Variáveis

Tipos de dados e constantes da linguagem Arduino.

Sintaxe

word var = val;

Parâmetros
var: nome da variável
val: valor a ser atribuído à variável

Conversão

byte() Converte um valor para o tipo de dado byte.

char() Converte um valor para o tipo de dado char.

float() Converte um valor para o tipo de dado float.

int() Converte um valor para o tipo de dado int.

long() Converte um valor para o tipo de dado long.

word() Converte um valor para o tipo de dado word.

Tipos de Dados

bool O tipo bool pode armazenar dois valores: true or false. (Cada variável bool ocupa um byte na memória.)

boolean boolean é um alias de tipos de dados não padrão para bool definido pelo Arduino. É recomendado usar em vez disso o tipo padrão bool, que é idêntico.

byte Uma variável 'byte' armazena valores numéricos de 8-bit sem sinal, de 0 a 255.

char Um tipo de dado usado para armazenar um caractere. Caracteres literais são escritos em aspas simples, dessa forma: 'A' (para múltiplos caracteres - ou seja, strings - use aspas duplas: "ABC").

double Número de Ponto flutuante Double. No UNO e outras placas baseadas no ATMEGA, ocupa 4 bytes. Isto é, nesses a impelmentação do double é exatamente a mesma do float, sem nenhum ganho em precisão.

float Tipo de dado para números de ponto flutuante, ou seja, um número racional.

int Ints (integer ou inteiros) são o tipo o tipo de dados primário para armazenamento de números.

long Variáveis long são variáveis de tamanho extendido para armazenamento de números, armazenam 32 bits (4 bytes),

short O tipo short é um tipo de dado 16-bit.

Sintaxe
short var = val;

=== Parâmetros var: nome da variável
val: valor a ser atribuído à variável

size_t size_t é um tipo de dado capaz de representar o tamanho de qualquer objeto em bytes. Exemplos do uso de size_t são o tipo do retorno de sizeof() e Serial.print().

Parâmetros
var: nome da variável
val: valor a ser atribuído à variável

string As strings de texto podem ser representadas de duas maneiras.

```
char Str1[15];
char Str2[8] = {'a', 'r', 'd', 'u', 'i', 'n', 'o'};
char Str3[8] = {'a', 'r', 'd', 'u', 'i', 'n', 'o', '\0'};
char Str4[] = "arduino";
char Str5[8] = "arduino";
char Str6[15] = "arduino";
```

String() Constrói uma instância da classe String. Há múltiplas versões que constroem Strings a partir de diferentes tipos de dados (ex. formatam-nos como uma sequência de caracteres), incluindo:

Sintaxe
String(val)
String(val, base)
String(val, decimalPlaces)

unsigned char Um tipo de dado sem sinal, que ocupa um byte na memória. O mesmo que o tipo de dado byte. O tipo de dado unsigned char armazena valores de 0 a 255.

Sintaxe
unsigned char var = val;

Parâmetros
var: nome da variável
val: valor a ser atribuído à variável

unsigned int

Sintaxe
unsigned int var = val;

Parâmetros
var: nome da variável
val: valor a ser atribuído à variável

Código de Exemplo
O trecho de código abaixo cria uma variável unsigned int chamada ledPin e a atribui o valor 13.

```
unsigned int ledPin = 13;
```

unsigned long

Variáveis unsigned long são variáveis de tamanho extendido para armazenamento de números, que armazenam 32 bits (4 bytes).
Diferentemente de longs padrão, unsigned longs não guardam números negativos,
o que faz com que possam armazenar valores de 0 a 4,294,967,295 (2^32 - 1).

vetor Um vetor (array) é uma coleção de variáveis que são acessadas com um número índice. Vetores na linguagem C++, na qual o Arduino é baseado, podem ser complicados, mas usar vetores simples é relativamente fácil.

void A palavra chave void é usada apenas em declarações de funções. Ela indica que é esperado que a função não retorne nenhuma informação para a função da qual foi chamada.

word Uma variável word armazena um número sem sinal de ao menos 16 bits, de 0 A 65535.

Escopo de Variáveis e Qualificadores

const A palavra-chave const é uma abreviação de constante. É um qualificador de variáveis que modifica o comportamento da variável, fazendo com que a variável seja de "apenas-leitura".

escopo Código de Exemplo

```
int gPWMval;  // qualquer função poderá acessar essa variável
```

static A palavra-chave static é usada para criar variáveis que são visíveis para apenas uma função.

volatile A palavra chave volatile é conhecida como um qualificador de variáveis, e é geralmente usada antes do tipo de dado da variável, para modificar a forma com qual o compilador e o programa tratam a variável.

Functions Digital I/O

digitalRead() Lê o valor de um pino digital especificado,

digitalWrite() Aciona um valor HIGH ou LOW em um pino digital.

pinMode() Configura o pino especificado para funcionar como uma entrada ou saída.

Entradas e Saídas Analógicas

analogRead() Lê o valor de um pino analógico especificado. Isso significa que este irá mapear tensões entre 0 e a tensão operacional (5V or 3.3V) para valores inteiros entre 0 e 1023.

analogReference() Configura a tensão de referência para a entrada analógica (o valor máximo do intervalo de entrada).

analogWrite() Aciona uma onda PWM (descrição (Em Ingês)) em um pino. Pode ser usada para variar o brilho de um LED ou acionar um motor a diversas velocidades.

Apenas Zero, Due e Família MKR

analogReadResolution() analogReadResolution() é um extensão da API Analog para o Arduino Due, Zero e família MKR.

analogWriteResolution() analogWriteResolution() é uma extensão da API Analog para os Arduinos Due, Zero e MKR.

Funções Matemáticas

abs() Calcula o módulo (ou valor absoluto) de um número.

constrain() Restringe um número a ficar dentro de um intervalo.

map() Restringe um número a ficar dentro de um intervalo.

max() Remapeia um número de um intervalo para outro.

min() Calcula o menor de dois números.

pow() Calcula o valor de um número elevado a uma potência.

sq() Calcula o quadrado de um número: o número multiplicado por si mesmo.

sqrt() Calcula a raiz quadrada de um número.

Entradas e Saídas Avançadas

noTone() Interrompe a geração de uma onda quadrada iniciada pela função tone(). Não tem nenhum efeito se nenhum tom está sendo gerado.

pulseIn() Captura a duração de um pulso em um pino (que pode ser HIGH ou LOW).

pulseInLong() pulseInLong() é uma alternativa à função pulseIn(), sendo melhor para lidar com pulsos longos e situações afetadas por interrupções.

shiftIn() pulseInLong() é uma alternativa à função pulseIn(), sendo melhor para lidar com pulsos longos e situações afetadas por interrupções.

shiftOut() Transfere um byte de dados um bit de cada vez.

tone() Gera uma onda quadrada na frequência especificada (e duty cycle 50%) em um pino. A duração pode ser especificada, do contrário a onda continua até uma chamada de noTone().

Funções Temporizadoras

delay() Pausa o programa por uma quantidade especificada de tempo (em milissegundos). Cada segundo equivale a 1000 milissegundos.

delayMicroseconds() Pausa o programa pela quantidade de tempo especificada como parâmetro (em microssegundos).

micros()

millis()

Estruturas
Os elementos da linguagem Arduino (C++).

Sketch

setup()

Descrição

A função setup() é chamada quando um sketch inicia. Use-a para inicializar variáveis, configurar o modo dos pinos(INPUT ou OUTPUT), inicializar bibliotecas, etc. A função setup() será executada apenas uma vez, apoós a placa ser alimentada ou acontecer um reset.

Código de Exemplo

```
int buttonPin = 3;

void setup() {
  // Inicializa a porta serial
  Serial.begin(9600);
  // configura o pino 3 como INPUT
  pinMode(buttonPin, INPUT);
}

void loop() {
  // ...
}
```

loop()

Código de Exemplo

```
int buttonPin = 3;

// setup inicializa a porta serial e o pino para o botão
void setup() {
  Serial.begin(9600);
  pinMode(buttonPin, INPUT);
}

// loop checa o estado do botão repetidamente, e envia
// pela serial um 'H' se este está sendo pressionado
void loop() {
  if (digitalRead(buttonPin) == HIGH) {
    Serial.write('H');
  }
  else {
    Serial.write('L');
  }

  delay(1000);
}
```

Outros Elementos da Sintaxe

#define (define)

#define é uma diretiva muito útil da linguagem C++ que permite ao programador dar um nome a um valor constante antes de o programa ser compilado

Código de Exemplo

```
#define pinoLED 3
// O compilador irá substituir qualquer menção de pinoLED com o valor 3 no tempo de compilação.
```

#include (include)

A diretiva #include é usada para incluir bibliotecas externas ao seu sketch

```
#include <Servo.h>
```

// comentário

;

Usado para encerrar um comando.

{} (chaves)

```
Funções
void minhafuncao(tipo argumento) {
  // comando(s)
}
Loops
while (expressão boolena) {
  // comando(s)
}

do {
  // comando(s)
} while (expressão boolena);

for (inicialização; condição; incremento) {
  // comando(s)
}
Estruturas Condicionais
if (expressão boolena) {
  // comando(s)
}

else if (expressão boolena) {
  // comando(s)
}
else {
  // comando(s)
```

/* */ (comentário em bloco)

Estruturas de Controle

break

break é usado usado para sair de um laço for, while ou do…while, ignorando a condição padrão do loop. Também é usada para sair do comando switch case.

continue

O comando continue "pula" o resto da iteração atual de um loop (for, while, ou do…while). Esse comando continua a checar a expressão condicional do loop, e procede com qualquer iterações subsequentes.

do...while

O loop do…while funciona da mesma forma que o loop while, com a exceção de a condição ser testada no final do loop, tal que o loop será executado pelo menos uma vez.

else

Sintaxe

```
if (condição1) {
  // faz coisa A
}
else if (condição2) {
  // faz coisa B
}
else {
  // faz coisa C
}
```

for

O comando for é usado para repetir um bloco de código envolvido por chaves.

Sintaxe

```
for (inicialização; condição; incremento) {
  //comando(s);
}
```

goto

goto rótulo; // envia o fluxo do programa de volta para o rótulo

if

O comando if checa uma condição e executas o comando a seguir ou um bloco de comandos delimitados por chaves, se a condição é verdadeira ('true').

Sintaxe

```
if (condição) {
  comentario //comando(s)
} if (x > 120) {
    digitalWrite(pinoLED, HIGH);
  }

  if (x > 120) {
    digitalWrite(pinoLED, HIGH);
  }

  if (x > 120) {
    digitalWrite(pinoLED, HIGH);
  }

  if (x > 120) {
    digitalWrite(pinoLED1, HIGH);
    digitalWrite(pinoLED2, HIGH);
  } // todas as formas acima estão corretas
```

return

Termina uma função e retorna um valor, caso desejado.

switch...case

Da mesma forma que o comando if, o comando switch case controla o fluxo do programa permitindo ao programador especificar código diferente para ser executado em várias condições.

Código de Exemplo

```
switch (var) {
  case 1:
    // faz algo quando var é igual a 1
    break;
```

while

Um loop while irá se repetir continuamente, e infinitamente, até a expressão dentro dos parênteses (), se torne falsa.

Operadores de Comparação

- != (diferente de)
- < (menor que)
- <= (menor que ou igual a)
- == (igual a)
- > (maior que)
- >= (maior que ou igual a)