Arduino

Pinos

OS PINOS DIGITAIS

São 14 pinos marcados com o nome **DIGITAL**. São numerados de 0 a 13 da direita para a esquerda. Esses pinos tem o valor de 0 ou 5 volts.

OS PINOS ANALÓGICOS

São 6 pinos em uma só barra com o nome **ANALOG IN**, localizada no lado oposto às barras dos pinos digitais. São numerados de 0 a 5, agora da esquerda para a direita. Esses pinos são usados para leitura de sinais analógicos de sensores conectados ao Arduino, e podem ser de quaisquer valores entre zero a 5 volts.

Retorno

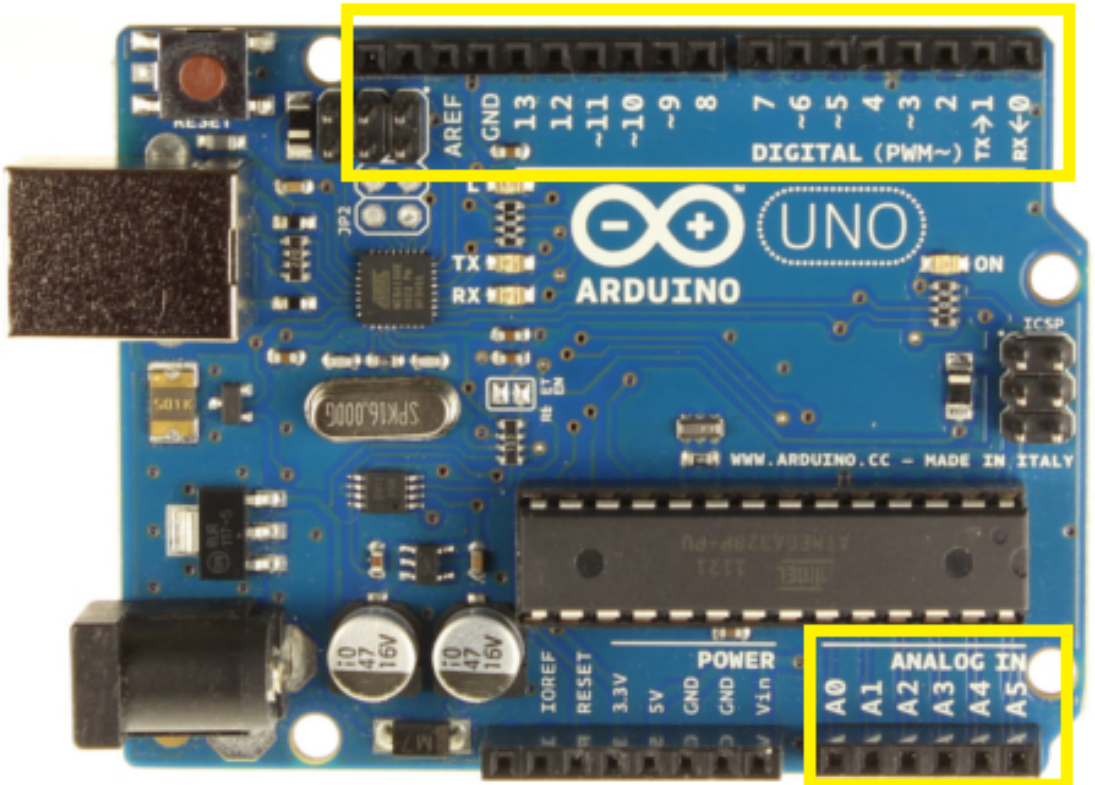
int (0 a 1023)

***PWM**

OS PINOS DE ALIMENTAÇÃO

Ficam na barra com 6 pinos, marcada como **POWER**, localizada ao lado dos pinos analógicos. O primeiro pino dessa barra, **RESET**, quando forçado ao potencial de terra serve para resetar o Arduino. Do outro lado, **Vin** é um pino que também pode servir para alimentar o Arduino se nele for aplicada uma tensão entre 9 e 15 volts.Pino **GND** é o terra. Pino **5V** é saida de tensão com corrente de 250mA e o Pino **3.3V** é saida de tensão de 50mA.

Entradas / Saídas Digitais



Entradas Analógicas

LInguaguem

É uma linguagem chamada wiring. É baseada no C e no C++. Foi desenvolvida junto com o projeto Arduino. Pode-se utilizar outras linguagens para o Arduino também.

Estrutura

A função setup() é chamada quando um programa começa a rodar. Use esta função para inicializar as sua variáveis, os modos dos pinos, etc. Esta função será executada apenas uma vez após a placa Arduino ser ligada ou ressetada.

Após criar uma fução setup() que declara os valores iniciais, a função loop() faz exatamente o que seu nome sugere, entra em looping (executa sempre o mesmo bloco de código). Use esta função para controlar ativamente a placa Arduino.

#define é um componente muito útil da linguagem C que permite ao programador dar um nome a uma constatante antes que o programa seja compilado.

#define nomeDeConstante valor

#include é usado para incluir outras bibliotecas no seu programa. Isto permite ao programador acessar um grande número de bibliotecas padrão da linguagem C, e também à bibliotecas desenvolvidas especificamente para o Arduino.

#include <biblioteca.h>

Operadores Comparativos

== (igual a)

!= (diferente)

< (menor que)

> (maior que)

<= (menor ou igual a)

>= (maior ou igual a)

Operadores Booleanos

&& (e)

|| (ou)

! (negação)

Operadores Aritméticos

= (atribuição)

+ (adição)

- subtração)

* (multiplicação)

/ (divisão)

% (módulo)

Operadores Compostos

++ (incremento)

-- (decremento)

+= (adição composta)

-= (subtração composta)

*= (multiplicação composta)

/= (divisão composta)

```
x += y;   // equivalente à expressão x = x + y;
x -= y;   // equivalente à expressão x = x - y;
x *= y;   // equivalente à expressão x = x * y;
x /= y;   // equivalente à expressão x = x / y;
```

Estrutura de Controle

- If
- If … Else
- Switch Case
- For
- While
- Do … While
- Break
- Continue
- Return

Show me the Code

```
if (algumaVariavel > 50){
   // faça algo aqui
}
```

```
if (input < 500){
   // fazer A
}else {
   // fazer B
}
```

```
switch (valor) {
   case 1:
      //fazer algo quando valor == 1
   break;
   case 2:
      //fazer algo quando valor == 2
   break;
   default:
      // se nada mais encaixa, fazer o padrão
}
```

```
for (inicialização; condição; incremento) {
   //instrução;
}

for(int x = 2; x < 100; x = x * 1.5){
   Serial.println(x);
}
```

```
int var = 0;

while(var < 200){
    // fazer algo repetitivo 200 vezes
    var++;
}
```

```
int var = 0;


do{
   // fazer algo repetitivo 200 vezes
   var++;
}while(var < 200);
```

Tipos de Dados

- Boolean
- Char
- Byte
- Int
- Unsigned Int
- Long
- Unsigned Long
- Float
- Double
- String
- Void

Boolean

O tipo boolean pode representar valores booleanos, verdadeiro(true) ou falso(false). Um tipo boolean ocupa um byte da memória.

Char

O tipo char armazena valres de 1 byte. Caracteres são codificados em um único byte e são especificados na tabela ASCII.

Byte

Armazena valores de 8 bits não sinalizados de 0 a 255.

Int

Inteiros são tipos primários de armazenamento. No Arduino Uno(e em outras placas baseadas em ATMEGA) um int armazena valores de 16 bits(2 bytes). Esse tipo compreende valores de -32768 a 32767.

Unsigned Int

No Arduino UNO e em outras placas baseadas em ATMEGA armazenam valores de 16 bits não sinalizados, ou seja, apenas valores negativos de 0 a 65535.

Long

O tipo de dado Long armazena valores inteiros sinalizados de 32 bits (4 bytes) que compreendem a faixa de -2147483648 a 2147483647

Unsigned Long

O tipo unsigned long armazena valores de 32 bits (4 bytes) não sinalizados que compreendem a faixa de 0 a 429967295.

Float

O tipo float armazena valor em ponto flutuante, ou seja, um valor que possui casas decimais. O tipo float armazena valores de 32 bits(4 bytes) e compreendem a faixa de -3,4028235 a 3,4028235.

Double

O tipo double também armazena valores de ponto flutuante, porém no Arduino Uno e outras placas baseadas em ATMEGA esse tipo é exatamente o mesmo que o tipo float, sem ganho de precisão.

Void

A palavra reservada void é usada em declarações de funções. Este tipo indica que a função não retorna nenhum valor quando é executada.

Converção

char()

byte()

int()

long()

float()

Constantes

Constantes são rótulos para certos valores, os quais são pré-definidos no compilador do Arduino.

HIGH | LOW

INPUT | OUTPUT

Funções

Digital I/O (entradas e saídas digitais)

- pinMode(pino, modo)
- digitalWrite(pino, valor)
- digitalRead(pino)

Analog I/O (entradas e saídas analógicas)

int analogRead(pino)

analogWrite(pino, valor) - PWM

Comunicação Serial

Usado para comunicação entre a placa Arduino e um computador ou outros dispositivos. Esta comunicação ocorre através dos conectores serial ou USB da placa Arduino e nos pinos digitais 0 (RX) e 1 (TX).

Serial.begin(int velocidade)

- Ajusta o taxa de transferência em bits por segundo (baud) para transmissão de dados pelo padrão serial.

Serial.available()

- Obtem o número de bytes (caracteres) disponíveis para leitura através da porta serial.

Serial.read()

- Lê dados que estejam entrando pela porta serial.

Serial.Print(Data)

- Envia dados pela porta serial.

Serial.Println(Data)

- Envia dados pela porta serial e quebra linha.