Co-funded by the
Erasmus+ Programme
of the European Union

UNIDADE 4
Tomada de decisão e funções de controlo

POLITEKNIKA IKASTEGIA TXORIERRI S.COOP.
isep Instituto Superior de Engenharia do Porto
TEI of Crete
Technological Educational Institute of Crete
apro

Objetivo e Conteúdos da Unidade 4

Objetivo

Estuda as funções de controlo de curso e de execução do programa.

Conteúdos

- Estudo **comparativo,** operadores **Booleanos e compostos**
- Apresenta diferente tipos de **funções de sequenciais**
  - **Função If ()**
  - **Função If() else**
  - **Função For()**
  - **Função While()**
- Apresenta diferentes tipos de **funções de control**

Co-funded by the
Erasmus+ Programme
of the European Union

OPERADORES DE COMPARAÇÃO

- O Arduino sabe como fazer comparações entre números ou resultados de determinadas funções.
- Aqui estão vários operadores de comparação assim como os sinais que as representam

OPERATOR	SINAL	OPERADOR	SINAL
Igual a	= =	Diferente de	!=
Menos do que	<	Maior do que	>
Menos do que ou igual a	< =	Maior do que ou igual a	>=

Co-funded by the Erasmus+ Programme of the European Union

OPERADORES BOOLEANOS

- É possível relacionar entre si algumas da comparações anteriores
- Existem apenas dois resultados possíveis quando duas ou mais expressões são relacionadas, utilizando estes operadores lógicos: “true” ou “false”.

OPERADOR	SINAL
NOT	!
AND	&&
OR	||

EXEMPLOS

```
(Letter == 'X') && (A > 10)                          //False
(A == 10+3) && (B >= 12345) && (Letter != 'Q')       //True
(B > 12300) || (PI = 3.1412)                         //True
(A == B) || (A > 10 + 4)                             //False
!(A == B)                                            //True
```

Co-funded by the Erasmus+ Programme of the European Union

OPERADORES DE COMPARAÇÃO

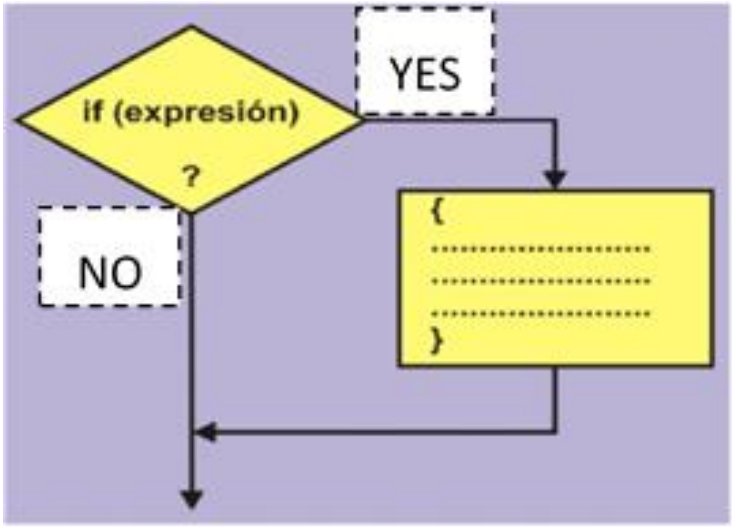
- Diversas vezes, irá efetuar operações muito simples com uma variável e o resultado vai sempre terminar na mesma variável.
- Lembre-se que pode utilizar os denominados “operadores compostos” para simplificar estas expressões.

OPERAÇÃO	OPERADOR	EXEMPLO	IGUAL A
++	aumenta uma unidade	X++	X= X + 1
- -	reduz um número	Y- -	Y = Y - 1
+ =	adição	X+=Y	X = X + Y
-=	subtração	X-= 3	X = X - 3
***=**	multiplicação	X *= Y	X = X * Y
/=	divisão	X /= 5	X = X / 5

Co-funded by the Erasmus+ Programme of the European Union

FUNÇÃO DE CONTROLO

- FUNÇÃO IF()
  - Esta é a função de control mais básica e a mais importante.
  - Se o resultado for “true”, executa todas as funções inseridas nas chavetas “{…}”. Se o resultado é “false” o controlador não as executa e o programa continua.



Co-funded by the
Erasmus+ Programme
of the European Union

FUNÇÃO DE CONTROLO

- FUNÇÃO IF()

if(expression)
{
….
}

expressão: estabelece a condição a que o controlador Arduino tem acesso.
chavetas: assemelham-se às duas fatias de pão numa *sandwich.*

EXEMPLO:

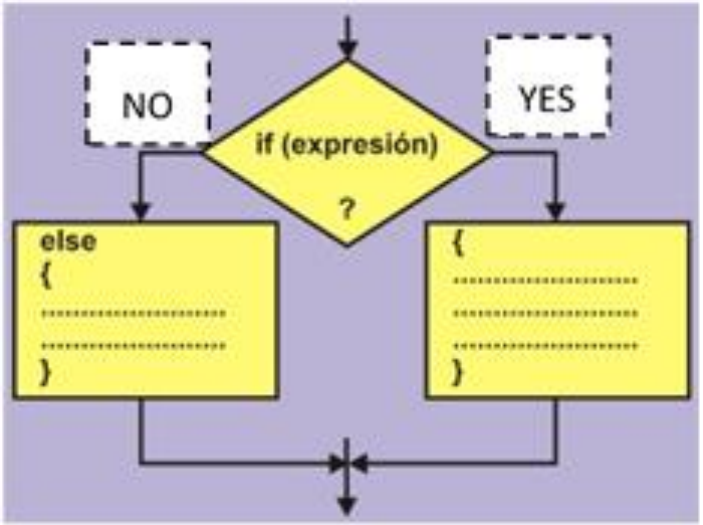
```
 void loop()
{
if((A>B) || (C < 25))                    //Se a condição for true…
{
 digitalWrite(6,HIGH);                   //Troca no pin 6

C=25;                                    //Um valor de 25 é armazenado na variável C
}
....                                     //Continua a execução....
}
```

Co-funded by the Erasmus+ Programme of the European Union

FUNÇÃO DE CONTROLO

- FUNÇÃO IF() ELSE
  - É uma derivada da função if(…) anterior.
  - Se “true”, todas as funções entre as chavetas “{…}” são executadas, à semelhança do que acontece com a função if(…).
  - Se a condição for “false” todas as funções para além das chavetas {…} são executadas. Assim que as funções tiverem sido executadas, quer a condição seja “true” ou não, o programa continua.



Co-funded by the
Erasmus+ Programme
of the European Union

FUNÇÃO DE CONTROLO

➢ FUNÇÃO IF()ELSE

```
if(expression)
{
  ….
}
else
{
….
}
```

expressão: estabelece a condição a que o controlador Arduino tem acesso.

chavetas: assemelham-se às duas fatias de pão numa *sandwich*

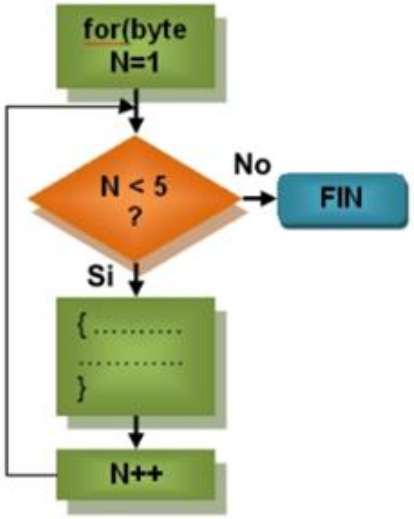
EXEMPLO:

```
if(digitalRead(4) == 1)          //Se o pin 4 está em "1" …
{
digitalWrite(6,HIGH);            //Ativa o pin 6
}
else                             //…e se não…
digitalWrite(6,LOW);             //Desativa o output 6
….                               //Continua a executar o programa
```

Co-funded by the Erasmus+ Programme of the European Union

FUNÇÃO DE CONTROLO

- FUNÇÃO FOR()
  - Esta função permite criar loops controlados
    - ✓ Declara-se um valor inicial
    - ✓ A variável do valor é alterada automaticamente
    - ✓ Se *true*, o campo da afirmação e o incremento são executados e a condição é testada mais uma vez
    - ✓ Quando a condição se torna *false*, o loop termina



Co-funded by the
Erasmus+ Programme
of the European Union

FUNÇÃO DE CONTROLO

- FUNÇÃO FOR()

for(initialização,condição,incremento)
{
….
….
}

initialização: torna possível estabelecer-se o valor de uma variável
condição: é a condição que é testada
incremento: torna possível alterar o valor da variável
chavetas: assemelham-se às duas fatias de pão numa sandwich

EXEMPLO:

```
for (int N = 1; N < 5; N=N+1)          //Estabelece as condições do loop
{
digitalWrite(6,HIGH);                  //Ativa o pin 6
delay (150);                           //Pausa o programa durante 0.15”
digitalWrite(6,LOW);                   //Desativa o pin 6
delay (1000);                          //Pausa o programa durante 1”
}
….                                     //Continua a executar o programa
```

Co-funded by the Erasmus+ Programme of the European Union

FUNÇÃO DE CONTROLO

- FUNÇÃO WHILE()
  - Os loops While são uma variação nas funções for()
  - Também são utilizados para loops enquanto um grupo de funções é executado um certo número de vezes

while(condição)

{

….

….

}

condição: esta é a expressão condicional. Está em loop continuamente e infinitamente, até a expressão dentro das chavetas, {}, se tornar *false*. Quando se torna, o loop termina e o programa continua.

Co-funded by the Erasmus+ Programme of the European Union

FUNÇÃO DE CONTROLO

- FUNÇÃO WHILE()

EXEMPLO:

```
int N = 6 while (N > 0)                 //Enquanto N é maior do que 0
…
{
digitalWrite(6,HIGH);                   //Ativa o pin 6
delay (150);                            //Pausa durante 0.15”
digitalWrite(6,LOW);                    //Desativa o pin 6
delay (1000);                           //Pausa durante 1”
n--;                                    //O valor seguinte de N ( N = N – 1)
}
….
….                                      //Continua a executar o programa
```

Co-funded by the Erasmus+ Programme of the European Union

SWITCH() / FUNÇÃO CASE

- Esta função permite optar entre diversas formas de “ways” executar vários grupos de funções
- A afirmação *switch* compara o valor da variável com os valores especificados em afirmações *case*.

“Se o valor da variável é X executa estas funções. Se o valor da variável é Y executa outras. Se for Z executa outras ainda, etc…”

Co-funded by the Erasmus+ Programme of the European Union

SWITCH() / FUNÇÃO CASE

```
switch(variável)
{
case X:
….;
break;
case n:
….;
….;
break;
default:
….;
}
```

variável: este é o valor da variável que irá ser comparado com os valores em cada um dos "case" mencionados.

case: fixa todos os valores que vão ser comparados aos conteúdos da variável.

default: é opcional. Se nenhum dos valores coincidir são executadas todas as funções.

Co-funded by the Erasmus+ Programme of the European Union

FUNÇÃO DE CONTROLO

Example:

```
switch(A)                          //Variable to be compared
{
case 1:                            //If the value of A is 1...
digitalWrite(6,HIGH);              //Enables pin 6
tone(2,200,200);                   //Tone on pin 2
break;                             //Exit
case 3:                            // If the value of A is 3…
B=digitalRead(7);                  //Reads pin 7
break;                             //Exit
case 124:                          //If the value of A is 124…
B=12*4;                            //The value of B is 48
digitalWrite(11,HIGH)              //Enables pin 11
break;                             //Exit
default:                           //If nothing else matches…
 digitalWrite(6.LOW);              //Disables pin 6
 digitalWrite(11,LOW);             //Disables pin 11
}
```

Co-funded by the
Erasmus+ Programme
of the European Union

OUTRAS FUNÇÕES DE CONTROL

- FUNÇÃO DO…WHILE()

O loop do funciona da mesma forma que o loop while(), com exceção de que a condição é testada no final do loop

```
do
{
….
} while(condition)
```

- FUNÇÃO BREAK

break is used to exit from a for(),while() or do() loop, bypassing the normal loop condition. It is also used to exit from a switch() / case statement

```
break;
```

- FUNÇÃO RETURN

Esta termina uma função e devolve um valor de qualquer função criada pelo utilizador pela função de chamada.

```
return;
return value;
```

value: este é o valor que a função devolve quando regressa ao programa que a solicita

- FUNÇÃO GO TO()

Transfere o curso do programa para um ponto etiquetado do programa.

```
test:
….
goto test:
```

Salta para onde quer que a etiqueta indique.

Co-funded by the Erasmus+ Programme of the European Union

Co-funded by the
Erasmus+ Programme
of the European Union

UNIDADE 4
Tomada de decisão e funções de controlo

Obrigada!

POLITEKNIKA IKASTEGIA TXORIERRI S.COOP.
isep Instituto Superior de Engenharia do Porto
TEI of Crete
Technological Educational Institute of Crete
apro