



UNINOVE

IOT (Internet of Things)





Conteúdo da Aula

- Síntax
- Comando pinMode (Out e In)
- Serial
- Digital ou Analog Read e Write
- Leitura Digital
- Leitura Analógica
- Condicionais com os valores
- Laço de repetição
- [Mais tipos de sensores](#)
- Desenvolvimento do Projeto





UNINOVE

Mestrando Prof. Rafael Rojas

E-mail: rafael.rojas@uni9.pro.br



Síntax

Afinal o que é a Síntax?

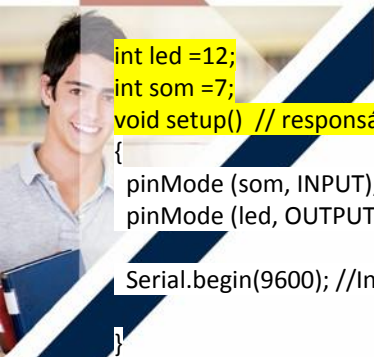
Assim como um idioma, as linguagens possuem algumas similaridades, como laço de repetição, condicionais e operadores lógicos, mas saber isso não basta para definir o quando você sabe daquela língua.

Por exemplo, para se declarar uma variável em JavaScript usamos o seguinte comando:

```
var "nome_da_variável"; // Repare que não declaramos o tipo de variável
```

Já na linguagem em arduino é necessário declarar qual o tipo da variável

```
int "nome_da_variável"; // Variável só poderá receber um número inteiro
```



```
int led =12;  
int som =7;
```

```
void setup() // responsável por definir todas as entradas e saídas das portas seriais e analógicas
```

```
{  
  pinMode (som, INPUT);  
  pinMode (led, OUTPUT);
```

```
  Serial.begin(9600); //Inicia a comunicação serial.
```

```
}
```

```
void loop() // responsável por efetuar os códigos de instrução repetidamente infinitamente
```

```
{
```

```
  int sensordesom = digitalRead(som); // Pedindo para que seja feita a leitura do sensor de som (PortaA0)
```

```
  //sensordesom = map(sensordesom, 0, 1023, 0, 100); // Utilizando a função Map para realizar a proporcionalidade dos valores lido pelo sensor.
```

```
  if (sensordesom == HIGH){
```

```
    digitalWrite(led, HIGH);
```

```
    Serial.println("HIGH");
```

```
    delay(1000);
```

```
  }else{
```

```
    Serial.println("LOW");
```

```
    digitalWrite(led, LOW);
```

```
  }
```

```
  delay(1);
```

```
}
```



Comando pinMode (Out e In)

Descrição

Configura o pino especificado para funcionar como uma entrada ou saída. Veja a descrição dos [pinos digitais \(em Inglês\)](#) para mais detalhes sobre a funcionalidade dos pinos.

Desde a versão 1.0.1, é possível ativar os resistores internos de pull-up como o modo INPUT_PULLUP. Adicionalmente, o modo INPUT explicitamente desativa os resistores pull-up internos.

Sintaxe

`pinMode(pino, modo)`

Parâmetros

pino: the número do pino do Arduino no qual se quer configurar o modo

modo: o modo do pino. Este pode ser [INPUT](#), [OUTPUT](#) ou [INPUT_PULLUP](#); que correspondem respectivamente a entrada, saída e entrada com pull-up ativado.

Serial

Description

Usada para comunicação entre uma placa Arduino e um computador ou outros dispositivos. Todas as placas Arduino possuem pelo menos uma porta serial (também conhecida como UART ou USART), enquanto alguns possuem várias .

No Uno, Nano, Mini, e Mega, pinos 0 e 1 são usados para comunicação com o computador. Conectar qualquer coisa a esses pinos pode interferir nessa comunicação, incluindo causar falhas na gravação da placa.

Você pode usar o monitor serial integrado na IDE do Arduino para comunicar-se com uma placa Arduino. Clique no botão do monitor serial (lupa) na barra de ferramentas e escolha a mesma taxa de transmissão usada na chamada de `begin()`.

Você pode usar o monitor serial presente na IDE do Arduino para se comunicar com uma placa Arduino. Clique no botão do monitor serial na barra de ferramentas e escolha e usar a mesma taxa de transferência (baud rate) usada na chamada de `begin()`.

Comunicação serial nos pinos TX/RX usa níveis lógicos TTL (5V ou 3.3V dependendo da placa). **Não conecte esses pinos diretamente em uma porta serial RS232; esses operam em +/- 12V e podem danificar sua placa Arduino.**

Para usar esses pinos para comunicar com seu computador pessoal, você vai precisar de um adaptador USB-serial extra, pois esses pinos não são conectados ao adaptador USB-serial do Arduino Mega. para usá-los para comunicar com um dispositivo serial TTL externo, conecte o pino TX do Arduino ao pino RX do dispositivo, o pino RX ao pino TX do dispositivo, e o ground de seu Mega ao ground do dispositivo.

O comunicador Serial possui algumas funções muito úteis, dentre elas focaremos nas funções de impressão.

Funções Serial:

```
if(Serial)  
available()  
availableForWrite()  
begin()  
end()  
find()  
findUntil()  
flush()  
parseFloat()  
parseInt()  
peek()  
print()  
println()  
read()  
readBytes()  
readBytesUntil()  
readString()  
readStringUntil()  
setTimeout()  
write()  
serialEvent()
```


Serial.print()

Imprime dados na porta serial em como texto ASCII (facilmente legível, diferentemente dos valores binários). Essa função pode assumir várias formas. números são impressos usando um caractere ASCII para cada dígito. Floats são similarmente impressos como dígitos ASCII, padronizados com duas casas decimais. Bytes são enviados como um único caractere. Caracteres e strings são enviados normalmente, pois já estão em formato ASCII. Por exemplo:

`Serial.print(78)` imprime "78"

`Serial.print(1.23456)` imprime "1.23"

`Serial.print('N')` imprime "N"

`Serial.print("Hello world.")` imprime "Hello world."

Obs. Para usar o Serial é necessário declará-lo no Setup como `Serial.begin(9600)` para abrir a porta Serial a 9600 bps.

if(Serial)

Descrição

Indica se a porta serial especificada está pronta.

Nas placas com USB nativa, `if (Serial)`, ou `if(SerialUSB)` no Due, indica se a conexão serial USB CDC está aberta. Para todas as outras placas, e portas seriais não USB CDC, isso irá sempre retornar `true`.

Essa função foi introduzida na IDE Arduino 1.0.1.

Sintaxe

`if (Serial)`

Parâmetros

Nenhum

Retorna

`bool` : retorna `true` se a porta serial especificada está disponível. Irá retornar `false` apenas caso verificada a conexão serial USB CDC do Leonardo antes que a mesma esteja pronta.



Serial.available()

Descrição

Retorna o número de bytes (caracteres) disponíveis para leitura da porta serial. Esses são dados que já chegaram e foram guardados no buffer de recebimento (o qual armazena 64 bytes).

Serial.availableForWrite()

Descrição

Retorna o número de bytes (caracteres) livres no buffer de transmissão serial que podem ser ocupados sem bloquear a operação de transmissão.

Serial.end()

Descrição

Desativa a comunicação serial, permitindo os pinos RX e TX serem usados novamente como entradas e saídas digitais. Para reativar a comunicação serial, chame Serial.begin().

Serial.find()

Descrição

Serial.find() lê dados do buffer serial de recebimento até a string especificada ser encontrada. A função retorna true se a string denominada é encontrada, false se ocorre time-out.

A função Serial.find() é herdada da classe Stream.

Sintaxe

Serial.find(target)

Serial.find(target, length)



Serial.findUntil()

Descrição

Serial.findUntil() lê dados do buffer serial até uma string especificada de dado tamanho ou um terminador serem encontrados.

A função retorna true se a string especificada é encontrada, false se ocorre time-out.

A função Serial.findUntil() é herdada da classe Stream.

Sintaxe

Serial.findUntil(str, terminal)

Serial.flush()

Descrição

Espera a transmissão de dados seriais enviados terminar.

(Antes do Arduino versão 1.0, em vez disso removia qualquer dados recebidos armazenados no buffer).

flush() herda da classe Stream.

Serial.parseFloat()

Descrição

Serial.parseFloat() retorna o primeiro número válido de ponto flutuante do buffer serial. parseFloat() é terminada pelo primeiro caractere que não é um número de ponto flutuante. A função retorna se acabar o tempo limite (veja Serial.setTimeout()).

A função Serial.parseFloat() é herdada da classe Stream.

Sintaxe

Serial.parseFloat() Serial.parseFloat(lookahead)

Serial.parseFloat(lookahead, ignore)





Serial.parseInt()

Descrição

Procura o próximo inteiro válido no buffer de recebimento serial. A função retorna se acabar o tempo limite (veja `Serial.setTimeout()`).

A função `Serial.parseInt()` é herdada da classe `Stream`.

Em particular:

Caracteres iniciais que não são dígitos ou um sinal de menos, são ignorados;

O Parsing termina quando caracteres não foram lidos em um time-out configurável, ou um valor não-dígito é lido;

Se nenhum dígito válido foi lido quando ocorre o time-out (ver `Serial.setTimeout()`), 0 é retornado;

Serial.peek()

Descrição

Retorna o próximo byte (caractere) de dados seriais recebidos sem o remover do buffer serial interno. Isto é, sucessivas chamadas de `peek()` irão retornar o mesmo caractere, assim como irá a próxima chamada de `read()`.

A função `Serial.peek()` é herdada da classe `link:../..stream[Stream]`.

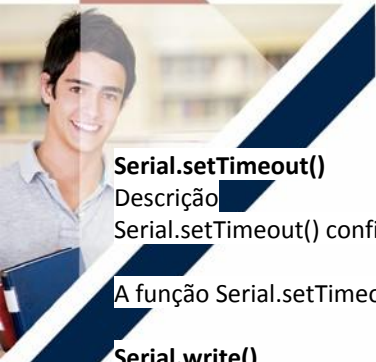
Serial.read()

Descrição

Lê dados recebidos na porta serial.

A função `Serial.read()` é herdada da classe `Stream`.





Serial.setTimeout()

Descrição

`Serial.setTimeout()` configura o número máximo de milissegundos a se esperar por dados seriais. O valor padrão é de 1000 milissegundos.

A função `Serial.setTimeout()` é herdada da classe `Stream`.

Serial.write()

Descrição

Escreve dados binários na porta serial. Esses dados são enviados como um byte ou séries de bytes; para enviar os caracteres representando os dígitos de um número, use em vez disso a função `print()`.

Sintaxe

`Serial.write(val)`

`Serial.write(str)`

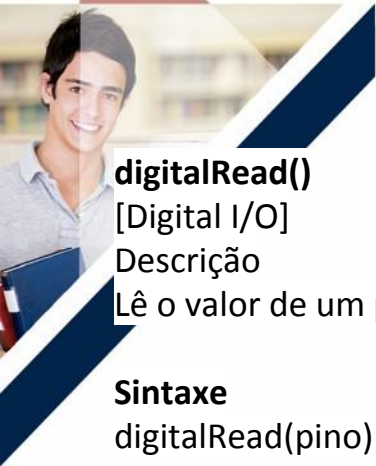
`Serial.write(buf, len)`

serialEvent()

Descrição

Chamada quando dados estão disponíveis no buffer serial. Use `Serial.read()` para ler esses dados.





Digital ou Analog Read e Write



digitalRead()

[Digital I/O]

Descrição

Lê o valor de um pino digital especificado, que pode ser HIGH ou LOW.

Sintaxe

`digitalRead(pino)`

Parâmetros

pino: o número do pino digital do Arduino que você quiser verificar

Retorna

HIGH ou LOW





digitalWrite()

[Digital I/O]

Descrição

Aciona um valor HIGH ou LOW em um pino digital.

Se o pino for configurado como saída (OUTPUT) com a função `pinMode()`, sua tensão será acionada para o valor correspondente: 5V (ou 3.3V em placas alimentadas com 3.3V como o DUE) para o valor HIGH, 0V (ou ground) para LOW.

Se o pino for configurado como entrada (INPUT), a função `digitalWrite()` irá ativar (HIGH) ou desativar (LOW) o resistor interno de pull-up no pino de entrada. É recomendado configurar `pinMode()` com `INPUT_PULLUP` para ativar o resistor interno de pull-up. Veja o tutorial sobre pinos digitais para mais informações.

Se você não configurar o pino com `pinMode()` e OUTPUT, e conectar um LED ao pino, quando chamar `digitalWrite(HIGH)`, o LED pode aparecer um pouco apagado. Sem configurar explicitamente `pinMode()`, `digitalWrite()` irá apenas ativar o resistor de pull-up interno, que age como um grande resistor limitador de corrente.

Sintaxe

`digitalWrite(pino, valor)`

Parâmetros

pino: o número do pino do Arduino

valor: HIGH ou LOW



analogRead()



Descrição

Lê o valor de um pino analógico especificado. A placa Arduino possui um conversor analógico-digital 10 bits de 6 canais (8 canais nos Mini e Nano, 16 no Mega, 7 canais em placas MKR). Isso significa que este irá mapear tensões entre 0 e a tensão operacional (5V or 3.3V) para valores inteiros entre 0 e 1023. No Arduino UNO, por exemplo, isso permite uma resolução entre leituras de: 5 volts / 1024 unidades, ou .0049 volts (4.9 mV) por unidade. Veja a tabela abaixo para os pinos utilizáveis, tensão de operação e resolução máxima para algumas placas Arduino.

Sintaxe

`analogRead(pino)`

Parâmetros

pino: o nome do pino de entrada analógica que se quer ler (A0 a A5 na maioria das placas, A0 a A6 em placas MKR, A0 a A7 no Mini e Nano, A0 a A15 no Mega).

Retorna

A leitura analógica no pino (int). No entanto esta é limitada a resolução do conversor analógico digital (0-1023 para 10 bits ou 0-4095 para 12 bits).



analogWrite()

Descrição

Aciona uma onda PWM (descrição (Em Inglês)) em um pino. Pode ser usada para variar o brilho de um LED ou acionar um motor a diversas velocidades. Após a função `analogWrite()` ser chamada, no pino haverá uma onda quadrada com o duty cycle (ciclo de trabalho) especificado até a próxima chamada de `analogWrite()` (ou uma chamada de `digitalRead()` ou `digitalWrite()` no mesmo pino). A frequência do sinal PWM na maioria dos pinos é aproximadamente 490 Hz. No Uno e placas similares, pinos 5 e 6 usam uma frequência de aproximadamente 980 Hz.

Sintaxe

`analogWrite(pino, valor)`

Parâmetros

pino: o pino escolhido do Arduino. Tipos de dados permitidos: `int`.

valor: o duty cycle: entre 0 (sempre desligado) and 255 (sempre ligado). Tipos de dados permitidos: `int`

Leitura Digital

Código de Exemplo

Aciona o pino 13 para o mesmo valor que o pino 7, declarado como entrada.

```
int ledPin = 13; // LED conectado ao pino digital 13
int inPin = 7;   // botão conectado ao pino digital 7
int val = 0;     // variável para guardar o valor lido

void setup() {
  pinMode(ledPin, OUTPUT); // configura o pino digital 13 como saída
  pinMode(inPin, INPUT);   // configura o pino digital 7 como entrada
}

void loop() {
  val = digitalRead(inPin); // lê o pino de entrada
  digitalWrite(ledPin, val); // aciona o LED com o valor lido do botão
}
```

Leitura Analógica

Código de Exemplo

O código abaixo lê o valor de um pino de entrada analógica e mostra seu valor na porta serial.

```
int analogPin = A3; // terminal do meio de um potenciômetro conectado ao pino analógico 3
                    // terminais mais externos são conectados um no ground e o outro em +5V
int val = 0;        // variável para guardar o valor lido

void setup() {
  Serial.begin(9600);    // configura a porta serial
}

void loop() {
  val = analogRead(analogPin); // lê o pino de entrada
  Serial.println(val);         // imprime o valor na porta serial
}
```

Condicionais com os valores

```
int a=10;
int b=9;
int c=8;
int d=6;
int e=5;
int f1=12;
void setup()
{
  pinMode(f1, INPUT);
  pinMode(a, OUTPUT);
  pinMode(b, OUTPUT);
  pinMode(c, OUTPUT);
  pinMode(d, OUTPUT);
  pinMode(e, OUTPUT);
  Serial.begin(9600);
}
void loop()
{
  // int status=digitalRead(f1);
```

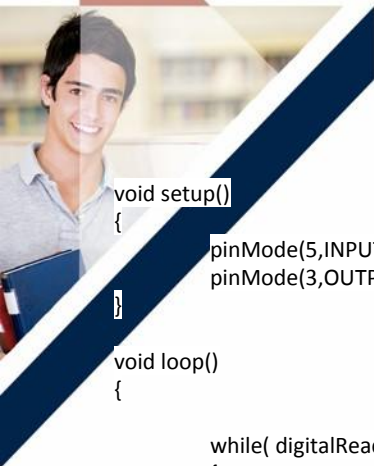
```
  if (digitalRead(f1)==1){
    delay(5000);

    digitalWrite(c, LOW);
    digitalWrite(b, HIGH);
    delay(3000);

    digitalWrite(b, LOW);
    digitalWrite(a, HIGH);
    digitalWrite(d, LOW);
    digitalWrite(e, HIGH);

    delay(5000);
  } else{
    digitalWrite(a, LOW);
    digitalWrite(b, LOW);
    digitalWrite(c, HIGH);
    digitalWrite(d, HIGH);
    digitalWrite(e, LOW);
  }
  delay(1);
}
```

Laço de repetição



```
void setup()
```

```
{
```

```
pinMode(5,INPUT); //pino do botão
```

```
pinMode(3,OUTPUT); //pino do LED
```

```
}
```

```
void loop()
```

```
{
```

```
while( digitalRead(5) == 1 ) //enquanto o botão estiver pressionado
```

```
{
```

```
    //blink
```

```
    digitalWrite(3,HIGH);
```

```
    delay(1000);
```

```
    digitalWrite(3,LOW);
```

```
    delay(1000);
```

```
}
```

```
}
```



Desenvolvimento do Projeto

Quizz

