```
// ---IMPORTS---
// ltc6804 imports
#include "src\LTC68041\LTC68041.h"
#include "src\LTC68041\Linduino.h"
#include "src\LTC68041\LT_SPI.h"
// can imports
#include <SPI.h>
#include "src/mcp_can/mcp_can.h"
#include "src/LTC68041/LTC68041.h"

char segment = 'A';

// voltage inits
uint16_t cell_codes[1][12];
float voltages[12];
uint8_t tx_cfg[1][6];

// temp inits
float temp_curve[][35] = {{0, 5, 10, 15, 20, 25, 30,
35, 40, 45, 50, 55, 60, 65, 70, 75, 80, 85, 90, 95,
100, 105, 110, 115, 120},\
                          {2.17, 2.11, 2.05, 1.99, 1.92,
1.86, 1.80, 1.74, 1.68, 1.63, 1.59, 1.55, 1.51, 1.48, 1.
45, 1.43, 1.40, 1.38, 1.37, 1.35, 1.34, 1.33, 1.32, 1.
31, 1.30}};
uint8_t temp_pins[] = {A0, A1, A2, A3, A4, A5, A6, A7,
A8, A9, A10, A11};
float temps[12];

// can inits
MCP_CAN CAN = MCP_CAN(3);
```

```
byte  can_messages[3][8];

void setup()
{
   Serial.begin(115200);

   // ltc6804 setup:
   LTC6804_initialize();
   tx_cfg[0][0] = 0xFE;
   tx_cfg[0][1] = 0x00;
   tx_cfg[0][2] = 0x00;
   tx_cfg[0][3] = 0x00;
   tx_cfg[0][4] = 0x00;
   tx_cfg[0][5] = 0x00;

   // can setup:
   while (CAN_OK != CAN.begin(CAN_1000KBPS, MCP_8MHz)){
     Serial.println("CAN BUS init Failed");
     delay(100);
   }
   Serial.println("CAN BUS Init OK!");
   CAN.setMode(MODE_NORMAL);
   pinMode(3, OUTPUT);

   delay(1000);
}

void loop()
{
   // balancing config
```

```
  // ltc6804 communication:
  wakeup_sleep();
  LTC6804_wrcfg(1, tx_cfg);
  wakeup_idle();
  LTC6804_adcv();
  delay(10);
  wakeup_idle();
  if (LTC6804_rdcv(0, 1, cell_codes) == -1){
    Serial.println("A PEC error was detected in the
received data");
  }

  // voltage readings:
  for(int cell = 0; cell < 12; cell++){
    voltages[cell] = (((float) cell_codes[0][cell]) /
10000.0);
  }

  // temp readings:
  for(int cell = 0; cell < 12; cell++){
    temps[cell] =
temp_conversion(analogRead(temp_pins[cell]) * (5.0 /
1023));
  }

  // print data:
  // print_voltages();
  // print_temps();

  // can message creation:
```

```
  for(int cell = 0; cell < 12; cell++){
    int voltage = (int) (((float) voltages[cell] - 1.8)
* 100.0);
    int temperature = (int) ((float) temps[cell] * 4);

    voltage = truncate_max_min(voltage, 255, 0);
    temperature = truncate_max_min(temperature, 255, 0);

    can_messages[cell/4][cell%4] = voltage;
    can_messages[cell/4][(cell%4)+4] = temperature;
  }

  CAN.sendMsgBuf(200 + (3*(segment - 65)), 0, 8,
can_messages[0]);
  CAN.sendMsgBuf(200 + (3*(segment - 65)) + 1, 0, 8,
can_messages[1]);
  CAN.sendMsgBuf(200 + (3*(segment - 65)) + 2, 0, 8,
can_messages[2]);

  delay(50);
}

// convert temperature probe voltage to temperature in
degrees celcius:
float temp_conversion(float voltage)
{
  int curve_point = 0;
  while(voltage < temp_curve[1][curve_point]){
    curve_point++;
  }
```

```
  float x1 = temp_curve[1][curve_point];
  float x2 = temp_curve[1][curve_point-1];
  float y1 = temp_curve[0][curve_point];
  float y2 = temp_curve[0][curve_point-1];

  return (y2 - y1)*(voltage - x2)/(x2 - x1) + y2;
}

// truncate value to range
int truncate_max_min(int input, int max, int min){
  if(input > max){
    return max;
  }else if(input < min){
    return min;
  }
  return input;
}

void print_voltages(){
  for(int cell = 0; cell < 12; cell++){
    Serial.print("V" + String(cell) + ": ");
    Serial.print(((float)cell_codes[0][cell]) * 0.0001);
    Serial.print(", ");
  }
  Serial.println();
}

void print_temps(){
  for(int cell = 0; cell < 12; cell++){
    Serial.print("T" + String(cell) + ": ");
    Serial.print(temps[cell]);
```

```
    Serial.print(", ");
  }
  Serial.println();
}
```