

Sistemas Embarcados

Desenvolvimento de códigos

Tiago Piovesan Vendruscolo



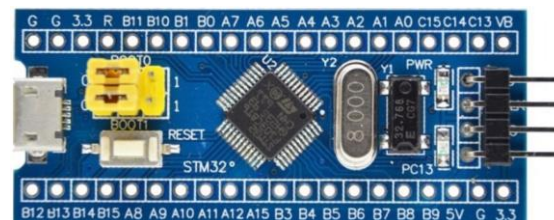
Esta licença permite que outros remixem, adaptem e criem a partir do trabalho para fins não comerciais, desde que atribuam o devido crédito aos autores originais. [4.0 international](https://creativecommons.org/licenses/by-nc-nd/4.0/)

Conversores A/D – leitura analógica



Arduíno Uno:

- Faixa dinâmica: 0 – 5V;
- ADC: 10 bits;
- Resolução: $\frac{5V}{2^{10}-1} = 4,88 \text{ mV}$



Blue Pill ARM Cortex-M3:

- Faixa dinâmica: 0 – 3.3V;
- ADC: 12 bits;
- Resolução: $\frac{3.3V}{2^{12}-1} = 0,8 \text{ mV}$

Conversores A/D

- Exemplo 1: Faça a leitura da porta analógica A0 (PA0) e imprima na serial. Mantenha um LED em 1Hz no pino 7

```
#define LED0 7
#define potenciometro A0

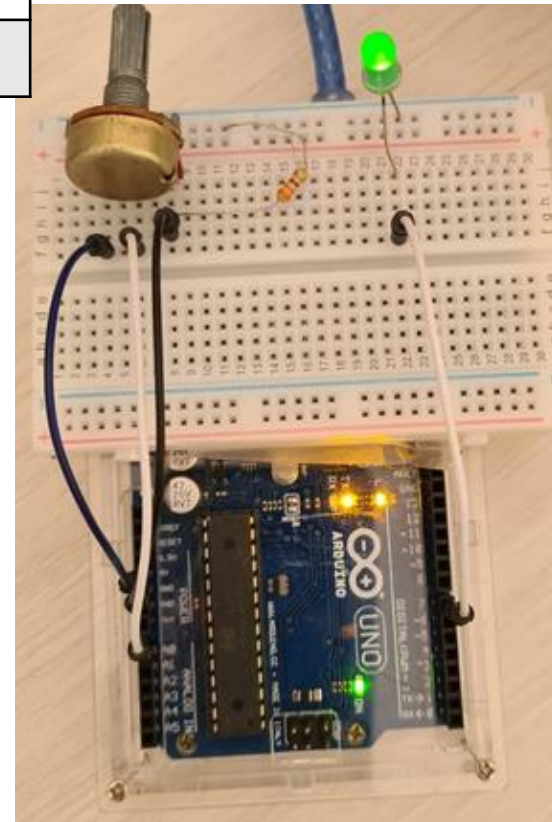
int pot_valor = 0;

void setup() {
  Serial.begin(9600);
  pinMode(LED0, OUTPUT);
}

void loop() {
  pot_valor = analogRead(potenciometro);
  digitalWrite(LED0, !digitalRead(LED0));
  delay(500);
  Serial.println(pot_valor);
}
```

Pin Potenciômetro	Local
1	5 V
2 (meio)	A0
3	GND

As entradas analógicas variam do pino A0 até A5 para o Arduino UNO.



Conversores A/D

- Fazendo dessa forma, o valor de saída do AD será sempre entre 0 até 1023, pois a resolução padrão é de 10 bits. Caso você esteja utilizando um microcontrolador com um conversor AD com mais de 10 bits, a resolução pode ser configurada da seguinte forma:

```
void setup() {  
    Serial.begin(9600);  
    pinMode(LED0, OUTPUT);  
    → analogReadResolution(12);  
}
```

- Nesse caso, a resolução foi alterada para 12 bits. Essa função não funciona com o arduíno UNO.

Gerando um sinal PWM

- Exercício 1: Faça a intensidade do LED0 variar de acordo com a leitura do potenciômetro. Utilize a saída de PWM do Arduino. A função utilizada é:

```
analogWrite(PWM_LED1, pot_valor);
```

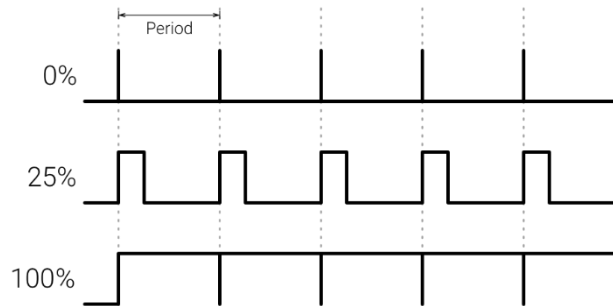
- Onde o primeiro parâmetro é o pino de saída, que pode ser os pinos 3, 5, 6, 9, 10, 11.

```
#define PWM_LED1 9
#define potenciometro A0

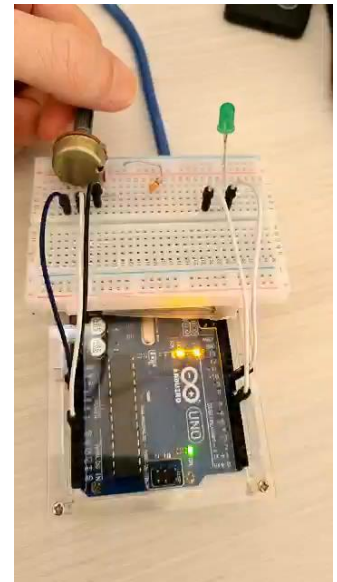
int pot_valor = 0;

void setup() {
  Serial.begin(9600);
}

void loop() {
  pot_valor = analogRead(potenciometro);
  Serial.println(pot_valor);
  analogWrite(PWM_LED1, pot_valor);
}
```



Funcionou corretamente?



Gerando um sinal PWM

- O Arduino UNO tem a frequência do PWM fixa em 244Hz e resolução de 8bits (0 – 255). Como o AD tem resolução de 10 bits (0 – 1023), por isso o controle do LED não funciona corretamente. Podemos utilizar a função MAP para deixar a saída do AD proporcional ao PWM.
- Exemplo 2: Adicione a função MAP.

```
#define PWM_LED1 9  
#define potenciometro A0
```

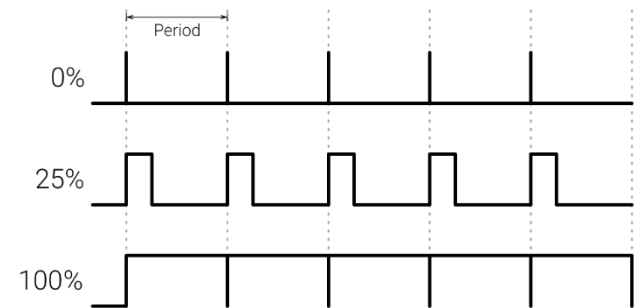
```
int pot_valor = 0;
```

```
void setup() {  
  Serial.begin(9600);  
}
```

```
void loop() {  
  pot_valor = analogRead(potenciometro);  
  pot_valor = map(pot_valor, 0, 1023, 0, 255);  
  Serial.println(pot_valor);  
  analogWrite(PWM_LED1, pot_valor);  
}
```

$$\text{Map} = (\text{val} - \text{in_min}) * (\text{out_max} - \text{out_min}) / (\text{in_max} - \text{in_min}) + \text{out_min};$$

Map(valor de entrada, menor valor da entrada, maior valor da entrada, menor valor da saída, maior valor da saída)



Gerando um sinal PWM

- Adicione um delay de 1 segundo no código conforme abaixo, como ficou o comportamento? Por que o delay não afetou o PWM?

```
void loop() {  
    pot_valor = analogRead(potenciometro);  
    pot_valor = map(pot_valor, 0, 1023, 0, 255);  
    Serial.println(pot_valor);  
    analogWrite(PWM_LED1, pot_valor);  
    delay(1000);  
}
```

Gerando um sinal PWM

- Dependendo qual microcontrolador utilizado, é possível alterar a frequência e a resolução facilmente com as funções abaixo: (não funciona para o Arduino UNO)

```
analogWriteFrequency(1000);  
analogWriteResolution (12);
```

- Para o Arduino UNO, é possível alterar a resolução para 10 bits fazendo alterações nos registradores dos TIMERS

```
void setup()  
{  
  // Set pwm clock divider  
  TCCR1B &= ~(1 << CS12);  
  TCCR1B |= (1 << CS11);  
  TCCR1B &= ~(1 << CS10);  
  
  // Set pwm resolution to mode 7 (10 bit)  
  
  TCCR1B &= ~(1 << WGM13);    // Timer B clear bit 4  
  TCCR1B |= (1 << WGM12);    // set bit 3  
  
  TCCR1A |= (1 << WGM11);    // Timer A set bit 2  
  TCCR1A |= (1 << WGM10);    // set bit 1  
}
```


Gerando um sinal PWM

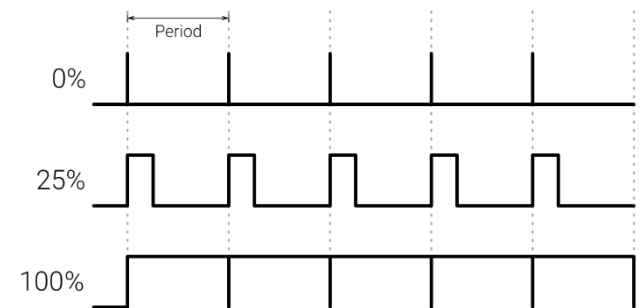
- Caso o microcontrolador não tenha mais PWM disponível, é possível fazer um novo canal via software.
- Exercício 2: Faça a intensidade do LED0 variar de acordo com a leitura do potenciômetro simulando um PWM. Use a função `delayMicroseconds`.

```
#define PWM_software_LED1 9
#define potenciometro A0

int pot_valor = 0;

void setup() {
    pinMode(PWM_software_LED1, OUTPUT);
}

void loop() {
    pot_valor = analogRead(potenciometro);
    digitalWrite(PWM_software_LED1, HIGH);
    delayMicroseconds(pot_valor);
    digitalWrite(PWM_software_LED1, LOW);
    delayMicroseconds(1023 - pot_valor);
}
```



Como o ciclo do PWM tem 1023 microssegundos, teremos a frequência do PWM em torno de 977 Hz

Utilizando o Timer

- O Arduino UNO possui 3 timers, sendo que o Timer1 e Timer3 são de 8 bits, enquanto que o Timer2 possui 16 bits. Cada timer possui funções específicas atreladas, então deve-se tomar cuidado qual timer irá utilizar para não prejudicar as funções. As principais funções são:
 - Timer0: `delay()`, `millis()` e `micros()`,
 - Timer1: biblioteca `servos`, para servos motores,
 - Timer2: função `tone`.
- A forma mais prática de utilizar o timer é através da biblioteca `TimerOne`. Sendo que todas as funções da biblioteca podem ser vistas [aqui](#). Essa biblioteca não tem por padrão na IDE Arduino e deve ser instalada. Essa biblioteca trabalha com o Timer1.
- Para instalar a biblioteca: Ferramentas > gerenciar bibliotecas
- Para utilizar a biblioteca, deve-se incluir
 - `#include <TimerOne.h>`

Utilizando o Timer

- Serão utilizadas as seguintes funções:
 - Inicializa o timer com um tempo padrão de 1 Segundo, pode ser alterado.

```
Timer1.initialize();
```

- Tempo de contagem, minimo 1 microsegundo e maximo 8,3 segundos.

```
Timer1.setPeriod(1000000);
```

- Função que será chamada quando atingir a contagem

```
Timer1.attachInterrupt(timer1segundo);
```

Utilizando o Timer

- Exemplo 3: Faça um timer que incremente um contador a cada 1 segundo e imprima a contagem na serial.

```
#include <TimerOne.h>

int contador =0;

void timer1segundo();

void setup() {
    Serial.begin(9600);
    Timer1.initialize();
    Timer1.setPeriod(1000000); // minimo 1 microsegundos e maximo 8,3 segundos
    Timer1.attachInterrupt(timer1segundo);
}

void loop() {
}

void timer1segundo () {
    contador++;
    Serial.println(contador);
}
```

Código no MOODLE

Utilizando o Timer

- Exercício 3: Faça um código em que quando um botão (interrupção – pino 2) for pressionado, o Timer conte até 5 segundos e mude o estado lógico de um LED no pino 7. Dica: Utilize a função `Timer1.stop()`.

```
#include <TimerOne.h>

#define LED0 7
#define BOTAO 2

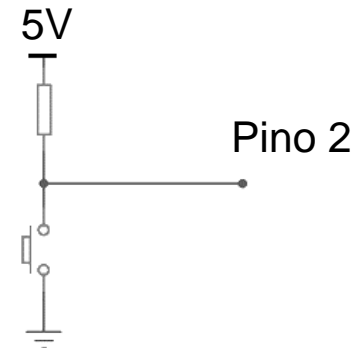
void aciona_led();
void timer1segundo();

void setup() {
    pinMode(LED0, OUTPUT);
    pinMode(BOTAO, INPUT_PULLUP);
    Serial.begin(9600);
    attachInterrupt(digitalPinToInterrupt(BOTAO), aciona_led, FALLING);
    Timer1.attachInterrupt(timer1segundo);
    Timer1.stop();
}

void loop() {
}

void aciona_led () {
    Timer1.initialize(5000000);
    Serial.println("botao pressionado");
}

void timer1segundo () {
    Timer1.stop();
    digitalWrite(LED0, !digitalRead(LED0));
    Serial.println("LED acionado");
}
```



Utilizando o Arduino – Serial

- Funções para a leitura da serial:
 - *Utilizado para verificar se possui uma nova informação na serial:*

```
Serial.available()
```

```
if(Serial.available()>0)
```

- *Função para ler a serial:*

```
Serial.read()
```

- *Função para ler um integer:*

```
Serial.parseInt();
```

```
x = Serial.parseInt();
```

- *Função para ler uma String*

```
x = Serial.readString();
```

Cuidado ao utilizar esses tipos de funções, pois ele ocupa o processador até ser concluído ou estourar o timeout.

Por padrão, tem o timeout de 1 segundo para empacotar a String (isso gera um atraso de 1 segundo no código). Pode ser alterado com `Serial.setTimeout(ms)` no setup. Na velocidade padrão de 9600, sendo 1 caractere = 8bit, pode receber até 1200 caracteres por segundo. Para receber mais, precisa aumentar o timeout, ou aumenta a velocidade de transmissão.

Utilizando o Arduino – Serial

- Exercício 4: Refaça o exercício 3, no entanto, no lugar do botão, inicie o Timer ao digitar ON na serial. Quando digitar OFF, irá desligar o LED. Teste diferentes valores para a função Serial.setTimeout(ms).

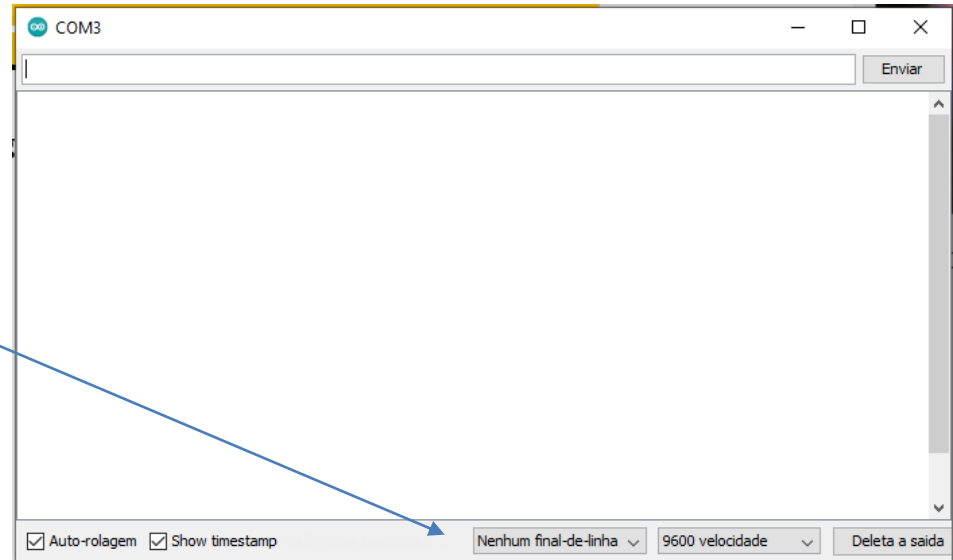
```
#include <TimerOne.h>

#define LED0 7
String x;
void aciona_led();
void timer1segundo();

void setup() {
    pinMode(LED0, OUTPUT);
    Serial.begin(9600);
    Timer1.attachInterrupt(timer1segundo);
    Timer1.stop();
    Serial.setTimeout(50);
}
```

Utilizando o Arduino – Serial

```
void loop() {  
  if (Serial.available() > 0) {  
    x = Serial.readString();  
    Serial.println(x);  
  }  
  if (x == "ON") {  
    aciona_led();  
    x = "";  
  }  
  if (x == "OFF") {  
    digitalWrite(LED0, LOW);  
    x = "";  
  }  
}  
  
void aciona_led () {  
  Timer1.initialize(5000000);  
  Serial.println("Timer iniciado");  
}  
  
void timer1segundo () {  
  Timer1.stop();  
  digitalWrite(LED0, !digitalRead(LED0));  
  Serial.println("LED acionado");  
}
```



Caso utilize os terminadores, por exemplo, nova linha e CR (carriage return), o código ficaria

```
if (x == "ON\r\n") {  
  aciona_led();  
  x = "";  
}
```

Ambos, NL e CR


Utilizando o Arduino – Serial

- Nesse caso, ele só funciona se receber ON ou OFF em maiúsculas. Para funcionar também para minúscula, pode adicionar o `.equalsIgnoreCase`

Para consultar mais funcionalidades

<https://www.arduino.cc/reference/en/language/variables/data-types/stringobject/>

```
void loop() {  
  if(Serial.available() > 0) {  
    x = Serial.readString();  
    Serial.println(x);  
  }  
  if (x == "ON") {  
    aciona_led();  
    x="";  
  }  
  if (x == "OFF") {  
    digitalWrite(LED0, LOW);  
    x="";  
  }  
}
```



```
  if (x.equalsIgnoreCase("ON")) {  
    aciona_led();  
    x="";  
  }  
  if (x.equalsIgnoreCase("OFF")) {  
    digitalWrite(LED0, LOW);  
    x="";  
  }  
}
```