#include "Wire.h"

#include "SPI.h"

#include "SD.h"

#include "TMRpcm.h"

#define SD\_ChipSelectPin 53

#define MAX9744\_I2CADDR 0x4B

#include "Adafruit\_TCS34725.h"

#include <Adafruit\_SSD1306.h>

#include <Adafruit\_GFX.h>

#define OLED\_ADDR 0x3C

Adafruit\_SSD1306 display(-1);

int value; //value for the "voltage" read at analog pin

float voltage; //actual voltage after doing voltage divider calculations

float R1 = 44.0; //Resistor 1 value in voltage divider this is 440k for this i have two 220k in series

float R2 = 47.0; //Resistor 2 value in voltage divider this is 470k

int fullyDischargedVoltage=7; //fuly discharged voltage value for vattery

int fullyChargedVoltage=9; //fuly charged voltage value for vattery

int percentage; //percentage of battery remaining

int pin5; //initialization of pin 7

int pin6; //initialization of pin 6

TMRpcm tmrpcm;

int8\_t thevol;

int val;

Adafruit\_TCS34725 tcs = Adafruit\_TCS34725(TCS34725\_INTEGRATIONTIME\_50MS, TCS34725\_GAIN\_1X);

//function to draw horizontal lines for battery

void drawLineH(int start, int end, int row) {

int i = start;

while (i <= end) {

display.drawPixel(i, row, WHITE); //command to display pixel (column, row, color)

i++; //We dont have color but it has to be there

}

}

//function to draw vertical lines for battery

void drawLineV(int start, int end, int column) {

int j = start;

while (j <= end) {

display.drawPixel(column, j, WHITE); //command to display pixel (column, row, color)

j++;

}

}

//function to draw battery border

void screenSetup(){

// code for the battery horizontal border

drawLineH(93, 126, 0);

drawLineH(93, 126, 7);

// code for the battery vertical border

drawLineV(1, 6, 92);

drawLineV(1, 6, 127);

// code for the battery tip

drawLineH(88, 91, 2);

drawLineH(88, 91, 3);

drawLineH(88, 91, 4);

drawLineH(88, 91, 5);

}

//function for the block 1 inside battery

void block1(){

drawLineH(95, 100, 2);

drawLineH(95, 100, 3);

drawLineH(95, 100, 4);

drawLineH(95, 100, 5);

}

//function for the block 2 inside battery

void block2(){

drawLineH(103, 108, 2);

drawLineH(103, 108, 3);

drawLineH(103, 108, 4);

drawLineH(103, 108, 5);

}

//function for the block 3 inside battery

void block3(){

drawLineH(111, 116, 2);

drawLineH(111, 116, 3);

drawLineH(111, 116, 4);

drawLineH(111, 116, 5);

}

//function for the block 4 inside battery

void block4(){

drawLineH(119, 124, 2);

drawLineH(119, 124, 3);

drawLineH(119, 124, 4);

drawLineH(119, 124, 5);

}

//function to measure voltage, do percentage calulcation, then display the corresponding number of blocks in the battery

void battery(){

value = analogRead(A0); //reads the value at the analog pin

voltage = 1.012 \* value \* (5.0/1024)\*((R1 + R2)/R2); //1.04 is correction factor, 5/1024 is to get the bits or whatever multiplied by reciprocal of voltage divider

percentage = (voltage - fullyDischargedVoltage) / (fullyChargedVoltage - fullyDischargedVoltage) \* 100; //calculated the percentage of the battery remaining

percentage = (percentage/10)\*10; //This makes the percentage increment in increments of 10

screenSetup(); //calls screenSetup function to put the borders of the battery on the screen

if (percentage >= 80){

block1();

block2();

block3();

block4();

}

else if (percentage < 80 && percentage >= 60){

block2();

block3();

block4();

}

else if (percentage < 60 && percentage >= 40){

block3();

block4();

}

else if (percentage < 40 && percentage >= 20){

block4();

}

else if (percentage < 20 && percentage >= 0){

//empty on purpose

}

}

class ColorName { //creates a private class color name which will be used to create our predetermined color value objects.

protected: //safe variables that cannot be alterted by the sketch/ComputeMSE

String \_name;

byte \_red;

byte \_green;

byte \_blue;

public: //duplicate variables safe for manipulation

ColorName(String name, byte red, byte green, byte blue){

\_name = name;

\_red = red;

\_green = green;

\_blue = blue;

}

byte getRed(){ //gets protected variables and passes them into public domain

return \_red;}

byte getGreen(){

return \_green;}

byte getBlue(){

return \_blue;}

String getName(){

return \_name;}

long computeMse(byte red, byte green, byte blue) { //computes error between defined color and measured color using Mean square error method

return (long) (((long)(red - \_red) \* (red - \_red) + (long)(green - \_green) \* (green - \_green) + (long)(blue - \_blue) \* (blue - \_blue)) / 3);

}

static ColorName\* getClosestColorName(ColorName colorNames[],

unsigned int numColors, byte red, byte green, byte blue){ //takes in predetermined color array and predetermined color

//array size. also takes in measured RGB values

ColorName\* \_bestMatchColorName = NULL; //initializes best color name match pointer

long bestMatchDistance = 65535;

for(int i = 0; i < numColors; i++){ //iterate through all predetermined color objects and their values

ColorName\* curColorName = &colorNames[i];

if(\_bestMatchColorName == NULL){

\_bestMatchColorName = &colorNames[0];

}

long meanSquareError = curColorName->computeMse(red, green, blue);

if(meanSquareError < bestMatchDistance){

\_bestMatchColorName = &colorNames[i];

bestMatchDistance = meanSquareError;

}

}

return \_bestMatchColorName;

}

};

const int NUM\_COLOR\_NAMES = 12; //defines the colors and their ideal values

ColorName \_colorNames[NUM\_COLOR\_NAMES] = { //this can be adjusted to add more colors and better accuracy

ColorName("Purple", 75, 71, 103),

ColorName("Yellow", 122, 79, 42),

ColorName("Blue", 41, 83, 123),

ColorName("Green", 62, 110, 69),

ColorName("Red", 160, 47, 48),

ColorName("Black", 84, 83, 78),

ColorName("Orange", 164, 48, 36),

ColorName("White", 80, 85, 78),

ColorName("Brown", 112, 75, 58),

ColorName("Cyan", 64, 89, 91),

ColorName("Maroon", 105, 74, 70),

ColorName("Pink", 151, 44, 56),

};

void setup() {

Serial.begin(9600); //Initializes Arduino

tmrpcm.speakerPin=46;

SD.begin(SD\_ChipSelectPin); //Initialized SD Card

Wire.begin();

pinMode(6, INPUT);

pinMode(5, INPUT);

setvolume(45);

// initialize and clear display

display.begin(SSD1306\_SWITCHCAPVCC, OLED\_ADDR);

display.clearDisplay();

display.display(); //updates the display need this anytime you want to put something on the screen

float sensedRed, sensedGreen, sensedBlue;

tcs.getRGB(&sensedRed, &sensedGreen, &sensedBlue);

int rawRed = (int)sensedRed;

int rawGreen = (int)sensedGreen;

int rawBlue = (int)sensedBlue;

}

boolean setvolume(int8\_t v) { //volume control function

if (v > 60) v = 60; // cant be higher than 63 or lower than 0

if (v < 0) v = 0;

Wire.beginTransmission(MAX9744\_I2CADDR);

Wire.write(v);

if (Wire.endTransmission() == 0)

return true;

else

return false;

}

char hex(int n) {

if (n < 10) {

return char(n + 48); // 48 is the ASCII code for '0'

} else {

return char(n + 55); // 55 is the ASCII code for 'A' - 10

}

}

String RGBtoHex(int r, int g, int b) {

String hexCode = "";

hexCode += hex(r / 16);

hexCode += hex(r % 16);

hexCode += hex(g / 16);

hexCode += hex(g % 16);

hexCode += hex(b / 16);

hexCode += hex(b % 16);

return hexCode;

}

ColorName \*closestColorName = nullptr; //added this so we can call it in the later if statements

String hexCode = "";

float R; float G; float B; int Rcomp; int Gcomp; int Bcomp;

void loop() {

pin5 = digitalRead(5);

pin6 = digitalRead(6);

if (pin6 == LOW && pin5 == HIGH) {

if (digitalRead(7) == HIGH) {

uint16\_t r, g, b, c; //variables for rgb and clear

tcs.getRawData(&r, &g, &b, &c); //stores the data inside the variables

R = (float)r / (float)c;

G = (float)g / (float)c;

B = (float)b / (float)c;

R \*= 255; G \*= 255; B \*= 255;

R = map(R, 42, 140, 0, 255); //USES THE MAP FUNCTION TO SKEW THE VALUES THE SENSOR MEASURES:

G = map(G, 46, 90, 0, 255); //THE 2ND VALUES OF THE MAP FUNCTION CAN BE FURTHER TUNED TO INSCREASE ACCURACY

B = map(B, 35, 95, 0, 255);

Rcomp = constrain(R,0,255); //CONSTRAINS VALUES BETWEEN 0 and 255 IN CASE MAP FUNCTION THROWS IT OUTSIDE THAT RANGE

Gcomp = constrain(G,0,255);

Bcomp = constrain(B,0,255);

hexCode = RGBtoHex((int)Rcomp,(int)Gcomp,(int)Bcomp);

delay(100);

}

display.clearDisplay();

battery(); //call battery function to display the battery

display.setTextSize(1); //set text size for the text to be printed

display.setTextColor(WHITE); //set color of text

display.setCursor(0,0); //sets the starting location of the text (column, row)

display.print("MODE: ADVANCED"); //text to display

display.setTextSize(1);

display.setTextColor(WHITE);

display.setCursor(0,9);

display.print("R:");

display.setCursor(17,9);

display.print((int)Rcomp);

display.setCursor(45,9);

display.print("G:");

display.setCursor(60,9);

display.print((int)Gcomp);

display.setCursor(85,9);

display.print("B:");

display.setCursor(100,9);

display.print((int)Bcomp);

display.setTextSize(1);

display.setTextColor(WHITE);

display.setCursor(0, 23);

display.print("HEXCODE:");

display.setTextSize(2);

display.setTextColor(WHITE);

display.setCursor(52, 18);

display.print(hexCode);

delay(1);

display.display();

}

else if (pin5 == LOW && pin6 == HIGH) {

if (digitalRead(7) == HIGH) {

float sensedRed, sensedGreen, sensedBlue;

tcs.getRGB(&sensedRed, &sensedGreen, &sensedBlue);

int rawRed = (int)sensedRed;

int rawGreen = (int)sensedGreen;

int rawBlue = (int)sensedBlue;

Serial.print("Red: ");

Serial.println(rawRed);

Serial.print("Green: ");

Serial.println(rawGreen);

Serial.print("Blue: ");

Serial.println(rawBlue);

closestColorName = ColorName::getClosestColorName(\_colorNames, NUM\_COLOR\_NAMES, rawRed, rawGreen, rawBlue);

Serial.println(closestColorName->getName());

String sound = closestColorName->getName() + ".wav";

char \*char\_array = new char[sound.length() + 1];

strcpy(char\_array, sound.c\_str());

tmrpcm.play(char\_array);

delay(50);

}

display.clearDisplay();

battery();

display.setTextSize(1);

display.setTextColor(WHITE);

display.setCursor(0, 0);

display.print("MODE:");

display.setTextSize(1);

display.setTextColor(WHITE);

display.setCursor(0, 8);

display.print("NORMAL");

display.setTextSize(1);

display.setTextColor(WHITE);

display.setCursor(0, 23);

display.print("COLOR:");

if (closestColorName != nullptr) { // Check if closestColorName has been initialized

display.setCursor(40, 23);

display.print(closestColorName->getName());

delay(1);

}

display.display();

}

}