/\*

\* Instituto Federal de Educação, Ciência e Tecnologia Minas Gerais

\* IFMG - Campus Avançado Conselheiro Lafaiete

\*

\* Código para monitoramento de pluviógrafo Versão 1

\*

\* Autor.............: Jonas Henrique Nascimento

\* Data de início....: 30/06/2018

\* Data da ultima atualização: 22/07/2018

\* Data de término...: 22/07/2018

\*

\* O código consiste em um leitor de dois botões que recebem nivel lógico zero ao serem pressionados pelo movimento da báscula.

\* Após a leitura dos botões é feito uma contagem precisa de tempo de 1 minuto, a partir do modulo RTC. Com isso, se obtém o valor da intensidade do pressionar dos botões

\* sobre o tempo decorrido.

\* Como se espera analisar os valores ao futuro, estes são armazenados em um cartão SD em um arquivo SVC.

\* A intencidade será calculada a partir do número de pulsos colhidos pelos botões por minuto, e esta será calculada e armazenada no cartão SD por forma de um datalogger.

\* Tal datalogger será formado por um modelo de agenda em forma de tabela, contendo a data referênte a medida de cada pulso, e uma contagem do numero total naquele minuto.

\* Paralelamente será medida a temperatura ambiente através de um leitor de temperatura já embutido no shield RTC, a qual será armazenada, também no datalogger.

\* Ademais, o sistema contará com a possibilidade de revisionamentos, podendo alterar a data referente ao RTC por meio de uma interface HM ( Homem-máquina ).

\* Tal interface poderá ser acessada através da comunicação serial entre o microcontrolador e o computador, por meio de um cabo USB e de um adaptador Serial-SPI. Ou por meio da

\* atualização do código no microprocessador 328p.

\*

\* Todo o código, esquemático do circuito eletrônico e demais informações estarão sempre contidas no endereço abaixo, para livre aperfeiçoamento do código e do circuito. Todavia

\* pede-se, por educação, que ao compartilharem o código, mantenham os autores originais, tão bem quanto o nome da instituição.

\*

\* https://github.com/W8jonas/pluviografo

\*

\*/

-----------------------------------------------------------------

// §§§§ Hardware §§§§ //

/\*

\* Consiste na comunicação SPI com o módulo cartão SD, na comuni-

\* cação I2C com o módulo DS3231. Além de possuir sua saída, ori-

\* ginalmente o pino 5 ligado a um resistor que é ligado à um LED

\* em current source. Ademais, os dois botões estão ligados em

\* pull up. Originalmente, nos digitais 6 e 7.

\* Para mais informações, consute os esquemáticos na raiz de todo

\* o projeto, seguindo o link do GitHub, apresentado no cabeçalho

\* deste projeto.

\*

\*/

-----------------------------------------------------------------

// §§§§ Declaração das bibliotecas §§§§ //

#include <avr/sleep.h>

#include <DS3231.h>

#include <SPI.h>

#include <SD.h>

-----------------------------------------------------------------

// §§§§ Definições de Hardware §§§§ //

#define entrada1 7

#define entrada2 6

#define saida 5

#define chip\_select 4

-----------------------------------------------------------------

// §§§§ Declaração das variáveis globais §§§§ //

bool estado1 = false;

bool estado2 = false;

bool flag1 = false;

bool flag2 = false;

unsigned int minuto\_antigo = 0;

unsigned int minuto\_atual = 0;

unsigned long contador\_de\_pulsos = 0;

-----------------------------------------------------------------

// §§§§ Declaração dos objetos §§§§ //

DS3231 rtc(SDA, SCL);

Time t;

File datalogger;

-----------------------------------------------------------------

// §§§§ Void Setup §§§§ //

void setup() {

pinMode(entrada1, INPUT\_PULLUP);

pinMode(entrada2, INPUT\_PULLUP);

pinMode(saida, OUTPUT);

Serial.begin(115200);

int i = 150;

rtc.begin();

while (!Serial) {;}

if (!SD.begin(chip\_select)) {

Serial.println("Erro ao ler cartao de memoria");

return;

}

//rtc.setDOW(FRIDAY); // Selecione o dia em ingles; Ex: (SUNDAY) - Domingo

//rtc.setTime(14, 30, 10); // Selecione a hora; Ex: (14, 30, 10) -- 14 horas, 30 minutos e 10 segundos

//rtc.setDate(20, 7, 2018); // Selecione a data; Ex: (20, 7, 2018) -- Dia 20 do mes 7 de 2018.

while(1){

digitalWrite(saida, HIGH);

delay(i);

i = i-10;

digitalWrite(saida, LOW);

delay(i);

if (i < 50) {i = i+9;}

if (i == 9) {break;}

}

digitalWrite(saida, HIGH);

delay(500);

digitalWrite(saida, LOW);

}

-----------------------------------------------------------------

// §§§§ Void Loop §§§§ //

void loop() {

t = rtc.getTime();

minuto\_atual = t.min;

if ( minuto\_antigo != minuto\_atual ){

minuto\_antigo = minuto\_atual;

Serial.print("Foi apertado o botao: ");

Serial.println(contador\_de\_pulsos);

datalogger = SD.open("Valores.svc", FILE\_WRITE);

if ( datalogger ) {

Serial.println("Atualizando datalogger");

// datalogger.println(" Data, | Hora, | Contagem, | Temperatura, |");

datalogger.print(rtc.getDateStr());

datalogger.print(", | ");

datalogger.print(rtc.getTimeStr());

datalogger.print(", | ");

if(contador\_de\_pulsos < 10) {datalogger.print("0");}

datalogger.print(contador\_de\_pulsos);

datalogger.print(", | ");

datalogger.print(rtc.getTemp());

datalogger.println(", |");

datalogger.close();

} else {

Serial.println("Erro ao abrir datalogger");

}

contador\_de\_pulsos = 0;

Serial.println("Atualizado");

}

estado1 = digitalRead(entrada1);

estado2 = digitalRead(entrada2);

if ( estado1 == LOW ) {

flag1 = true;

}

if ( estado2 == LOW ) {

flag2 = true;

}

if ( (estado1 == HIGH) && (flag1 == true) ){ // botão 1 trocou de estado -- botao apertado

digitalWrite(saida, LOW);

contador\_de\_pulsos ++;

digitalWrite(saida, HIGH);

delay(20);

digitalWrite(saida, LOW);

delay(20);

flag1 = false;

}

if ( (estado2 == HIGH) && (flag2 == true) ){ // botão 2 trocou de estado -- botao apertado

digitalWrite(saida, HIGH);

contador\_de\_pulsos ++;

digitalWrite(saida, HIGH);

delay(20);

digitalWrite(saida, LOW);

delay(20);

flag2 = false;

}

if ((estado1 == HIGH) && (estado2 == HIGH) || (flag1 == true) && (flag2 == true)) { // Erro, pois não se pode ter os dois botões ou as duas flags setadas ao mesmo tempo.

// reset

}

}