hours per interval).

Each interval’s charge contribution *Qi*

*Qi=I⋅Δt=200 mA×13600 hours=0.0556 mAh*

Accumulating these over time gives us the remaining capacity after each measurement.

**How the Machine Learning Approach Will Work for Battery Monitoring:**

The ML-based method uses **data-driven models** to predict the **State of Charge (SoC)** and improve **battery calibration**. Here's a step-by-step explanation of how the process works:

**1. Data Collection (Sensors & Inputs)**

The system collects real-time battery data using sensors connected to your battery. These inputs form the dataset for training and prediction:

* **Voltage (V)**: The electrical potential of the battery.
* **Current (I)**: The flow of electric charge.
* **Temperature (T)**: Environmental or battery temperature (optional but useful for modeling).
* **Charging/Discharging State**: Whether the battery is charging or discharging.

This data is gathered during normal operation and logged continuously over time.

**2. Data Preprocessing**

Before feeding the collected data into the ML model:

* **Filtering**: Removes noisy or spurious sensor readings.
* **Feature Engineering**: Derives additional parameters, such as:
  + Rate of voltage change (dV/dt)
  + Cumulative charge consumed (∑I⋅Δt).
  + Power (P=V×I).
* **Normalization**: Ensures data scales are consistent for training.

**3. ML Model Training**

Using historical data, an ML model is trained to recognize patterns in battery behavior.

* **Example Input-Output Relationship**:
  + Input: Voltage, Current, Temperature, and Time.
  + Output: Predicted SoC or recalibrated battery capacity (Cnominal ​).

The model learns:

* How the battery’s voltage drops over time.
* How quickly charge is consumed at different currents.
* Nonlinear effects like aging, temperature dependencies, and variations between charging and discharging.

**4. Prediction and Real-Time Estimation**

Once trained, the model is deployed on a microcontroller or edge device (like an ESP32). It predicts the SoC in real-time using live sensor data:

1. **Input:** Real-time voltage, current, and optional temperature data.
2. **Processing:** The ML model uses this data to estimate:
   * SoC as a percentage (0%–100%).
   * Recalibrated capacity (Cnominal ​) if discrepancies arise.
3. **Output:** Accurate, real-time SoC and capacity readings.

**5. Recalibration Over Time**

As batteries degrade, their capacity decreases. The ML system continuously adjusts predictions:

* **Continuous Learning:** With every charge/discharge cycle, the system updates its model using new data.
* **Recalibration:** When a full charge or discharge cycle occurs, the ML model recalculates the battery’s true capacity. This allows for more accurate predictions in future cycles.

**SOH Estimation via Voltage and Coulomb Counting**

Incorporating **voltage** can improve the accuracy of SOH estimation:

1. **Voltage Plateau Analysis**: Certain voltage ranges correspond to the battery's state of charge (SOC) during discharge. By correlating the voltage data with the SOC and the measured CactualC\_{\text{actual}}Cactual​, SOH can be refined.
2. **Dynamic Corrections**: Use real-time voltage measurements to adjust the integration of current based on the battery's discharge characteristics.

**Considerations**

* **Temperature Effects**: Include temperature compensation, as battery capacity decreases in cold environments and can artificially lower SOH.
* **Aging Models**: You can improve SOH estimation by integrating machine learning models or empirical degradation equations tailored to your battery chemistry.

### **Summary**

In summary, Coulomb counting integrates current over time to keep track of battery charge, using periodic recalibration and accurate sensors to improve reliability. This method is efficient for continuous, real-time battery monitoring across various applications.

### **Materials Needed**

1. **Microcontroller**: Arduino Nano or ESP-32 wifi version
2. **Current Sensor**: INA219 (https://robu.in/product/cjmcu-219-ina219-i2c-interface-no-drift-bi-directional-current-power-monitoring-sensor-module/?gad\_source=1&gclid=CjwKCAiAudG5BhAREiwAWMlSjLjCiCL3MT25XsoquOnIDtWcX\_gpDM0NKlDFyzBzE\_LrHS6yRfmqThoC0jwQAvD\_BwE)
3. **External Battery**: A lithium-ion battery or any rechargeable battery(For Poetable mode)
4. **RGB LED**: RGB LED strip or a single RGB LED(https://robu.in/product/ws2812b-rgb-addressable-led-module/?gad\_source=1&gclid=CjwKCAiAudG5BhAREiwAWMlSjCbZH2rVO5goVqxa2OSFwK9HVfoPNGVL\_041n6rSmo9mFLWtynbcNBoC9nMQAvD\_BwE) or any tother w2818B led strip or (http://www.world-semi.com/ws2812-family/)
5. **Resistors**: Current-limiting resistors for the LED
6. **USB Charger Module**: For charging the device(https://robu.in/product/type-c-usb-5v-2a-step-up-boost-converter-with-usb-charger/?gad\_source=1)
7. Capacitors for smoothing power fluctuations if needed

**References**

**-** [**https://www.circuitbasics.com/build-an-arduino-controlled-power-outlet/**](https://www.circuitbasics.com/build-an-arduino-controlled-power-outlet/)

**-** [**https://www.powertechsystems.eu/home/tech-corner/lithium-ion-state-of-charge-soc-measurement/**](https://www.powertechsystems.eu/home/tech-corner/lithium-ion-state-of-charge-soc-measurement/)

**- https://www.circuitbasics.com/arduino/**