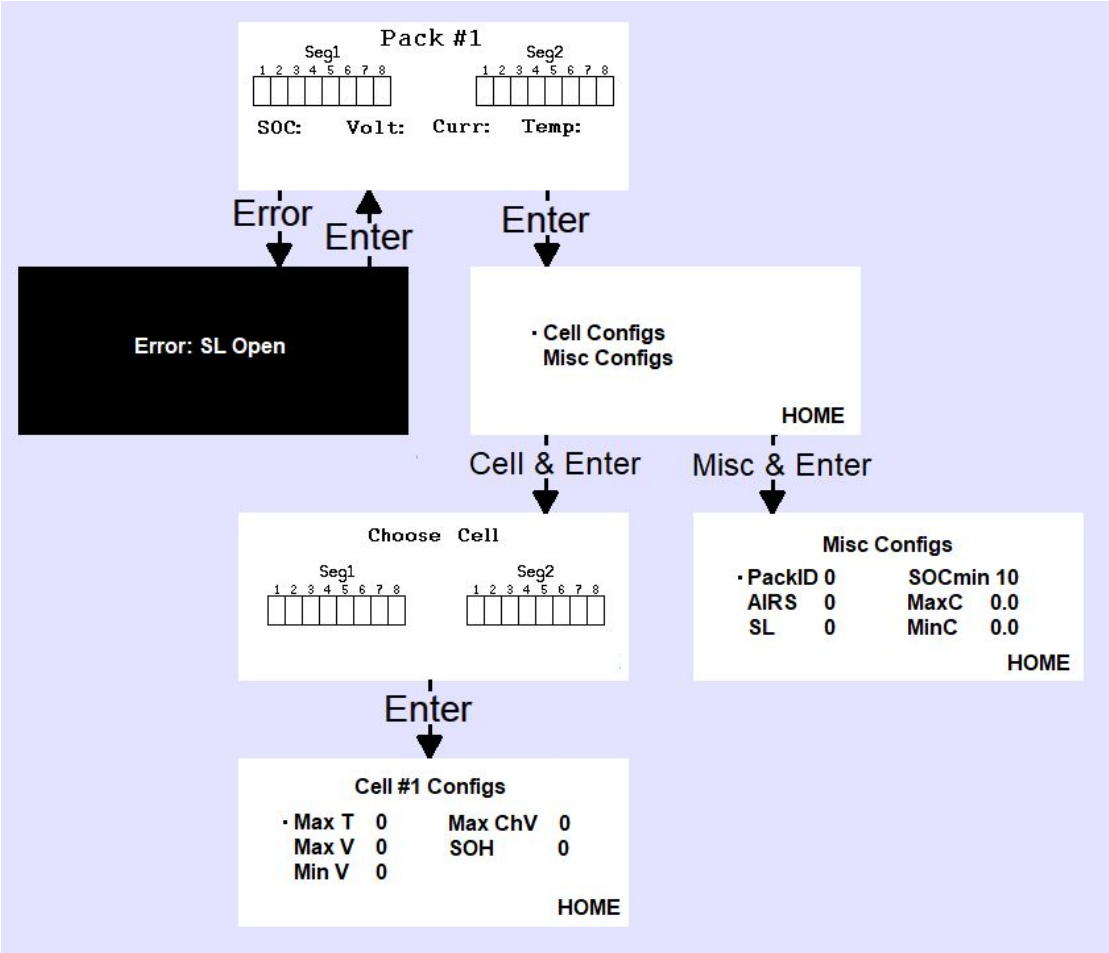
TSV Firmware CDR

November 20, 2019, 5:30pm, ECE 400

Proposed Design:



Minutes:

- The design review consisted of Simone demonstrating how to navigate through the display and then allowing Professor Nadovich to go through it on his own.
- Change initialization for the choose cell screen so it starts at zero
- Have confirmation screen to change configurations
- Make all configurations the same type (float)
    - Create a for loop that goes through an array with name, value (type)
- Add arrows underneath the cell and have it automatically go to a bad cell/cell with an error
- Change format to choose cell, view or configure it
- Make sure to include charging option
- Change pack ID number to CAN Address

Software:

```
//Display Firmware
//Simone Khalifa


#include "GxGDEH029A1/GxGDEH029A1.cpp"
#include "GxIO/GxIO_SPI/GxIO_SPI.cpp"
#include "GxIO/GxIO.cpp"
#include "BitmapGraphics.h"

#include "Fonts/FreeSansBold24pt7b.h"
#include "Fonts/FreeSansBold12pt7b.h"
#include "Fonts/FreeSansBold9pt7b.h"


GxIO_Class io(SPI, 15, 22, 21);
GxEPD_Class display(io, 21, 16);

#define CENTER_BUTTON 17
#define UP_BUTTON 18
#define DOWN_BUTTON 19
#define LEFT_BUTTON 25
#define RIGHT_BUTTON 26

void setup() {

  Serial.begin(9600);
  display.init();

  setUpMain(true); //id

  const GFXfont* f = &FreeSansBold9pt7b;
  display.setFont(f);
  display.setTextColor(GxEPD_BLACK);

  pinMode(CENTER_BUTTON, INPUT); //button
  pinMode(UP_BUTTON, INPUT); //button
  pinMode(DOWN_BUTTON, INPUT); //button
  pinMode(LEFT_BUTTON, INPUT); //button
  pinMode(RIGHT_BUTTON, INPUT); //button
  attachInterrupt(digitalPinToInterrupt(CENTER_BUTTON), CButton, RISING);
  attachInterrupt(digitalPinToInterrupt(UP_BUTTON), UButton, RISING);
  attachInterrupt(digitalPinToInterrupt(DOWN_BUTTON), DButton, RISING);
  attachInterrupt(digitalPinToInterrupt(LEFT_BUTTON), LButton, RISING);
  attachInterrupt(digitalPinToInterrupt(RIGHT_BUTTON), RButton, RISING);
}
```

```
//misc configs
boolean id; boolean sl; int soc=50;; float max_pc; float min_pc; //need to set
the equal to inputs but give defaults

void loop() {

  voltage1 = voltage1 + 1; //input from core1
  current1 = current1 + 1; //input from core1
  temp1 = temp1 + 1;  //input from core1
  soc_test = soc_test - 1;

  fsm();

  delay(5000);
}

boolean centerPress = false;
boolean leftPress = false;
boolean rightPress = false;
boolean upPress = false;
boolean downPress = false;
uint8_t dbDelay = 1000;

void CButton() //interrupt with debounce
{
  volatile static unsigned long last_interrupt_time = 0;
  unsigned long interrupt_time = millis();
  if (interrupt_time - last_interrupt_time > dbDelay) { centerPress = true;
Serial.println("c"); }
  else {centerPress = false;}
  last_interrupt_time = interrupt_time;
}

void LButton() //interrupt with debounce
{
  volatile static unsigned long last_interrupt_time = 0;
  unsigned long interrupt_time = millis();
  if (interrupt_time - last_interrupt_time > dbDelay) { leftPress = true;
Serial.println("l");}
  else {leftPress = false;}
  last_interrupt_time = interrupt_time;
}

void RButton() //interrupt with debounce
{
  volatile static unsigned long last_interrupt_time = 0;
  unsigned long interrupt_time = millis();
```

```
  if (interrupt_time - last_interrupt_time > dbDelay) { rightPress = true;
Serial.println("r");}
  else {rightPress = false;}
  last_interrupt_time = interrupt_time;
}

void UButton() //interrupt with debounce
{
  volatile static unsigned long last_interrupt_time = 0;
  unsigned long interrupt_time = millis();
  if (interrupt_time - last_interrupt_time > dbDelay) { upPress = true;
Serial.println("u");}
  else {upPress = false;}
  last_interrupt_time = interrupt_time;
}

void DButton() //interrupt with debounce
{
  volatile static unsigned long last_interrupt_time = 0;
  unsigned long interrupt_time = millis();
  if (interrupt_time - last_interrupt_time > dbDelay) { downPress = true;
Serial.println("d");}
  else {downPress = false;}
  last_interrupt_time = interrupt_time;
}

float voltage1 = 0;  //to be removed
float current1 = 0;   //to be removed
float temp1 = 0;      //to be removed
uint16_t soc_test = 100;  //to be remove

  typedef struct
  {
    int max_temp;
    long max_voltage;
    long min_voltage;
    long max_charge_voltage;
    boolean SOH; //true = good cell, false = bad cell--> if SOH is false, no
longer allow cell to cause faults
  } Configs;


#define NUM_CELLS 16
Configs configs[NUM_CELLS];

void defineCellConfigs(int maxTemp, long maxV, long minV, long maxCV, boolean
soh, int index) //pretty much set I think
{
```

```
  configs[index].max_temp = maxTemp; //=input[]
  configs[index].max_voltage = maxV;
  configs[index].min_voltage = minV;
  configs[index].max_charge_voltage = maxCV;
  configs[index].SOH = soh;
}

  typedef struct
  {
    boolean pack_id; //true = Pack 1, false = Pack 2  //change all to floats
    boolean airs_state; //true = closed, false = open
    boolean sl_state; //true = closed, false = open
    uint16_t SOC_min;
    float max_pack_current;
    float min_pack_current;
  } Misc_Configs;

Misc_Configs misc_configs[1] = {{id, airs, sl, soc, max_pc, min_pc}};


void setUpMain(boolean id) {
  display.setRotation(0);
  if (id) {
    display.drawExampleBitmap(gImage_main1, 0, 0, 128, 296, GxEPD_BLACK);
//true = Pack 1
  }
  else {
    display.drawExampleBitmap(gImage_main2, 0, 0, 128, 296, GxEPD_BLACK);
//false = Pack 2
  }
  display.update();
  display.update();
  display.setRotation(45);
}

void mainPartialUpdate(float temperature, uint16_t soc, float volt, float curr)
{
  const GFXfont* f = &FreeSansBold9pt7b;  //set font
  display.setFont(f);
  display.setTextColor(GxEPD_BLACK);

  String temp = String(String(temperature, 1) + " C"); //convert to strings
  String voltage = String(String(volt, 1) + " V");
  String current = String(String(curr, 1) + " A");
  String SOC = String(String(soc, DEC) + " %");

  uint16_t box_x = 15;  //set update window
  uint16_t box_y = 110;
```

```
  uint16_t box_w = 296 - box_x * 2;
  uint16_t box_h = 20;
  uint16_t indent = box_w / 4;

  display.fillRect(box_x, box_y - box_h, box_w, box_h, GxEPD_WHITE);

  display.setCursor(box_x, box_y);  //print SOC
  display.print(SOC);

  display.setCursor(box_x + indent, box_y); //print V
  display.print(voltage);

  display.setCursor(box_x + 2 * indent, box_y); //print current
  display.print(current);

  display.setCursor(box_x + 3 * indent, box_y); //print temp
  display.print(temp);

  display.updateWindow(128 - box_y, box_x, box_h, box_w, false);
}


void cellPartialUpdate(int errorType, int cellNum)
{
  uint16_t box_w = 14;
  uint16_t box_h = 23;
  uint16_t box_y = 52;
  uint16_t box_x = 0;

  cellNum = cellNum - 1; //start index at 1

  if (cellNum < 8) { //seg1
    box_x = 11 + (box_w - 1) * cellNum;
  }
  else { //seg2
    box_x = 180 + (box_w - 1) * cellNum;
  }

  //seg variables
  if (errorType == 1) { //temp
    display.fillRect(box_x, box_y - box_h, box_w, box_h, GxEPD_BLACK);
    display.setTextColor(GxEPD_WHITE);
    display.setCursor(box_x + 1, box_y - 6);
    display.print("V");
  }
  else if (errorType == 2) { //soh bad
    display.fillRect(box_x, box_y - box_h, box_w, box_h, GxEPD_BLACK);
  }
```

```
  else if (errorType == 3) { //high voltage
    display.fillRect(box_x, box_y - box_h, box_w, box_h, GxEPD_BLACK);
    display.setTextColor(GxEPD_WHITE);
    display.setCursor(box_x + 1, box_y - 6);
    display.print("T");
  }

  display.updateWindow(128 - box_y, box_x, box_h, box_w, false);
}

void faults(int errorType)
{
  const GFXfont* font = &FreeSansBold24pt7b;
  display.setFont(font);

  display.fillScreen(GxEPD_BLACK);
  display.setTextColor(GxEPD_WHITE);
  display.setCursor(30, 75);

  if (errorType == 1) { //SL Open
    display.print(" SL Open ");
  }
  else if (errorType == 2) { //Airs Open
    display.print("AIRS Open");
  }
  else if (errorType == 3) { //Dangerous temp
    display.print("High Temp");
  }
  else if (errorType == 4) { //Dangerous voltage
    display.print("High Volt");
  }
  else if (errorType == 5) { //Low SOC
    display.print("Low SOC");
  }
  display.update();
}

void mainConfigScreen()
{
  const GFXfont* font = &FreeSansBold12pt7b;
  display.setFont(font);

  display.fillScreen(GxEPD_WHITE);
  display.setTextColor(GxEPD_BLACK);
  display.setCursor(30, 50);
  display.print("Cell Configs");
  display.setCursor(30, 80);
  display.print("Misc Configs");
```

```
  display.fillRect(20, 41, 4, 4, GxEPD_BLACK);
  display.updateWindow(5, 5, 118, 286, false);
}

void configPartial(boolean index) {
  int top = 45;
  int bottom = 75;
  int older;
  int newer;
  if (index) {
    newer = top;  //choose cell
    older = bottom;
  }
  else {
    newer = bottom;  //choose misc
    older = top;
  }
  //draw underline
  display.fillRect(20, newer - 4, 4, 4, GxEPD_BLACK);
  display.fillRect(20, older - 4, 4, 4, GxEPD_WHITE);
  display.updateWindow(128 - bottom, 20, bottom - top + 4, 4, false);
}

void chooseCellScreen(uint8_t cellNum)
{
  display.setRotation(0);
  display.drawExampleBitmap(gImage_chooseCell, 0, 0, 128, 296, GxEPD_BLACK);

  display.setRotation(45);

  uint16_t box_w = 14;
  uint16_t box_h = 23;
  uint16_t box_y = 78;
  uint16_t box_x = 25;

  display.fillRect(box_x, box_y - box_h, box_w, box_h, GxEPD_BLACK);
  display.updateWindow(5, 5, 118, 286, false);
}

void partialChooseCell(uint8_t cellNum) {
  display.setRotation(0);
  display.drawExampleBitmap(gImage_chooseCell, 0, 0, 128, 296, GxEPD_BLACK,
GxEPD::bm_default | GxEPD::bm_partial_update);
  display.setRotation(45);

  uint16_t box_w = 14;
  uint16_t box_h = 23;
  uint16_t box_y = 78;
```

```
  uint16_t box_x = 25;

  if (cellNum < 8)  {
    box_x = 25 + (box_w - 1) * cellNum;
  }
  else {
    box_x = 166 + (box_w - 1) * (cellNum - 8);
  }

  display.fillRect(box_x, box_y - box_h, box_w, box_h, GxEPD_BLACK);
  display.updateWindow(128 - box_y, 25, box_h, 246, false);
}

struct button
{
  const uint8_t   pin;
  uint8_t         buttonState;
  uint8_t         lastButtonState;
  unsigned long   lastDebounceTime;
};

struct button buttons[] =
{
  { CENTER_BUTTON, LOW, LOW, 0},
  { UP_BUTTON, LOW, LOW, 0},
  { DOWN_BUTTON, LOW, LOW, 0},
  { LEFT_BUTTON, LOW, LOW, 0},
  { RIGHT_BUTTON, LOW, LOW, 0}
};

void updateCellConfig(uint8_t cellNum, uint8_t cellConfig, boolean direction)
{
  //change value of config
  if (cellConfig == 0) { //temp
    if (direction) {
      configs[cellNum].max_temp += 1;
    }
    else {
      configs[cellNum].max_temp -= 1;
    }
  }
  else if (cellConfig == 1) { //temp
    if (direction) {
      configs[cellNum].max_voltage += 0.1;
    }
    else {
      configs[cellNum].max_voltage -= 0.1;
    }
```

```
  }
  else if (cellConfig == 2) { //temp
    if (direction) {
      configs[cellNum].min_voltage += 0.1;
    }
    else {
      configs[cellNum].min_voltage -= 0.1;
    }
  }
  else if (cellConfig == 3) { //temp
    if (direction) {
      configs[cellNum].max_charge_voltage += 0.1;
    }
    else {
      configs[cellNum].max_charge_voltage -= 0.1;
    }
  }
  else if (cellConfig == 4) { //temp
    configs[cellNum].SOH = !configs[cellNum].SOH;
  }
  //partially update screen
}

#define NUM_CELL_CONFIGS 5

void printCellConfigs(uint8_t cellNum)
{
  //print each
  const GFXfont* font = &FreeSansBold9pt7b;
  display.setFont(font);

  uint8_t left = 10;
  uint8_t right = (296/2);
  uint8_t top = 60;
  uint8_t line = 20;
  uint8_t y_point = top;

  String num = String("Cell #" + String(cellNum+1, DEC));
  String mtemp = String("Max T " + String(configs[cellNum].max_temp, DEC));
  String maxv = String("Max V " + String(configs[cellNum].max_voltage, 1));
  String minv = String("Min V " + String(configs[cellNum].min_voltage, 1));
  String maxcv = String("Max ChV " +
String(configs[cellNum].max_charge_voltage, 1));
  String soh = String("SOH " + String(true,DEC));
  display.fillScreen(GxEPD_WHITE);
  display.setTextColor(GxEPD_BLACK);
  display.setCursor(110, 30);
  display.print(num);
```

```
  display.setCursor(235, 120);
  display.print("HOME");
  display.setCursor(left, y_point);
  display.fillRect(left-5, y_point-8, 4, 4, GxEPD_BLACK);
  y_point+=line;
  display.print(mtemp);
  display.setCursor(left, y_point);
  y_point+=line;
  display.print(maxv);
  display.setCursor(left, y_point);
  y_point=top;
  display.print(minv);
  display.setCursor(right, y_point);
  y_point+=line;
  display.print(maxcv);
  display.setCursor(right, y_point);
  y_point+=line;
  display.print(soh);
  display.updateWindow(5, 5, 118, 286, false);
}

uint8_t last_s = 5;
uint8_t last_y = 52;

void cellConfigs(uint8_t cellNum)
{
  //update screen to print all options for cell

  last_s = 5;
  last_y = 52;
  printCellConfigs(cellNum);

  uint8_t cell_config = 0;
  centerPress=false; upPress=false; downPress=false; leftPress=false;
rightPress=false;

  while (1) {

    if (cell_config == NUM_CELL_CONFIGS && centerPress)  { //exit
      Serial.println("center");
      centerPress = false;
      delay(50);
      break;
    }
    else if (leftPress) {
      leftPress = false;
      delay(50);
      Serial.println("left");
```

```
      if (cell_config == 0) {
        cell_config = NUM_CELL_CONFIGS;
      }
      else {
        cell_config--;
      }
      moveCellConfig(cell_config);
    }
    else if (rightPress) {
      rightPress = false;
      delay(50);
      Serial.println("right");
      if (cell_config == NUM_CELL_CONFIGS) {
        cell_config = 0;
      }
      else {
        cell_config++;
      }
      moveCellConfig(cell_config);
    }
    else if (upPress) {
      Serial.println("up");
      upPress = false;
      delay(50);
      updateCellConfig(cellNum, cell_config, true);
    }
    else if (downPress) {
      Serial.println("down");
      downPress = false;
      delay(50);
      updateCellConfig(cellNum, cell_config, false);
    }
  }
}


void moveCellConfig(uint8_t cellConfig)
{
    //change position of bullet point
  uint8_t left = 5;
  uint8_t right = (296/2)-5;
  uint8_t side = left;
  uint8_t top = 60;
  uint8_t line = 20;
  uint8_t y_point = top-8;
  if (cellConfig<=2) {
    y_point = top-8+20*cellConfig;
    side = left;
```

```
  }
  else if (cellConfig>2 && cellConfig<NUM_CELL_CONFIGS) {
    y_point = top-8+20*(cellConfig-3);
    side = right;
  }
  else {
    y_point = 112;
    side = 230;
  }
  display.fillRect(last_s, last_y, 4, 4, GxEPD_WHITE);
  display.fillRect(side, y_point, 4, 4, GxEPD_BLACK);
  display.updateWindow(5, 5, 118, 286, false);

  last_s = side;
  last_y = y_point;
}

uint8_t last_sm = 5;
uint8_t last_ym = 42;

#define NUM_MISC_CONFIGS 6
void miscConfigs()
{
    last_sm = 5;
    last_ym = 42;
    printMiscConfigs();
  centerPress=false; upPress=false; downPress=false; leftPress=false;
rightPress=false;
  uint8_t misc_config = 0;

  while (1) {
    if (misc_config == NUM_MISC_CONFIGS && centerPress)  {//exit
      Serial.println("center");
      centerPress = false;
      delay(50);
      break;
    }
    else if (leftPress) {
      Serial.println("left");
      leftPress = false;
      delay(50);
      if (misc_config == 0) {
        misc_config = NUM_MISC_CONFIGS;
      }
      else {
        misc_config--;
      }
      moveMiscConfig(misc_config);
```

```
    }
    else if (rightPress) {
      Serial.println("right");
      rightPress = false;
      delay(50);
      if (misc_config == NUM_MISC_CONFIGS) {
        misc_config = 0;
      }
      else {
        misc_config++;
      }
      moveMiscConfig(misc_config);
    }
    else if (upPress) {
      Serial.println("up");
      upPress = false;
      delay(50);
      updateMiscConfig(misc_config, true);
    }
    else if (downPress) {
      Serial.println("down");
      downPress = false;
      delay(50);
      updateMiscConfig(misc_config, false);
    }
  }
}

void updateMiscConfig(uint8_t miscConfig, boolean direction)  //finish this
{
    //change value of config
  if (miscConfig == 0) { //pack id
      misc_configs[0].pack_id = !misc_configs[0].pack_id;
  }
  else if (miscConfig == 1) { //airs
    misc_configs[0].airs_state = !misc_configs[0].airs_state;
  }
  else if (miscConfig == 2) { //sl
    misc_configs[0].sl_state = !misc_configs[0].sl_state;
  }
  else if (miscConfig == 3) { //soc min
    if (direction) {
      misc_configs[0].SOC_min += 1;
    }
    else {
      misc_configs[0].SOC_min -= 1;
    }
  }
```

```
  else if (miscConfig == 4) { //max current
    if (direction) {
      misc_configs[0].max_pack_current += 0.1;
    }
    else {
      misc_configs[0].max_pack_current -= 0.1;
    }
  }
  else if (miscConfig == 5) { //min current
    if (direction) {
      misc_configs[0].min_pack_current += 0.1;
    }
    else {
      misc_configs[0].min_pack_current -= 0.1;
    }
  }
  printMiscConfigs2(miscConfig);
}

void printMiscConfigs()
{
    //print each
  const GFXfont* font = &FreeSansBold9pt7b;
  display.setFont(font);

  uint8_t left = 10;
  uint8_t right = (296/2);
  uint8_t top = 50;
  uint8_t line = 20;
  uint8_t y_point = top;

  String pack = String("PackID " + String(misc_configs[0].pack_id, DEC));
  String air = String("AIRS " + String(misc_configs[0].airs_state, DEC));
  String sl = String("SL " + String(misc_configs[0].sl_state, DEC));
  String minSOC = String("SOCmin " + String(misc_configs[0].SOC_min, DEC));
  String maxC =String("Max C " + String(misc_configs[0].max_pack_current, 1));
  String minC =String("Min C " + String(misc_configs[0].max_pack_current, 1));

  display.fillScreen(GxEPD_WHITE);
  display.setTextColor(GxEPD_BLACK);
  display.setCursor(235, 120);
  display.print("HOME");

  display.setCursor(left, y_point);
  display.fillRect(left-5, y_point-8, 4, 4, GxEPD_BLACK);
  y_point+=line;
  display.print(pack);
  display.setCursor(left, y_point);
```

```
  y_point+=line;
  display.print(air);
  display.setCursor(left, y_point);
  y_point=top;
  display.print(sl);

  display.setCursor(right, y_point);
  y_point+=line;
  display.print(minSOC);
  display.setCursor(right, y_point);
  y_point+=line;
  display.print(maxC);
  display.setCursor(right, y_point);
  y_point+=line;
  display.print(minC);
  display.updateWindow(5, 5, 118, 286, false);
}

void printMiscConfigs2(uint8_t config_num)
{
    //print each
  const GFXfont* font = &FreeSansBold9pt7b;
  display.setFont(font);

  uint8_t left = 10;
  uint8_t right = (296/2);
  uint8_t top = 50;
  uint8_t line = 20;
  uint8_t y_point = top;

  uint8_t side = left;
  if (config_num>=3){side = right;}

  String pack = String("PackID " + String(misc_configs[0].pack_id, DEC));
  String air = String("AIRS " + String(misc_configs[0].airs_state, DEC));
  String sl = String("SL " + String(misc_configs[0].sl_state, DEC));
  String minSOC = String("SOCmin " + String(misc_configs[0].SOC_min, DEC));
  String maxC =String("Max C " + String(misc_configs[0].max_pack_current, 1));
  String minC =String("Min C " + String(misc_configs[0].max_pack_current, 1));

  display.fillScreen(GxEPD_WHITE);
  display.setTextColor(GxEPD_BLACK);
  display.setCursor(235, 120);
  display.print("HOME");

  display.setCursor(left, y_point);
  y_point+=line;
  display.print(pack);
```

```
  display.setCursor(left, y_point);
  y_point+=line;
  display.print(air);
  display.setCursor(left, y_point);
  y_point=top;
  display.print(sl);

  display.setCursor(right, y_point);
  y_point+=line;
  display.print(minSOC);
  display.setCursor(right, y_point);
  y_point+=line;
  display.print(maxC);
  display.setCursor(right, y_point);
  y_point+=line;
  display.print(minC);
  display.updateWindow(15, side, 118, 140, false);
}



void moveMiscConfig(uint8_t miscConfig)
{
  //change position of bullet point
  uint8_t left = 5;
  uint8_t right = (296/2)-5;
  uint8_t side = left;
  uint8_t top = 50;
  uint8_t line = 20;
  uint8_t y_point = top-8;
  if (miscConfig<=2) {
    y_point = top-8+20*miscConfig;
    side = left;
  }
  else if (miscConfig>2 && miscConfig<NUM_MISC_CONFIGS) {
    y_point = top-8+20*(miscConfig-3);
    side = right;
  }
  else {
    y_point = 112;
    side = 230;
  }
  display.fillRect(last_sm, last_ym, 4, 4, GxEPD_WHITE);
  display.fillRect(side, y_point, 4, 4, GxEPD_BLACK);
  display.updateWindow(5, 5, 118, 286, false);

  last_sm = side;
  last_ym = y_point;
```

```
}

unsigned long debounceDelay = 40; // the debounce time; increase if the output
flickers

typedef enum {
  Main,
  Config_State,
  Misc_Configurations,
  Choose_Cell,
  Cell_Configs
} State;

void fsm() {

  boolean config_index = true;
  uint8_t cell_index = 0;
  uint8_t misc_index = 0;

  State nextState = Main;
  State state = Main;

  centerPress=false; upPress=false; downPress=false; leftPress=false;
rightPress=false;

  while (1) {
    switch (nextState) {
      case Main: {  //done
          if (centerPress) {
            Serial.println("center");  //testing
            centerPress=false;
            nextState = Config_State;
          }
          else if (state != Main) {
            setUpMain(id);
          }
          mainPartialUpdate(1, 2, 3, 4);  //fill w variables
          state = Main;
        }
        break;

      case Config_State: {   //done
          if (centerPress && config_index == true) {
            Serial.println("center");    //testing
            centerPress=false;
            nextState = Choose_Cell;
          }
          else if (centerPress && config_index == false) {
```

```
      Serial.println("center");    //testing
      centerPress=false;
      nextState = Misc_Configurations;
    }
    else if (upPress || downPress || leftPress || rightPress) {
      Serial.println("up/down/left/right");
      upPress=false; downPress=false; leftPress=false; rightPress=false;
      config_index = !config_index;
      configPartial(config_index);
    }
    else if (state != Config_State) {
      config_index = true;
      mainConfigScreen();
    }
    state = Config_State;
  }
  break;

case Misc_Configurations: {   //to be finished
    miscConfigs(); //exits function if on home button
    nextState = Main;
  }
  break;

case Choose_Cell: {  //done
    if (centerPress) {
      Serial.println("center");
      centerPress=false;
      nextState = Cell_Configs;
    }
    else if (leftPress) {
      Serial.println("left");
      leftPress=false;
      if (cell_index == 0) {
        cell_index = 15;
      }
      else {
        cell_index--;
      }
      partialChooseCell(cell_index);
    }
    else if (rightPress) {
      Serial.println("right");
      rightPress=false;
      if (cell_index == 15) {
        cell_index = 0;
      }
      else {
```

```
            cell_index++;
          }
          partialChooseCell(cell_index);
        }
        else if (state != Choose_Cell) {
          chooseCellScreen(cell_index);
        }
        state = Choose_Cell;
      }
      break;

    case Cell_Configs: {   //to be finished
        cellConfigs(cell_index); //exits function if on home button
        nextState = Main;
      }
      break;
  }
 }
}
```