#include <MD\_Parola.h>

#include <MD\_MAX72xx.h>

#include <DHT.h>

#include <SPI.h>

#include <Wire.h>

#include "Font7Seg.h"

#define HARDWARE\_TYPE MD\_MAX72XX::PAROLA\_HW

#define MAX\_DEVICES 4 // Define the number of displays connected

#define CLK\_PIN 13 // CLK or SCK

#define DATA\_PIN 11 // DATA or MOSI

#define CS\_PIN 10 // CS or SS

#define SPEED\_TIME 75 // Speed of the transition

#define PAUSE\_TIME 0

#define MAX\_MESG 20

// These are for the clock

#define DS1307\_ADDRESS 0x68

// These are for the temperature

#define DHTPIN 2

#define DHTTYPE DHT22

#define TIMEDHT 1000

// Global variables

uint8\_t wday, mday, month, year;

uint8\_t hours, minutes, seconds;

char szTime[9]; // mm:ss\0

char szMesg[MAX\_MESG + 1] = "";

float humidity, celsius, fahrenheit;

uint8\_t degC[] = { 6, 3, 3, 56, 68, 68, 68 }; // Deg C

uint8\_t degF[] = { 6, 3, 3, 124, 20, 20, 4 }; // Deg F

uint8\_t clear = 0x00;

uint32\_t timerDHT = TIMEDHT;

DHT dht(DHTPIN, DHTTYPE);

// Hardware SPI connection

MD\_Parola P = MD\_Parola(HARDWARE\_TYPE, CS\_PIN, MAX\_DEVICES);

void beginDS1307()

{

// Read the values (date and time) of the DS1307 module

**Wire**.beginTransmission(DS1307\_ADDRESS);

**Wire**.write(clear);

**Wire**.endTransmission();

**Wire**.requestFrom(DS1307\_ADDRESS, 0x07);

}

uint8\_t decToBcd(uint8\_t value)

{

return ((value / 10 \* 16) + (value % 10));

}

uint8\_t bcdToDec(uint8\_t value)

{

return ((value / 16 \* 10) + (value % 16));

}

// Code for get Temperature

void getTemperature()

{

// Wait for a time between measurements

if ((millis() - timerDHT) > TIMEDHT) {

// Update the timer

timerDHT = millis();

// Reading temperature or humidity takes about 250 milliseconds!

// Sensor readings may also be up to 2 seconds 'old' (its a very slow sensor)

humidity = dht.readHumidity();

// Read temperature as Celsius (the default)

celsius = dht.readTemperature();

// Read temperature as Fahrenheit (isFahrenheit = true)

fahrenheit = dht.readTemperature(true);

// Check if any reads failed and exit early (to try again)

if (isnan(humidity) || isnan(celsius) || isnan(fahrenheit)) {

**Serial**.println("Failed to read from DHT sensor!");

return;

}

}

}

void setup(void)

{

**Wire**.begin();

P.begin(2);

P.setInvert(false);

P.setZone(0, MAX\_DEVICES - 4, MAX\_DEVICES - 1);

P.setZone(1, MAX\_DEVICES - 4, MAX\_DEVICES - 1);

P.displayZoneText(1, szTime, PA\_CENTER, SPEED\_TIME, PAUSE\_TIME, PA\_PRINT, PA\_NO\_EFFECT);

P.displayZoneText(0, szMesg, PA\_CENTER, SPEED\_TIME, 0, PA\_PRINT , PA\_NO\_EFFECT);

P.addChar('$', degC);

P.addChar('&', degF);

dht.begin();

}

void loop(void)

{

static uint32\_t lastTime = 0; // Memory (ms)

static uint8\_t display = 0; // Current display mode

static bool flasher = false; // Seconds passing flasher

beginDS1307();

getTemperature();

P.displayAnimate();

if (P.getZoneStatus(0))

{

switch (display)

{

case 0: // Temperature deg Celsius

P.setPause(0, 1000);

P.setTextEffect(0, PA\_SCROLL\_LEFT, PA\_SCROLL\_UP);

display++;

dtostrf(celsius, 3, 1, szMesg);

strcat(szMesg, "$");

break;

case 1: // Temperature deg Fahrenheit

P.setTextEffect(0, PA\_SCROLL\_UP, PA\_SCROLL\_DOWN);

display++;

dtostrf(fahrenheit, 3, 1, szMesg);

strcat(szMesg, "&");

break;

case 2: // Humidity

P.setTextEffect(0, PA\_SCROLL\_DOWN, PA\_SCROLL\_LEFT);

display++;

dtostrf(humidity, 3, 0, szMesg);

strcat(szMesg, "%UR");

display=0;

break;

}

P.displayReset(0); // Rest zone zero

}

}