ESTGOH
Escola Superior de Tecnologia e Gestão de Oliveira do Hospital
Instituto Politécnico de Coimbra

Licenciatura em Engenharia Informática

Tecnologia e Arquitetura de Computadores 2022/2023

Trabalho Prático nº 2

Realizado em: 25/05/2023
Elaborado em: 10/06/2023

Mariana Magalhães - a2022147454

Índice

1. Introdução

Este trabalho tem como objetivo criar um programa em que os dados de temperatura são recolhidos de 15 em 15 segundos sendo que o valor máximo, mínimo e médio deve ser escrito no monitor série a cada 60 segundos. Para demonstrar o seu funcionamento o circuito terá um led intermitente, o seu tempo de intermitência é configurado através de um potenciómetro (1-10 segundos). Qualquer tipo de alteração da configuração do potenciómetro é imediatamente refletido no led.

Este sistema esta equipado com alertas luminosos e sonoros. A informação luminosa é simulada com 1 led (vermelho). O valor de temperatura a que é ligado o led vermelho e ativa o alarme sonoro (piezo) é configurado através de um potenciómetro (0-50 °C). Uma vez que o led vermelho é ligado apenas deve ser desligado através de um switch button (sw1) e para isso usamos **interrups**. O alarme sonoro é ativado por um período de 5 segundos. O número de vezes que o alarme sonoro é ativado é registado num display de 7 seguementos.

Uma vez que temos de usar **interrups** o que foi utilizado neste programa foi o **attachInterrupt(digitalPinToInterrupt(pino), ISR, modo)**.

interrupt: o número da interrupção (int)

pin: o número do pino

ISR: o ISR a ser chamado quando ocorrer a interrupção, esta função não deve ter parâmetros e não retornar nada. Às vezes, essa função é chamada de rotina de serviço de interrupção.

mode: define quando a interrupção deve ser acionada e apenas permite quatro constantes que podem ter o valor de:

- LOW - para acionar a interrupção sempre que o pino estiver baixo,
- CHANGE - para acionar a interrupção sempre que o pino mudar de valor
- RISING - para disparar quando o pino vai de baixo para alto,
- FALLING - para quando o pino vai de alto para baixo.
- HIGH - para acionar a interrupção sempre que o pino estiver alto

Como o botão vem configurado como **HIGH** para evitar que ele seja contado duas vezes devemos usar o modo **FALLING** para quando o pino vai de alto para baixo.

Devido ao reset do display tinha de clicar nos 2 botões ao mesmo tempo e para não atrapalhar com o código nem com o interrupt, eu usei valores lógicos booleanos que só podem ter dois valores true ou false.

2. Métodos

O trabalho foi realizado no decorrer das 3 horas nas últimas duas aulas de Tecnologia e Arquitetura de Computadores (TAC) e cerca de 10 horas fora de aula.

Para a realização deste trabalho foi utilizado o **Tinkercad**, um programa de modelagem tridimensional (3D) online e gratuito que é executado num navegador da web, conhecido por ser simples e fácil de utilizar, sendo este usado para projetar o circuito. Para além do Tinkercad, foi usado o **Arduino IDE**, uma plataforma de prototipagem eletrónica de hardware livre e de placa única, projetada com um microcontrolador com suporte de entrada/saída embutido, uma das linguagens de programação padrão usada no programa é C/C++, neste caso essa linguagem é usada para o desenvolvimento do código. Para a realização do fluxograma foi usado o draw.io, um editor gráfico online no qual é possível desenvolver desenhos, gráficos, entre outros, sem a necessidade de usar um software caro e pesado. Todos este programas foram desenvolvidos num computador com um processador Intel(R) Core(TM) i5-1035G1 CPU @ 1.00GHz 1.19 GHz, também foram usados os materiais representados na tabela 1.

Nome	Quantidade	Componente
U1	1	Arduino Uno
U2	1	Sensor de temperatura [TMP36]
D1	1	Amarelo LED
R1, R2	2	560 Ω Resistor
D3 D6	2	Amarelo LED
R1, R2,	2	Resistências 560Ω
R7,R8	2	Resistência de 1K Ω
Rpotpt12, Rpotpt2	2	10 kΩ Potenciômetro
D2	1	Vermelho LED
PIEZO1	1	Piezo
Ssw1 Ssw2	2	Botão
Digit1	1	Catódica Visor de sete segmentos
R5 R6 R7 R8 R9 R10 R11	7	220 Ω Resistor

Tabela 1 - Materiais

3. Resultados

Comecei por fazer o projeto do circuito no Tinkercad (Figura 1) e através dessa montagem foi obtido o diagrama do circuito (figura 2 e 3), em seguida foram desenvolvidos um algoritmo e um fluxograma (Figura 4 e 5) para o desenvolvimento do código no Arduíno.

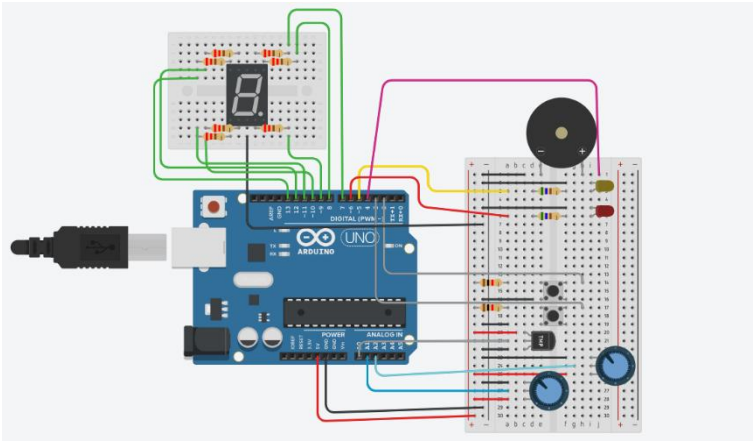

Figura 1 - tinkercad 1

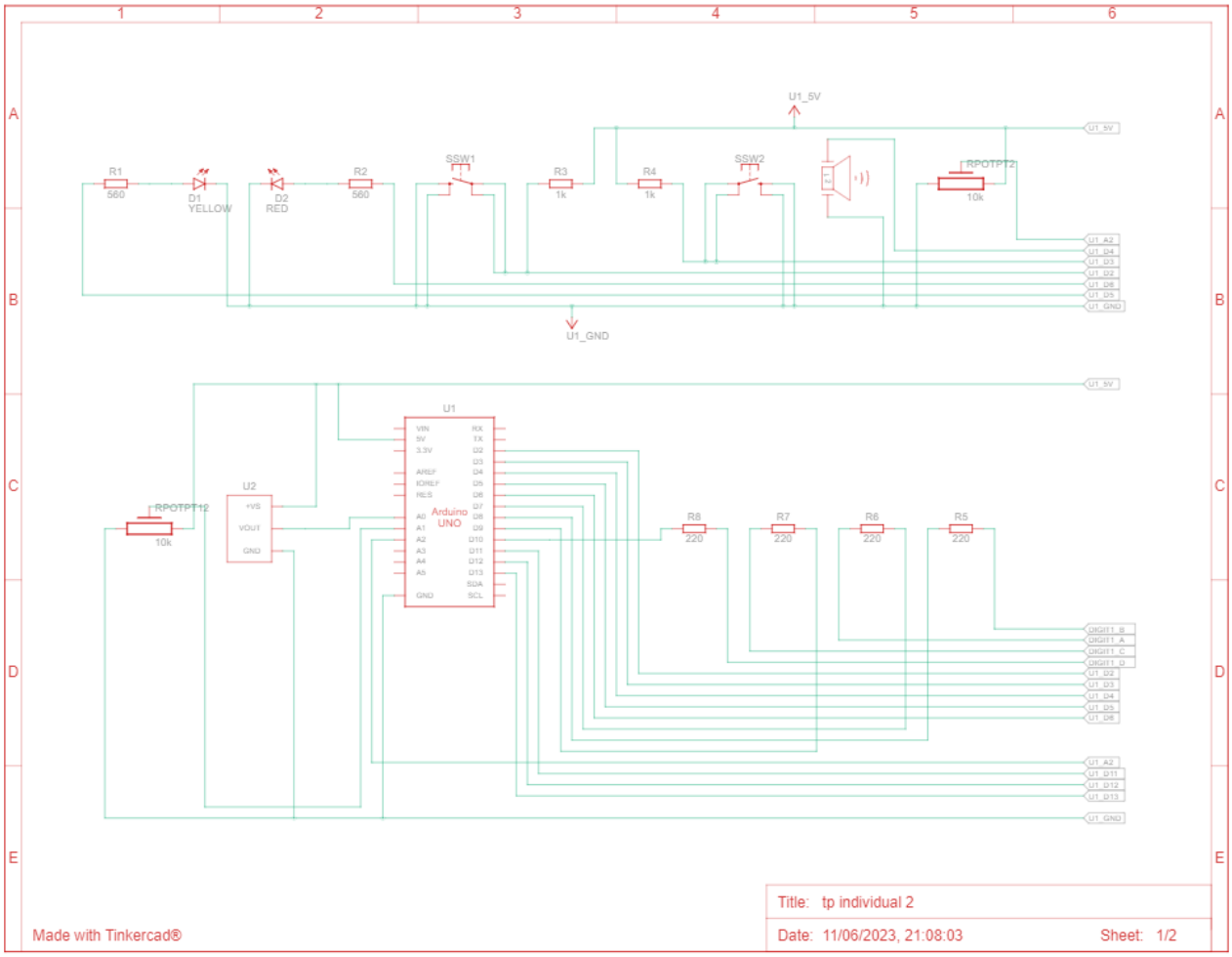

Figura 2 - diagrama do circuito folha 1

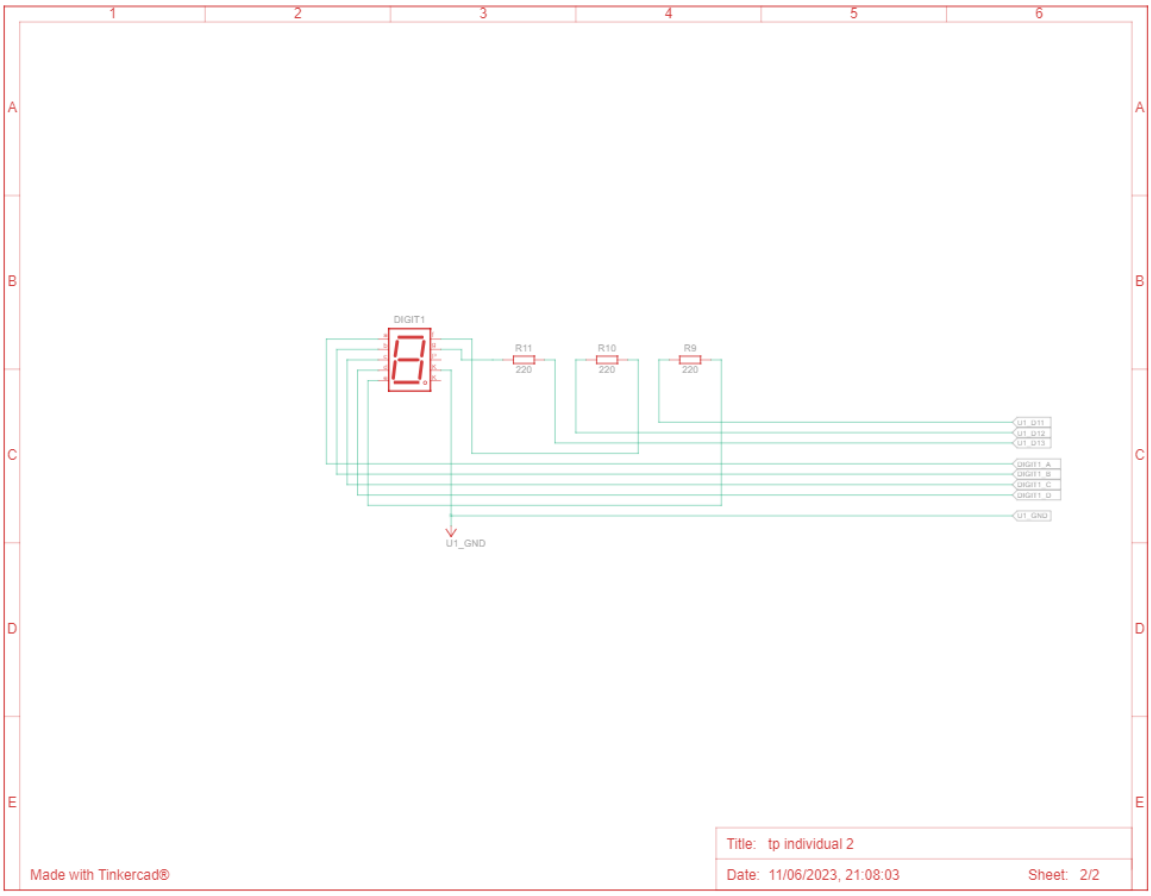

Figura 3- diagrama do circuito folha2 1

Fluxogramas do código:

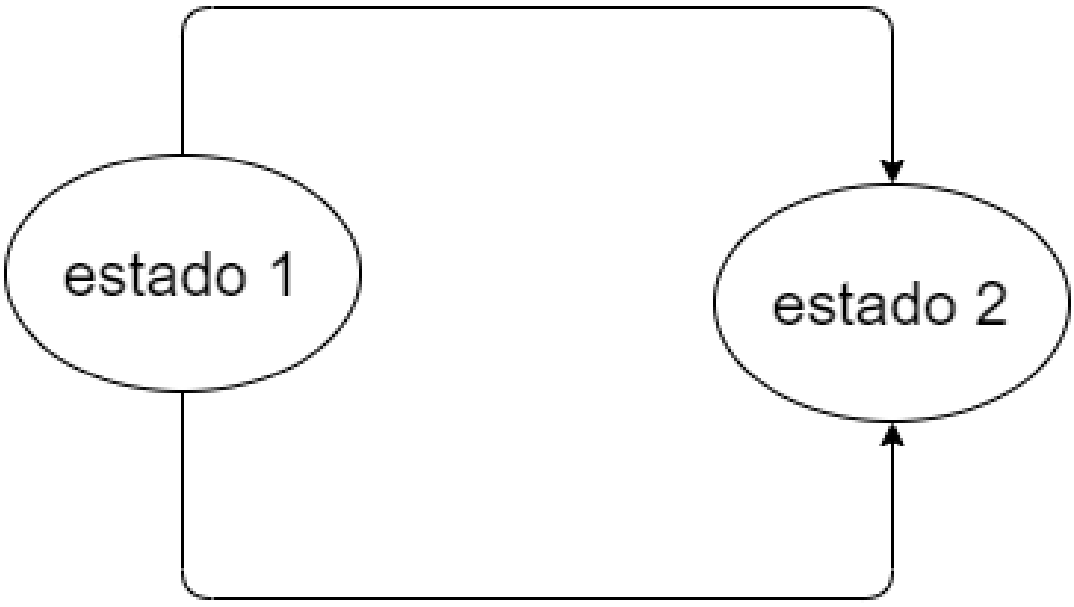

Figura 4 - fluxograma máquina de estados

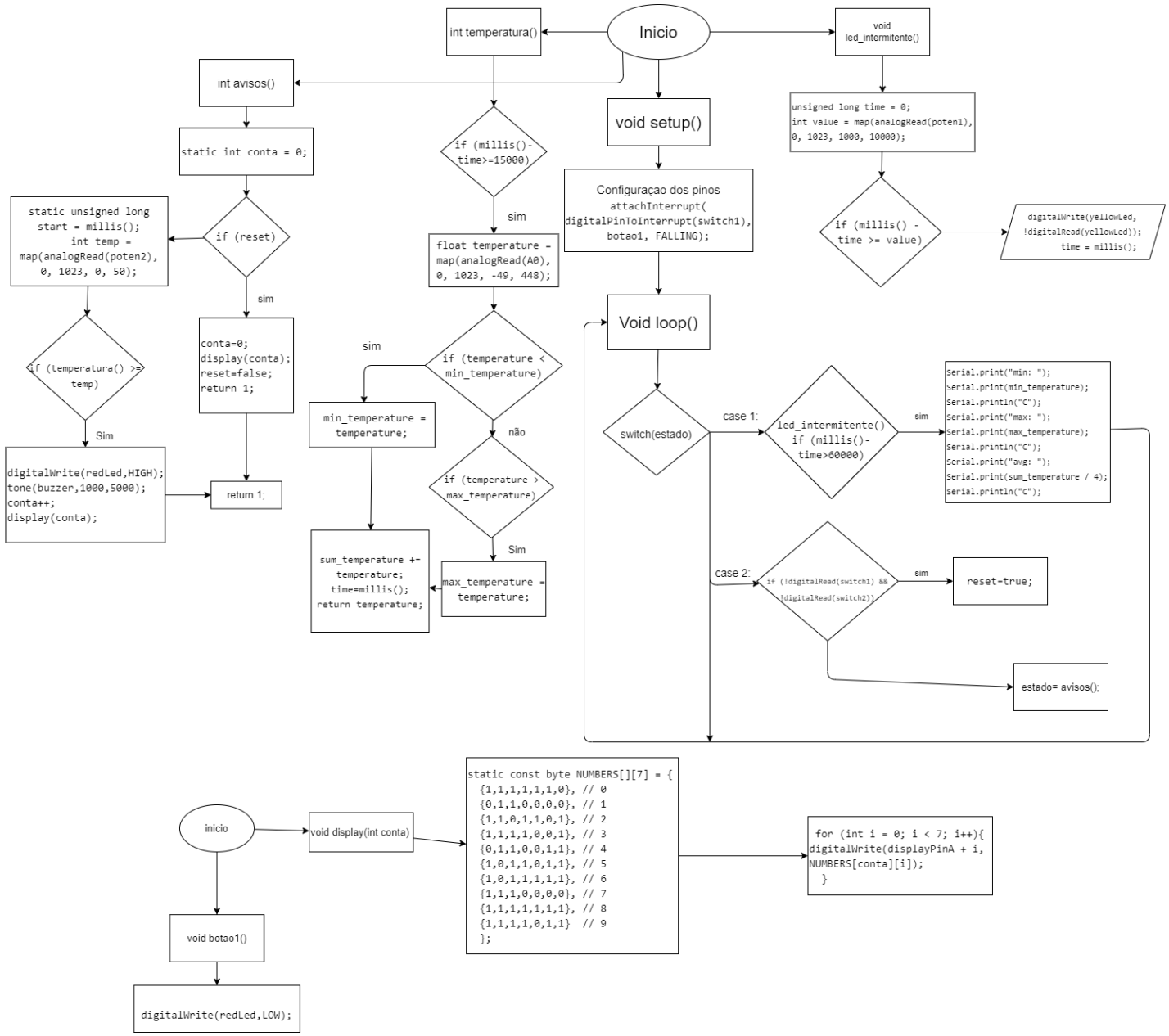

Figura 5 - fluxograma do código

Algoritmo do programa:
1. Configuração inicial e definição de pinos

2. Função **temperatura()**:

- Lê o valor analógico do pino A0 para obter a leitura da temperatura.
- Mapeia o valor lido para o intervalo de temperatura desejado.
- Atualiza os valores mínimo, máximo e a soma da temperatura.
- Retorna o valor da temperatura.

3. Função **led_intermitente()**:

- Lê o valor analógico do pino poten1 para determinar o intervalo de tempo para o LED amarelo piscar.
- Se o tempo decorrido for maior ou igual ao intervalo definido, inverte o estado do LED amarelo.

4. Função **botao1()**:

- Quando o botão 1 é pressionado (queda de sinal), aciona o pino redLed.

5. Função **display(conta)**:

- Mostra o valor passado como argumento no display de 7 segmentos.
- Utiliza uma matriz predefinida para representar os números de 0 a 9.

6. Função **avisos()**:

- Verifica se o valor da temperatura excede o valor definido pelo poten2.
- Se a temperatura exceder o limite, aciona o pino redLed, o buzzer e incrementa a variável "conta".
- Atualiza o display com o valor de "conta".

7. Função **setup()**:
   - Configuração inicial do programa.
   - Define os pinos como entrada ou saída.
   - Inicializa a comunicação serial.

8. 8. Função **loop()**:
   - Loop principal do programa.
   - Executa diferentes estados de acordo com a variável "estado".
   - Os estados são:
     - Estado 1: Pisca o LED amarelo em intervalos regulares e exibe as estatísticas da temperatura a cada minuto.
     - Estado 2: Verifica se os botões 1 e 2 são pressionados para reiniciar as estatísticas da temperatura e o contador "conta".

Abaixo segue-se o código utilizado:

```
#define temperaturePin A0    // Pino analógico para leitura da temperatura
#define yellowLed  5         // Pino para o LED amarelo
#define redLed 6
#define poten1  A1          //potenciometro 1
#define poten2 A2          //potenciometro 2
#define buzzer 4          //buzzer
#define switch1 2          // botao 1
#define switch2 3          //botao 2
#define displayPinA 7
#define displayPinB 8
#define displayPinC 9
#define displayPinD 10
#define displayPinE 11
#define displayPinF 12
#define displayPinG 13
float min_temperature = 100;
float max_temperature = 0;
float sum_temperature = 0;
bool reset =false;
int estado=1;
int temperatura(){
    static unsigned long time = 0;
    if (millis()-time>=15000)
    {
      //float temperature = map(analogRead(A0), 0, 1023, -1, 498);
       float temperature = map(analogRead(A0), 0, 1023, -49, 448);
        Serial.print(temperature);
        Serial.println("C");
      if (temperature < min_temperature) {
            min_temperature = temperature;
        }
      else if (temperature > max_temperature) {
            max_temperature = temperature;
        }
           sum_temperature += temperature;
      time=millis();
      return temperature;
    }

}
void led_intermitente(){
    static unsigned long time = 0;
    int value = map(analogRead(poten1), 0, 1023, 1000, 10000);
    if (millis() - time >= value) {
        Serial.println("entrou");
    digitalWrite(yellowLed, !digitalRead(yellowLed));
       time = millis();
    }
}
```

```
void setup() {
    for (int i = 4; i <= 13; i++){
        pinMode(i,OUTPUT);
    }
    for (int i = 2; i <= 3; i++)
        pinMode(i , INPUT_PULLUP);
    attachInterrupt(digitalPinToInterrupt(switch1), botao1, FALLING);
    Serial.begin(9600);
}
void botao1(){
    Serial.println("enu");
    digitalWrite(redLed,LOW);
}
void display(int conta){
static const byte NUMBERS[][7] = {
  {1,1,1,1,1,1,0}, // 0
  {0,1,1,0,0,0,0}, // 1
  {1,1,0,1,1,0,1}, // 2
  {1,1,1,1,0,0,1}, // 3
  {0,1,1,0,0,1,1}, // 4
  {1,0,1,1,0,1,1}, // 5
  {1,0,1,1,1,1,1}, // 6
  {1,1,1,0,0,0,0}, // 7
  {1,1,1,1,1,1,1}, // 8
  {1,1,1,1,0,1,1}  // 9
  };
  for (int i = 0; i < 7; i++){
    digitalWrite(displayPinA + i, NUMBERS[conta][i]);

  }
}
int avisos() {
    static int conta = 0;
    if (reset)
    {
        conta=0;
        display(conta);
        reset=false;
        return 1;
    }
    static unsigned long start = millis();
    int temp = map(analogRead(poten2), 0, 1023, 0, 50); // Armazena o
valor da temperatura
    if (temperatura() >= temp) {

        digitalWrite(redLed, HIGH);
        tone(buzzer, 1000,5000);
        conta++;
        display(conta);
    }
```

```
    return 1;
}
void loop() {

switch (estado)
{
case 1:
    static unsigned long time = 0;
    led_intermitente();
    if (millis()-time>60000)
    {
        Serial.println("--------------------");
        Serial.print("min: ");
        Serial.print(min_temperature);
        Serial.println("C");
        Serial.print("max: ");
        Serial.print(max_temperature);
        Serial.println("C");
        Serial.print("avg: ");
        Serial.print(sum_temperature / 4);
        Serial.println("C");
        Serial.println("--------------------");
        Serial.println();
    time = millis();

    }
    estado =2;
    break;
case 2:
    if (!digitalRead(switch1) && !digitalRead(switch2))
        {
            reset=true;
        }
    estado =avisos();
    break;
}
}
```

4. Discussão

Uma das coisas a ter em atenção neste trabalho, foi o facto da função **delay()** não ser a função mais apropriada para se utilizar neste programa, porque o que ela vai fazer é "congelar" o programa numa determinada parte do código por um tempo especificado em milissegundos e durante o período em que o código está parado, não pode ocorrer nenhuma leitura de sensores, cálculos matemáticos ou manipulação de pinos, enquanto que a função **millis()** retorna um número indicando há quantos milissegundos o Arduino está ligado , ou seja, ao invés de interromper o sistema durante um tempo determinado usando a função **delay()**, iremos trabalhar com o valor retornado pela função **millis()** e calcular indiretamente o tempo decorrido.

Outra coisa em ter em atenção é ligar o **TMP36** (sensor de temperatura) e os potenciómetros a uma porta analógica do arduino porque, as entradas digitais só podem assumir dois estados, **HIGH** e **LOW**, ou seja, 0 V ou 5 V. Dessa forma só é possível ler apenas dois estados, mas em muitas situações a variação das grandezas envolvidas acontece de forma analógica, ou seja, variam continuamente em relação ao tempo e podem assumir valores infinitos dentro de uma faixa, neste caso o valor temperatura vai assumir vários valores ao longo do tempo então nos vamos ter que ligar a uma porta analógica.

Uma das coisas que também facilitou o trabalho foi a utilização da função **map()**, ou seja, em vez de se usar formulas para se calcular a temperatura usamos a função **map()** ele converte imediatamente o valor da voltagem para temperatura.

Uma vez que vamos usar **interrupts**, mais concretamente o **attachInterrupt()** é preciso ter em atenção que esta função só pode ser usada em dois pinos do arduino o 2 e o 3.

5. Conclusão

Assim sendo podemos dizer que cumprimos com o objetivo e que este programa é um exemplo simples de sistemas de segurança, estes podem ser encontrados em fábricas, por exemplo, neste exemplo o led amarelo funciona de forma a informar os utilizadores que a medição da temperatura está a ser feita e quando por exemplo uma máquina chega a uma certa temperatura que é específica, toca não só o alarme, mas também existe um alarme visual.

Referências

Jorge, Ricardo. "Função Switch E a Máquina de Estado - State Machine." Panorama Blog Space, 29 Dec. 2020, www.panoramablog.space/blog/funcao-switch-e-a-maquina-de-estado-state-machine/.Accessed 20 Apr. 2023.

Lara, Silvio Garbes. "Função Millis() No Arduino: Aprenda Como Utilizar." MakerHero, 28 Jan. 2020, www.makerhero.com/blog/subtituindo-delay-por-millis-no-arduino/.Accessed 13 Apr. 2023.

"Millis() - Arduino Reference." Www.arduino.cc, 16 Apr. 2020, www.arduino.cc/reference/en/language/functions/time/millis/. Accessed 14 Apr. 2023.

BillWagner. "Tipo Bool - Referência C#." Learn.microsoft.com, 10 May 2023, learn.microsoft.com/pt-pt/dotnet/csharp/language-reference/builtin-types/bool.Accessed 8 June 2023.

Contador de 0 a 9 Com Display de 7 Segmentos – Blog Da Robótica. 15 May 2020, www.blogdarobotica.com/2020/05/15/contador-de-0-a-9-com-display-de-7-segmentos/ Accessed 8 June 2023.

"AttachInterrupt() - Documentação de Referência Do Arduino." Www.arduino.cc, 8 June 2021, www.arduino.cc/reference/pt/language/functions/externalinterrupts/attachinterrupt/.Accessed 8 June 2023.

"Map() - Arduino Reference." Reference.arduino.cc, 6 Sept. 2022, reference.arduino.cc/reference/en/language/functions/math/map/.Accessed 8 June 2023.

Tinkercad:
https://www.tinkercad.com/things/2gnra6JgjlH