MICROPROCESSADORES E SISTEMAS EMBEBIDOS

LAB. 9

André Silva, Gabriel Medeiros e Rui Correia

CIRCUITO FÍSICO:

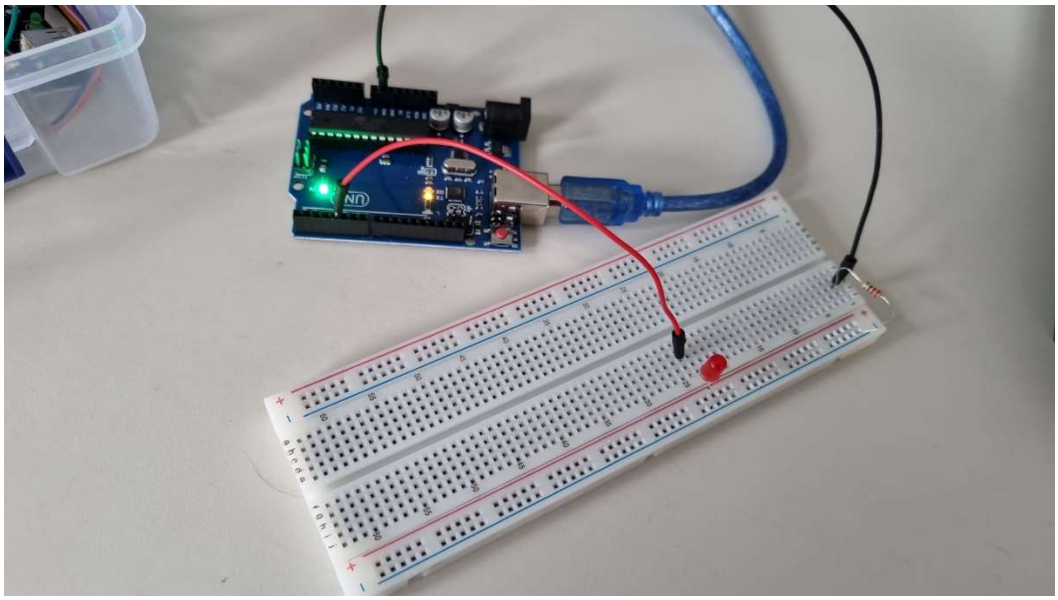
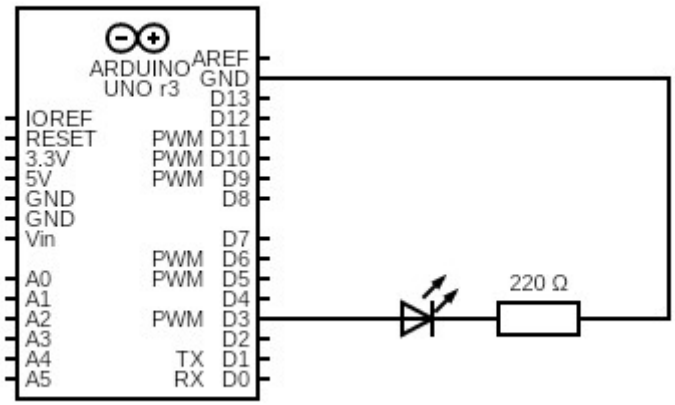


DIAGRAMA DO CIRCUITO:



CÓDIGO 1:

```
// ###################
// 0. Bibliotecas
// ###################

// Incluir a biblioteca TimerOne
#include <TimerOne.h>

// ###################
// 1. Variaveis
// ###################

// 1.1 Pino do LED <- 3
#define pinLed 3
// 1.2 Estado do LED <- HIGH
volatile bool stateLed = HIGH;

// ###################
// 2. Função Setup
// ###################

void setup()
{
  // 2.1 Definir a taxa de comunicação em bits/s
  Serial.begin(9600);
  // 2.2 Inicializar o Timer1 com o delay desejado
  Timer1.initialize(1000000);
  // 2.3 Associar a Função Pisca()
  Timer1.attachInterrupt(piscaLed);
}

// ###################
// 3. Função Loop
// ###################

void loop()
{

}

// ###################
// 4. Função Pisca()
// ###################

void piscaLed(){
  // 4.1 Acende ou Apaga o LED (dependendo do estado)
  digitalWrite(pinLed, stateLed);
  // 4.2 Inverte o valor do estado
  stateLed = !stateLed;
}
```

CÓDIGO 2:

```
// ####################
// 0. Bibliotecas
// ####################

// Incluir a biblioteca TimerOne
#include <TimerOne.h>

// ####################
// 1. Variaveis
// ####################

// 1.1 Pino do LED <- 3
#define pinLed 3
// 1.2 Estado do LED <- HIGH
volatile bool stateLed = HIGH;
// 1.3 Contador = 0
unsigned long cont = 0;
// 1.4 Tempo atual <- 0
unsigned long time_now = 0;
// 1.5 alarm_cont <- 0
unsigned long alarm_cont = 0;
// 1.6 Delay_cont <- 1000
int delay_cont = 1000;

// ####################
// 2. Função Setup
// ####################

void setup()
{
  // 2.1 Definir a taxa de comunicação em bits/s
  Serial.begin(9600);
  // 2.2 Inicializar o Timer1 com o delay desejado
  Timer1.initialize(1000000);
  // 2.3 Associar a Função Pisca()
  Timer1.attachInterrupt(piscaLed);
}

// ####################
// 3. Função Loop
// ####################

void loop()
{
  // 3.1 time_now <- millis()
  time_now = millis();

  // 3.1 Incrementar a variável cont
  cont++;

  // 3.2 SE time_now for maior ou igual a alarm_cont, ENTAO:
  if(time_now >= alarm_cont)
  {
    // 3.2.1 alarm_cont = alarm_cont + delay_cont
    alarm_cont += delay_cont;
    // 3.2.2 Imprimir cont
    Serial.println(cont);
    // 3.2.3 Resetar cont
    cont = 0;
  }
}

// ####################
// 4. Função Pisca()
// ####################

void piscaLed(){
  // 4.1 Acende ou Apaga o LED (dependendo do estado)
  digitalWrite(pinLed, stateLed);
  // 4.2 Inverte o valor do estado
  stateLed = !stateLed;
}
```

Resultado:

```
318135
317809
318135
317809
318135
```

CÓDIGO 3:

```
// ####################
// 0. Bibliotecas
// ####################

// Incluir a biblioteca TimerOne
#include <TimerOne.h>

// ####################
// 1. Variaveis
// ####################

// 1.1 Pino do LED <- 3
#define pinLed 3
// 1.2 Estado do LED <- HIGH
volatile bool stateLed = HIGH;
// 1.3 Contador = 0
volatile long cont = 0;

// ####################
// 2. Função Setup
// ####################

void setup()
{
  // 2.1 Definir a taxa de comunicação em bits/s
  Serial.begin(9600);
  // 2.2 Inicializar o Timer1 com o delay desejado
  Timer1.initialize(1000000);
  // 2.3 Associar a Função Pisca()
  Timer1.attachInterrupt(piscaLed);
}

// ####################
// 3. Função Loop
// ####################

void loop()
{
  // 3.1 Incrementar a variável cont
  cont++;
}

// ####################
// 4. Função Pisca()
// ####################

void piscaLed(){
  // 4.1 Acende ou Apaga o LED (dependendo do estado)
  digitalWrite(pinLed, stateLed);
  // 4.2 Inverte o valor do estado
  stateLed = !stateLed;
  // 4.3 Imprimir cont
  Serial.println(cont);
  // 4.4 Resetar a variável cont
  cont = 0;
}
```

Resultado:

```
662662
662617
1325006
1987454
2649898
```

Média = 1457527