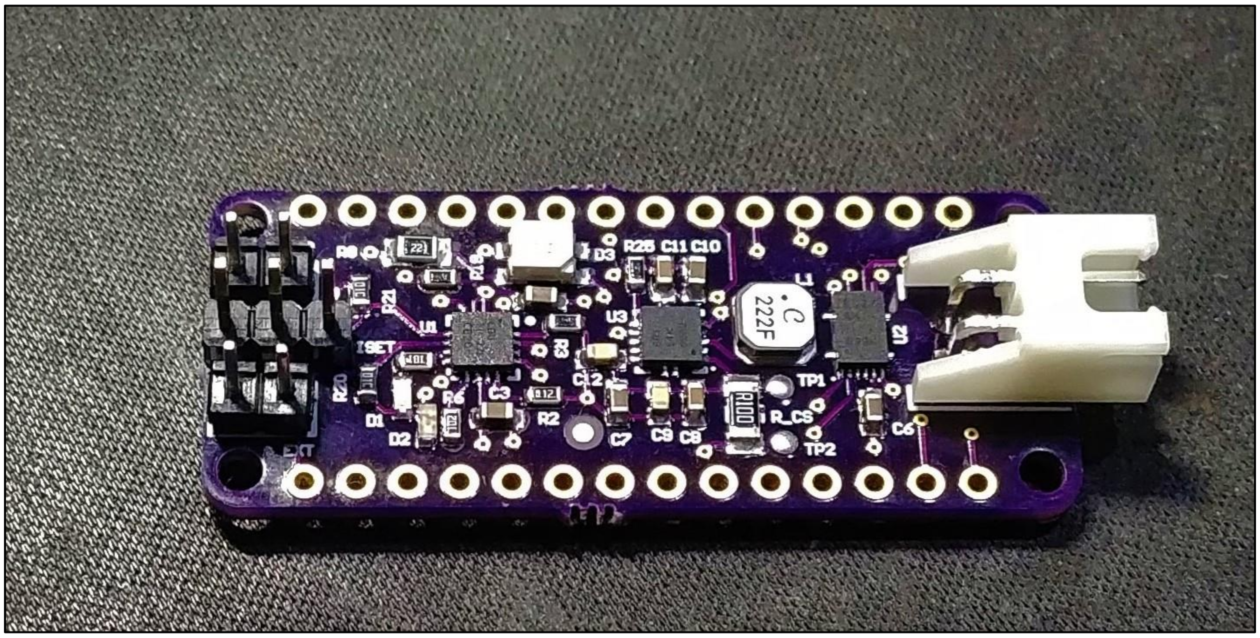
ELE391: Final Project Report - Metro Mini Battery Shield

By: Peter Campellone, Zack Duclos, Cole Wright

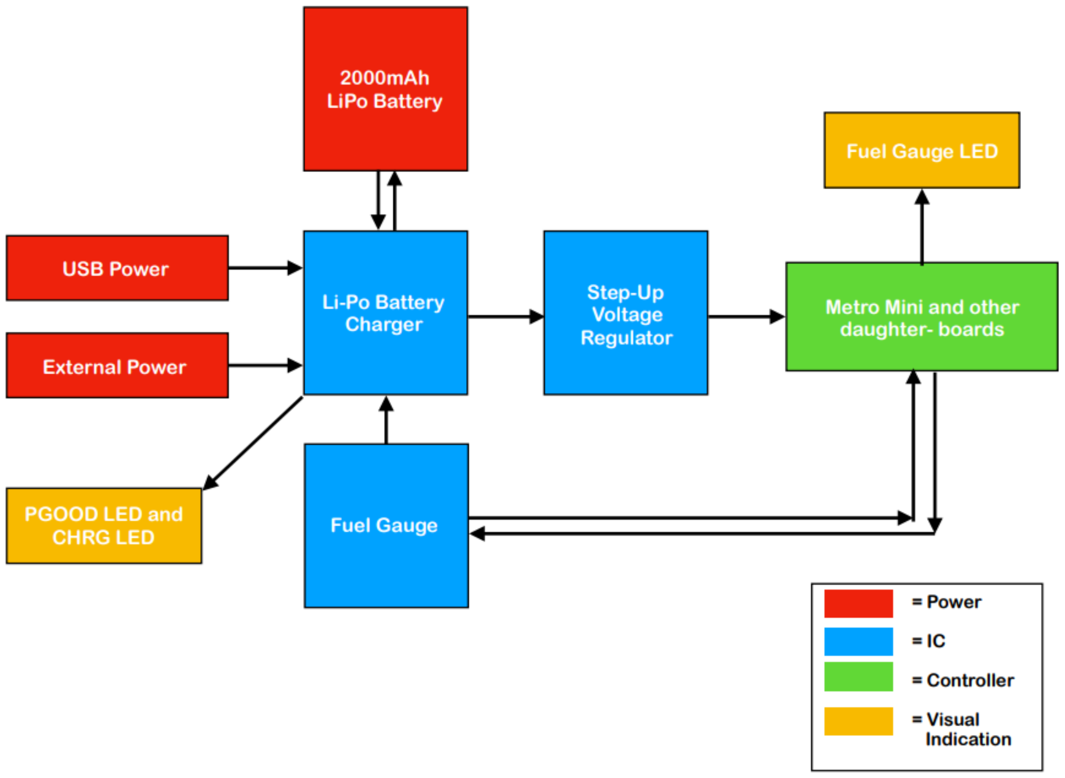
May 3, 2018



Board Requirements

- Be able to supply approximately 1A of continuous current in order to power Metro Mini and up to three other “daughter” boards
- Be able to charge an externally connected LiPo battery
- Be able to charge an externally connected LiPo battery and still power the Metro Mini and up to three other “daughter” boards all at the same time
- Be able to take power in from either an external source or from the USB cable connected to the Metro Mini board
- Be able to switch between USB power and battery power automatically in the case that either one is not available
- Be able to measure current draw using a built-in current sense resistor
- Be able to set the charge rate in order to select different charge rates based on different power inputs (USB, external ,etc)
- Feature an On/Off switch in order to disable the charger completely if necessary
- The fuel gauge needs to be able to report battery statistics to the Metro Mini over I2C at a regular interval such that the tri-color LED can be used as a visual indication of the “fuel” level of the battery
- Feature an built-in 5V step-up regulator that can take a wide range of input voltage and output a constant 5V (primarily used for stepping up LiPo 3.7V to 5V)

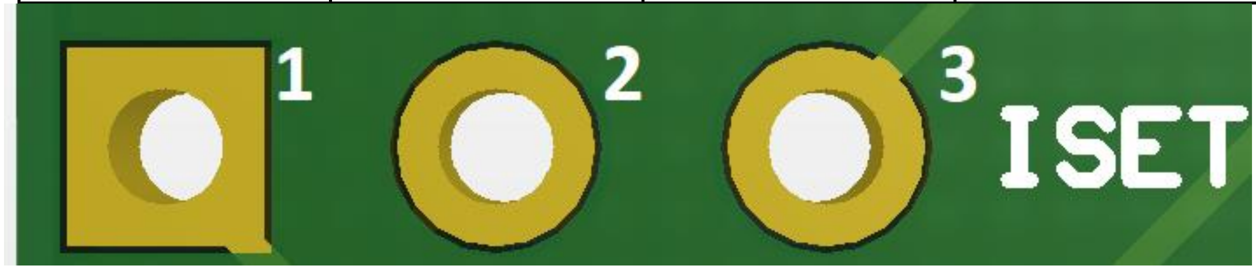
Implementation



The flowchart above displays the connections of the Metro Mini Battery Shield and how the different components interface with one another. The operation is described below:

1. Input power is either supplied by the USB connection on the Metro Mini or by an external power connection on the Battery Shield
2. From here the battery charger will direct that power depending on whether or not a battery is connected.
    a. If a battery IS connected then the battery charger will charge that battery and power the Metro Mini simultaneously using a built-in Texas Instruments feature called “Dynamic Power Path Management”
    b. If a battery IS NOT connected then the battery charger will directly route that power to the Metro Mini
3. Some considerations relevant to the charge and power management processes:
    a. The battery charger is only enabled when the PWR jumper is shorted
    b. External power input allowable range is 4.35 - 6.4V
    c. The ISET jumper defines the charge rate of the battery and connecting the jumper according to the table and image below will define the charge rate

Pin 1	Pin 2 (GND)	Pin 3	Charge Mode
Shorted	Shorted	Shorted	100mA
Shorted	Shorted	Open	500mA
Open	Shorted	Shorted	972mA (max based on voltage regulator)
Open	Open	Open	Standby (default)



4. If the battery charging process is started and proceeding successfully then the CHRG LED will illuminate
5. If a battery is connected correctly and the battery charger has recognized it then the PGOOD LED will illuminate
6. The power that passes through the battery charger is sent to the Step-Up voltage regulator in order to output a constant +5V for use by the Metro Mini and other daughter boards. All power will be sent through the step-up regulator regardless of the power input type or whether or not a battery is connected.
7. While all of this is happening the fuel gauge will directly measure the battery and multiple statistics relevant to the battery. These statistics are sent back to the Metro Mini over I2C using Arduino code that was written specifically to be used to interface with the BQ27441 fuel gauge.
8. Part of the Arduino code will drive one of the LEDs on the tri-color LED based on the charge percentage of the battery (a.k.a. State of charge). The color of the LED is based on the following table and state of charge value:

State of Charge	LED Color
Less than 30% (inclusive)	Red
Between 30% and 70%	Yellow
Greater than 70% (inclusive)	Green

Testing Performed

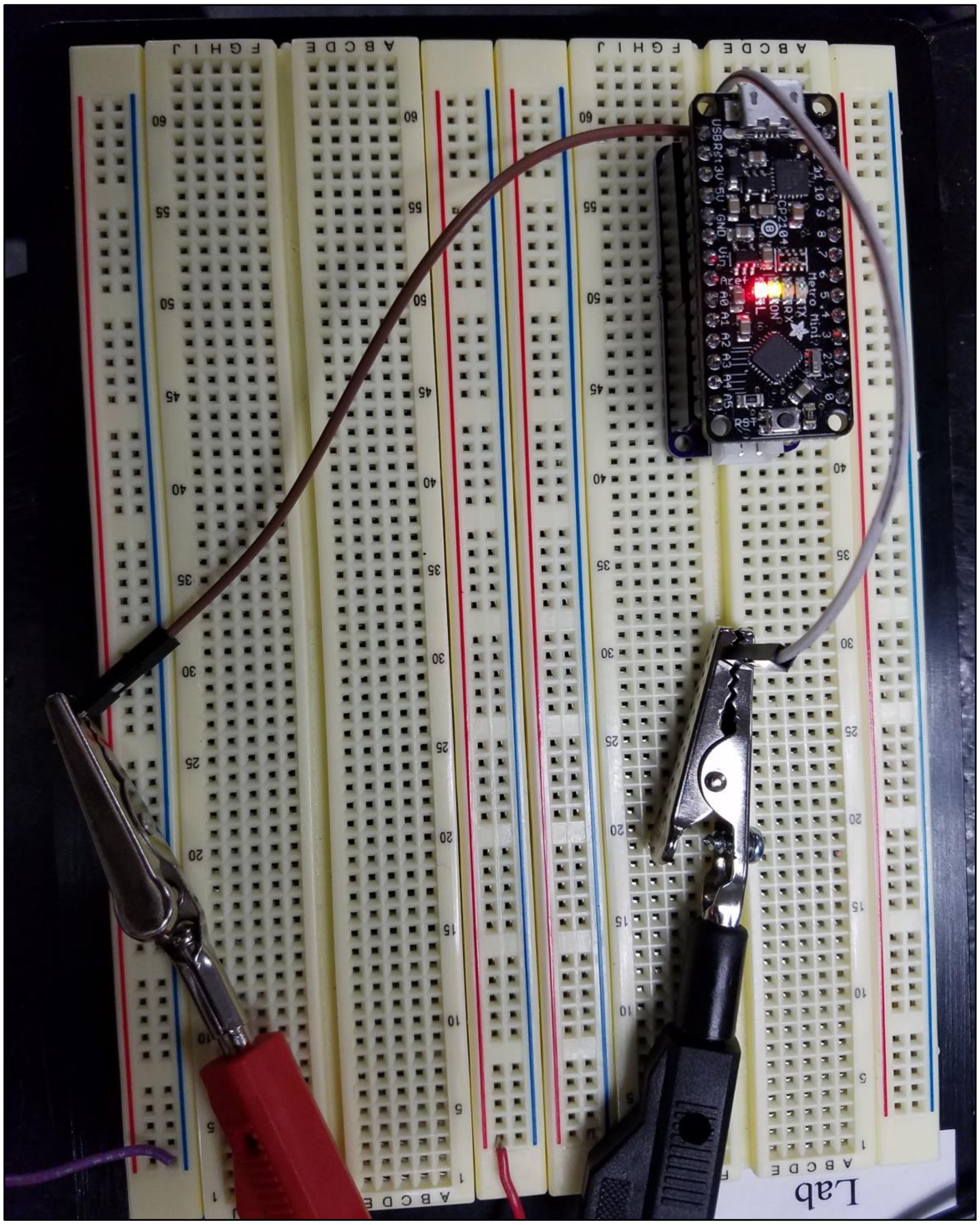


Figure 1: Testing PCB with external power input, successfully powering Metro Mini

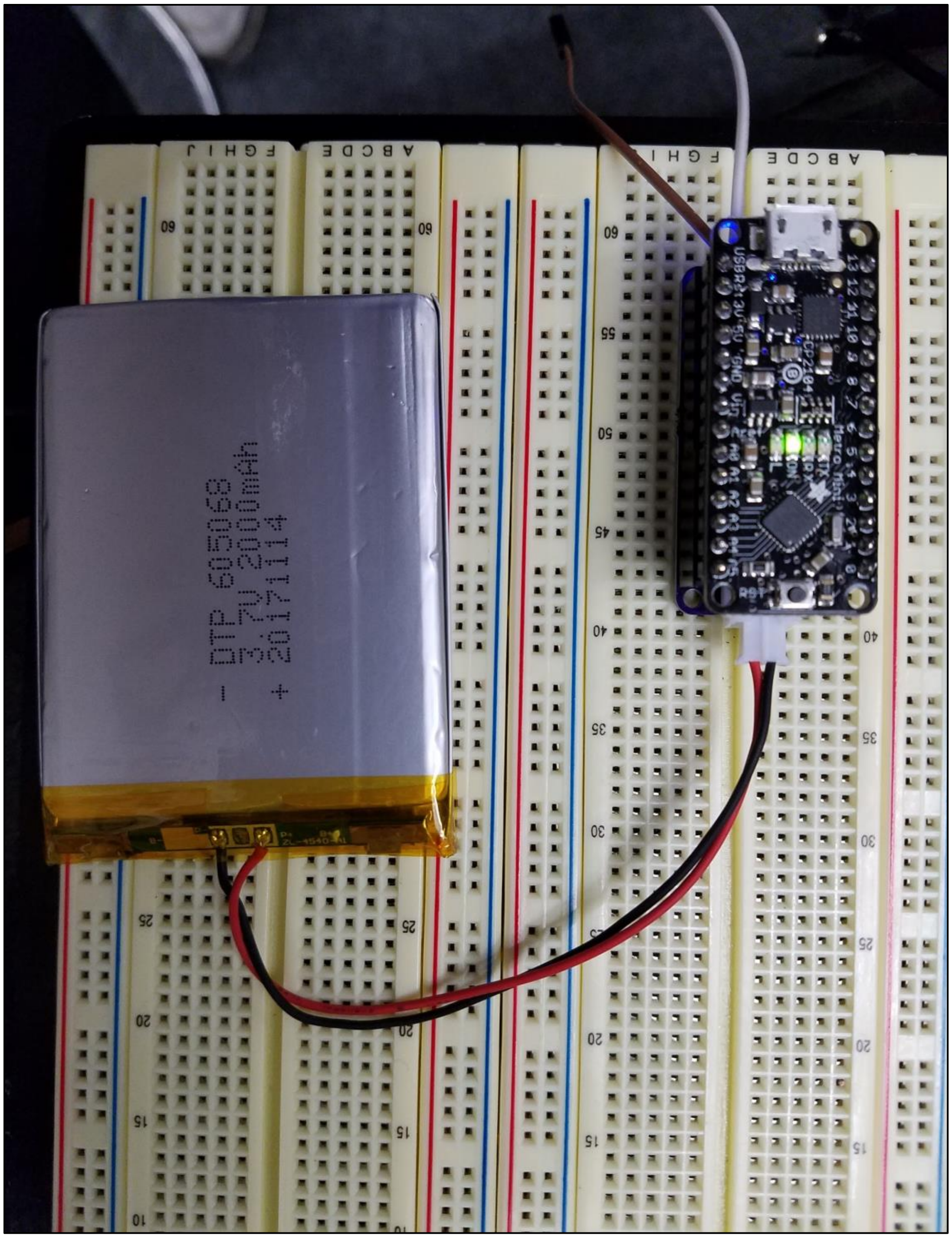


Figure 2: Testing PCB with battery power, successfully powering Metro Mini, PGOOD LED illuminated indicated battery is recognized by battery charger

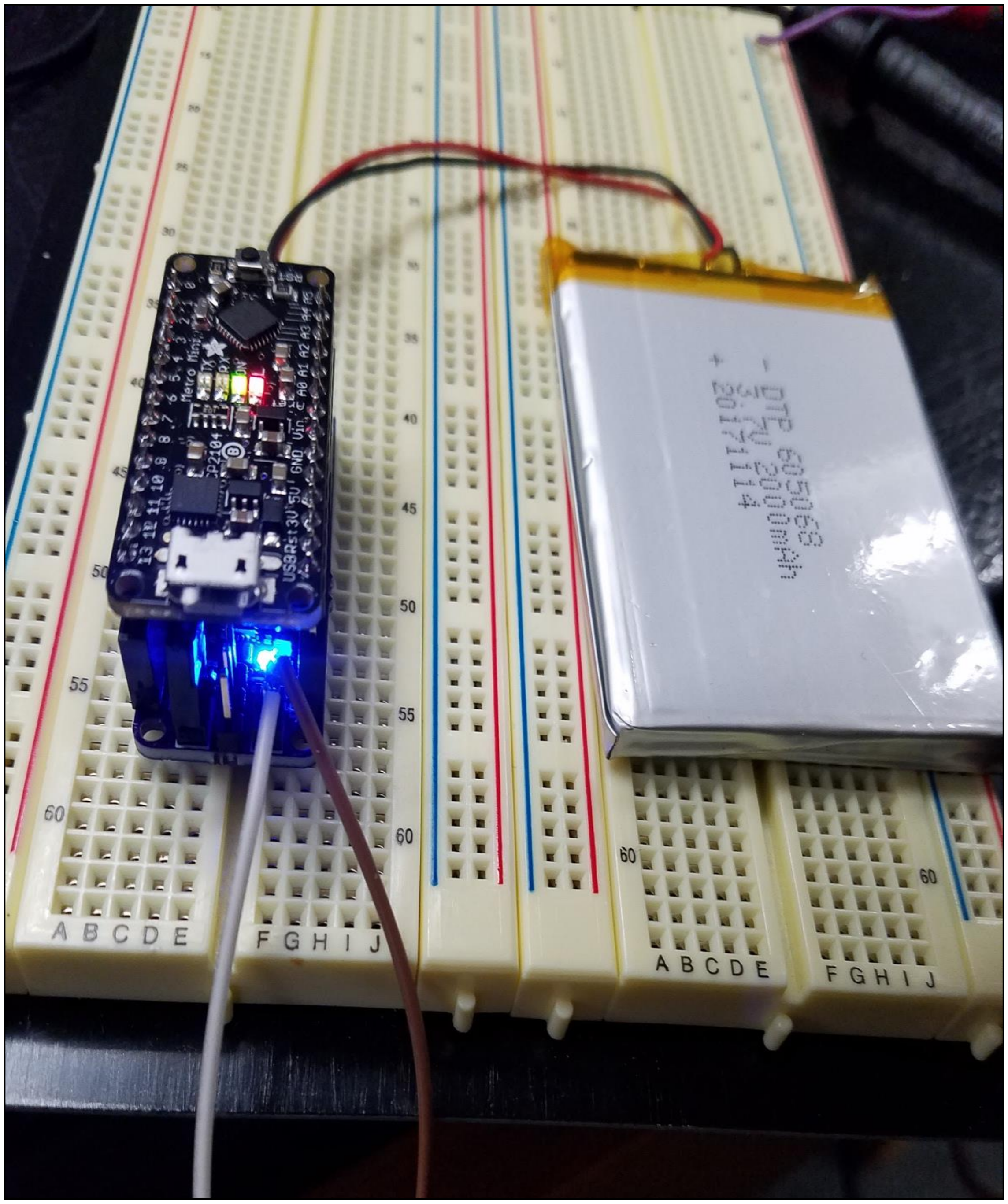


Figure 3: Testing PCB with battery power, successfully powering Metro Mini, PGOOD LED illuminated indicating that battery charger has recognized battery

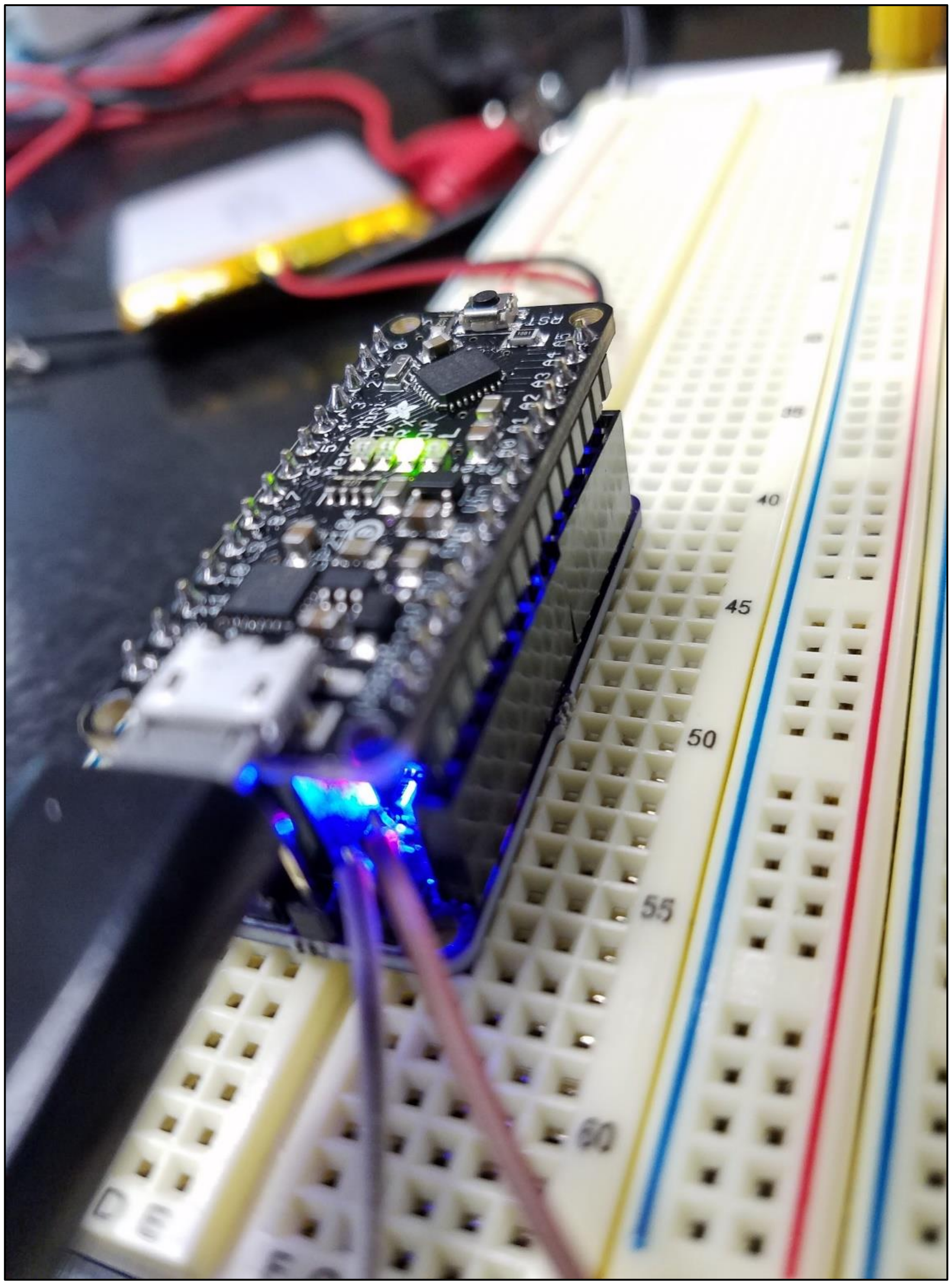


Figure 4: Testing PCB with USB connection, successfully powering Metro Mini, PGOOD LED illuminated indicating that battery charger has recognized battery, CHRG LED illuminated indicating that battery is being charged using USB power

In each test, the PCB was tested with all four of the charge rate settings to verify operation and expected current draw

Conclusion

In reference to the board requirements listed above only one part of the entire PCB did not work. Specifically, the tri-color LED used as a visual indication of the fuel level did not work after assembly. After examining the PCB, schematics, and datasheet it was determined that the tri-color LED was common anode although it had been connected as if it was common cathode. This caused the LED to be reversed bias and therefore not illuminate. In order to fix this the ground connection to the common anode pin on the LED needed to be replaced with a connection to the +5V rail. Further, the initial design featured three digital pins on the Metro Mini's Atmega328 to be used in order to source current to power each of the individual LEDs although with the incorrect common anode connection driving current from the digital pins would not complete the circuit. After our initial testing we tried cutting a trace to ground from the resistor in the tri-color LED circuit and then wiring in a jumper wire to a 5V bus in order to source current through the digital pins for each LED to complete the circuit. While this did work, the solution was not ideal and the modifications were revered to normal in order prevent the PCB from becoming non-functional or damaged.

Regardless of this non-functioning part of the circuit, the rest of the PCB did function as expected. The PCB did meet all of the other board requirements as mentioned above. Although, there were a number of errors noticed and subsequent solutions that could be made in a second revision of the board. These errors and their corresponding solutions are listed below:

Error/Solution Log

Error: Tri-Color LED wired as common cathode instead of common anode.
Solution: Replace GND connection to tri-color LED anode with +5V bus connection

Error: Jumpers for PWR, EXT, and ISET are difficult to access when Metro Mini is installed on top
Solution: Replace Jumpers with right angle connectors or something easily accessible from side or bottom

Error: CHG and PGOOD LED silkscreens are reversed
Solution: Change net labels in schematic

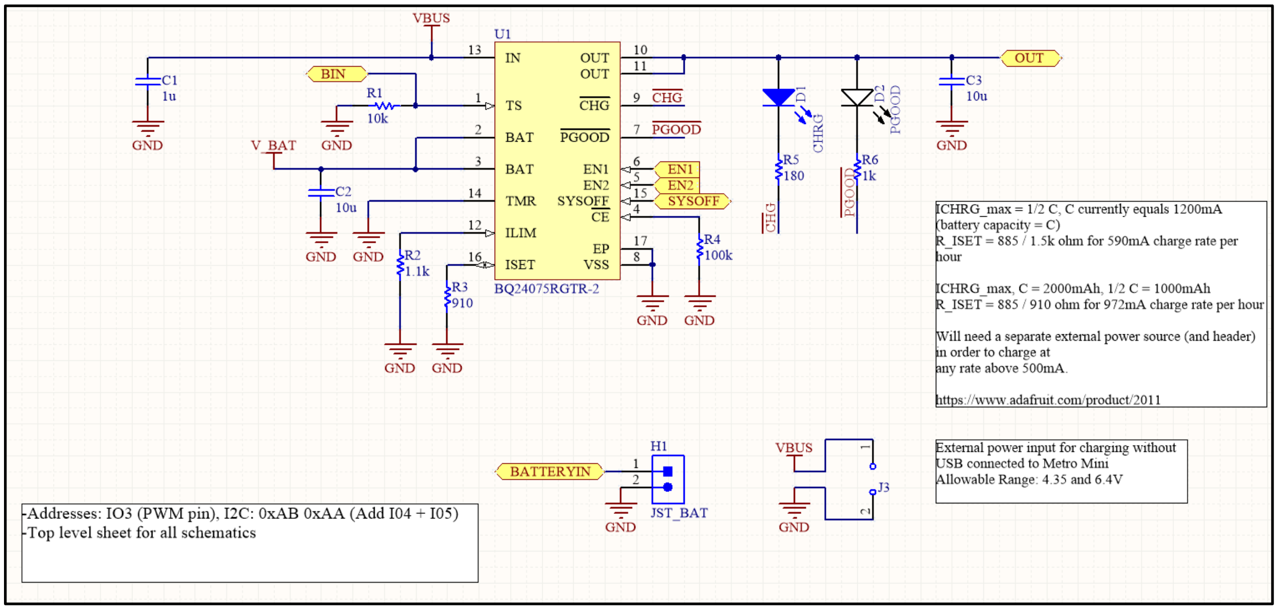
Error: Adjust resistor value for PGOOD LED
Solution: Re-calculate resistor value for PGOOD LED (red) or replace with a different color LED to have comparable brightness to CHRG (blue) LED

Error: Resistor value on tri-color LED is too large and LED brightness is too low
Solution: Replace with a new resistor, recalculate optimal value (likely 100-120 ohm)

Error: JST Connector is wrong physical size, pins are too big for battery connector
Solution: Replace JST connector with correct female part to match physical size and pin size of male JST battery connector

Error: JST connector does not sit flat on PCB and is easily bent when inserting battery connector
Solution: Replace with JST connector with different physical design that sits more securely and fits better on the board

Appendix A: Schematics



ICHRG_max = 1/2 C, C currently equals 1200mA (battery capacity = C)
R_ISET = 885 / 1.5k ohm for 590mA charge rate per hour

ICHRG_max, C = 2000mAh, 1/2 C = 1000mAh
R_ISET = 885 / 910 ohm for 972mA charge rate per hour

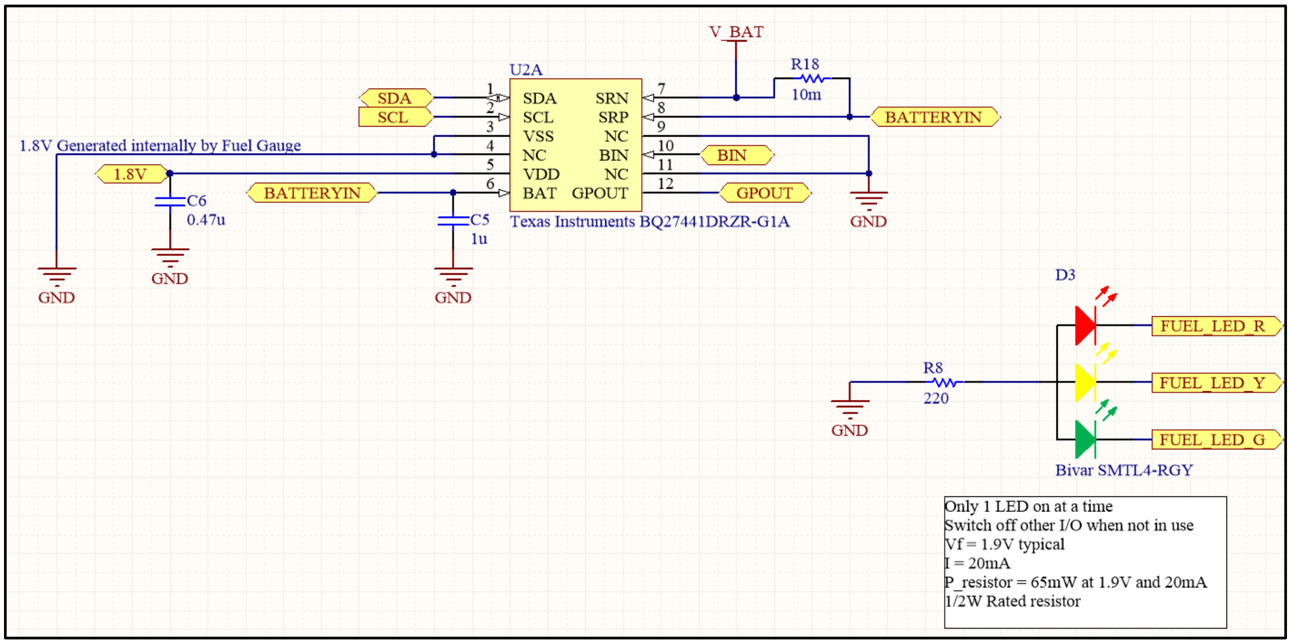
Will need a separate external power source (and header) in order to charge at any rate above 500mA.

https://www.adafruit.com/product/2011

External power input for charging without USB connected to Metro Mini
Allowable Range: 4.35 and 6.4V

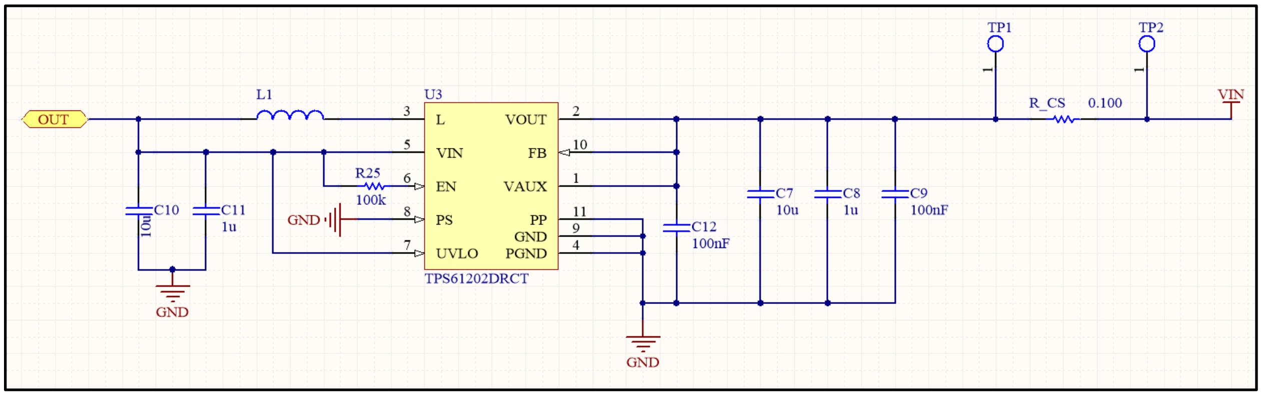
-Addresses: IO3 (PWM pin), I2C: 0xAB 0xAA (Add I04 + I05)
-Top level sheet for all schematics

Schematic #1: Lithium Ion Charger

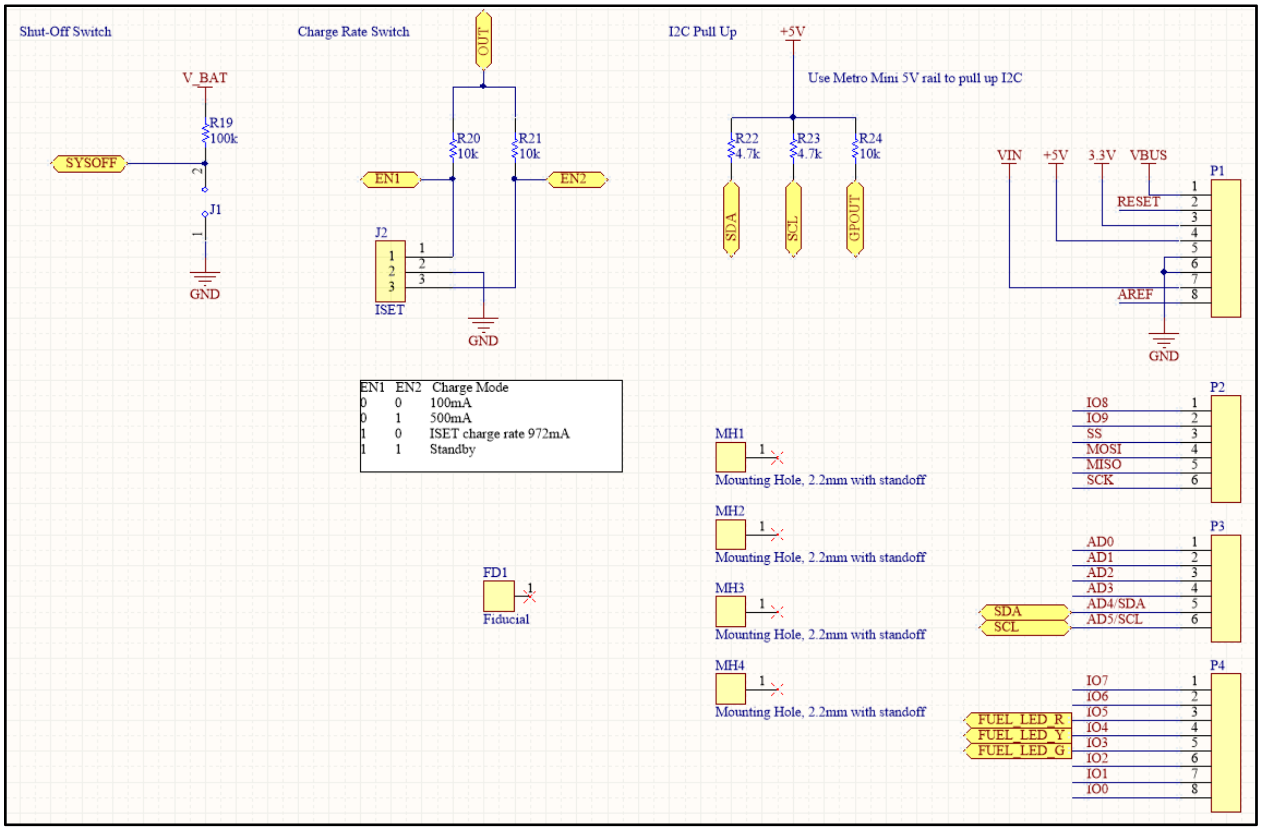


Only 1 LED on at a time
Switch off other I/O when not in use
Vf = 1.9V typical
I = 20mA
P_resistor = 65mW at 1.9V and 20mA
1/2W Rated resistor

Schematic #2: Fuel Gauge



Schematic #3: Step Up Voltage Regulator



Schematic #4: Headers and Misc. Circuits

Appendix B: Arduino Code

```
/******************************************************************************
BQ27441_Basic
BQ27441 Library Basic Example
Jim Lindblom @ SparkFun Electronics
May 9, 2016
https://github.com/sparkfun/SparkFun_BQ27441_Arduino_Library

Modified by: Peter Campellone, University of Rhode Island
April 20, 2018

Demonstrates how to set up the BQ27441 and read state-of-charge (soc),
battery voltage, average current, remaining capacity, average power, and
state-of-health (soh).
******************************************************************************/
#include <SparkFunBQ27441.h>

// Set BATTERY_CAPACITY to the design capacity of your battery.
const unsigned int BATTERY_CAPACITY = 2000; // e.g. 2000mAh battery

void setup()
{
  pinMode(3, OUTPUT);  //Setup pin #3, green LED, as output for sourcing current
  pinMode(4, OUTPUT);  //Setup pin #4, yellow LED, as output for sourcing current
  pinMode(5, OUTPUT);  //Setup pin #5, red LED, as output for sourcing current

  Serial.begin(115200); //Start serial monitor at 115200 bps
  setupBQ27441(); //Connect BQ27441
}

void loop()
{
  int stateOfCharge = printBatteryStats(); //Print battery stats to serial monitor and return SOC for use with
                                           //LEDGauge()
  //int stateOfCharge = 85; //for testing led directly
  LEDGauge(stateOfCharge); //Displays SOC using tri-color LED
  delay(1000); //Updates every 1s
}
```

```
void setupBQ27441(void)
{
  // Use lipo.begin() to initialize the BQ27441-G1A and confirm that it's
  // connected and communicating.
  if (!lipo.begin()) // begin() will return true if communication is successful
  {
  // If communication fails, print an error message and loop forever.
    Serial.println("Error: Unable to communicate with BQ27441.");
    Serial.println("Check wiring and try again.");
    Serial.println("Battery must be plugged in for fuel gauge to work!");
    while (1) ;
  }
  Serial.println("Connected to BQ27441!");

   // Uset lipo.setCapacity(BATTERY_CAPACITY) to set the design capacity
   // of your battery.
   lipo.setCapacity(BATTERY_CAPACITY);
}

int printBatteryStats()
{
  // Read battery stats from the BQ27441-G1A
  unsigned int soc = lipo.soc();  // Read state-of-charge (%)
  unsigned int volts = lipo.voltage(); // Read battery voltage (mV)
  int current = lipo.current(AVG); // Read average current (mA)
  unsigned int fullCapacity = lipo.capacity(FULL); // Read full capacity (mAh)
  unsigned int capacity = lipo.capacity(REMAIN); // Read remaining capacity (mAh)
  int power = lipo.power(); // Read average power draw (mW)
  int health = lipo.soh(); // Read state-of-health (%)

  // Now print out those values:
  String toPrint = String(soc) + "% | ";
  toPrint += String(volts) + " mV | ";
  toPrint += String(current) + " mA | ";
  toPrint += String(capacity) + " / ";
  toPrint += String(fullCapacity) + " mAh | ";
  toPrint += String(power) + " mW | ";
  toPrint += String(health) + "%";

  Serial.println(toPrint);

  return soc;
}
```

```
void LEDGauge(int stateOfCharge) //Displays charge state using tri-color LED,
                                //SOC < 30%: Red, 30% < SOC < 70%: Yellow, SOC >= 70%: Green
{
  if(stateOfCharge <= 30){
    digitalWrite(3,LOW);  //Turn off Green
    digitalWrite(4,LOW); //Turn off Yellow
    digitalWrite(5,HIGH); //Turn on Red

  } else if (stateOfCharge > 30 and stateOfCharge < 70){
    digitalWrite(3,LOW); //Turn off Green
    digitalWrite(4,HIGH); //Turn on Yellow
    digitalWrite(5,LOW); //Turn off Red

  } else if (stateOfCharge >= 70){
    digitalWrite(3,HIGH); //Turn on Green
    digitalWrite(4,LOW); //Turn off Yellow
    digitalWrite(5,LOW); //Turn off Red

  } else
    Serial.println("ERROR: State of charge outside range (0-100)");

}


void LEDBoot() //Run on boot to test tri-color LED, for use when tri-color LED is wired correctly
{
 for(int i = 0; i++; i <=20) //Total runtime ~= 50ms * 3 * 10 = 1.5s
  {
    digitalWrite(3,LOW);  //Turn off Green
    digitalWrite(4,LOW); //Turn off Yellow
    digitalWrite(5,HIGH); //Turn on Red
    delay(50); //Delay

    digitalWrite(3,LOW);  //Turn off Green
    digitalWrite(4,HIGH);  //Turn off Yellow
    digitalWrite(5,LOW); //Turn on Red
    delay(50); //Delay //

    digitalWrite(3,HIGH);  //Turn off Green
    digitalWrite(4,LOW); //Turn off Yellow
    digitalWrite(5,LOW); //Turn on Red
    delay(50); //Delay
  }
}
```