****

**Licenciatura em Engenharia Informática**

**Tecnologia e Arquitetura de Computadores 2022/2023**

**Trabalho Prático nº 6**

**Millis()**

**Realizado em: 18/05/2023**

**Elaborado em: 19/05/2023**

**Grupo: 5**

**António Dinis - a2021157297**

**Francisco Figueiras - a2021155919**

**Mariana Magalhães - a2022147454**

**Índice**

[**1. Introdução** 3](#_Toc135518706)

[**2. Métodos** 4](#_Toc135518707)

[**3. Resultados** 5](#_Toc135518708)

[**3.1. Exercício 1** 5](#_Toc135518709)

[**3.2. Exercício 2** 7](#_Toc135518710)

[**3.3. Exercício 3** 10](#_Toc135518711)

[**3.4. Exercício 4** 13](#_Toc135518712)

[**3.5. Exercício 5** 16](#_Toc135518713)

[**4. Discussão** 19](#_Toc135518714)

[**5. Conclusão** 19](#_Toc135518715)

[**6. Referências** 20](#_Toc135518716)

**1. Introdução**

Este trabalho tem como objetivo fazer 5 exercícios utilizando a função **millis()** que 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.

**2. Métodos**

O trabalho foi realizado no decorrer das 3 horas de aula de **Tecnologia e Arquitetura de Computadores (TAC)** 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. Todos este programas foram desenvolvidos num computador com um processador AMD Ryzenn 7 5800H With Radeon Graphics e também foram usados os materiais representados nas tabelas seguintes.

|  |  |  |
| --- | --- | --- |
| Nome | Quantidade | Componente |
| U1 | 1 | Arduino Uno R3 |

Tabela - materiais do exercício 1

|  |  |  |
| --- | --- | --- |
| Nome | Quantidade | Componente |
| U1 | 1 | Arduino Uno R3 |
| D1 | 1 | blue LED |
| D2 | 1 | White LED |
| R1 | 1 | 560Ω Resistor |

Tabela - materiais do exercício 2

|  |  |  |
| --- | --- | --- |
| Nome | Quantidade | Componente |
| U1 | 1 | Arduino Uno R3 |
| PIEZO1 | 1 | Piezo |
| PIR1 | 1 | Sensor PIR |

Tabela - materiais do exercício 3

|  |  |  |
| --- | --- | --- |
| Nome | Quantidade | Componente |
| U1 | 1 | Arduino Uno R3 |
| PIEZO1 | 1 | Piezo |
| PIR1 | 1 | Sensor PIR |
| D1 | 1 | Vermelho LED |
| R1 | 1 | 560Ω Resistor |

Tabela - materiais dos exercicios 4 e 5

**3. Resultados**

**3.1. Exercício 1**

O objetivo neste exercício é medir o tempo necessário para ler/escrever de um digital INPUT.

Começámos por fazer o projeto do circuito no Tinkercad (Figura 1) e através dessa montagem foi obtido o diagrama do circuito (figura 2), em seguida foi desenvolvido um algoritmo para o desenvolvimento do código no Arduíno e por fim a sua montagem na breadbord.

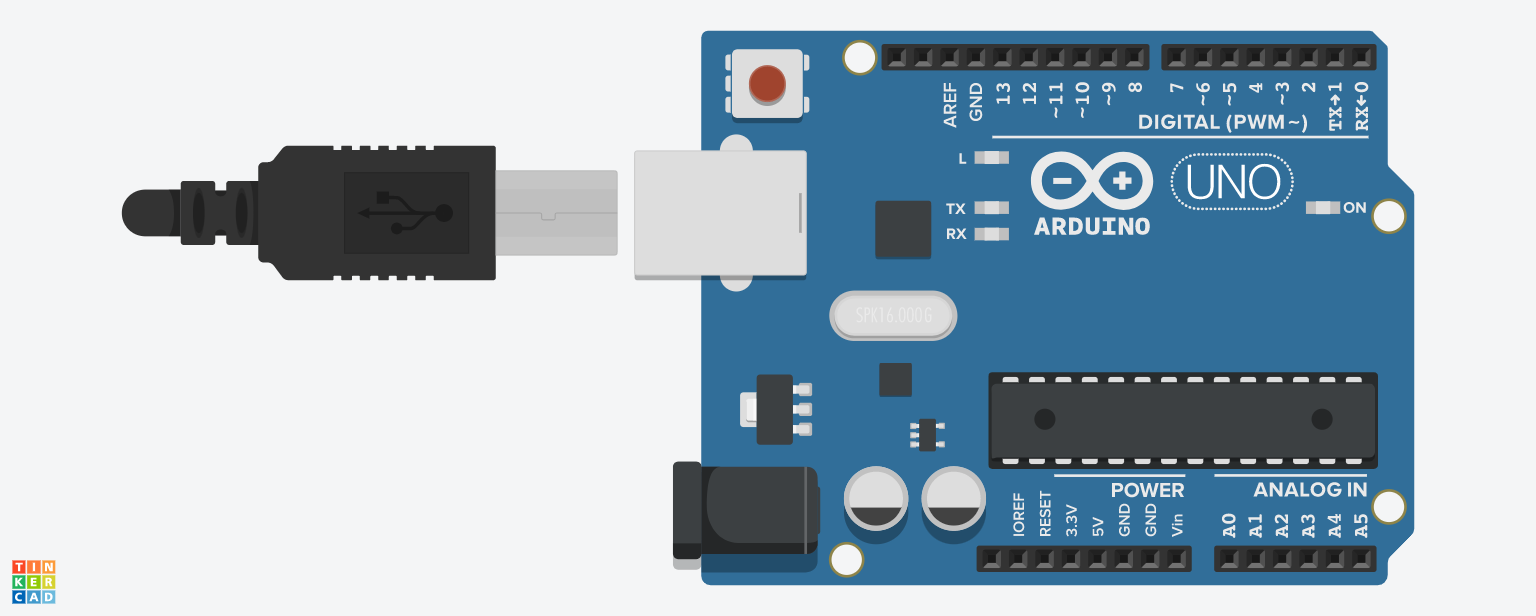


Figura - montagem no tinkercad

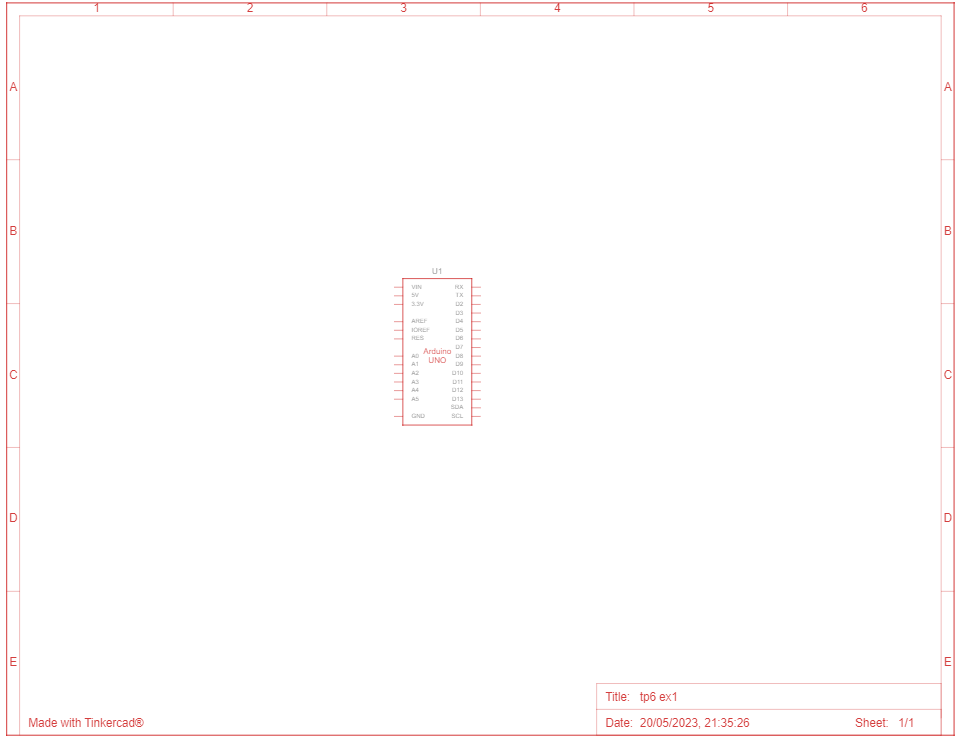


Figura - diagrama do circuito

Algoritmo do programa:

1. Inicializar as entradas e saídas necessárias
2. Mede o tempo atual em microssegundos.
3. Define o pino 4 como HIGH (liga).
4. Calcula a diferença entre o tempo atual e o tempo medido anteriormente como o tempo de escrita.
5. Imprime o tempo de escrita no monitor serial.
6. Mede o tempo atual em microssegundos.
7. Lê o estado do pino 4.
8. Calcula a diferença entre o tempo atual e o tempo medido anteriormente como o tempo de leitura.
9. Imprime o tempo de leitura no monitor serial.
10. Aguarda 1 segundo.

O código representado abaixo foi o programa utilizado, este código vai medir o tempo para se obter a velocidade de escrita e leitura em um pino digital específico.

unsigned writeTime = 0;

unsigned readTime = 0;

void setup() {

  Serial.begin(9600);

  pinMode(4, INPUT);

}

void loop() {

  unsigned long time = micros();

  digitalWrite(4, HIGH);

  writeTime = micros() - time;

  Serial.print("Write time: ");

  Serial.println(writeTime);

  time = micros();

  digitalRead(4);

  readTime = micros() - time;

  Serial.print("Read time: ");

  Serial.println(readTime);

  delay(1000);

}

**3.2. Exercício 2**

O objetivo neste exercício é piscar dois leds em frequências diferentes. Um led deve piscar a cada segundo e o outro a cada 300 milissegundos.

Começamos por fazer o projeto do circuito no Tinkercad (Figura 3) e através dessa montagem foi obtido o diagrama do circuito (figura 4), em seguida foi desenvolvido um algoritmo para o desenvolvimento do código no Arduíno e por fim a sua montagem na breadbord (figura 5).

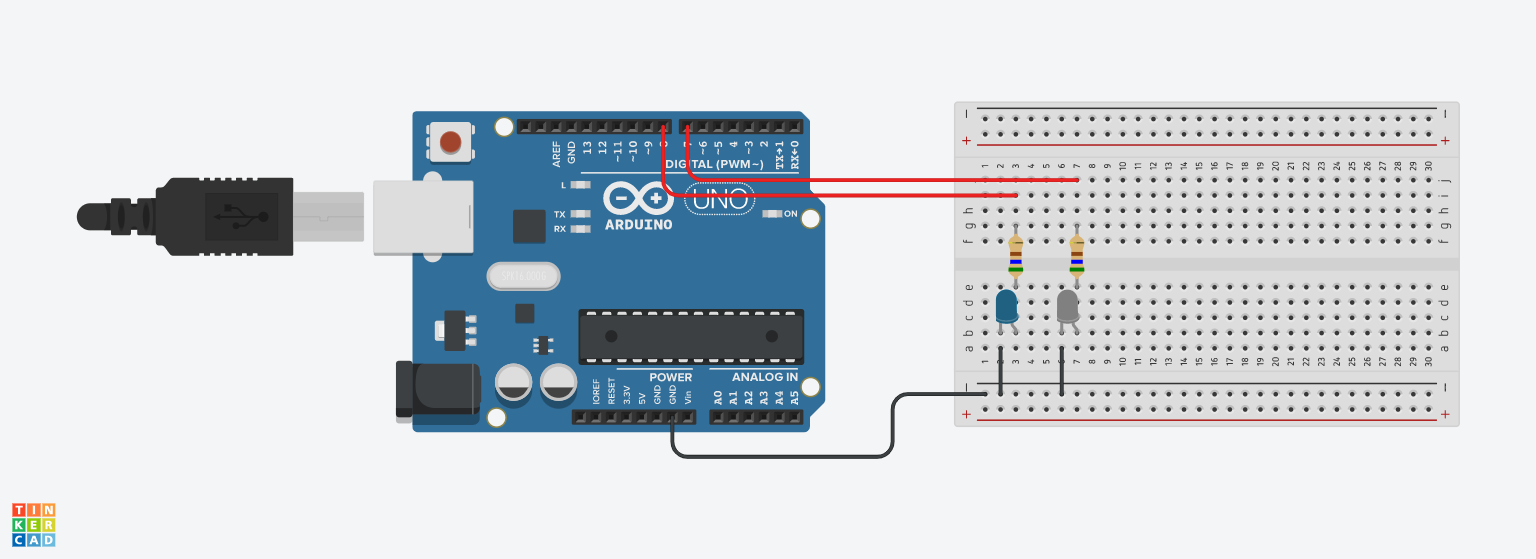


Figura - montagem tinkercad

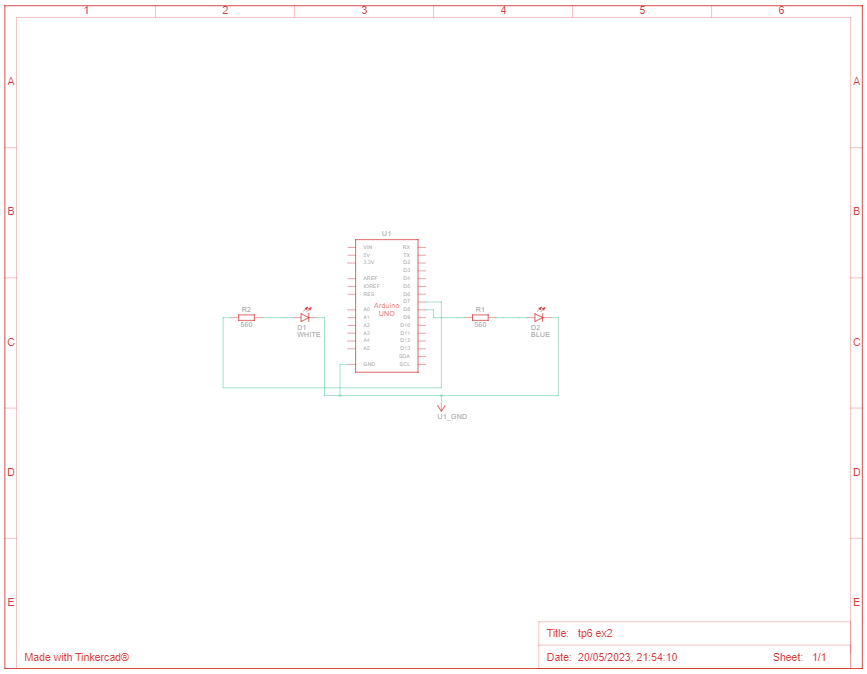


Figura - diagrama do circuito

Algoritmo do programa:

1. Declara e inicializa as variáveis `time1` e `time2` com o valor atual de `millis()`, que representa o tempo decorrido em milissegundos desde o início do programa.
2. Define a função `blink1` que controla o piscar do primeiro LED. Ela recebe dois parâmetros: o pino do LED (`pin`) e o tempo de atraso entre os estados HIGH e LOW (`delayTime`).
3. Dentro da função `blink1`, verifica se o tempo decorrido desde o último acionamento do `time1` é menor que o `delayTime`. Se for, define o pino do LED como HIGH (aceso).
4. Caso contrário, verifica se o tempo decorrido é menor que o dobro do `delayTime`. Se isso acontecer, define o pino do LED como LOW (apagado).
5. Se nenhum dos casos anteriores for verdadeiro, significa que passou tempo suficiente para o LED piscar duas vezes. Nesse caso, atualiza o `time1` com o valor atual de `millis()`.
6. Define a função `blink2` que controla o piscar do segundo LED, esta é similar à função `blink1`, mas usa a variável `time2` para controlar o tempo.
7. No método `setup()`, configura os pinos 7 e 8 como saídas (OUTPUT) para controlar os LEDs.
8. No método `loop()`, chama a função `blink1` para piscar o LED no pino 7 com um atraso de 1 segundo (1000 milissegundos) entre os estados HIGH e LOW.
9. Em seguida, chama a função `blink2` para piscar o LED no pino 8 com um atraso de 300 milissegundos entre os estados HIGH e LOW.

O código abaixo utiliza a função **millis()** para controlar o tempo e alternar os estados dos LEDs com base nos atrasos especificados.

unsigned long time1 = millis();

unsigned long time2 = millis();

void blink1(int pin, unsigned long delayTime) {

  if (millis() - time1 < delayTime) {

    digitalWrite(pin, HIGH);

  } else if (millis() - time1 < delayTime \* 2) {

    digitalWrite(pin, LOW);

  } else {

    time1 = millis();

  }

}

void blink2(int pin, unsigned long delayTime) {

  if (millis() - time2 < delayTime) {

    digitalWrite(pin, HIGH);

  } else if (millis() - time2 < delayTime \* 2) {

    digitalWrite(pin, LOW);

  } else {

    time2 = millis();

  }

}

void setup() {

  for (int i = 7; i <= 8; i++) {

    pinMode(i, OUTPUT);

  }

}

void loop() {

  blink1(7, 1000);

  blink2(8, 300);

}

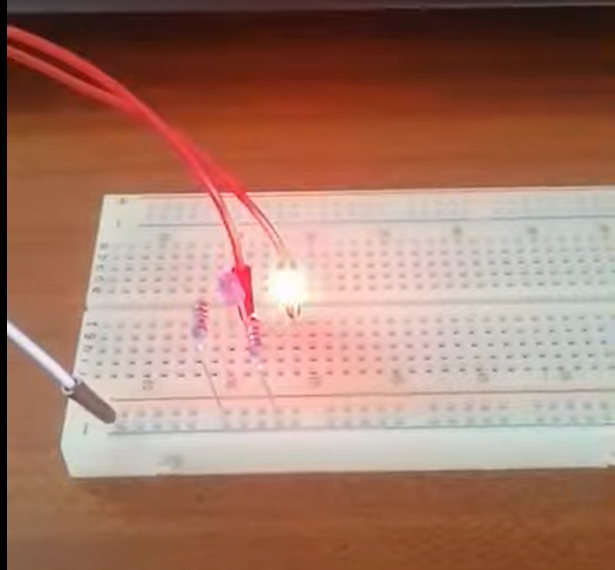


Figura - breadbord

**3.3. Exercício 3**

O objetivo neste exercício é criar um alarme de sensor de movimento. Para isso usamos um sensor PIR e quando for detetado movimento, o buzzer será ativado por 10 segundos.

Começamos por fazer o projeto do circuito no Tinkercad (Figura 6) e através dessa montagem foi obtido o diagrama do circuito (figura 7), em seguida foi desenvolvido um algoritmo para o desenvolvimento do código no Arduíno e por fim a sua montagem na breadbord (figura 8).

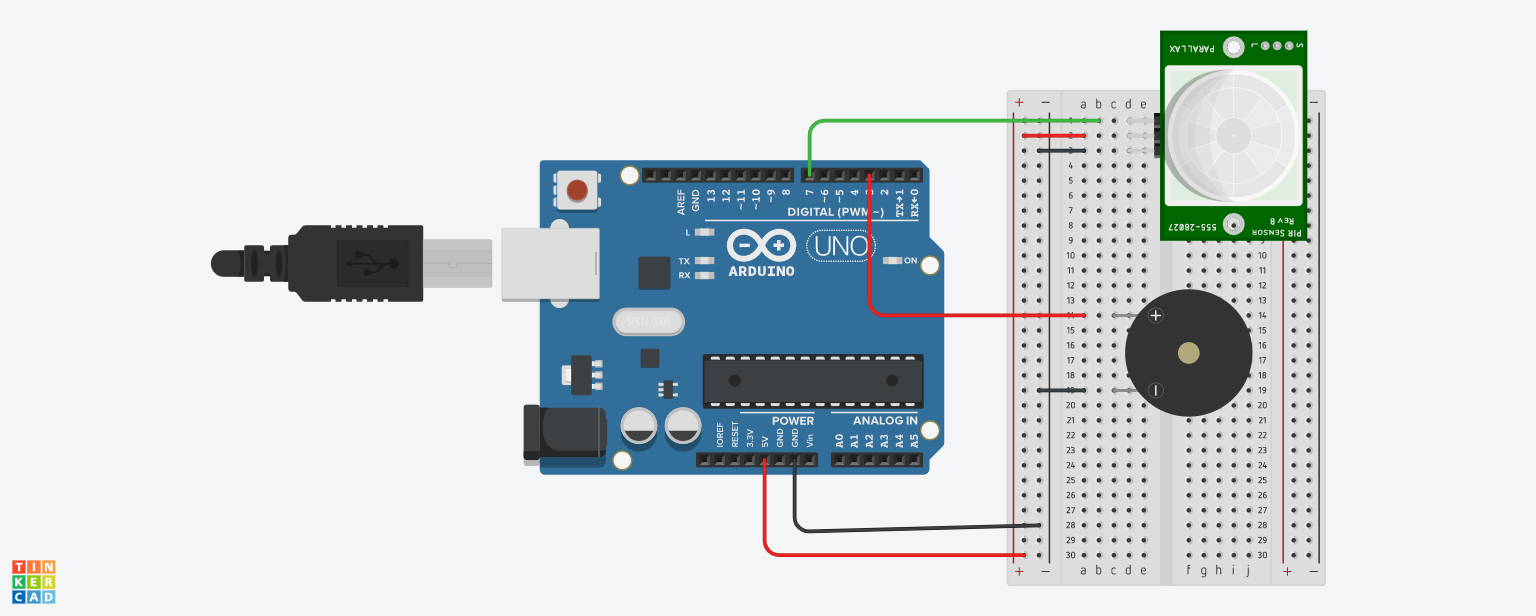


Figura - montagem tinkercad

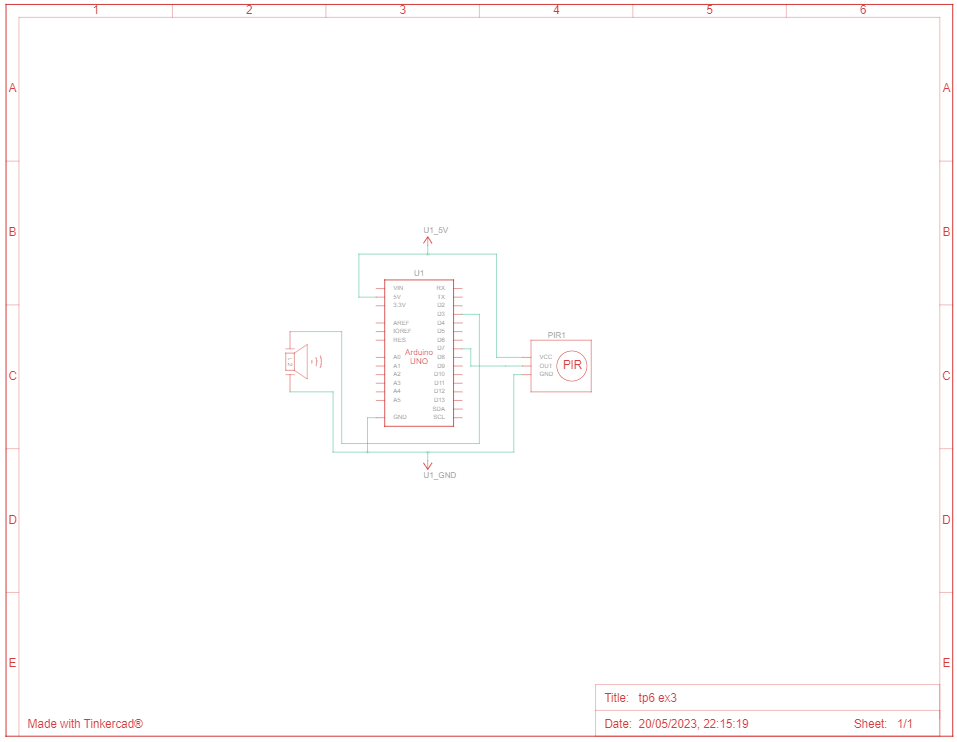


Figura - diagrama do circuito

Algoritmo do programa:

1. Declara a variável `sensor` para armazenar o estado do sensor PIR (HIGH ou LOW).

2. Declara a variável `time` para armazenar o valor de `millis()` quando o movimento é detetado.

3. Declara a variável booleana `state` para controlar o estado atual de deteção de movimento.

4. No método `setup()`, inicia a comunicação serial e define os pinos 3 (buzzer) como saída (OUTPUT) e 7 (sensor PIR) como entrada (INPUT).

5. No método `loop()`, lê o estado do sensor PIR através da função `digitalRead()` e armazena o valor na variável `sensor`.

6. Se o sensor estiver em estado HIGH, indica que o movimento foi detetado.

- Atualiza a variável `time` com o valor atual de `millis()`.

- Se `state` for false, ou seja, se não estava a detetar movimento anteriormente:

- Imprime "Motion detected!" no monitor serial.

- Atualiza `state` para true.

- Define o pino 3 (buzzer) como HIGH (ligado).

7. Caso contrário, se o sensor estiver em estado LOW, indica que não há movimento.

- Se `state` for true, ou seja, se estava a detetar movimento anteriormente:

- Imprime "Motion stopped!" no monitor serial.

- Atualiza `state` para false.

- Verifica se o tempo decorrido desde a deteção do movimento é maior ou igual a 10.000 milissegundos (10 segundos).

- Se for, define o pino 3 (buzzer) como LOW (desligado).

O código abaixo foi o utilizado, onde se verifica continuamente o estado do sensor PIR e, com base nesse estado, controla o acionamento do buzzer. Quando o movimento é detetado, o buzzer é ligado e permanece nesse estado por 10 segundos após o último movimento ser detetado pelo sensor.

int sensor;

unsigned long time;

bool state = false;

void setup() {

  Serial.begin(9600);

  pinMode(3, OUTPUT); // buzzer

  pinMode(7, INPUT); // pir sensor

}

void loop() {

  sensor = digitalRead(7);

  if (sensor == HIGH) {

    time = millis();

    if (state == false) {

      Serial.println("Motion detected!");

      state = true;

    }

    digitalWrite(3, HIGH);

  } else {

    if (state == true) {

      Serial.println("Motion stopped!");

      state = false;

    }

    if (millis() - time >= 10000) {

      digitalWrite(3, LOW);

    }

  }

}

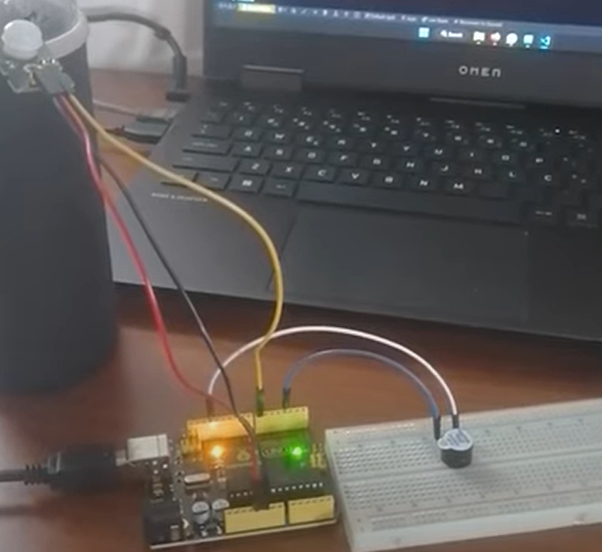


Figura - breadbord

**3.4. Exercício 4**

O objetivo neste exercício é acrescentar um led ao exercício anterior, que acende por 30 segundos ao detetar movimento.

Começamos por fazer o projeto do circuito no Tinkercad (Figura 9) e através dessa montagem foi obtido o diagrama do circuito (figura 10), em seguida foi desenvolvido um algoritmo para o desenvolvimento do código no Arduíno e por fim a sua montagem na breadbord (figura 11).

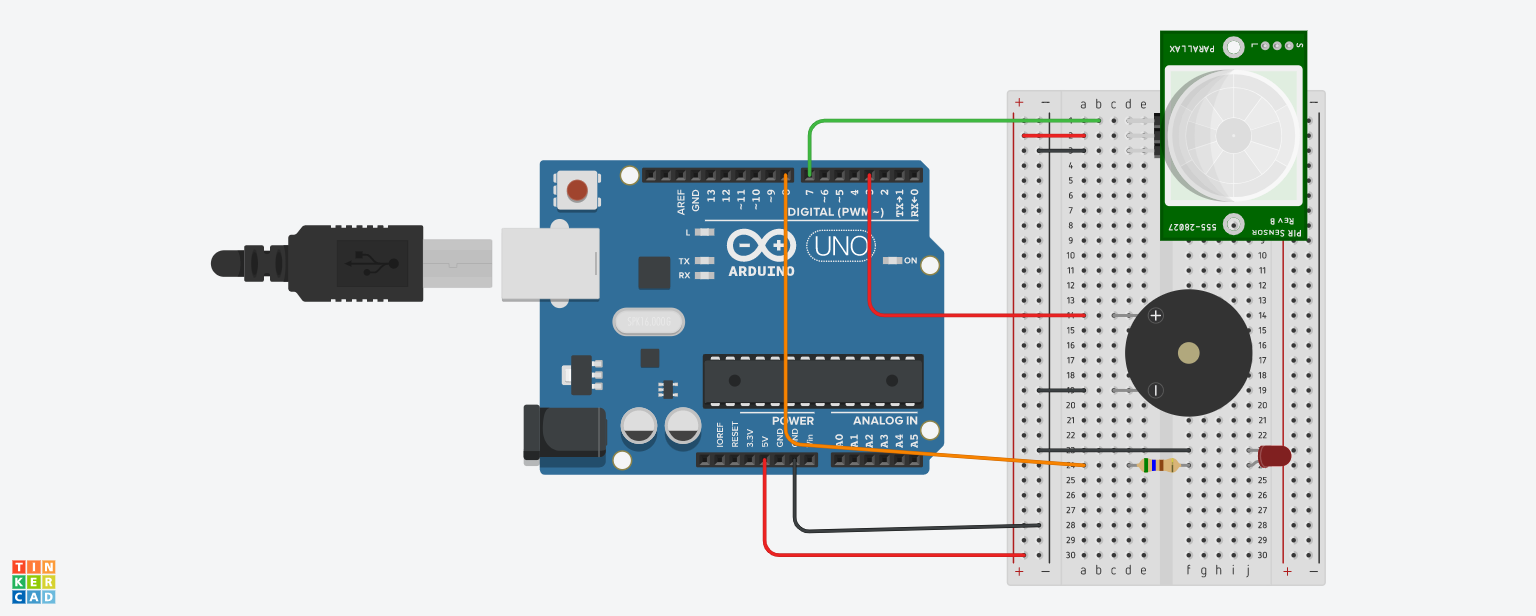


Figura - montagem tinkercad

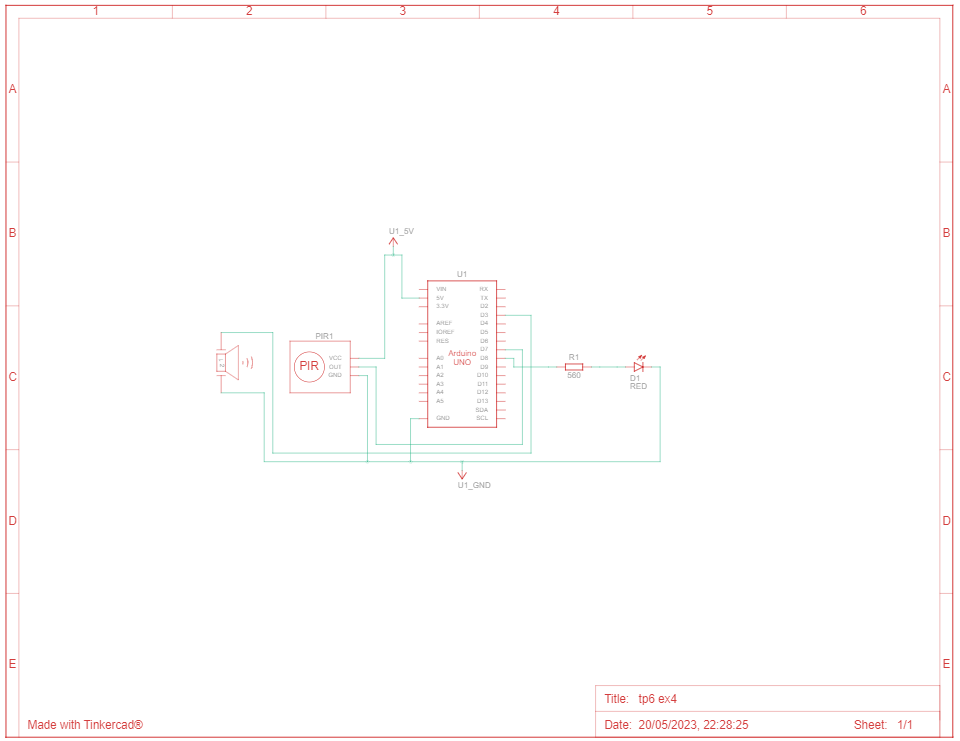


Figura - diagrama do circuito

Algoritmo do programa:

1. Declara a variável `sensor` para armazenar o estado do sensor PIR (HIGH ou LOW).

2. Declara a variável `time` para armazenar o valor de `millis()` quando algum movimento é detetado.

3. Declara a variável booleana `state` para controlar o estado atual da deteção de movimento.

4. No método `setup()`, inicia a comunicação serial e define os pinos 3 (buzzer), 7 (sensor PIR) e 8 (LED) como saída (OUTPUT).

5. No método `loop()`, lê o estado do sensor PIR através da função `digitalRead()` e armazena o valor na variável `sensor`.

6. Se o sensor estiver em estado HIGH, indica que o movimento foi detetado.

- Atualiza a variável `time` com o valor atual de `millis()`.

- Se `state` for false, ou seja, se não estava a detetar movimento anteriormente:

- Imprime "Motion detected!" no monitor serial.

- Atualiza `state` para true.

- Define os pinos 3 (buzzer) e 8 (LED) como HIGH (ligados).

7. Caso contrário, se o sensor estiver em estado LOW, indica que não há movimento.

- Se `state` for true, ou seja, se estava a detetar movimento anteriormente:

- Imprime "Motion stopped!" no monitor serial.

- Atualiza `state` para false.

- Verifica se o tempo decorrido desde a deteção do movimento é maior ou igual a 10.000 milissegundos (10 segundos).

- Se for, define o pino 3 (buzzer) como LOW (desligado).

- Verifica se o tempo decorrido desde a deteção do movimento é maior ou igual a 30.000 milissegundos (30 segundos).

- Se for, define o pino 8 (LED) como LOW (desligado).

O código abaixo representa o código utilizado, monitorando o estado do sensor PIR, controlando o acionamento do buzzer e o do LED. Quando o movimento é detetado, ambos são ligados e permanecem ligados por 10 segundos e 30 segundos, respetivamente.

int sensor;

unsigned long time;

bool state = false;

void setup() {

  Serial.begin(9600);

  pinMode(3, OUTPUT); // buzzer

  pinMode(7, INPUT); // pir sensor

  pinMode(8, OUTPUT); // led

}

void loop() {

  sensor = digitalRead(7);

  if (sensor == HIGH) {

    time = millis();

    if (state == false) {

      Serial.println("Motion detected!");

      state = true;

    }

    digitalWrite(3, HIGH);

    digitalWrite(8, HIGH);

  } else {

    if (state == true) {

      Serial.println("Motion stopped!");

      state = false;

    }

    if (millis() - time >= 10000) {

      digitalWrite(3, LOW);

    }

    if (millis() - time >= 30000) {

      digitalWrite(8, LOW);

    }

  }

}

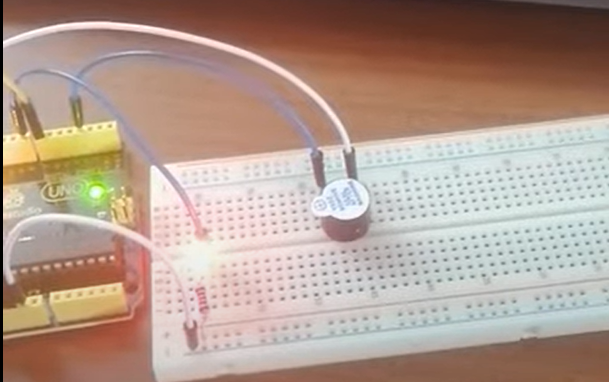


Figura - breadbord

**3.5. Exercício 5**

O objetivo deste exercício é semelhante ao anterior, porém é utilizada uma máquina de estados.

Neste exercício utilizamos o mesmo circuito (figura 12), alterando apenas o código.

Algoritmo do programa:

1. São definidos os pinos utilizados (`BUZZER\_PIN`, `PIR\_PIN` e `LED\_PIN`) e os limiares de tempo para o buzzer e LED (`TIME\_THRESHOLD\_BUZZER` e `TIME\_THRESHOLD\_LED`).

2. É definida a enumeração `DETECTION\_STATE` para representar os estados da máquina de estados.

3. São declarados os protótipos das funções utilizadas.

4. No método `setup()`, é iniciada a comunicação serial e configurados os pinos como saídas ou entradas.

5. No método `loop()`, a função `state\_machine()` é chamada repetidamente para executar a máquina de estados.

6. São definidas as funções `check\_idle()`, `check\_motion\_detected()` e `check\_motion\_stopped()` para realizar as verificações e ações correspondentes a cada estado da máquina de estados.

7. A função `state\_machine()` implementa a lógica da máquina de estados, esta utiliza uma variável estática `state` para controlar o estado atual e uma variável `time` para armazenar o tempo de deteção do movimento.

8. Dentro de `state\_machine()`, um switch-case é usado para selecionar a ação correspondente ao estado atual. Para cada estado, a função correspondente é chamada e o estado é atualizado de acordo com o resultado.

9. A função `check\_idle()` verifica se o sensor PIR detetou movimento. Se sim, imprime "Motion detected!", atualiza `time` com o valor atual de `millis()` e retorna o estado `MOTION\_DETECTED`, caso contrário, retorna ao estado `IDLE`.

10. A função `check\_motion\_detected()` verifica se o movimento foi interrompido. Se o sensor PIR não estiver a detetar movimento e o tempo decorrido desde a deteção for maior que o limiar de tempo para o LED, imprime "Motion stopped!" e retorna ao estado `IDLE`, caso contrário, controla o acionamento do buzzer e LED com base no tempo decorrido desde a deteção.

11. O programa utiliza as funções `digitalRead()` e `digitalWrite()` para ler o estado do sensor PIR e controlar os pinos do buzzer e LED.

O código implementado corresponde à máquina de estados, permitindo que o sistema responda aos eventos de deteção de movimento e realize as ações correspondentes.

#define BUZZER\_PIN 3

#define PIR\_PIN 7

#define LED\_PIN 8

#define TIME\_THRESHOLD\_BUZZER 10000ul // 10000ul = 10000 unsigned long

#define TIME\_THRESHOLD\_LED 30000ul // 30000ul = 30000 unsigned long

// --- states ---

enum class DETECTION\_STATE {

  IDLE,

  MOTION\_DETECTED

};

// --- prototypes ---

void state\_machine();

DETECTION\_STATE check\_idle(unsigned long &time);

DETECTION\_STATE check\_motion\_detected(unsigned long time);

DETECTION\_STATE check\_motion\_stopped(unsigned long time);

// --- start of the program ---

void setup() {

  Serial.begin(9600);

  pinMode(BUZZER\_PIN, OUTPUT);

  pinMode(PIR\_PIN, INPUT);

  pinMode(LED\_PIN, OUTPUT);

}

void loop() {

  state\_machine();

}

// --- functions ---

DETECTION\_STATE check\_idle(unsigned long &time) {

  bool sensor = digitalRead(PIR\_PIN);

  if (sensor) {

    Serial.println("Motion detected!");

    time = millis();

    return DETECTION\_STATE::MOTION\_DETECTED;

  }

  return DETECTION\_STATE::IDLE;

}

DETECTION\_STATE check\_motion\_detected(unsigned long time) {

  bool sensor = digitalRead(PIR\_PIN);

  if (!sensor && millis() - time > TIME\_THRESHOLD\_LED) {

    Serial.println("Motion stopped!");

    return DETECTION\_STATE::IDLE;

  }

  digitalWrite(BUZZER\_PIN, millis() - time < TIME\_THRESHOLD\_BUZZER ? HIGH : LOW);

  digitalWrite(LED\_PIN, millis() - time < TIME\_THRESHOLD\_LED ? HIGH : LOW);

  return DETECTION\_STATE::MOTION\_DETECTED;

}

void state\_machine() {

  static DETECTION\_STATE state = DETECTION\_STATE::IDLE;

  static unsigned long time = millis();

  switch (state) {

    case DETECTION\_STATE::IDLE: state = check\_idle(time);

      break;

    case DETECTION\_STATE::MOTION\_DETECTED: state = check\_motion\_detected(time);

      break;

    default: Serial.println("ERROR: Invalid state");

      break;

  }

}

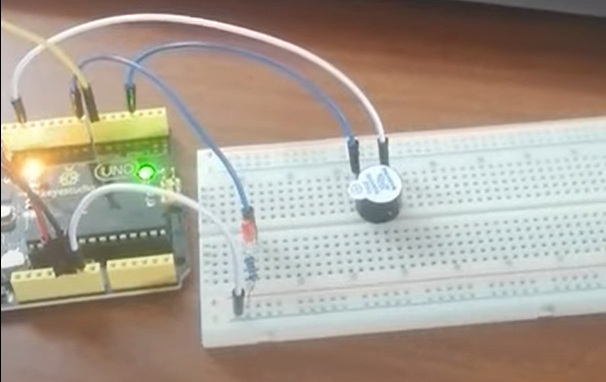


Figura – breadbord

**4. Discussão**

Neste trabalho um dos problemas que surgiu foi com sensor de movimento que estava sempre a detetar movimento, o que depois fazia com que o buzzer estivesse sempre a tocar, sendo assim e uma vez que tínhamos material em casa decidimos fazer a montagem com esse material e aí sim obtivemos resultados esperados.

**5. Conclusão**

Assim sendo podemos concluir que cumprimos os objetivos de todos os exercício e que foram bons exemplos de como usar a função **millis()** e a maquina de estados.

**6. Referências**

Lara, Silvio Garbes. “Função Millis() No Arduino: Aprenda Como Utilizar.” MakerHero, 28 Jan. 2020, [www.makerhero.com/blog/subtituindo-delay-por-millis-no-arduino/](http://www.makerhero.com/blog/subtituindo-delay-por-millis-no-arduino/).Accessed 18 May 2023.

Como Utilizar O Sensor de Presença/Movimento HC-SR501 PIR Com Arduino – Blog Da Robótica. 30 June 2022, [www.blogdarobotica.com/2022/06/30/como-utilizar-o-sensor-de-presenca-movimento-hc-sr501-pir-com-arduino/](http://www.blogdarobotica.com/2022/06/30/como-utilizar-o-sensor-de-presenca-movimento-hc-sr501-pir-com-arduino/).Accessed 18 May 2023.

Vídeos:

<https://www.youtube.com/playlist?list=PLhFikcfQ_ihFoGx9bbTMN5HVwPROLVBga>