Introdução à Programação de Sistemas Embarcados

Prof. Roberto Hiramatsu [http://www.lsi.usp.br/~kenji/]
Prof. João Pimentel [www.cin.ufpe.br/~jhcp]

Aula 1
Turma Março/2018
Introdução à Programação de Sistemas Embarcados

26 a 30 de março de 2018
Unidade Acadêmica do Cabo de Santo Agostinho
Universidade Federal Rural de Pernambuco

Instrutores

Prof. Roberto Hiramatsu

Doutor em Engenharia Elétrica pela USP

Site pessoal: http://www.lsi.usp.br/~kenji/

Currículo: http://lattes.cnpq.br/3420705447094049

Prof. João Pimentel

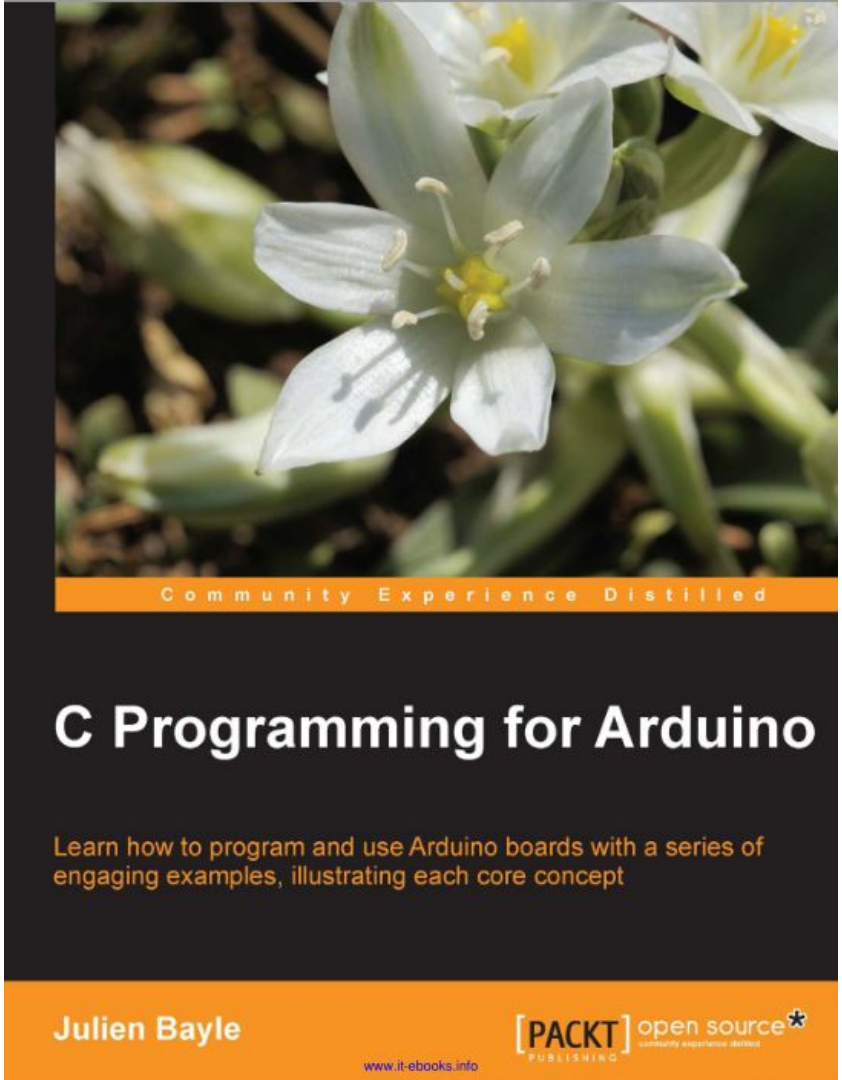
Doutor em Ciência da Computação pela UFPE

Site pessoal: www.cin.ufpe.br/~jhcp

Currículo: http://lattes.cnpq.br/8257035194560179

Livro de apoio

C Programming for Arduino
por Julien Bayle



Objetivos do curso

Que vocês conheçam as peculiaridades do desenvolvimento de sistemas embarcados

e que aprendam estratégias para lidar com eles

Este *não é* um curso de introdução a programação

Este *não é* um curso de Arduino

Este *não é* um curso de decorar

Este *não é* um curso em que você ficará sentada(o) só assistindo

Para hoje, esperamos que vocês

... tenham uma noção do que são sistemas embarcados e para que servem

... tenham uma noção de como é o desenvolvimento de software e comecem a utilizar algumas boas práticas

... formem as duplas e recebam o material necessário

... conheçam alguns detalhes da linguagem C

... comecem a executar programas no Arduino

... entendam como é possível integrar um software com o mundo real

COMEÇANDO


12:00
Panasonic


XONIX
LIGHT
ST/STP
22 SU
PM
7:48
CHRONO
ALARM
CHIME
32
MODE
SET/RST
WR 100M

Electronic Injections
Check Control
Speed Control
Central Locking
...

Electronic Gear Control
Electronic Air Condition
ASC Anti Slip Control
ABS Anti Blocking Sys.
Telephone
Seat Heating Control
Autom. Mirror Dimming
...

Navigation System
CD-Changer
ACC Active Cruise Control
Airbags
DSC Dynamic Stability Control
Adaptive Gear Control
Xenon Light
BMW Assist
RDS/TMC
Speech Recognition
Emergency Call
...

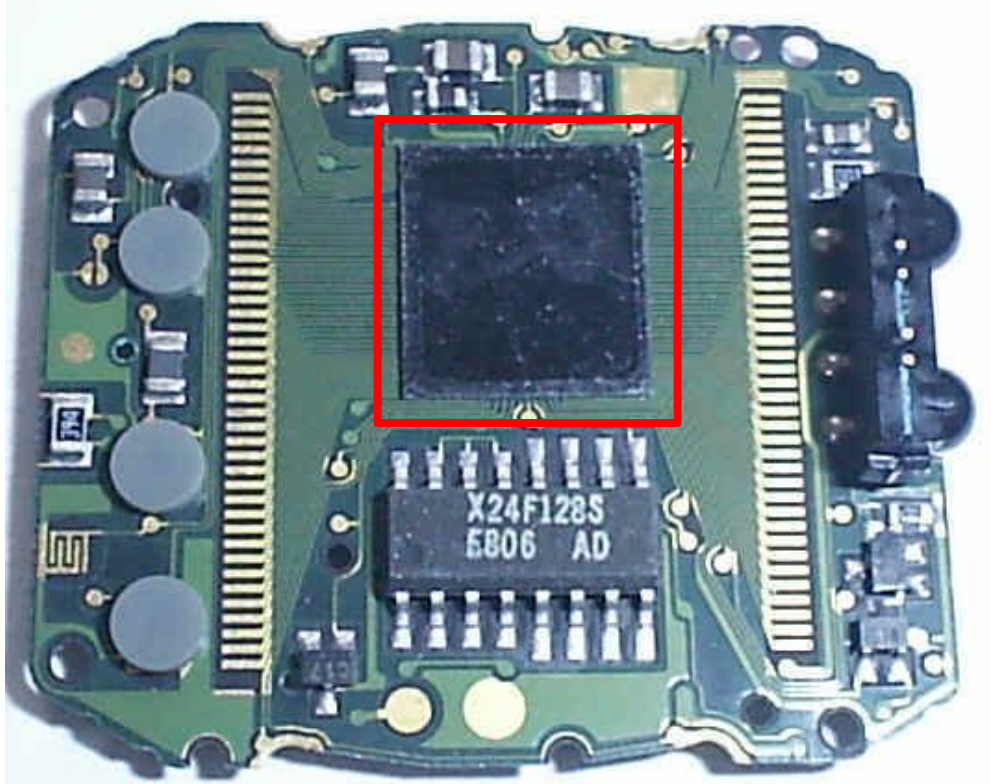
ACC Stop&Go
BFD
ALC
KSG
42-Voltage
Internet Portal
GPRS, UMTS
Telematics
Online Services
Blue-Tooth
Car Office
Local Hazard Warning
Integrated Safety System
Steer/Brake-By-Wire
I-Drive
Lane Keeping Assist.
Personalization
Software Update
Force Feedback Pedal

1970 1980 1990 2000

Degree of Interdependence

Source: [Frischkorn04] http://aswsd.ucsd.edu/2004/pdfs/Frischkorn_Keynote_Workshop_SanDiego2004vers1.2.pdf


SUNDAY
2000
1- 3
7:52 02


X24F128S
5806 AD

Sistemas embarcados - Embedded Systems

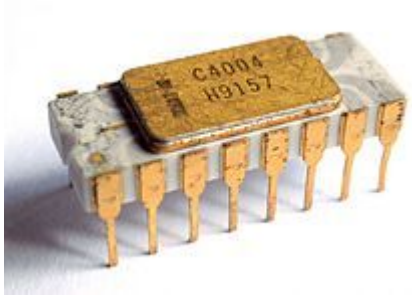
“Um sistema embarcado é uma combinação de hardware e software, e talvez outras partes (mecânicas ou de outro tipo), projetado para realizar uma função específica.”

“Um sistema computadorizado dedicado a realizar um conjunto específico de funções no mundo real”

“Um sistema embarcado é qualquer sistema computadorizado escondido em um produto que não seja um computador”

Sistemas embarcados - Origem

1971 - Intel lança o primeiro processador disponível comercialmente



Sistemas embarcados - Conflitos entre requisitos

Custo

Energia

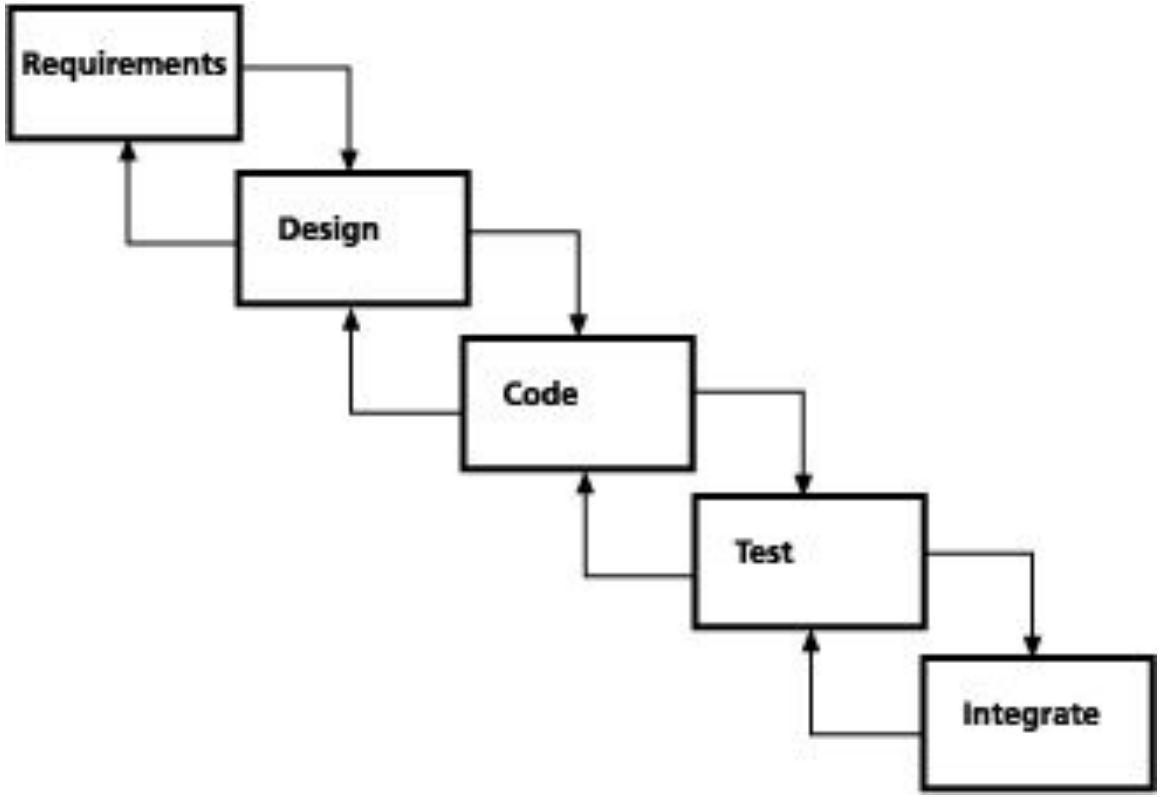
Tamanho

Temperatura

Precisão

Robustez

Engenharia de software



Programar não é escrever código

---

1) Entender o problema
2) Pensar em soluções diferentes e compará-las
3) Escolher uma solução e detalhar o seu passo-a-passo (algoritmo)
4) Elaborar testes
5) Escrever o código
6) Testar

Programação - boas práticas que vocês irão praticar

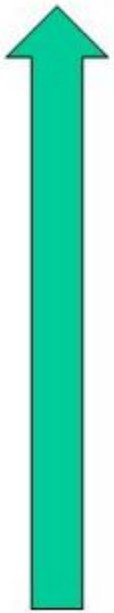
~~Inspeção de código~~

Programação em pares

Test-driven development

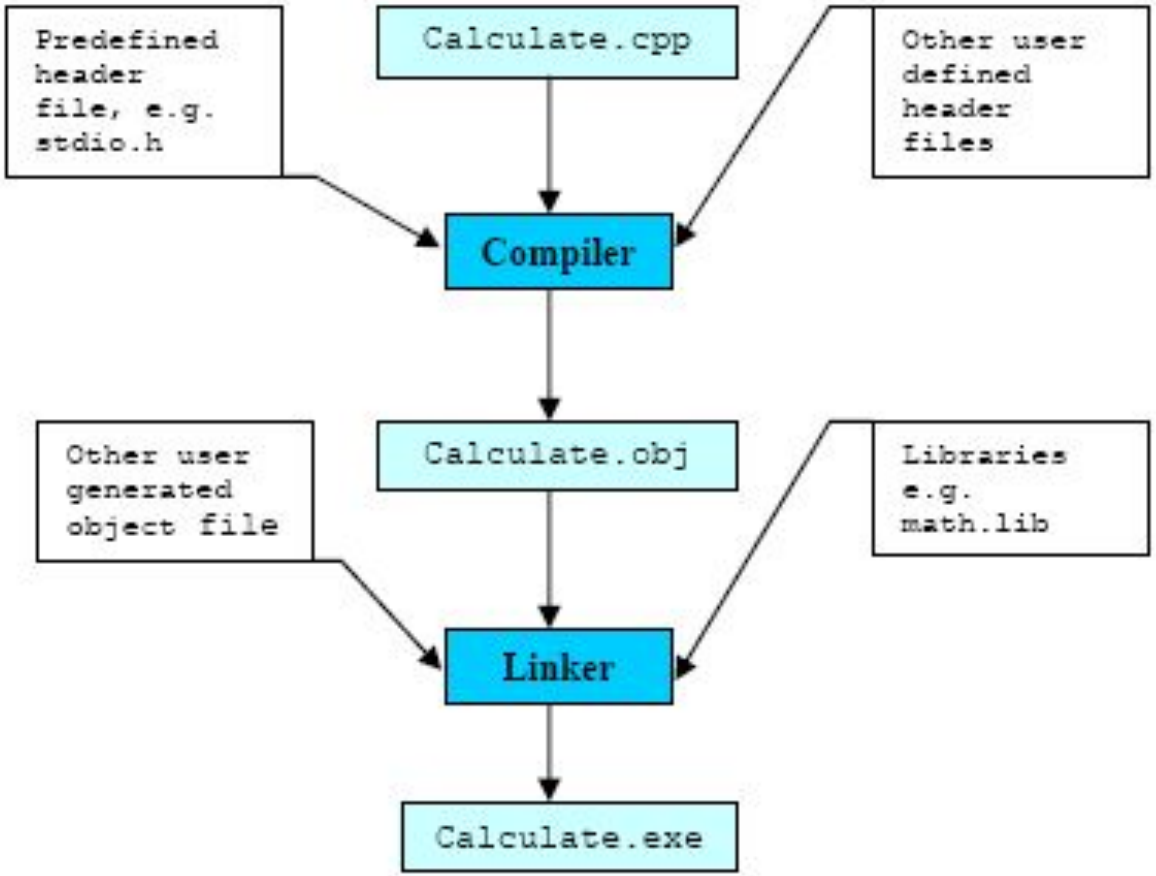
Desenvolvimento iterativo

Nível de abstração



Haskell, Prolog	sum[1..100]
Scheme, Java	mynum.add(5)
C	i++;
Assembly language	iadd
Machine language	10111001010110

Compilação


Predefined header file, e.g. stdio.h
Calculate.cpp
Other user defined header files
Compiler
Other user generated object file
Calculate.obj
Libraries e.g. math.lib
Linker
Calculate.exe

C e C++

C: 1972 (Bell Laboratories)

ANSI C: 1989

C++: publicado em 1985 (Bell Laboratories)

ISO/IEC 14882:1998

Ranking das linguagens

Mar 2018	Mar 2017	Change	Programming Language	Ratings	Change
1	1		Java	14.941%	-1.44%
2	2		C	12.760%	+5.02%
3	3		C++	6.452%	+1.27%
4	5	^	Python	5.869%	+1.95%
5	4	v	C#	5.067%	+0.66%

AMBIENTE DE DESENVOLVIMENTO (Roberto)

```
/*
  Exemplo para Curso de Introdução à Programação de Sistemas Embarcados
*/

const byte pino_do_led = 13;

// a função setup é executada uma vez quando você liga ou reinicia o Arduino
void setup() {
  pinMode(pino_do_led, OUTPUT);  //inicializa o pino_do_led como saída
}

// a função loop é executada repetidamente, para sempre (até o Arduino ser desligado)
void loop() {
  digitalWrite(pino_do_led, HIGH);   // liga o LED (sinal HIGH)
  delay(1000);                       // espera por um tempo
  digitalWrite(pino_do_led, LOW);    // desliga o LED (sinal LOW)
  delay(1000);                       // espera por um tempo
}
```

```
/*
  Exemplo para Curso de Introdução à Programação de Sistemas Embarcados
*/

const byte pino_do_led = 13;

// a função setup é executada uma vez quando você liga ou reinicia o Arduino
void setup() {
  pinMode(pino_do_led, OUTPUT);  //inicializa o pino_do_led como saída
}

// a função loop é executada repetidamente, para sempre (até o Arduino ser desligado)
void loop() {
  digitalWrite(pino_do_led, HIGH);    // liga o LED (sinal HIGH)
  delay(1000);                        // espera por um tempo
  digitalWrite(pino_do_led, LOW);     // desliga o LED (sinal LOW)
  delay(1000);                        // espera por um tempo
}
```

```
//comentário

/*   comentário    */
```

Comentários são muito úteis para ajudar outras pessoas (e você mesmo!) a entender o código.

- Explicar o que um trecho do código faz
- Explicar o porquê

```
/*
  Exemplo para Curso de Introdução à Programação de Sistemas Embarcados
*/

const byte pino_do_led = 13;

// a função setup é executada uma vez quando você liga ou reinicia o Arduino
void setup() {
  pinMode(pino_do_led, OUTPUT); //inicializa o pino_do_led como saída
}

// a função loop é executada repetidamente, para sempre (até o Arduino ser desligado)
void loop() {
  digitalWrite(pino_do_led, HIGH);   // liga o LED (sinal HIGH)
  delay(1000);                       // espera por um tempo
  digitalWrite(pino_do_led, LOW);    // desliga o LED (sinal LOW)
  delay(1000);                       // espera por um tempo
}
```

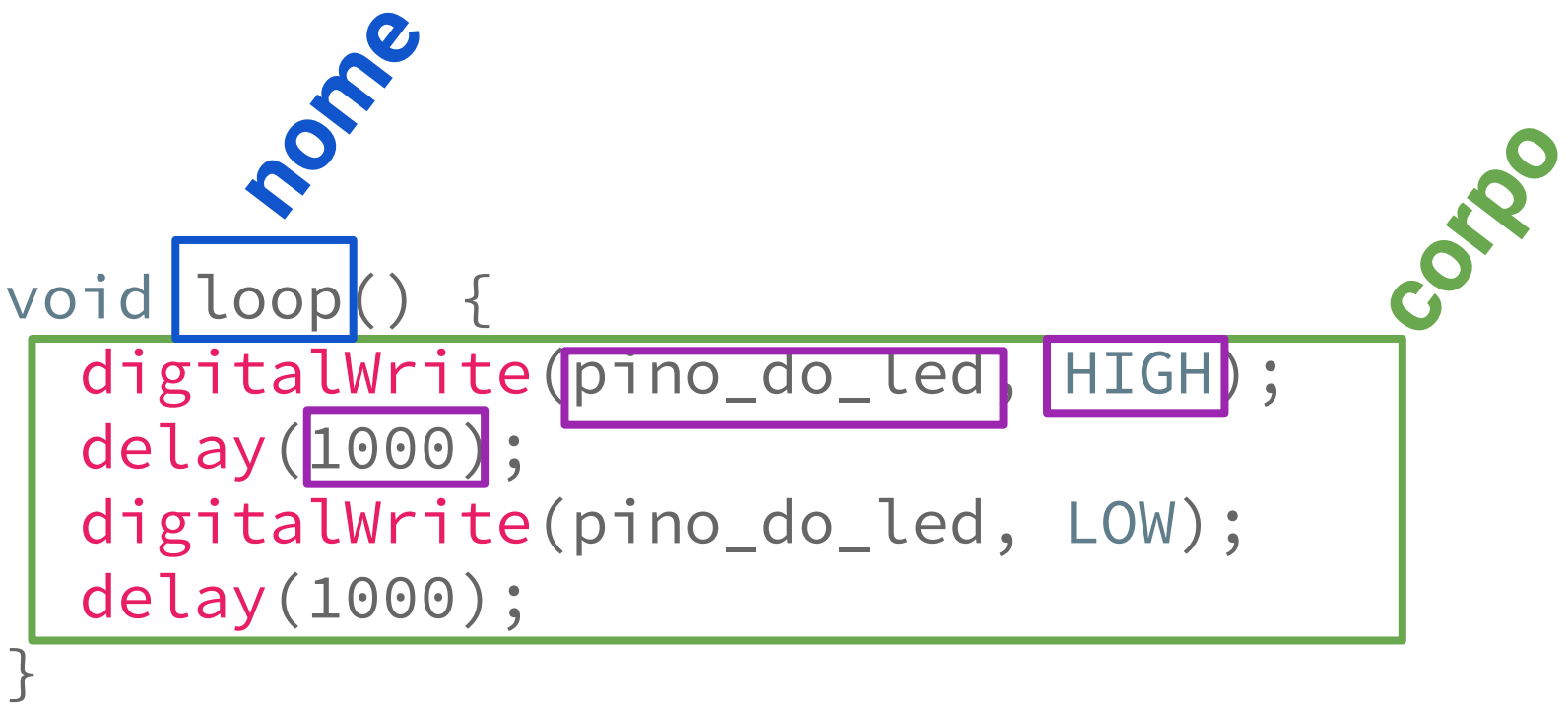
```
/*
  Exemplo para Curso de Introdução à Programação de Sistemas Embarcados
*/

const byte pino_do_led = 13;

// a função setup é executada uma vez quando você liga ou reinicia o Arduino
void setup() {
  pinMode(pino_do_led, OUTPUT);  //inicializa o pino_do_led como saída
}

// a função loop é executada repetidamente, para sempre (até o Arduino ser desligado)
void loop() {
  digitalWrite(pino_do_led, HIGH);   // liga o LED (sinal HIGH)
  delay(1000);                       // espera por um tempo
  digitalWrite(pino_do_led, LOW);    // desliga o LED (sinal LOW)
  delay(1000);                       // espera por um tempo
}
```

Este código está definindo a função chamada setup

```
void setup() {
  pinMode(pino_do_led, OUTPUT);
}
```

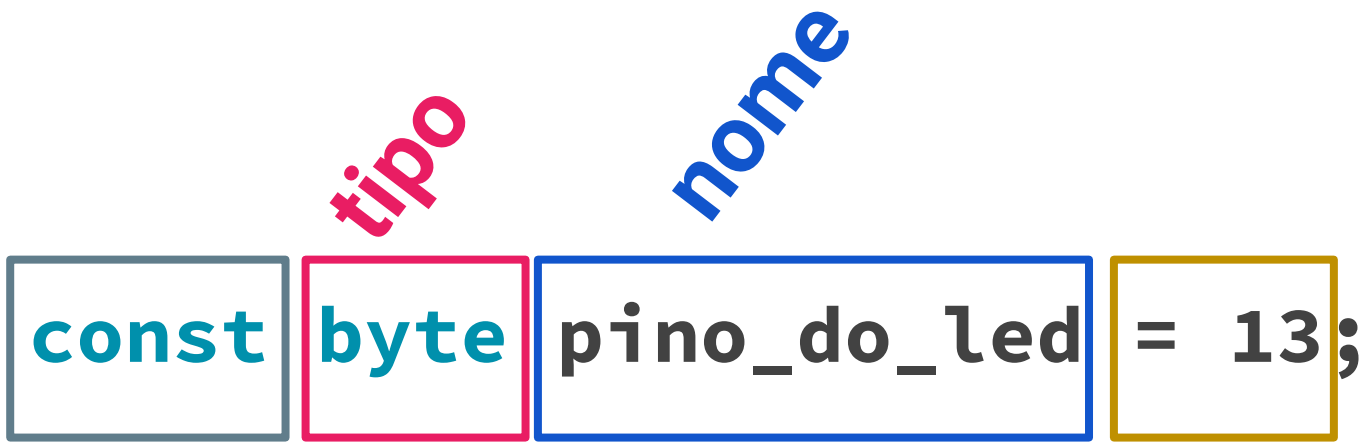
Este código está definindo a função chamada loop

```
void loop() {
  digitalWrite(pino_do_led, HIGH);
  delay(1000);
  digitalWrite(pino_do_led, LOW);
  delay(1000);
}
```

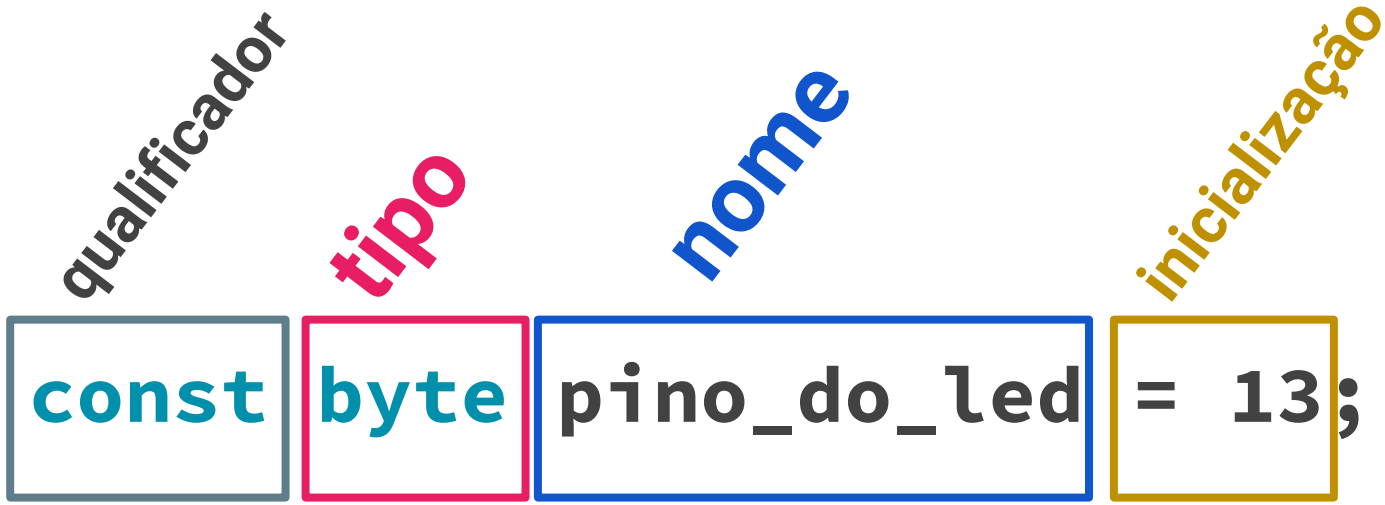

nome
corpo
void loop() {
digitalWrite(pino_do_led, HIGH);
delay(1000);
digitalWrite(pino_do_led, LOW);
delay(1000);
}

```
/*
  Exemplo para Curso de Introdução à Programação de Sistemas Embarcados
*/

const byte pino_do_led = 13;

// a função setup é executada uma vez quando você liga ou reinicia o Arduino
void setup() {
  pinMode(pino_do_led, OUTPUT);  //inicializa o pino_do_led como saída
}

// a função loop é executada repetidamente, para sempre (até o Arduino ser desligado)
void loop() {
  digitalWrite(pino_do_led, HIGH);   // liga o LED (sinal HIGH)
  delay(1000);                       // espera por um tempo
  digitalWrite(pino_do_led, LOW);    // desliga o LED (sinal LOW)
  delay(1000);                       // espera por um tempo
}
```

Variável



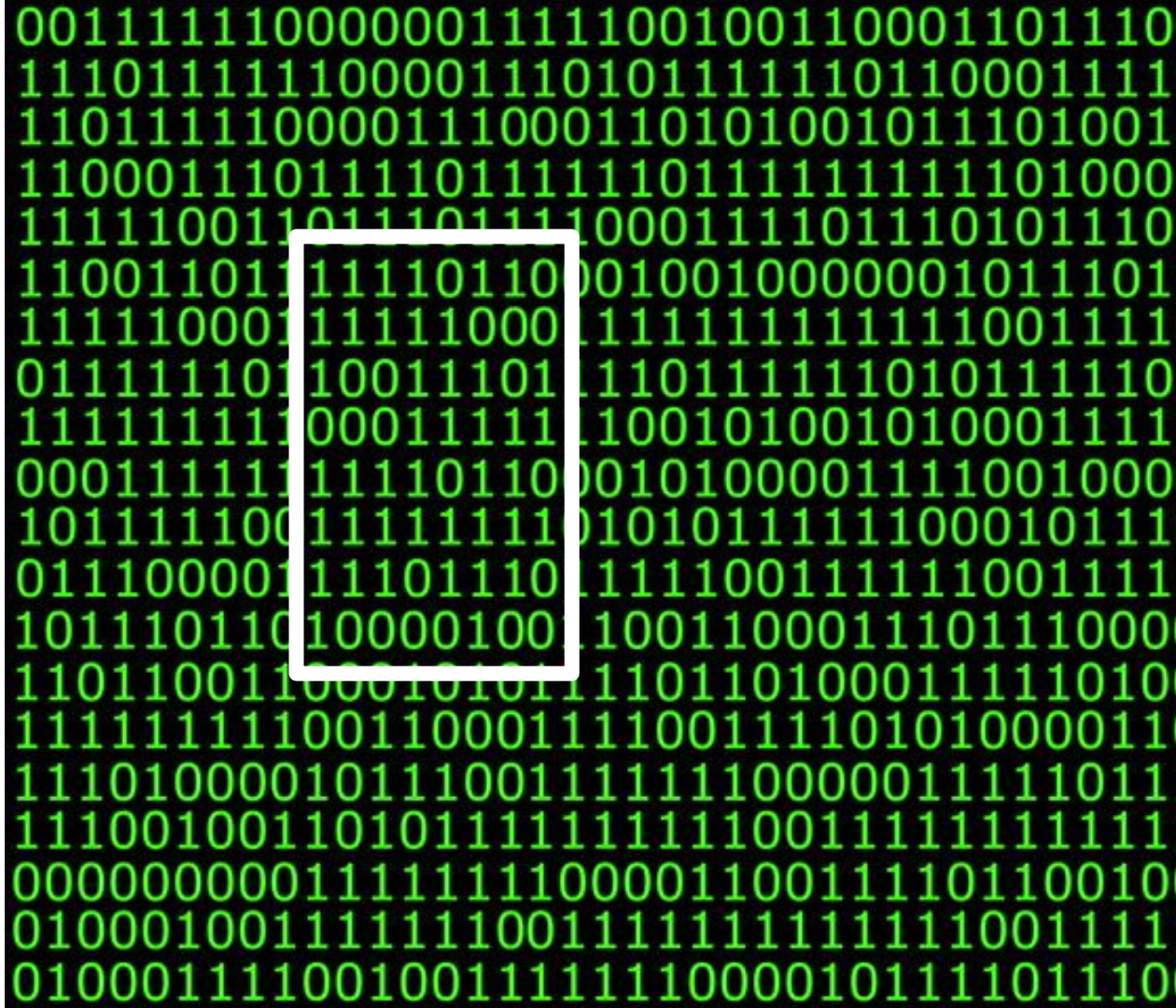
Variável


qualificador
tipo
nome
inicialização
const byte pino_do_led = 13;

Variáveis

nome

tipo



Variáveis

Declaração de variáveis:

```
tipo nome = valor;
```

```
int x = 2;
```

```
int y = 2;
```

Principais tipos de variáveis

Tipo		Tamanho	Exemplo
boolean	Valor false ou true	1 byte	boolean x = true;
char	Um caractere	1 byte	char x = ‘a’;
byte	Um número inteiro de 0 a 255	1 byte	byte x = 100;
int	Um número inteiro de -32,768 to 32,767	2 bytes	int x = -100;
float	Um número com casas decimais de -3.4028235E + 38 to 3.4028235E + 38	4 bytes	float x = 3.14;
String	Um texto	variável	String x =

Atribuição

```
x = 3
```

```
x = 3
```

```
x = 5
```

Operadores numéricos

+	Soma
-	Subtração
*	Multiplicação
/	Divisão
%	Módulo (resto da divisão)

Ordem das operações (numéricas)

---

1. Parênteses e colchetes
2. Multiplicação, divisão e módulo
3. Soma e subtração
4. Atribuição (=)

```
void setup() {
  Serial.begin(9600);
}
void loop() {
  Serial.println(8+5-3*4);
}
```

```
void setup() {
  Serial.begin(9600);
}
void loop() {
  Serial.println(10 + 17/10 -3);
}
```

```
void setup() {
  Serial.begin(9600);
}
void loop() {
  Serial.println(10 + 17.0/10 -3);
}
```

PRÁTICA

Escreva um programa que calcule a seguinte expressão matemática:

$$\frac{10-3\times2}{4+1}+\frac{3}{6}$$

Notação abreviada

A notação abreviada é apenas uma forma reduzida de escrever algumas operações muito comuns:

```
int x = 0;

x++; //equivalente a x = x + 1. Ou seja, x agora vale 1

x--; //equivalente a x = x - 1. Ou seja, x agora vale 0
```

Notação abreviada

A notação abreviada é apenas uma forma reduzida de escrever algumas operações muito comuns:

```
int x = 0;

x++; //incremento

x--; //decremento
```

Comparação de valores

==	igual
!=	diferente
<	menor
>	maior
<=	Menor ou igual
>=	Maior ou igual

Comparação de valores - exemplo

```
int x = 5

x > 3;

x < 3;

x != 3;
```

Operadores booleanos - expressões lógicas

&&	E (and)
||	OU INCLUSIVO (or)
!	NÃO (not)

```
int x = 5;
(x == 5) && (x > 10);
(x == 5) || (x > 10);
```

Operadores booleanos - tabela verdade

Operador E Lógico em C

A	B	A && B
V	V	V
V	F	F
F	V	F
F	F	F

Operador OU Lógico em C

A	B	A || B
V	V	V
V	F	V
F	V	V
F	F	F

Operador NÃO Lógico em C

A	! A
V	F
F	V

Ordem das operações (atualizado)

---

1. Parênteses e colchetes
2. Multiplicação, divisão e módulo
3. Soma e subtração
4. <, <=, >, >=
5. ==, !=
6. &&
7. ||
8. Atribuição (=)

Condicionais - se / senão

```
if (expression) {

// code executed only if expression is true

}

else {

// code executed only if expression is false

}
```

Condicionais - somente o se

```
if (expression) {

// code executed only if expression is true

}
```

Condicionais - múltiplas condições

```
if (expression1) {
// code executed only if expression1 is true
}
else if (expression2) {
// code executed only if expression1 is false
// and expression2 is true
}
else {
// code executed only if expression1 and expression2
//are false
}
```

Laços

Um laço é uma série de ações que se repete.

Tipos de laços que iremos utilizar:

- for
- while

For - estrutura

```
for (declaration & definition ; condition ; increment) {

// statements

}
```

For - exemplo

```
for (int i = 0 ; i < 100 ; i++) {
  Serial.println(i);
}
```

Let's break the loop for the first two and last two i values and see what happens. The values of the integer variable i for the first and second iteration is shown as follows:

- i = 0 , is i smaller than 100 ? yes, println(0) , increment i
- i = 1 , is i smaller than 100 ? yes, println(1) , increment i

For the last two iterations the value of i is shown as

For - escopo

```
for (int i = 0 ; i < 100 ; i++) {
  Serial.println(i);
}

int i = 0;
for ( ; i < 100 ; i++) {
  Serial.println(i);
}
```

For - Mudando o incremento

```
for (int i = 0 ; i < 50 ; i = 2 * i + 1) {

  Serial.println(i);
}
```

For - Decremento

```
for (int i = 50 ; i > 0 ; i = 2 * i - 1) {

  Serial.println(i);
}
```

Loops aninhados (um dentro do outro)

```
for (int x = 1 ; x <= 10 ; x++) {
  for (int y = 1 ; y <= 10 ; y++) {
    Serial.println(x*y);
  }
}
```

While - estrutura

```
while (expression) {

// statements

}
```

The expression is evaluated as a Boolean, true or false . While the expression is true , statements are executed, then as soon as it will be false , the loop will end.

While - exemplo

```
int i = 0;

while (i < 100) {

  Serial.println(x*y);

  i++;

}
```

Cuidado com loops infinitos

Cuidado - não crie loops infinitos sem querer