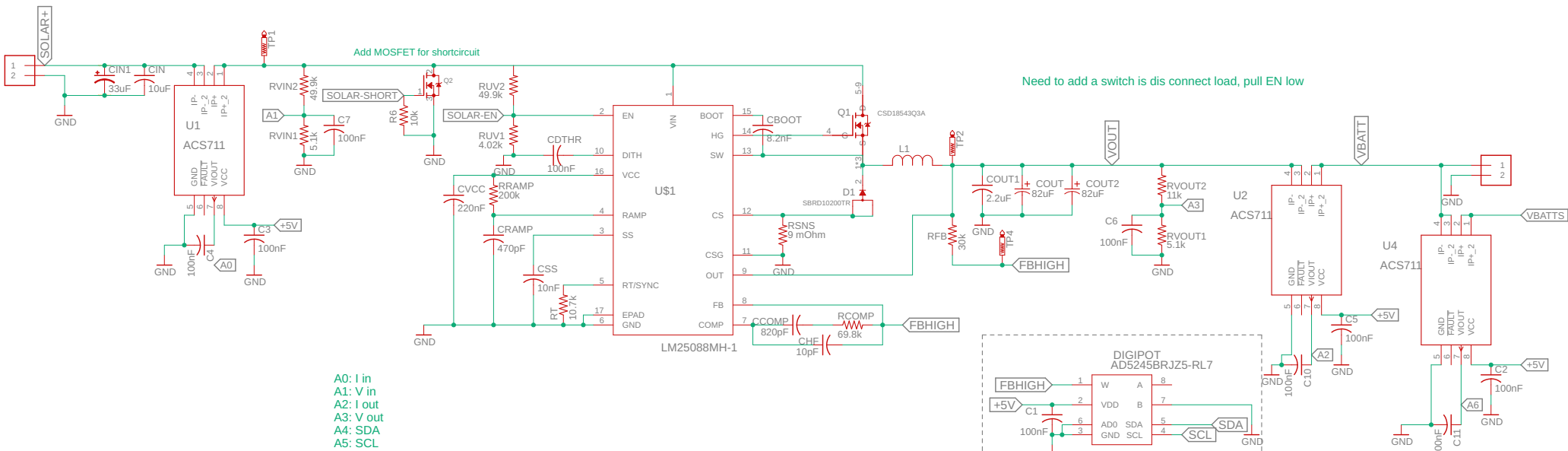
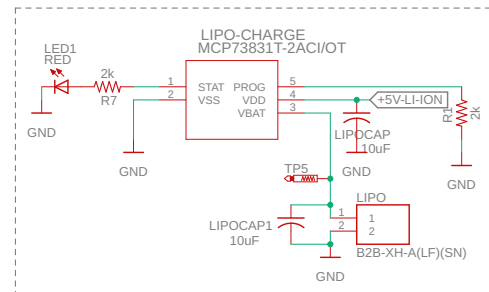
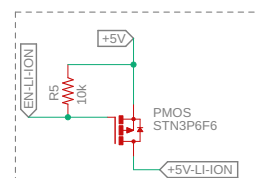
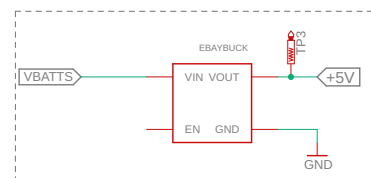
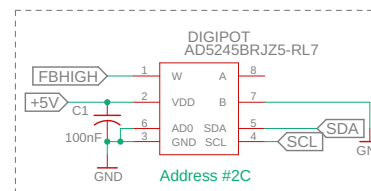
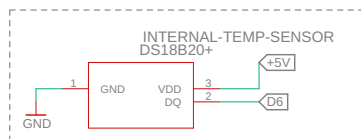
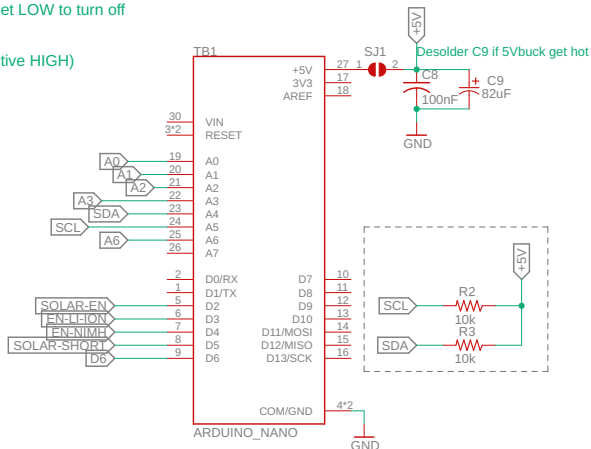
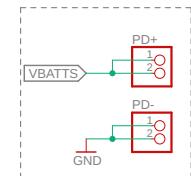
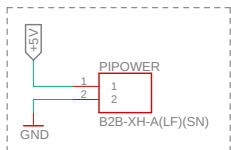
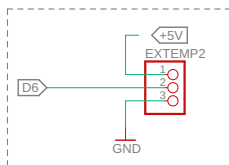
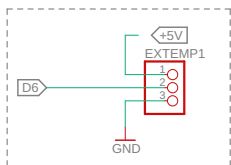
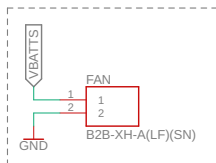
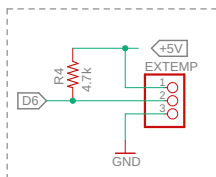


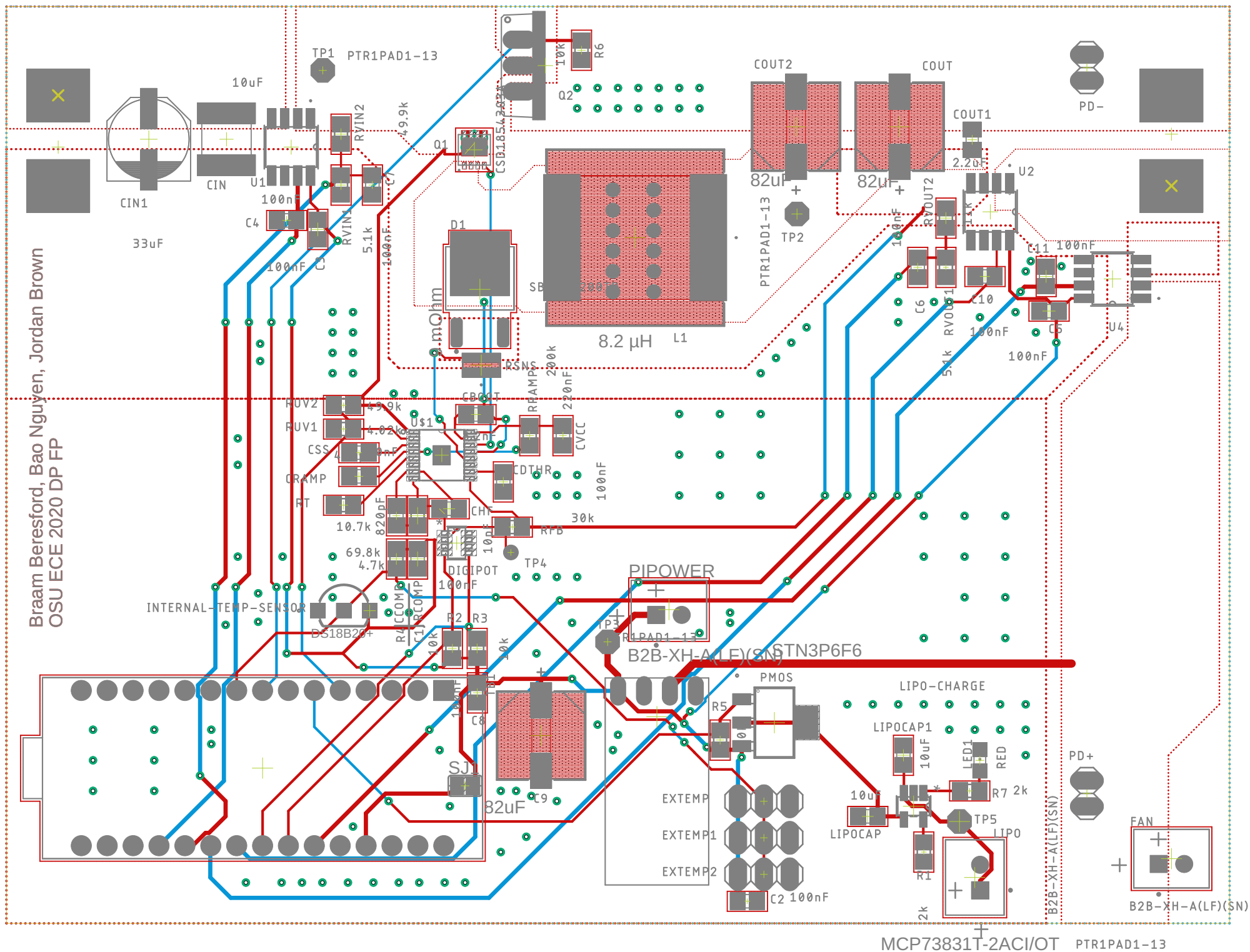
SOLAR 17 DOCUMENTATION

Braam Beresford, Bao Nguyen, Jordan Brown

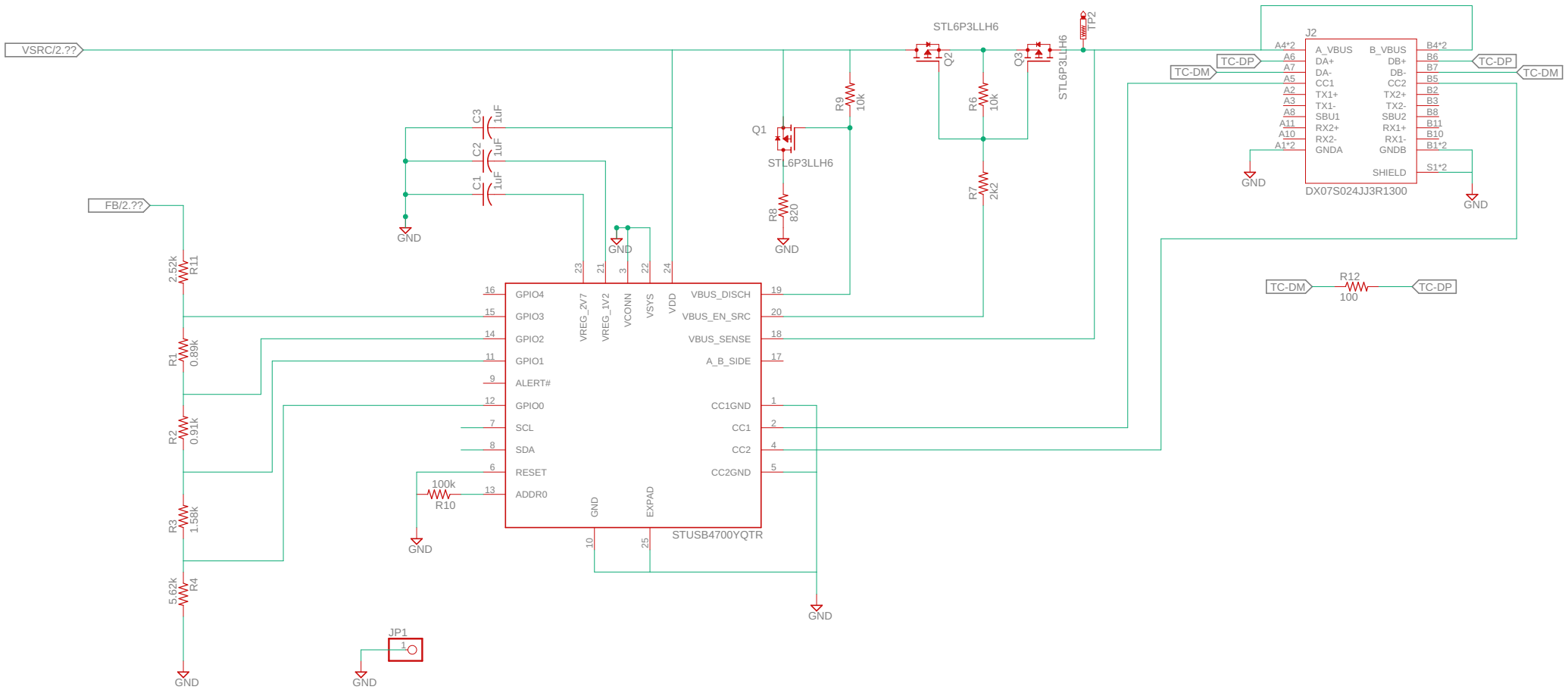


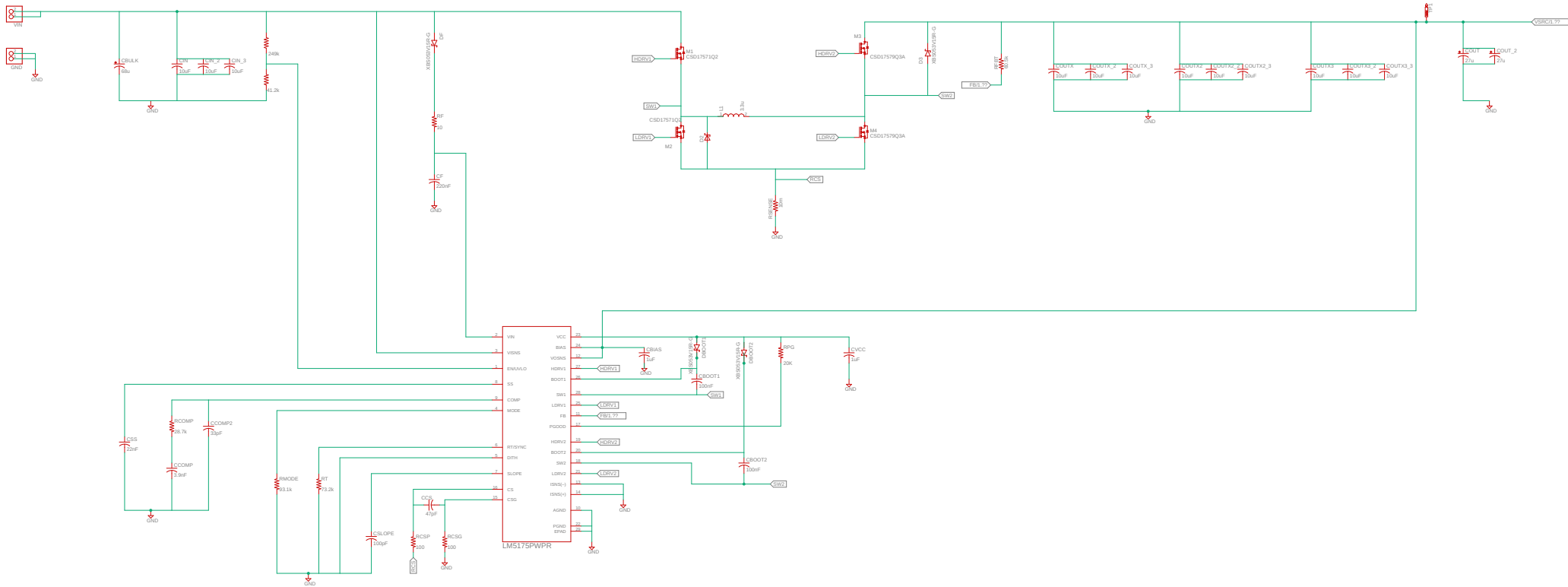
- D2: Solar-Enable (active HighZ), set LOW to turn off
D3: ENable Li-ion (active LOW)
D4: ENable NiMH (active LOW)
D5: Solar Short Circuit ENable (active HIGH)
D6: External Temperature Sensor



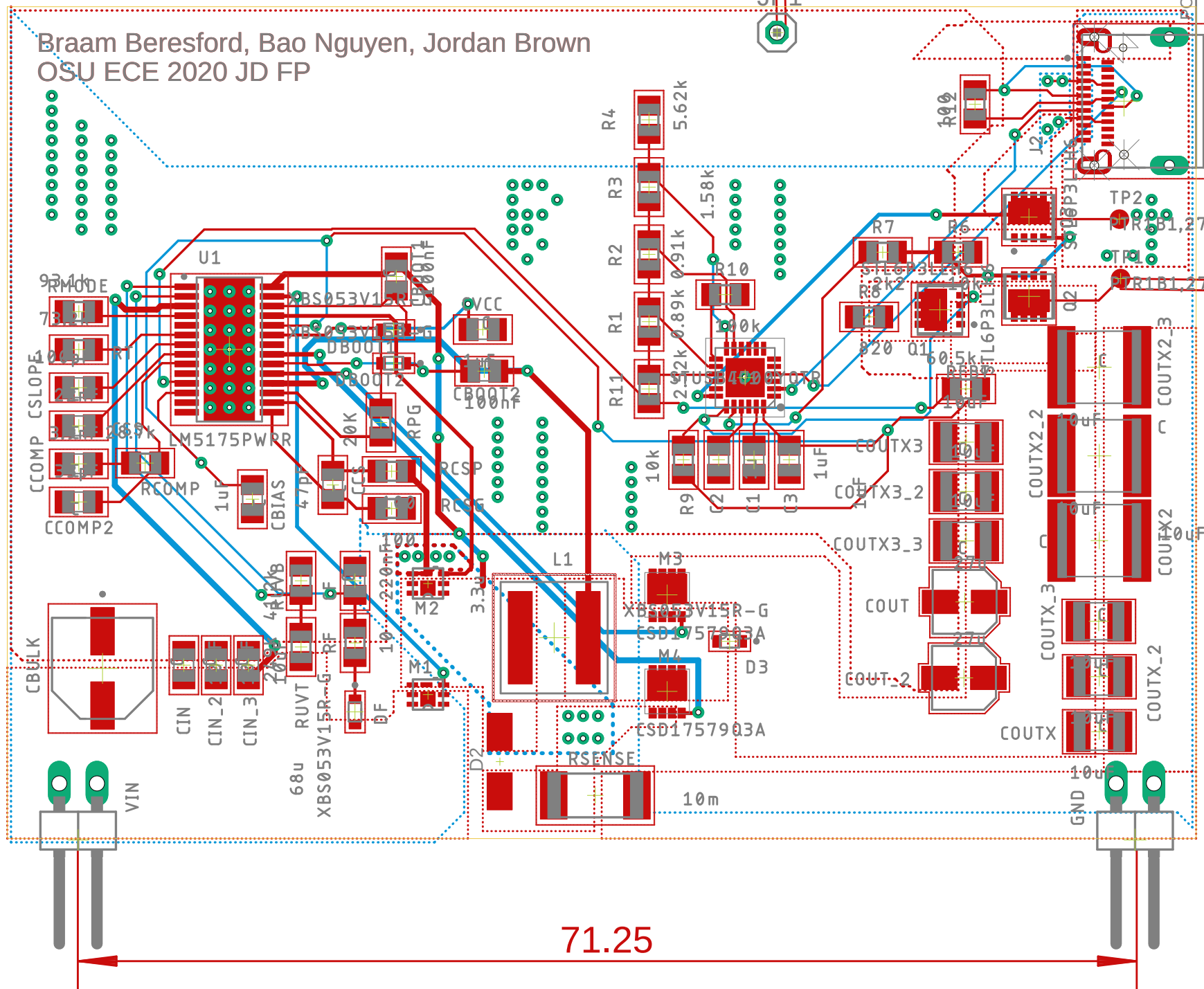


USB-PD BOARD





Braam Beresford, Bao Nguyen, Jordan Brown
OSU ECE 2020 JD FP



ARDUINO CODE


```

/**
 * Junior Design Solar Group 17
 * Arduino code.
 */
#include <Wire.h>
#include <DallasTemperature.h>

// Data wire is plugged into digital pin D6 on the Arduino
#define ONE_WIRE_BUS 6
#define CONSTANT_CHARGING_VOLTAGE 14.2
#define CONSTANT_CHARGING_CURRENT 2
#define FLOATING_BATT_VOLTAGE 13.5

// Setup a oneWire instance to communicate with any OneWire device
OneWire oneWire(ONE_WIRE_BUS);

// Pass oneWire reference to DallasTemperature library
DallasTemperature temperatureSensor(&oneWire);

// Address of temp sensor
uint8_t temperatureSensor1[8] = { 0x28, 0x6A, 0x24, 0xFC, 0x0B, 0x00, 0x00, 0x57 };
//Internal
uint8_t temperatureSensor2[8] = { 0x28, 0x33, 0x09, 0x79, 0x97, 0x15, 0x03, 0xC1 };
//LeadAcid
uint8_t temperatureSensor3[8] = {0x28, 0x00, 0x6B, 0xFD, 0x0B, 0x00, 0x00, 0x52 };
//Lipo

const int solarEnPin      = 2;
const int enLiIonPin      = 3;          // LOW = battery charging is working, HIGH =
Stop charging
const int solarShortPin   = 5;
const int systemTempLEDPin = 8;
const int liIonTempLEDPin = 9;
const int currentInPin    = A0;
const int voltageInPin    = A1;
const int currentOutPin   = A2;
const int voltageBattPin  = A3;
const int currentLoadPin  = A6;

// Global vars

//Open Current
float vOpen;

//Solar panel short current
float iShort;

```

```

//Current voltage from panel
float voltageFromPanel;

//Immediate current from panel
float currentFromPanel;

float currentToSystem;

float internalTemp;
float leadacidTemp;
float liIonTemp;

float voltageToBatt;
float currentToBatt;          //CurrentToSystem - CurrentToLoad
float currentToLoad;          //This is the current of load

int currentPanelTemporary;
int voltageFromPanelTemporary;
int currentOutTemporary;
int voltageToBatteryTemporary;
int currentLoadTemporary;

byte wiper;

typedef enum BattStates {BULK,TRICKLE, FLOATING} BattStates;

void setup() {
  pinMode(solarShortPin, OUTPUT);
  digitalWrite(solarShortPin, LOW); //Default OFF, there is pull down resistor

  pinMode(enLiIonPin, OUTPUT);
  pinMode(solarEnPin, OUTPUT);      //Default ON, there is pull up resistor
  digitalWrite(solarEnPin, HIGH);
  pinMode(currentInPin, INPUT);
  pinMode(currentOutPin, INPUT);
  pinMode(voltageBattPin, INPUT);
  pinMode(currentLoadPin, INPUT);

  // turn on LiIon charger
  digitalWrite(liIonTempLEDPin, LOW);
  digitalWrite(enLiIonPin, LOW);

  // declare global variable

```

```

wiper = 155;    //init

//Initialize Communication - Only need to perform once
Serial.begin(9600);
Wire.begin();
temperatureSensor.begin();
randomSeed(analogRead(0));
}

/*
 * name: changePot()
 * return: void
 * Pretty code - I2C communication to change feedback voltage
 */
void changePot(byte newWiperValue){
    //wiper value should be a number between 0 and 256, because there are 255 positions
    on the pot. The value of the pot can be measured from the wiper to point A or B.
    //Measuring from point A to the wiper will cause the resistance of the pot to
    decrease as the decimal value sent to it increases. Measuring from B to the wiper will
    cause the opposite to happen, but in this code we will
    //be measuring from A to the wiper.
    // In this case, 255 correlates to 119.5 ohms while 0 correlates to 5100. Each step
    of the decimal value increases the resistance by roughly 19.5 ohms.
    //((value/256)*5000) + 100 will calculate the value of the pot based on the
    decimal value sent to it.
    Wire.beginTransmission(44); //44 corresponds to the binary value that will begin the
    transmission to the pot. If AD0 is connected to 5V then 45 would be used.
    Wire.write(byte(0x00)); //the next byte sent doesn't matter, but the first bit
    determines whether to permanently burn the resistance value onto the pot. We want a 0
    here so we can change it again.
    Wire.write(newWiperValue); //the third byte is the decimal value used to determine
    the resistance. This value should be between 0 and 255
    Wire.endTransmission();
    delay(50);
}

/*
 * bulkChangingBangBang()
 * return: void
 * Description: Bumps current until hits 2 amps
 */
void bulkChangingBangBang(){
    //do I need to shut off the power to the battery to take an accurate reading of its
    resting voltage.
    if(currentToBatt > 1.2){

```

```

        wiper+= 1;                //CHECK if this decrease charging voltage;
        changePot(wiper);
    }

    else if (currentToBatt < 0.8){
        wiper -= 1;
        changePot(wiper);
    }
}

/*
 * name: readValues()
 * return: void
 * description: Read in analog values and update to global vars
 */
void determineBattValues(){//returns the state of the circuit and how the battery
should be charged.
    currentPanelTemporary      = analogRead(currentInPin);    //from solar
    voltageFromPanelTemporary  = analogRead(voltageInPin);    //from solar
    currentOutTemporary        = analogRead(currentOutPin);    // Ibatt + Iload
    voltageToBatteryTemporary   = analogRead(voltageBattPin); // Vbatt
    currentLoadTemporary       = analogRead(currentLoadPin); // Iload
    //Current to battery = SolarOutput - Iload

    //scale all of our values back
    // Update Global Vars
    currentFromPanel  = -0.0277*currentPanelTemporary+15.052;
    voltageFromPanel  = 0.981*(voltageFromPanelTemporary * .0526577) + 0.0069-.17;
    currentToSystem   = -0.0284*currentOutTemporary+15.15;
    currentToLoad      = -0.029*currentLoadTemporary+15.38;
    currentToBatt      = currentToSystem - currentToLoad;
    voltageToBatt      = voltageToBatteryTemporary *.01541436 - 0.4 +0.3;
}

BattStates BattState = BULK;
//Handles logic for charging battery
void chargeBatt(){

    //If under charged
    switch(BattState){
        case BULK:
            bulkChangingBangBang();
            if(voltageToBatt > CONSTANT_CHARGING_VOLTAGE){
                BattState = TRICKLE;
            }
            break;

```

```

    case TRICKLE:
        if(currentToBatt <= 0.2 * CONSTANT_CHARGING_CURRENT){
            BattState = FLOATING;
        }
        if(currentToBatt>0.5){
            BattState = BULK;
            wiper = 155;
        }

        break;
    case FLOATING:
        setBattVoltageBangBang(FLOATING_BATT_VOLTAGE);
        if(currentToBatt>0.5){
            BattState = BULK;
        }

        break;
    }
}

/*
 * tempSensingAndShutoff()
 * return: void
 * Read in Temp for all temp sensor and decide if the system need to be shutdown or
not
 * CHECK: Do the system need to recover?
 */
void tempSensingAndShutoff(){
    temperatureSensor.requestTemperatures();

    internalTemp = temperatureSensor.getTempC(temperatureSensor1)-1;
    leadacidTemp = temperatureSensor.getTempC(temperatureSensor2);
    liIonTemp = temperatureSensor.getTempC(temperatureSensor3);
    // overTempMain include internal temperature sensors and lead acid battery
    // these device provide/sink large amount of current
    bool overTempMain = (internalTemp > 40) || (leadacidTemp > 40);
    bool overTempLiIon = liIonTemp > 40;

    if(overTempMain == 1){
        //Over temp LED signal
        digitalWrite(systemTempLEDPin, HIGH);
        //Shut down voltage regulator
        digitalWrite(solarEnPin, LOW);
        digitalWrite(enLiIonPin, HIGH);
        Serial.print("Over Temp: 113F");
    }
}

```

```

    else{
        //Turn off LED iftemp nothing is wrong
        digitalWrite(systemTempLEDPin, LOW);
        //Serial.print("skip broken led");
    }

    if(overTempLiIon == 1){
        Serial.println("Over Temp");
        //turn off charging until reset
        digitalWrite(enLiIonPin, HIGH);
        digitalWrite(liIonTempLEDPin, HIGH);
    }
}

/*
 * shortCurrentOpenVoltage()
 * return: void
 * Turn of regulator to measure Vopen, then short circuit to measure Ishort
 * CHECK: Do the system need to recover?
 */
void shortCurrentOpenVoltage(){ //sends the status of the panel and how much power it
is producing
    //Perform shutdown voltage regulator
    digitalWrite(solarEnPin, LOW);
    delay(1000);

    vOpen = 0.981*(analogRead(voltageInPin) * .0526577) + 0.0069-.17;
    //Turn on short MOSFET

    delay(100);
    digitalWrite(solarShortPin, HIGH);
    delay(100);
    iShort = -0.0277*analogRead(currentInPin)+15.052;
    delay(100);

    //Turn off short MOSFET
    //Turn voltage regulator back on
    digitalWrite(solarShortPin, LOW);
    digitalWrite(solarEnPin, HIGH);
}

void sendShortOpen(){
    Serial.println(vOpen-2.);
    // Serial.println(iShort);
    Serial.println(2.0 + random(12, 77)/100.);
}

```

```

/*
 * sendChargingData()
 * return: void
 * print out to serial battery voltage and current
 */
void sendChargingData(){
    Serial.println(voltageFromPanel+1);
    Serial.println(currentFromPanel);
    Serial.println(voltageFromPanel+1*currentFromPanel); // power from panel
    Serial.println(internalTemp);
    Serial.println(leadacidTemp+1);
    Serial.println(liIonTemp);
    Serial.println(voltageToBatt);
    Serial.println(currentToBatt);
    Serial.println(currentToLoad);

}

/*
 * name: sendSystemInfo()
 * return: void
 * print out to serial requested data Vcurr, Icurr, Pcurr, Vopen, Ishort
 */
void sendSystemInfo(){ //send the voltage and current the battery sees.
    int requestCode = 0;
    if(Serial.available() > 0){
        requestCode = (Serial.read() - '0');
        delay(50);

        if(requestCode == 0){ //Code 0 mean regular charge data Vcurr, Icurr, Pcurr
            sendChargingData();
        }

        else if(requestCode == 1){ //Code 1 mean perform Open-Short test for solar panel
            shortCurrentOpenVoltage();
            sendShortOpen();
        }

        else{ //something was received through serial that wasn't one of those 2 strings.
            // Serial.print(requestCode);
        }
    }
}

```

```

/*
 * name: sendSystemInfo()
 * return: void
 * print out to serial requested data Vcurr, Icurr, Pcurr, Vopen, Ishort
 */
void setBattVoltageBangBang(float target){
    //As wiper increases votage decreases
    if(voltageToBatt > target){
        wiper++;           //CHECK if this decrease charging voltage;
        changePot(wiper);
    }

    else if (voltageToBatt < target){
        wiper--;
        changePot(wiper);
    }
}

void loop() {

    tempSensingAndShutoff();
    determineBattValues();
    sendSystemInfo();
    chargeBatt();
}

```


PYTHON CODE

```

from tkinter import*
import serial
import time
# Solar 17 – Bao Nguyen
# Program will only get V open – I short if button is pressed
# I-curr, V-curr, Pnow will be updated every 5 seconds

# global vOpen, iShort, pCurr, vCurr, iCurr
pCurr = vCurr= iCurr= iShort =vOpen = "NA"

ser=serial.Serial("/dev/ttyACM0", 9600)

# Send serial request to Arduino to get Vcurr, Icurr
# Update global variables, update vCurr, iCurr, pCurr
# Exit function

# Let's create the Tkinter window
window = Tk()
window.config(bg='white')

vOpenText = Label(window, text="V-Open: ", fg="#333", bg="#fff").grid(column=0, row
= 0, sticky = W)      # Not changing so can do .grid at the same time
vOpenDis = Label(window, text="1", fg="#333", bg="#fff")
vOpenDis.grid(column=1, row=0, sticky = W)      #
Changing so need to call .grid seperate

iShortText = Label(window, text="I-Short: ", fg="#333", bg="#fff").grid(column=0,
row = 1, sticky = W)
iShortDis = Label(window, text="2", fg="#333", bg="#fff")
iShortDis.grid(column=1, row=1, sticky = W)

vCurrText = Label(window, text="V-Curr: ", fg="#333", bg="#fff").grid(column=0, row
= 2, sticky = W)
vCurrDis = Label(window, text="3", fg="#333", bg="#fff")
vCurrDis.grid(column=1, row=2, sticky = W)

iCurrText = Label(window, text="I-Curr: ", fg="#333", bg="#fff").grid(column=0, row
= 3, sticky = W)
iCurrDis = Label(window, text="4", fg="#333", bg="#fff")
iCurrDis.grid(column=1, row=3, sticky = W)

pCurrText = Label(window, text="P-now: ", fg="#333", bg="#fff").grid(column=0, row
= 4, sticky = W)
pCurrDis = Label(window, text="5", fg="#333", bg="#fff")
pCurrDis.grid(column=1, row=4, sticky = W)

# Reuse Label multiple times
vSymbol = lambda:Label(window, text="V", fg="#333", bg="#fff")

```

```

vSymbol().grid(column=3, row=0)
vSymbol().grid(column=3, row=2)

iSymbol = lambda:Label(window, text="I", fg="#333", bg="#fff")
iSymbol().grid(column=3, row=1)
iSymbol().grid(column=3, row=3)

pSymbol = Label(window, text="W", fg="#333", bg="#fff").grid(column=3, row=4)

def startSerial():
    time.sleep(2)

def updateData():
    global vOpen, iShort, pCurr, vCurr, iCurr
    #global counter #debug
    #counter +=1 #debug
    #vCurrDis.configure(text=str(counter)) #debug
    vCurrDis.configure(text="...")
    iCurrDis.configure(text="...")
    pCurrDis.configure(text="...")

    ser.write(b'\0')
    vCurr = ser.readline().decode("utf-8").rstrip("\r\n")
    iCurr = ser.readline().decode("utf-8").rstrip("\r\n")
    pCurr = ser.readline().decode("utf-8").rstrip("\r\n")

    vCurrDis.configure(text=str(vCurr))
    iCurrDis.configure(text=str(iCurr))
    pCurrDis.configure(text=str(pCurr))

    print("HELLO")

    # each after call would create it own infinity 500ms delay loop (need to find
    another way)
    window.after(1000, updateData) # seem to be asynchronous

# Send serial request to Arduino if button is pressed
# to get Open Voltage and Short Circuit data
# Display "Waiting" until data is received, update vOpen, reading a byte from
serialiShort
# Exit function
def getShortOpen():
    print("HI")
    global vOpen, iShort, pCurr, vCurr, iCurr
    # Null all data for pretty
    # vOpenDis.configure(text="..." )
    # iShortDis.configure(text="...")
    # vCurrDis.configure(text="...")

```

```

# iCurrDis.configure(text="...")
# pCurrDis.configure(text="...")

#ser.write("shortAndOpen\r".encode())
ser.write(b'1')

vOpen = ser.readline().decode("utf-8").rstrip("\r\n")
iShort = ser.readline().decode("utf-8").rstrip("\r\n")

vOpenDis.configure(text=str(vOpen))
iShortDis.configure(text=str(iShort))
vCurrDis.configure(text=str(vCurr))
iCurrDis.configure(text=str(iCurr))
pCurrDis.configure(text=str(pCurr))

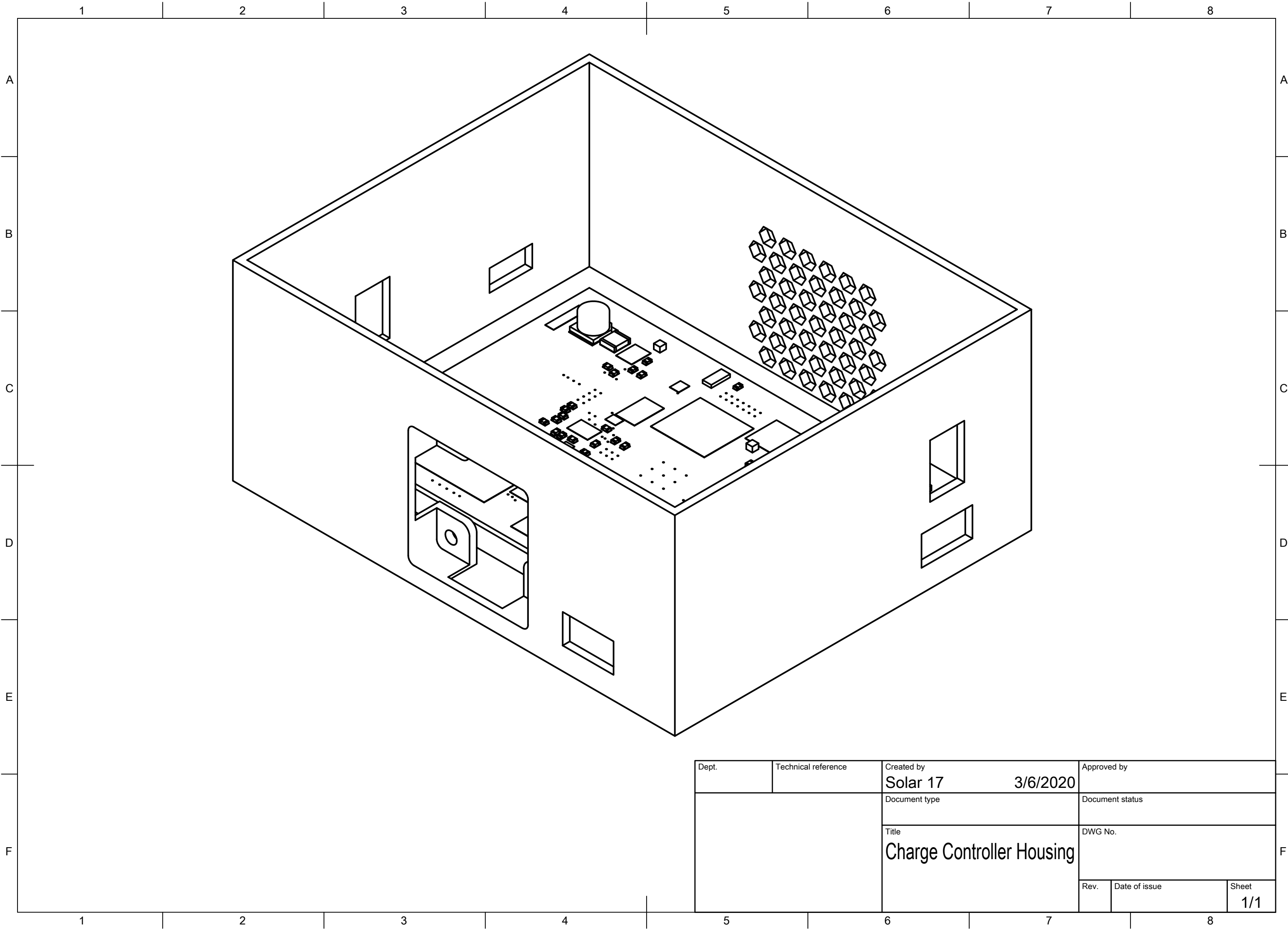
#window.after(0, getShortOpen2)
#iShort = ser.readline().decode("utf-8").rstrip("\r\n")

btn = Button(window, text="Reset Solar Info", width = 15, height= 2,
command=getShortOpen, border="0", bg="#E85B2A", activebackground = "#fff",
activeforeground = "#E85B2A", fg = "#fff", font=('Helvetica', '10'))
btn.grid(column=0, columnspan=2, row = 5, padx=12, sticky = EW)

window.after(0,startSerial)
window.after(2,updateData)
window.title("Solar Charger 1.0")
window.geometry('350x200')
window.mainloop()

```

MECHANICAL DRAWINGS



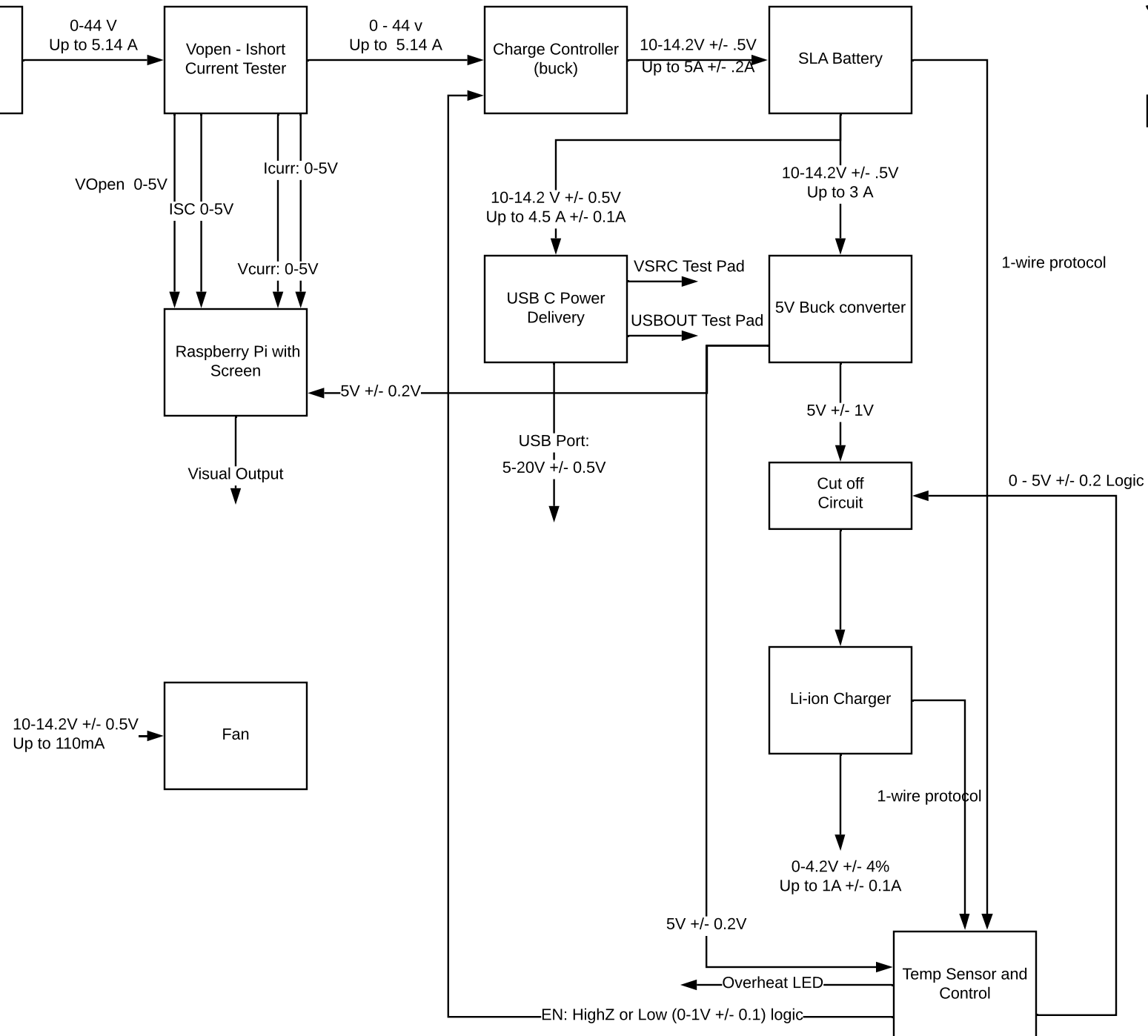
Dept.	Technical reference	Created by Solar 17	3/6/2020		Approved by
		Document type		Document status	
		Title Charge Controller Housing		DWG No.	
		Rev.	Date of issue		Sheet 1/1

MECHANICAL DRAWINGS

BLOCK DIAGRAM

Solar 17 Block Diagram

Braam Beresford
Jordan Brown
Bao Nguyen



INTERFACES

Interface Definition for Junior Design 2 FP, Solar Battery

Solar Panel	<ul style="list-style-type: none"> • To Open/Short/Power measurement module • Voltage: 0-44V • Current: Up to 5.14A • MC4 connector
Open/Short/Power measurement	<ul style="list-style-type: none"> • To raspberry pi: <ul style="list-style-type: none"> ◦ Vopen: up to 5V ◦ Isc: up to 5V ◦ Vcurr: up to 5V ◦ Icurr: up to 5V • To charge controller (buck): <ul style="list-style-type: none"> ◦ Voltage: 0-44V ◦ Current: Up to 5.14A
Raspberry Pi	<ul style="list-style-type: none"> • Input: From Open/Short/Power measurement <ul style="list-style-type: none"> ◦ Vopen: up to 5V ◦ Isc: up to 5V ◦ Vcurr: up to 5V ◦ Icurr: up to 5V ◦ All negligible currents • Input: Power: <ul style="list-style-type: none"> ◦ 5V +/- 0.2V ◦ Up to 2.5A draw • Input: Button: <ul style="list-style-type: none"> ◦ Mechanical button to flip through info • Output: LCD Display: <ul style="list-style-type: none"> ◦ Current Power draw (Up to 5V) ◦ Open voltage (Up to 5V) ◦ Short circuit current (Up to 5V) ◦ With 5% accuracy
Charge Controller, Buck	<ul style="list-style-type: none"> • Input: Open/Short/Power measurement: <ul style="list-style-type: none"> ◦ 0 to 44V ◦ Up to 5.14A • ENable: High Impedance (HighZ) or LOW logic (0-0.5V) from Temperature Sensor and Control • Output: SLA battery: <ul style="list-style-type: none"> ◦ 10-14.2V +/- 0.5V ◦ Up to 5A +/- 0.2A • Test points: <ul style="list-style-type: none"> ◦ 1.32mm +/- 0.1mm via diameter for Vin ◦ 1.32mm +/- 0.1mm via diameter for Vout
USB PD Power Delivery	<ul style="list-style-type: none"> • Input: SLA Battery power: <ul style="list-style-type: none"> ◦ 10-14.2V +/- 0.5V

	<ul style="list-style-type: none"> ○ Up to 4.5A +/- 0.1A ● Output: USB Port: <ul style="list-style-type: none"> ○ 5-20V +/- 0.5V ● Output: GND Via: <ul style="list-style-type: none"> ○ Must be at least 1mm in diameter. Allow ground witch hat to attach ● Output: VSRC voltage Test Pad: <ul style="list-style-type: none"> ○ Test pad must be at least 1.2mm in diameter. Allow testing of VSRC voltage ● Output: USBOUT voltage Test Pad: <ul style="list-style-type: none"> ○ Test pad must be at least 1.2mm in diameter. Allow testing of USBOUT voltage
Lead Acid Battery	<ul style="list-style-type: none"> ● Input: Charge controller: <ul style="list-style-type: none"> ○ 10-14.2V +/- 0.1V ○ Up to 2A +/- 0.2A ● Output: 5V buck: <ul style="list-style-type: none"> ○ 10-14.2V +/- 0.1V ○ Up to 3A ● Output: USB Power Regulator: <ul style="list-style-type: none"> ○ 10-14.2V +/- 0.1V ○ Up to 1A +/- 0.1A
5V Buck converter	<ul style="list-style-type: none"> ● Input: Power from SLA: <ul style="list-style-type: none"> ○ 10-14.2V +/- 0.1V ○ Up to 2A ● Output: Raspberry pi <ul style="list-style-type: none"> ○ 5V +/- 1V ○ Up to 1A ● Output: Cut off: <ul style="list-style-type: none"> ○ 5V +/- 1V ○ Up to 1.5A ● Output: Temp sensor: <ul style="list-style-type: none"> ○ 5V +/- 0.25V ○ Minimal current
Cut off circuit	<ul style="list-style-type: none"> ● Input: Power 5V +/- 2% up to 2A ● Input: Control Signal from uController ● Output: Power 5V +/- 0.25V up to 2A
Li-ion Charger	<ul style="list-style-type: none"> ● Input power: 5V +/- 2% up to 1100mA ● Output: up to 4.2V +/- 4%, up to 1100mA ● Output LED: Red and Blue for charging status
Temperature Sensor and Control	<ul style="list-style-type: none"> ● Input power: 5V +/- 2% up to 100mA ● 1-Wire data (0-5V logic) from sensors-Serial +/- 1°C ● Output: LEDs for over temp indicator 0 to 5V +/- .1A

	<ul style="list-style-type: none">• 0-20mA to LEDs• Power to sensors 5V +/- .1V, 0-.5mA
Fan	<ul style="list-style-type: none">• Input power: 10-14.2V and up to 110mA

BILL OF MATERIALS

Part	Link	Part Number	Quantity Needed	Unit Price	Ind. Price (\$)	Quantity to Order	Description
Cboot	Digikey	C2012COG1H822K060AA	1	0.25	0.25	3	Cap: 8.2 nF Total Derated Cap: 8.2 nF VDC: 50 V ESR: 0 Ω Package: 0805
Ccomp	Digikey	CC0805KRX7R9BB821	1	0.15	0.15	3	Cap: 820 pF Total Derated Cap: 820 pF VDC: 50 V ESR: 1 m Ω Package: 0805
Cdthr	Digikey	C0805C104M3RACTU	1	0.1	0.1	3	Cap: 100 nF Total Derated Cap: 100 nF VDC: 25 V ESR: 0 Ω Package: 0805
Chf	Digikey	CC0805JRNPO9BN100	1	0.11	0.11	3	Cap: 10 pF Total Derated Cap: 10 pF VDC: 50 V ESR: 0 Ω Package: 0805
Cin	Digikey	CGA9N3X7S2A106K230KB	1	2.71	2.71	2	Cap: 10uF 100V
Cin1	Digikey	EEE-2AA330P	1	0.62	0.62		Cap: 33 uF VDC 100 V
Cout Cout2 C9	Digikey	355VPF82M	3	1.83	5.49	6	Cap: 82 μ F Total Derated Cap: 160 μ F VDC: 35 V ESR: 20 m Ω Package: 8x12
Cout1	Digikey	C0805C225K3PAC7800	1	0.3	0.3	3	2.2uF, 25v Rating
Cramp	Digikey	CC0805KRX7R9BB471	1	0.11	0.11	3	Cap: 470 pF Total Derated Cap: 470 pF VDC: 50 V ESR: 1 m Ω Package: 0805
Css	Digikey	08055C103KAT2A	1	0.13	0.13	3	Cap: 10 nF Total Derated Cap: 10 nF VDC: 50 V ESR: 78 m Ω Package: 0805
Cvcc	Digikey	C0805C224K5RACTU	1	0.2	0.2	3	Cap: 220 nF Total Derated Cap: 220 nF VDC: 50 V ESR: 46 m Ω Package: 0805
D1	Digikey	SBRD10200TR	1	0.49	0.49	2	Type: Schottky VRRM: 200 V Io: 10 A
L1	Digikey	IHLP6767GZER8R2M01	1	5.87	5.87	2	L: 8.2 μ H DCR: 10.1 m Ω IDC: 18 A
M1	Digikey	CSD18543Q3A	1	0.84	0.84	2	VdsMax: 60 V IdsMax: 35 Amps
Rfbt	Digikey	ERA-6AEB303V	1	0.38	0.38	3	Resistance: 30 k Ω Tolerance: 0.1% Power: 125 mW
Rramp	Digikey	RC0805FR-07200KL	1	0.1	0.1	3	Resistance: 200 k Ω Tolerance: 1.0% Power: 100 mW
Rcomp	Digikey	CRCW060369K8FKEA	1	0.1	0.1	3	Resistance: 69.8 k Ω Tolerance: 1.0% Power: 100 mW
Rsns	Digikey	PRL1632-R009-F-T1	1	0.2	0.2	3	Resistance: 9 m Ω Tolerance: 1.0% Power: 1 W
Rt	Digikey	ERJ-6ENF1072V	1	0.1	0.1	3	Resistance: 10.7 k Ω Tolerance: 1.0% Power: 100 mW
Ruv1	Digikey	ERJ-6ENF4021V	1	0.1	0.1	3	Resistance: 4.02 k Ω Tolerance: 1.0% Power: 125 mW
Ruv2	Digikey	CRCW080549K9FKEA	1	0.1	0.1	3	Resistance: 49.9 k Ω Tolerance: 1.0% Power: 125 mW
R2 R3 R5 R6	Digikey	RC0805FR-0710KL	5	0.1	0.5	10	Resistance: 10 k Ω Tolerance: 1.0% Power: 125 mW
MOS-Short	Digikey	IRL540PBF	1	1.88	1.88	2	N-Channel 100V 28A (Tc) 150W (Tc) Through Hole TO-220AB
Lipo-charge	Digikey	MCP73831T-2ACI/OT	1	0.56	0.56	2	SOT23-5
R1, R7	Digikey	311-2.00KCRCT-ND	2	0.1	0.2	5	2k
LiPocap1 LiPocap2	Digikey	399-8012-1-ND	2	0.19	0.38	5	10uF 16V
LEDs	Digikey	LTST-C171GKT	1	0.28	0.28	3	0805 SMD LED
Pmos	Digikey	497-13537-1-ND	1	0.74	0.74	2	STN3P6F6
U1 U2 U4	Digikey	ACS711KLCTR-12AB-T	3	3.04	9.12	5	Current sensor
U\$1	Digikey	LM25088MH-1/NOPB	1	3.5	3.5	2	Buck converter, external switch
TB1	Ebay	Arduino Nano	1	3.95	3.95		
C1 C2 C3 C4 C5 C6	Digikey	C0805C104M3RACTU	10	0.1	1	25	Cap: 100 nF Total Derated Cap: 100 nF VDC: 25 V ESR: 0 Ω Package: 0805
Digipot	Digikey	AD5245BRJZ5-RL7	1	2.01	2.01	2	Digital Potentialmeter I2C 5k 256 taps
Rvin2	Digikey	RNCP0805FTD49K9	1	0.1	0.1	3	49.9k 1% 1/4W
Rvin1, Rvout1, Rvout2	Digikey	CRG0805F5K1	3	0.1	0.3	8	5.1k 1% 1/8W
Rvout2	Digikey	CRG0805F11K	1	0.1	0.1	3	11k 1% 1/8W
10A Rocker Switch	Digikey	KRE2ANA1BBD	2	2.82	5.64	2	
	Resistore		1		1 N/A		
ExternalTemp	Ebay	DS18B20	1		3.99	2	External for battery temperature measurement

InternalTemp	Digikey	Ds18B20	2	3.34	6.68	4	
5V buck	Ebay		1		5.75	N/A	Power regulator for arduino, sensors, and battery charger
D2	Digikey	B220A-13-F	1	0.46	0.46	3	Value: B220A-13-F Package: B220A-13-F
RF	Digikey	CRCW080510R0FKEAC	1	0.1	0.1	3	Value: 10 Package: 0805
R12, RCSG, RC	Digikey	CRCW0805100RFKEAC	3	0.1	0.3	8	Value: 100 Package: 0805
R10	Digikey	ASC0805-100KF1	1	0.29	0.29	3	Value: 100k Package: 0805
CBOOT1,CBOO	Digikey	08053C104JAZ2A	2	0.38	0.76	5	Value: 100nF Package: 0805
CSLOPE	Digikey	GRM2165C2A101JA01D	1	0.21	0.21	3	Value: 100pF Package: 0805
R6, R9	Digikey	RC0805FR-0710KL	2	0.1	0.2	5	Value: 10k Package: 0805
RSense	Digikey	PMR100HZPFU10L0	1	0.56	0.56	3	Value: 10m Package: 2512
CIN, CIN_2, CIN	Digikey	C2012X5R1V685K125AC	3	0.77	2.31	7	Value: 10uF Package: 0805
COUTX, COUTX	Digikey	GRM32ER71H106MA12	5	1.06	5.3	12	Value: 10uF Package: 1210
COUTX2, COUT	Digikey	C5750X7S2A106M230KB	3	2.58	7.74	3	Value: 10uF Package: 2220
COUTX3	Digikey	GRM32ER71H106MA12	1	1.06	1.06	3	Value: 10uF Package: 1210
R11	Digikey	RT0805BRD0713KL	1	0.42	0.42	3	Value: 12.99k Package: 0805
C1,C2,C3,CBIAS	Digikey	GMK212B7105KG-T	4	0.24	0.96	10	Value: 1uF Package: 0805
CVCC	Digikey	C0805C105K4RACTU	1	0.15	0.15	3	Value: 1uF Package: 0805
RFBT	Digikey	ERJ-6ENF4753V	1	0.1	0.1	3	Value: 200k Package: 0805
RPG	Digikey	CRCW080520K0FKEAHF	1	0.25	0.25	3	Value: 20k Package: 0805
CF	Digikey	GRM21BR71H224KA01L	1	0.33	0.33	3	Value: 220nF Package: 0805
CSS	Digikey	CGA4J2C0G1H223J125A	1	0.35	0.35	3	Value: 22nF Package: 0805
RUVT	Digikey	CRCW0805249KFKEA	1	0.1	0.1	3	Value: 249k Package: 0805
COUT, COUT_2	Digikey	25SVPF27MX	2	1.02	2.04	5	Value: 27uF Package: custom
RCOMP	Digikey	CRCW080528K0FKEA	1	0.1	0.1	3	Value: 28.7k Package: 0805
R7	Digikey	RC0805FR-072K2L	1	0.1	0.1	3	Value: 2.2K Package: 0805
CCOMP	Digikey	CGA4C2C0G1H392J060A	1	0.25	0.25	3	Value: 3.9nF Package: 0805
R4	Digikey	ERA-6AEB3322V	1	0.36	0.36	3	Value: 33.19k Package: 0805
CCOMP2	Digikey	CL21C330JBANNNC	1	0.1	0.1	3	Value: 33pF Package: 0805
R1	Digikey	RC0805FR-074K7L	1	0.1	0.1	3	Value: 4.71k Package: 0805
R2	Digikey	ERA-6AEB4991V	1	0.36	0.36	3	Value: 4.93k (acutally 4.99k) Package: 0805
RUVB	Digikey	CRCW080541K2FKEA	1	0.1	0.1	3	Value: 41.2k Package: 0805
Ccs	Digikey	GQM2195C1H470JB01D	1	1.18	1.18	3	Value: 47pF Package: 0805
RT	Digikey	CRCW080573K2FKEA	1	0.1	0.1	3	Value: 73.2k Package: 0805
R3	Digikey	RNCF0805BTE8K76	1	0.4	0.4	3	Value: 8.73k (8.76k) Package: 0805
R8	Digikey	ERA-6AEB821V	1	0.36	0.36	3	Value: 820 Package: 0805
RMODE	Digikey	CRCW080593K1FKEA	1		0	3	Value: 93.1k Package: 0805
M1,M4	Digikey	CSD17571Q2	2	0.51	1.02	5	Value: CSD17571Q2 Package: CSD17571Q2
M3, M4	Digikey		2	0.56	1.12	5	Value: CSD17579Q3A Package: CSD17579Q3A
USB C Port (J2)	Digikey	DX07S024JJ3R1300	1	2.52	2.52	2	Value: DX07S024JJ3R1300 Package: DX07S024JJ3R1300
CBULK	Digikey	EEE-FK1E680P	1	0.49	0.49	2	Value: EEE-FK1E680P Package: EEE-FK1E680P

U1	Digikey	LM5175	1	6.69	6.69	2	Value: LM5175 Package: LM5175
Q1,Q2,Q3	Digikey		3	1.5	4.5	7	Value: STL6P3LLH6 Package: STL6P3LLH6
USB PD Controll	Digikey	STUSB4700QTR	1	3.58	3.58	2	Value: STUSB4700QTR Package:
L1	Mouser	XAL7070-332MEB	1	3.14	3.14	2	Value: XAL7070-332MEB Package: 3.3uH
DBOOT1, DBOC	Mouser	XBS053V15R-G	4	0.5	2	10	Value: XBS053V15R-G Package: XBS053V15R-G

TIMESHEET

Date	Time In	Time Out	Time Spent (Hours)	Members	Task	
12/20	10:00 AM	12:00 PM	2	All	Research	
12/23	1:00 PM	2:00 PM	1	All	Research	
12/28	2:00 PM	4:00 PM	2	All	Research	
1/2	6:00 PM	9:00 PM	3	All	Research	
1/5	2:00 PM	4:00 PM	2	All	Research	
1/7	1:00 PM	4:00 PM	3	All	Research & development	
1/9	10:00 AM	2:00 PM	4	All	Research & development	
1/11	9:00 AM	10:00 AM	1	ALL	Research & development	
1/12	1:00 PM	4:00 PM	3	ALL	Research & development	
1/15	4:00 PM	7:00 PM	3	ALL	Research & development	
1/18	2:00 PM	4:00 PM	2	ALL	Research & development	
1/20	5:00 PM	7:00 PM	2	ALL	Research & development	
1/22	4:00 PM	8:00 PM	4	ALL	Research & development	
1/27	2:00 PM	6:00 PM	4	ALL	Research & development	
1/29	3:00 PM	4:00 PM	1	ALL	Laying PCB	
1/30	4:00 PM	6:00 PM	2	ALL	Laying PCB	
2/2	6:00 PM	7:00 PM	1	ALL	Laying PCB	
2/5	4:00 PM	7:00 PM	3	ALL	Laying PCB	
2/6	5:00 PM	8:00 PM	3	ALL	Laying PCB	
2/9	6:00 PM	8:00 PM	2	ALL	Laying PCB	
2/10	5:00 PM	7:00 PM	2	ALL	Ordering	
2/15	6:00 PM	7:00 PM	1	ALL	Design	
2/18	10:00 AM	4:00 PM	6	ALL	Design	
2/20	12:00 PM	2:00 PM	2	ALL	Design	
2/21	11:00 AM	5:00 PM	6	ALL	Assembly	
2/24	3:00 PM	7:00 PM	4	Bao and Braam	Assembly	
2/26	5:00 PM	7:00 PM	2	ALL	Troubleshooting	
2/28	6:00 PM	8:00 PM	2	ALL	Troubleshooting	
3/1	1:00 PM	5:00 PM	4	ALL	Troubleshooting	
3/3	5:00 PM	7:00 PM	2	ALL	Code - Charge Controller	
3/4	12:00 PM	5:00 PM	5	ALL	Code - Charge Controller	
3/5	11:00 AM	3:00 PM	4	ALL	Code - Charge Controller	
3/6	3:00 PM	7:00 PM	4	ALL	Code - Charge Controller	
3/7	2:00 PM	7:00 PM	5	ALL	Code - Charge Controller / Troubleshooting	
3/8	1:00 PM	8:00 PM	7	ALL	Code - Charge Controller / Troubleshooting	
3/9	2:00 PM	4:00 AM	2	ALL	Code - Charge Controller / Troubleshooting	