

# Microprocessadores e Microcontroladores

## Introdução ao Arduino

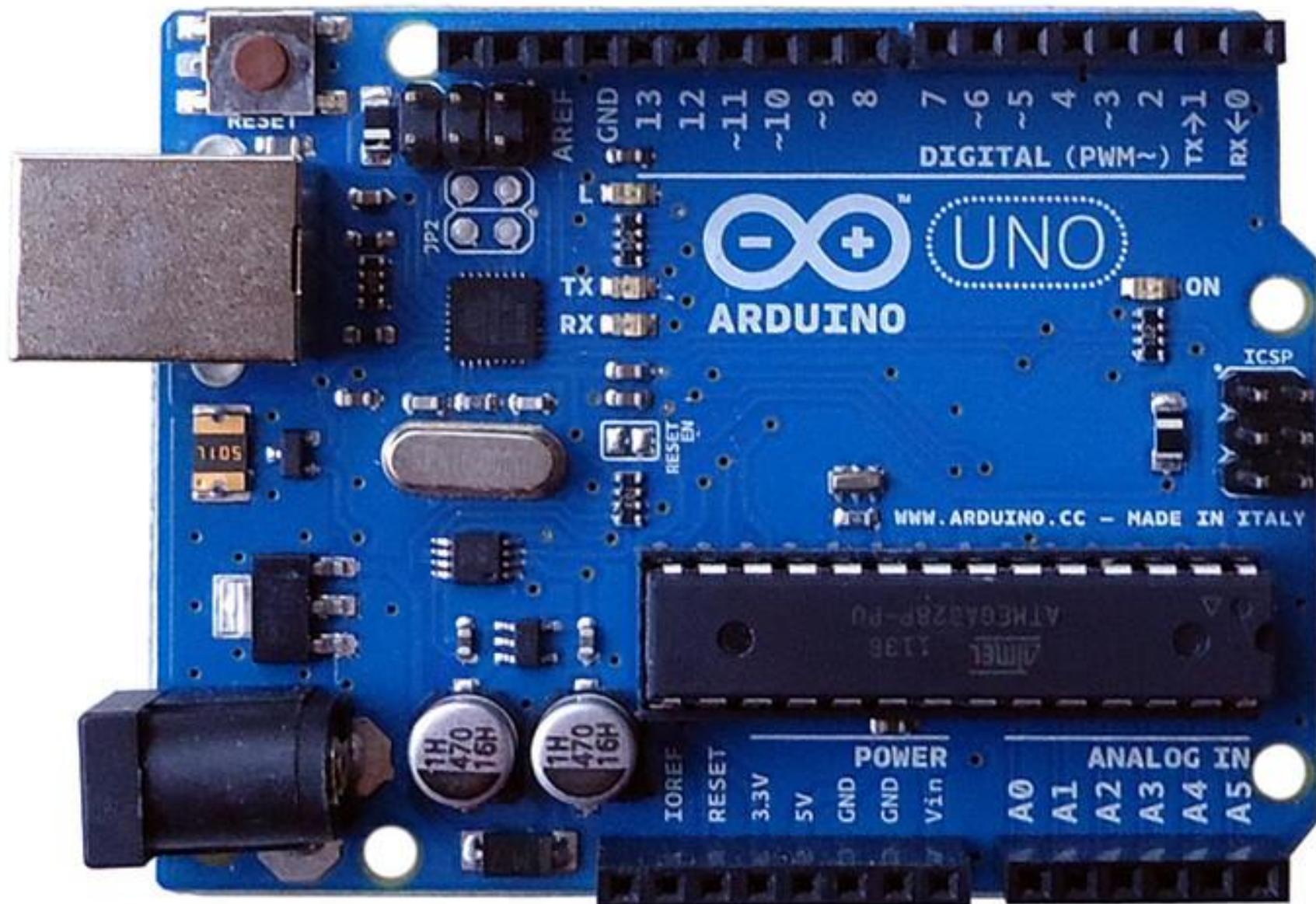
José Tarcísio Franco de Camargo

# Estudo do Arduino Uno R3

# Introdução ao Arduino Uno R3

- O que é?
  - É uma plataforma de prototipação microcontrolada de baixo custo.
- Qual microcontrolador é utilizado?
  - ATMEL ATmega328
- Quais são as entradas e saídas?
  - 6 portas de entrada analógicas.
  - 14 portas digitais configuráveis como entrada/saída, PWM ou comunicação serial.
  - 2 terminais de comunicação serial padrão I2C.

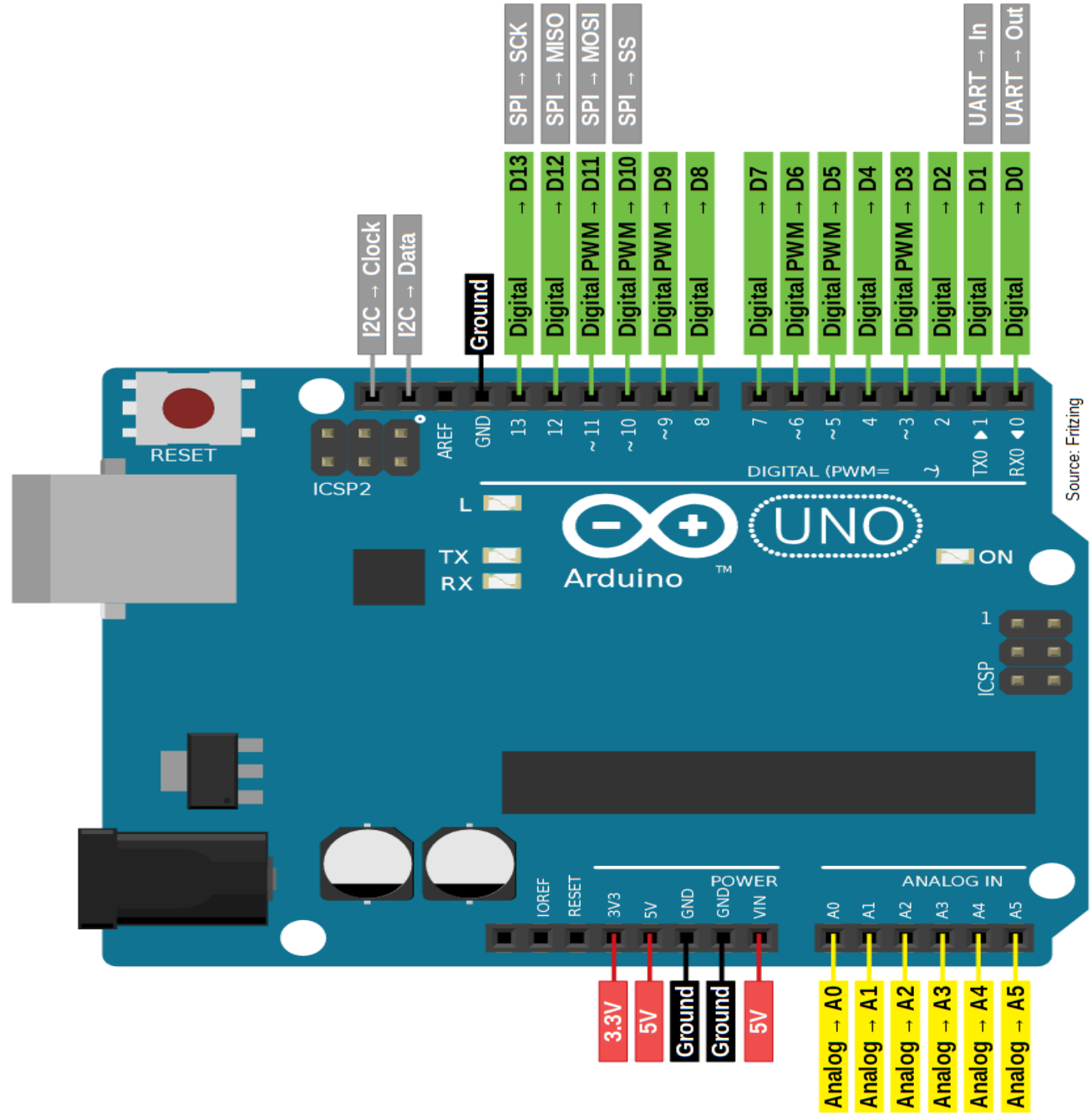
# Visão geral



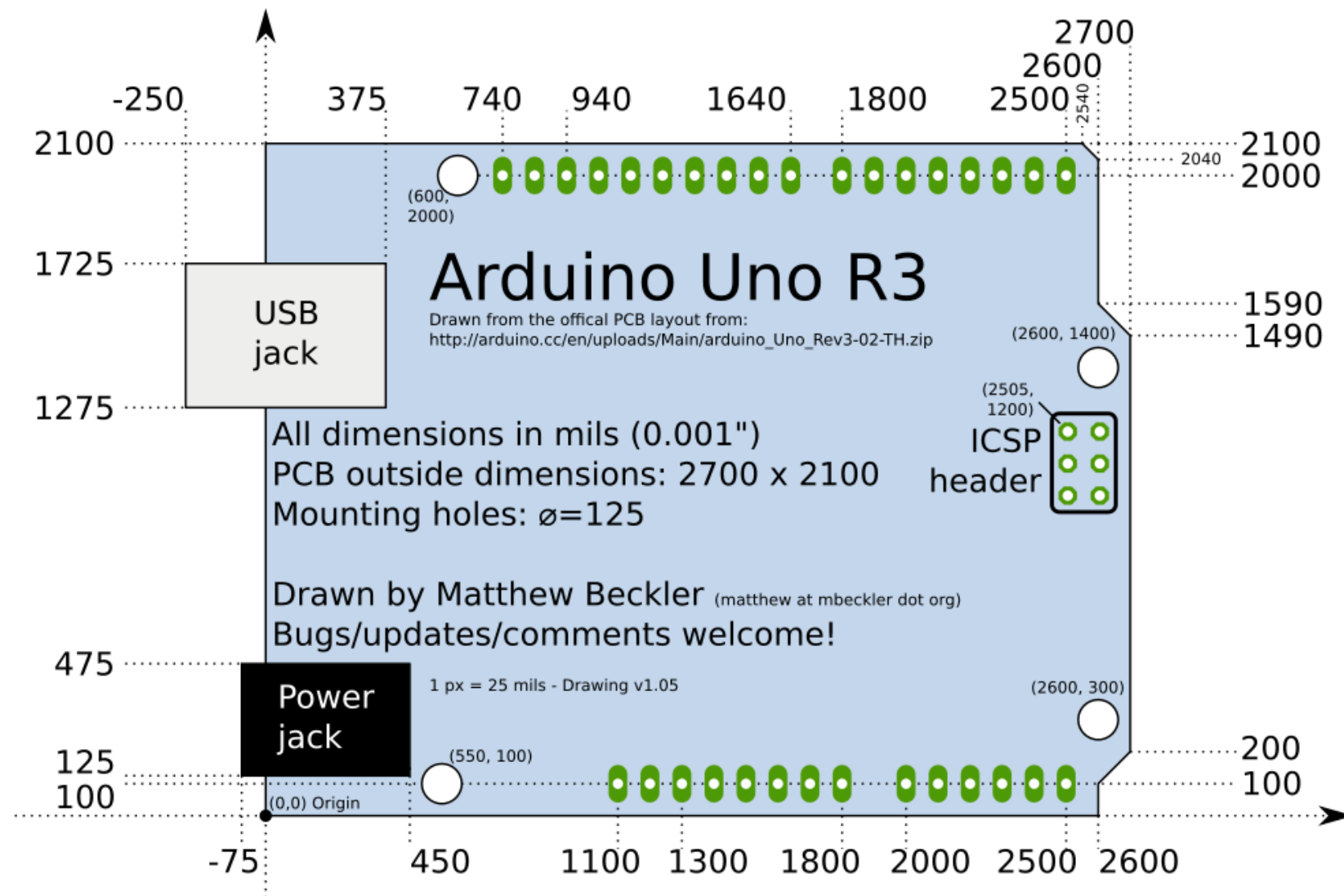
# Visão geral



# Visão geral



# Dimensões



# Pinos

- 14 pinos de entrada e saída digital (pinos 0-13):
  - Esses pinos podem ser utilizados como entradas ou saídas digitais de acordo com a necessidade do projeto.
  - Desses 14 pinos, 6 deles (pinos 3, 5, 6, 9, 10 e 11) podem ser programados para saídas PWM (Modulação por Largura de Pulso).
- 6 pinos de entradas analógicas (pinos A0 - A5):
  - Esses pinos são dedicados a receber valores analógicos, por exemplo, a tensão de um sensor. O valor a ser lido deve estar na faixa de 0 a 5 volts, que serão convertidos para valores entre 0 e 1023.
- A alimentação da placa pode ser feita a partir da porta USB do computador ou através de um adaptador AC. Para o adaptador AC recomenda-se uma tensão de 9 a 12 volts.



# Usando os pinos digitais

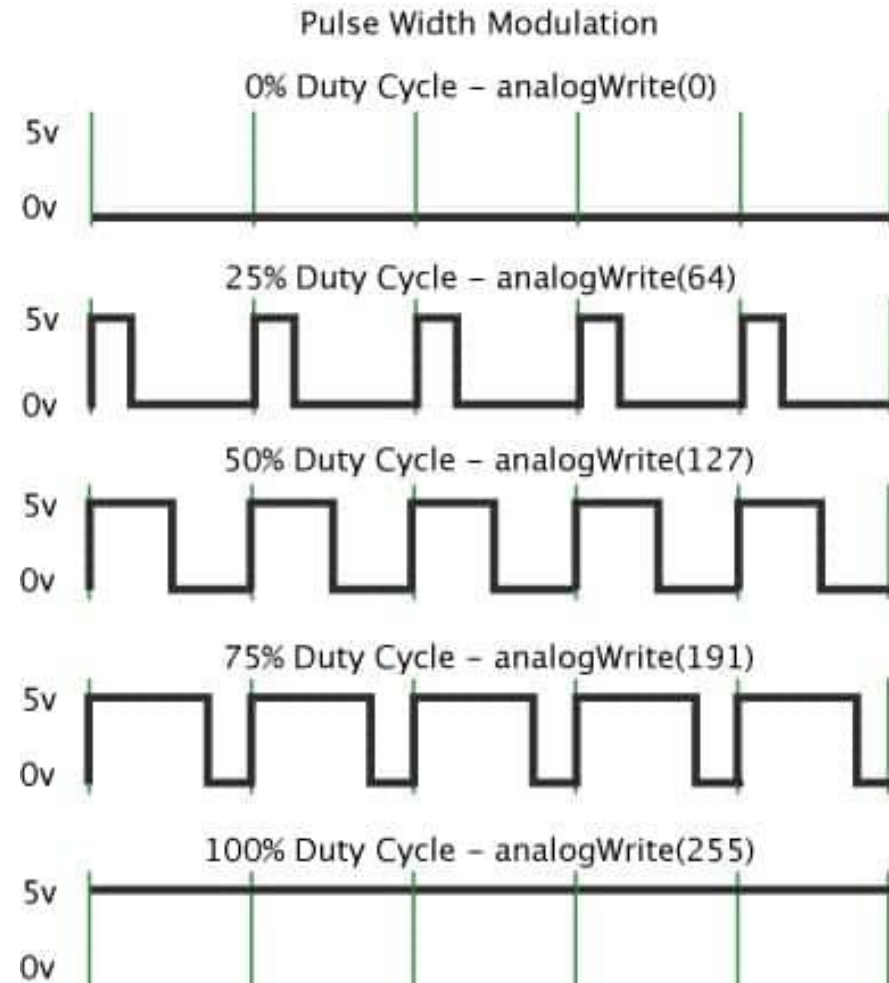
- Os pinos digitais podem ser utilizados como entrada ou saída (digital), devendo ser habilitados para tanto através de programação específica.
- Quando configurados como entrada digital, os pinos possuem alta impedância de entrada (100 M ohms).
- Quando configurado como saída digital, a corrente máxima que um pino pode fornecer é de 40mA. A soma das correntes de todos os pinos não pode ser superior a 200mA.
- Os pinos digitais também podem ser habilitados como saída PWM.

# Usando os pinos digitais como saída PWM

- PWM, do inglês *Pulse Width Modulation*, é uma técnica utilizada por sistemas digitais para variação do valor médio de uma forma de onda periódica.
- A técnica consiste em manter a frequência de uma onda quadrada fixa e variar o tempo que o sinal fica em nível lógico alto.
- Esse tempo é chamado de *duty cycle*, ou seja, o ciclo ativo da forma de onda.

# Usando os pinos digitais como saída PWM

- No gráfico abaixo são exibidas algumas modulações PWM:



# Usando os pinos digitais como saída PWM

- Analisando as formas de onda nota-se que a frequência da forma de onda tem o mesmo valor, variando apenas o *duty cycle* da forma de onda.
- Quando o duty cycle está em 0% o valor médio da saída encontra-se em 0 V.
- Consequentemente para um *duty cycle* de 100% a saída assume seu valor máximo, que no caso é 5V.
- Para um *duty cycle* de 50% a saída assumirá 50% do valor da tensão, ou seja, 2,5 V, e assim sucessivamente para cada variação no *duty cycle*.

# A função `analogWrite()`

- A função **`analogWrite()`** escreve um valor de PWM em um pino digital que possui a função PWM.
- Após a chamada dessa função, o pino passa a operar com uma onda quadrada de frequência fixa e com *duty cycle* conforme valor passado pela função.
- A frequência dessa onda, na maioria dos pinos é em torno de 490 Hz, porém, os pinos 5 e 6 da Arduino UNO operam em 980 Hz.

# Usando as entradas analógicas

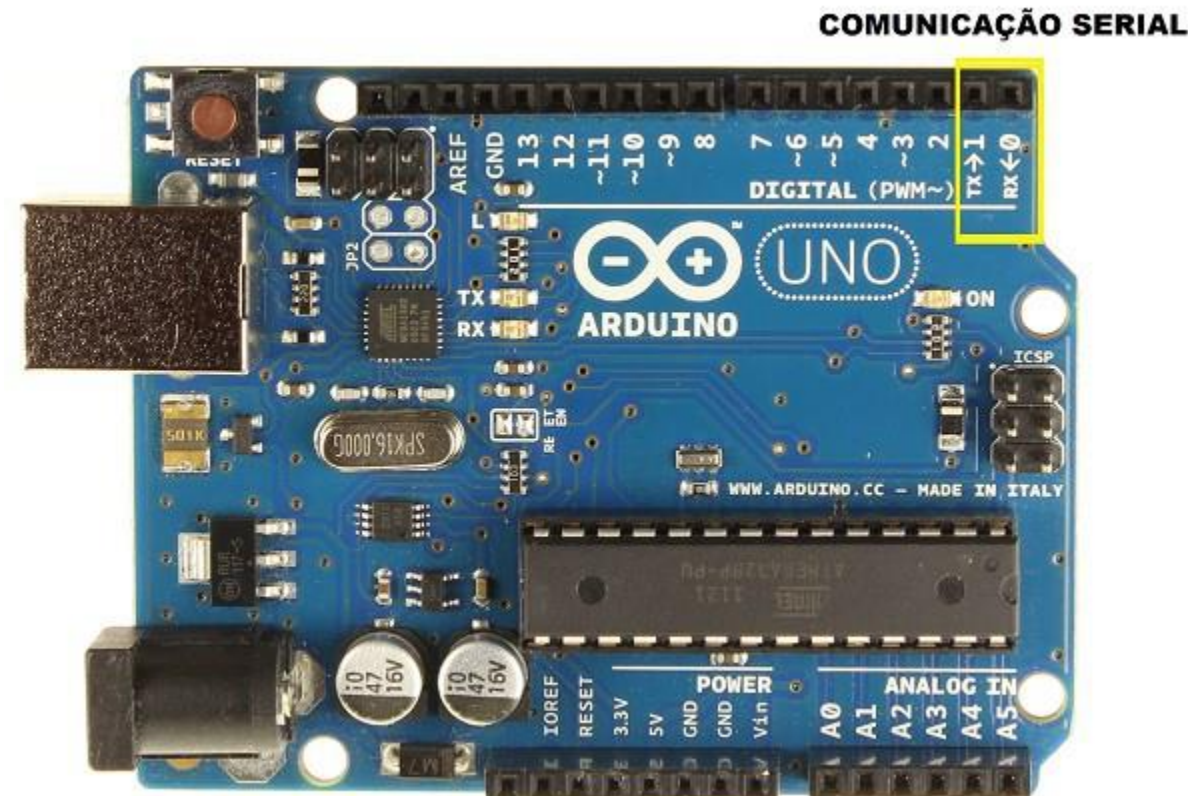
- As entradas digitais só podem assumir dois estados, HIGH e LOW, ou seja, 0 V ou 5 V.
- Porém alguns projetos necessitam a leitura de dados analógicos, que variam de forma contínua dentro de uma determinada faixa. Esta é a função das entradas analógicas.
- Os dados de entrada analógicos são internamente convertidos pelo Arduino em formato digital.
- Essa conversão é feita pelo conversor A/D (analógico/digital) do Arduino.

# Conversor A/D do Arduino

- O conversor A/D do microcontrolador ATmega328 possui 10 bits de resolução.
- A sua tensão de entrada pode variar de 0 V até 5V.
- A tensão de entrada (de 0V a 5V) será convertida pelo Arduino para um valor digital que pode variar de 0 (0V) a 1023 (5V).
- Lembre-se que  $1023_D = 1111111111_B$ .

# Comunicação serial do Arduino

- A placa Arduino UNO possui um canal de comunicação por hardware.
- Esse canal está ligado aos pinos digitais 0 (RX) e 1 (TX).
- Observe a figura ao lado:





# Comunicação serial do Arduino

- O sinal de comunicação na placa Arduino UNO é um sinal TTL de 5V.
- Para a comunicação com um computador ou outro dispositivo que não tenha o mesmo nível de tensão é necessário um conversor de nível.
- Existem várias opções de conversores, como por exemplo TTL/RS232, TTL/RS485, TTL/USB, entre outros.
- A seguir são exibidos alguns modelos de conversores.

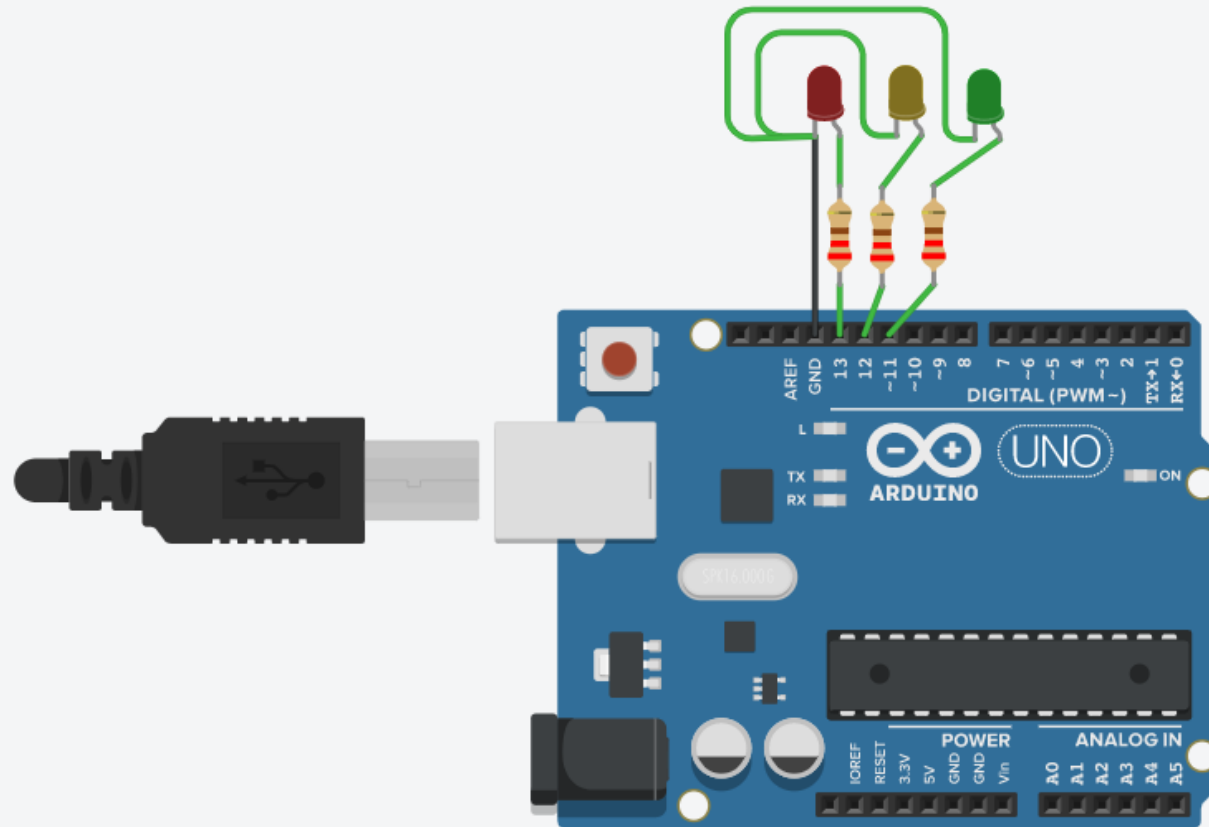
# Comunicação serial do Arduino



# Referências

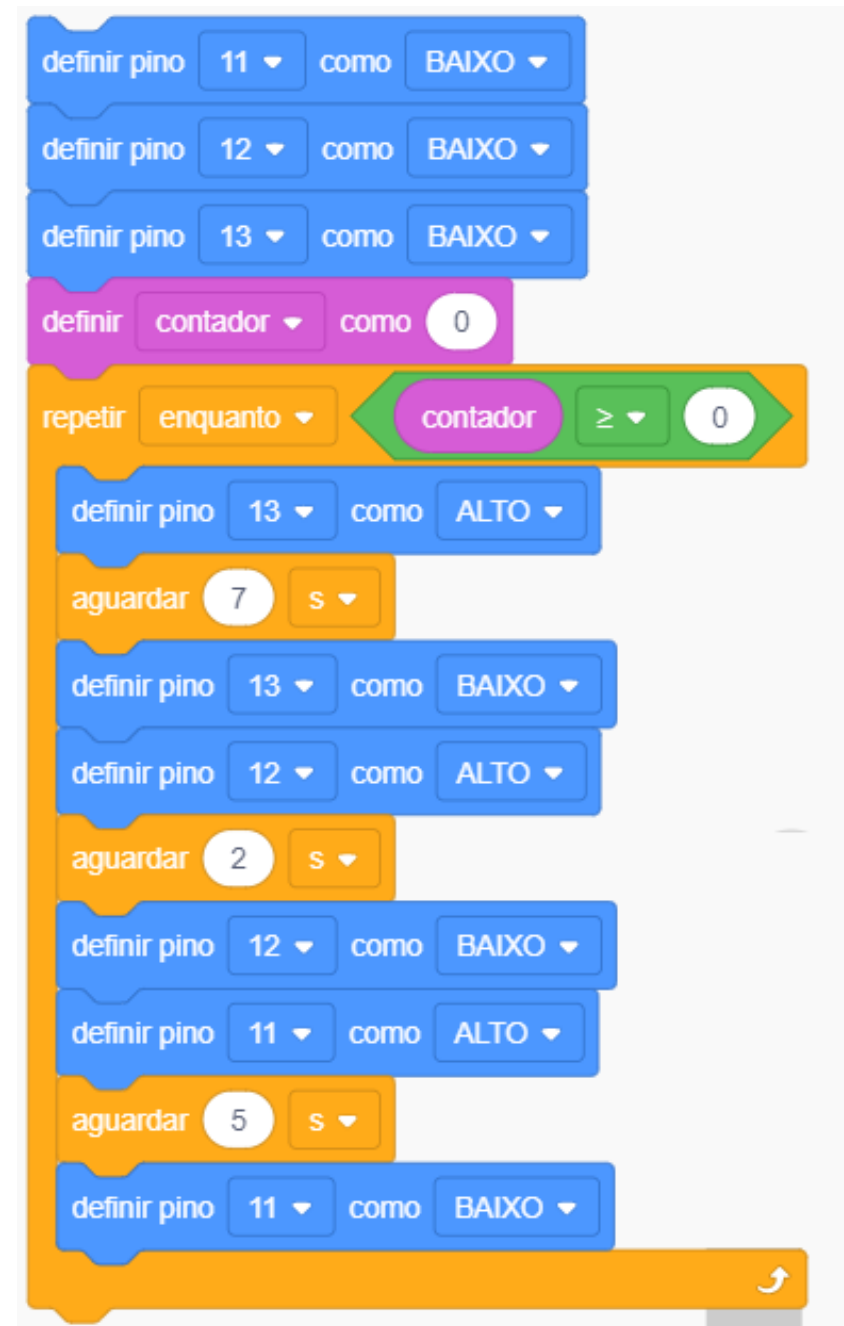
- [Primeiros Passos com Arduino](#)
- [Usando os pinos digitais do Arduino](#)
- [Entendendo as Entradas Analógicas do Arduino](#)
- [Usando as saídas PWM do Arduino](#)
- [Arduino - Comunicação Serial](#)
- [Criando suas próprias bibliotecas para Arduino](#)

# Exemplo com o TinkerCad – Um semáforo

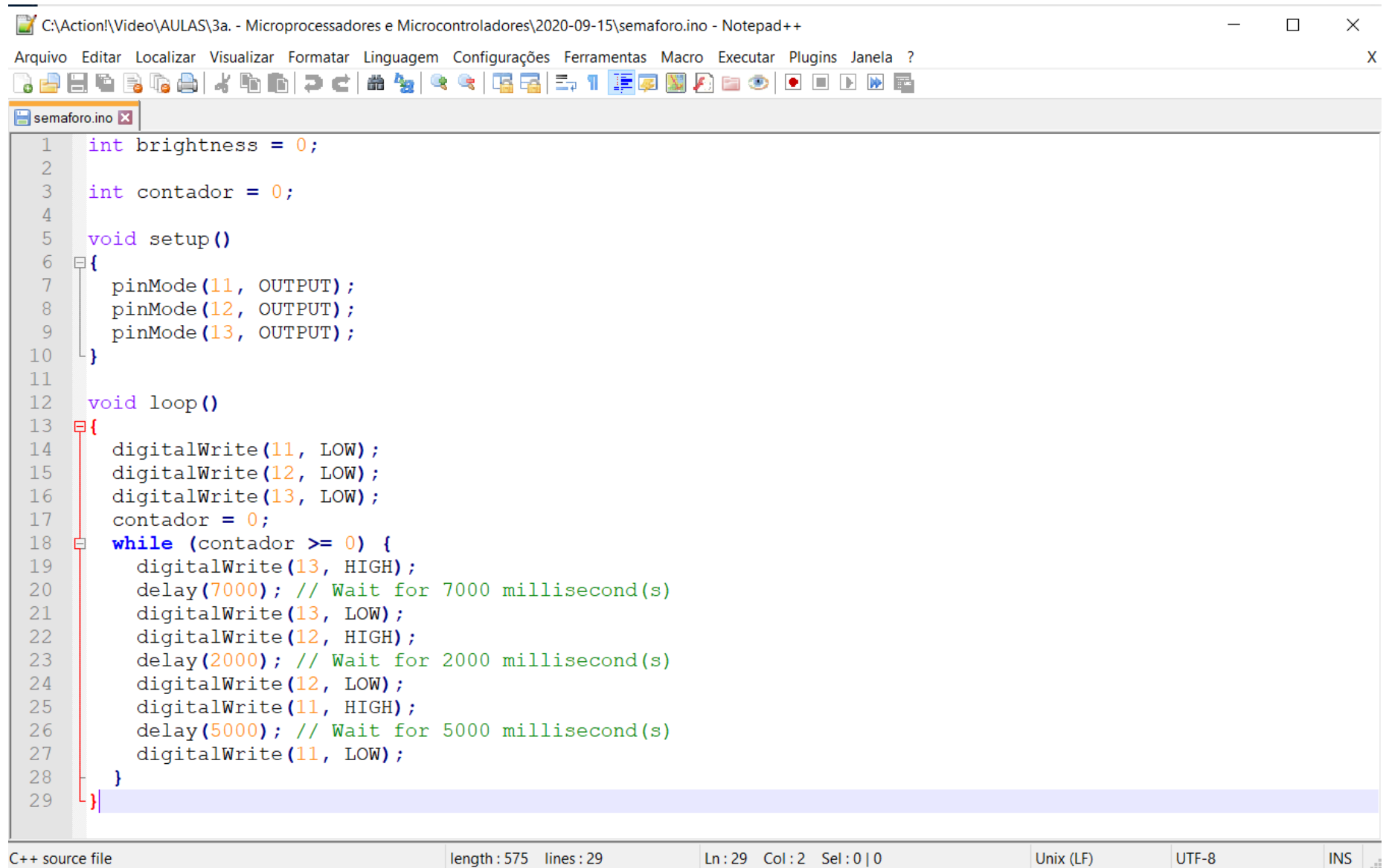


# Exemplo

## Programação em blocos



# Exemplo – Programação original



The image shows a Notepad++ window with the title bar "C:\Action!\Video\AULAS\3a. - Microprocessadores e Microcontroladores\2020-09-15\semaforo.ino - Notepad++". The menu bar includes "Arquivo", "Editar", "Localizar", "Visualizar", "Formatar", "Linguagem", "Configurações", "Ferramentas", "Macro", "Executar", "Plugins", "Janela", and "?". The toolbar contains various icons for file operations and editing. The editor window shows a C++ source file named "semaforo.ino" with the following code:

```
1  int brightness = 0;
2
3  int contador = 0;
4
5  void setup()
6  {
7      pinMode(11, OUTPUT);
8      pinMode(12, OUTPUT);
9      pinMode(13, OUTPUT);
10 }
11
12 void loop()
13 {
14     digitalWrite(11, LOW);
15     digitalWrite(12, LOW);
16     digitalWrite(13, LOW);
17     contador = 0;
18     while (contador >= 0) {
19         digitalWrite(13, HIGH);
20         delay(7000); // Wait for 7000 millisecond(s)
21         digitalWrite(13, LOW);
22         digitalWrite(12, HIGH);
23         delay(2000); // Wait for 2000 millisecond(s)
24         digitalWrite(12, LOW);
25         digitalWrite(11, HIGH);
26         delay(5000); // Wait for 5000 millisecond(s)
27         digitalWrite(11, LOW);
28     }
29 }
```

The status bar at the bottom displays "C++ source file", "length : 575 lines : 29", "Ln : 29 Col : 2 Sel : 0 | 0", "Unix (LF)", "UTF-8", and "INS".

# Sua tarefa!

- Faça a simulação do funcionamento de dois semáforos de trânsito com uma placa Arduino.