/\* Cronômetro

\* Paul Badger 2008

\* Demonstra o uso de millis(), resistores pull-up,

\* como fazer duas coisas acontecerem ao mesmo tempo,

\* imprime frações

\*

\* Montagem física: interruptor momentâneo ao pino 4 e ao

\* terra, e o LED com um resistor em série entre o pino 13

\* e o terra

\*/

#define ledPin 13 // LED ao pino digital 13

#define buttonPin 4 // botão ao pino 4

int value = LOW; // valor anterior do LED

int buttonState; // variável a guardar estado do botão

int lastButtonState; // variável a guardar o último estado do botão

int blinking; // condição para piscamento - o cronômetro venceu

long interval = 100; // intervalo de piscamento - mude à vontade

long previousMillis = 0; // variável a guardar o último momento de atualização do LED

long startTime ; // momento de partida do cronômetro

long elapsedTime ; // tempo decorrido no cronômetro

int fractional; // variável a guardar parte fracionária do tempo

void setup()

{

Serial.begin(9600);

pinMode(ledPin, OUTPUT); // ajuste o pino digital como saída

pinMode(buttonPin, INPUT); // linha não muito necessária, já que os pinos são por padrão entradas

digitalWrite(buttonPin, HIGH); // ligue resistores de pull-up

}

void loop()

{

// aguarde pressionamento do botão

buttonState = digitalRead(buttonPin); // leia e guarde estado do botão

if (buttonState == LOW && lastButtonState == HIGH && blinking == false){

// veja se houve transição de HIGH para LOW

// em caso positivo, então trata-se de um novo pressionamento enquanto o relógio não corre: parta o relógio

startTime = millis(); // guarde o tempo de partida

blinking = true; // ligue o piscar enquanto corre o relógio

delay(5); // pequena pausa para filtrar ruído mecânico do botão

lastButtonState = buttonState; // guarde buttonState em lastButtonState, para que possa comparar mais tarde

}

else if (buttonState == LOW && lastButtonState == HIGH && blinking == true){

// veja se houve transição de HIGH para LOW

// em caso positivo, trata-se de um novo pressionamento enquanto o relógio corre: pare o relógio e exiba mensagem

elapsedTime = millis() - startTime; // guarde tempo decorrido

blinking = false; // desligue o pisca-pisca

lastButtonState = buttonState; // guarde buttonState em lastButtonState, para comparar mais tarde

// rotina para exibir tempo decorrido

Serial.print( (int)(elapsedTime / 1000L)); // divida por 1000 para converter a segundos - e então converta a inteiro

Serial.print("."); // imprima ponto decimal

// use operador módulo para obter parte fracionária part do tempo

fractional = (int)(elapsedTime % 1000L);

// preencha com zeros à esquerda - não seria ótimo se

// a linguage Arduino tivesse algo específico para tratar disso? :)

if (fractional == 0)

Serial.print("000"); // inclua 3 zeros

else if (fractional < 10)

Serial.print("00"); // inclua 2 zeros

else if (fractional < 100)

Serial.print("0"); // inclua 1 zero

Serial.println(fractional); // imprima parte fracionária do tempo

}

else{

lastButtonState = buttonState; // guarde buttonState em lastButtonState, para comparar mais tarde

}

// rotina de piscamento - pisca LED enquanto conta o tempo

// verifique se já é hora de piscar LED; isto é, a diferença

// entre o tempo atual e o último momento em que piscamos o LED é maior que

// o intervalo ao qual queremos piscá-lo?

if ( (millis() - previousMillis > interval) ) {

if (blinking == true){

previousMillis = millis(); // lembre-se da última vez que piscou-se o LED

// se LED está desligado, ligue-o e vice-versa

if (value == LOW)

value = HIGH;

else

value = LOW;

digitalWrite(ledPin, value);

}

else{

digitalWrite(ledPin, LOW); // desligue o LED caso não esteja a piscar

}

}

}