/\* Sysc4907 Team 11 - Power and Battery Management

\* Supervisor: Professor Ibnkahla

\* Co-Supervisor: Zied Bouida

\* Members: Jacob Sharkey, Shawn Khadem, June Looi, Andre Yim, Ashley Hooker

\*/

#include <Wire.h>

#include <LiFuelGauge.h>

#define TCAADDR 0x70 //establishes TCA i2c mux address

LiFuelGauge gauge(MAX17043); //creates LiFuelGauge object

void tcaselect(uint8\_t i) { //function used to select ports in i2c mux

if (i > 3) return;

Wire.beginTransmission(TCAADDR);

Wire.write(1 << i);

Wire.endTransmission();

}

double bat0 = 2000, bat1 = 150, bat2 = 150; // mAh of batteres connected to ports 0-2

double SOCTotal = bat0 + bat1 + bat2;

double percent0 = bat0/SOCTotal, percent1 = bat1/SOCTotal, percent2 = bat2/SOCTotal;

double SOC0 = 0, SOC1 = 0, SOC2 = 0;

int timeOff = 0;

int bootupTime = 5;

int timeOn = 10;

int inMinutes = 1000;

int SerialBaudRate = 9600;

bool twoIoTDevs = false;

double minSOCSum = 15;

double getSOC0(){

tcaselect(0);

delay(250);

//gauge.reset();

//delay(100);

return gauge.getSOC();

}

double getSOC1(){

tcaselect(1);

delay(250);

//gauge.reset();

//delay(100);

return gauge.getSOC();

}

double getSOC2(){

tcaselect(2);

delay(250);

//gauge.reset();

//delay(100);

return gauge.getSOC();

}

void setup() {

pinMode(7, OUTPUT);

pinMode(8, OUTPUT);

// Serial.begin(SerialBaudRate);

// while(!Serial);

// Serial.println("Communication OK\n");

//loop to reset all fuel gauges

// for(int t=0; t<2; t++){ //loop to reset all fuel gauges

// tcaselect(t);

// gauge.reset();

// delay(250);

// }

// Serial.println("Gauges reset");

//initial measurements

//Serial.println("Initial measurements");

// SOC0 = getSOC0();

// Serial.print("Port 0: "); Serial.print(SOC0); Serial.println("%");

// SOC1 = getSOC1();

// Serial.print("Port 1: "); Serial.print(SOC1); Serial.println("%");

// SOC2 = getSOC2();

// Serial.print("Port 2: "); Serial.print(SOC2); Serial.println("%");

}

void loop() {

for(int t=0; t<2; t++){ //loop to reset all fuel gauges

tcaselect(t);

gauge.reset();

delay(250);

}

//Get SOCs

SOC0 = getSOC0();

SOC1 = getSOC1();

SOC2 = getSOC2();

//"Sum" SOC

double SOCSum = SOC0\*percent0 + SOC1\*percent1 + SOC2\*percent2;

if (SOCSum>minSOCSum){ //check for low batteries -> turns on IoT device(s) if higher than minSOCSum

//turn on IoT device(s)

digitalWrite(7,HIGH); //1st IoT device on

delay (bootupTime\*inMinutes);

//send initial SOCs

Serial.begin(SerialBaudRate);

while(!Serial);

Serial.println(SOC0); delay(150);

Serial.println(SOC1); delay(150);

Serial.println(SOC2); delay(150);

Serial.print("SOCSum: "); Serial.println(SOCSum);

Serial.print("On for: "); Serial.println(timeOn);

delay(timeOn\*inMinutes); //1st IoT device on

digitalWrite(7,LOW); //1st IoT device off

if (twoIoTDevs){

digitalWrite(8,HIGH); //2nd IoT device on

delay (bootupTime\*inMinutes);

delay(timeOn\*inMinutes);

digitalWrite(8, LOW); //2nd IoT device off

}

//Acquire new SOCs

SOC0 = getSOC0();

SOC1 = getSOC1();

SOC2 = getSOC2();

//"Sum" SOC

double SOCSum = SOC1\*percent1 + SOC1\*percent1 + SOC2\*percent2;

//Make decision of timeOff duration based on new SOCs

if (SOCSum > 85){

timeOff = 15;

}

else if (85 < SOCSum < 50){

timeOff = 30;

}

else if (25 < SOCSum < 50){

timeOff = 45;

}

else {

timeOff = 60; //default case

}

} else { //default case if batteries are low

timeOff = 60;

}

Serial.print("Off for: "); Serial.println(timeOff); Serial.println("");

delay (timeOff\*inMinutes);

}