Tecnologia da Informática

Licenciatura em Engenharia Informática

Universidade de Coimbra

2023/2024

Da aula anterior...

- I/O digital
    - Função `digitalRead(pin)`
    - Ligação em Pull Down
    - Ligação em Pull Up com resistências do Arduino:

```
pinMode(pin, INPUT_PULLUP)
```

- Execução condicional:

```
if (condição){ }
```

Resolução dos exercícios 1.6 e 1.10

Exercício 1.6 - Assuma que montou um circuito com dois LED (um ligado no pino 13 e outro no 12) e um botão de pressão (ligado ao pino 7). Faça um programa para alternar os dois LEDs conforme o estado do botão.

Exercício 1.10 - Tendo presente o exercício 1.5, altere o programa de forma a ligar um LED, sempre que o número de execuções for múltiplo de 3. O programa deve informar também o número de vezes que acendeu o LED. Caso não seja múltiplo de 3, deve desligar o LED.

Para hoje

- *Debouncing*
- Ciclos *for* e *while*
- Operadores bit a bit (*bitwise*)
- Leitura da porta série

A arte de lidar com botões

Exercícios recomendados
(da ficha de exercícios)

Exercício 1.7 - *Debouncing*

a) - Analise o exemplo *debounce* da categoria *Digital*, incluído com o Arduino IDE e tente compreender em que é que consiste o fenómeno de *bouncing*. Averigue como funciona a função *millis()*.

b) - Partindo de um circuito idêntico ao do exercício 1.6, mas agora reduzido a apenas um LED, escreva um programa que acenda e apague um LED sempre que o botão seja premido 3 vezes seguidas.

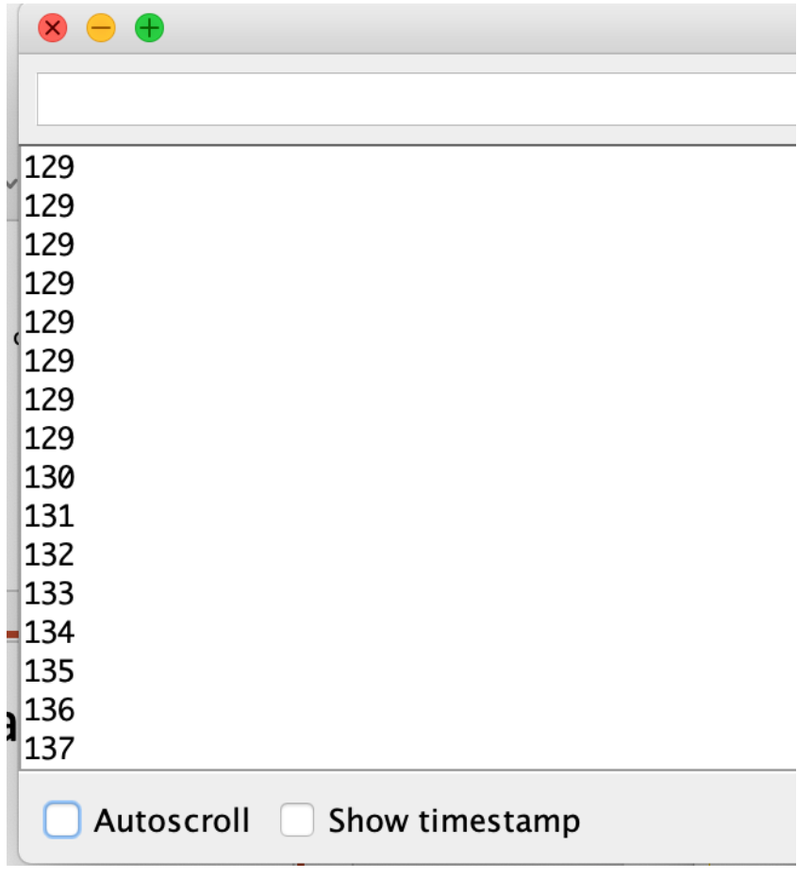
Isto resulta ?

```
int contaprime=0;

void setup(){
  pinMode(7, INPUT_PULLUP);
  Serial.begin(9600);
}

void loop(){ }
  int reading=!digitalRead(7);

  if (reading==true) contaprime=contaprime+1;
  Serial.println(contaprime);
}
```

Isto resulta ?

```
int contaprime=0;

void setup(){
  pinMode(7, INPUT_PULLUP);
  Serial.begin(9600);
}

void loop(){ }
  int reading=!digitalRead(7);

  if (reading==true) contaprime=contaprime+1;
  Serial.println(contaprime);
}
```

Não!



Porquê ?

E como poderia resolver o problema ?

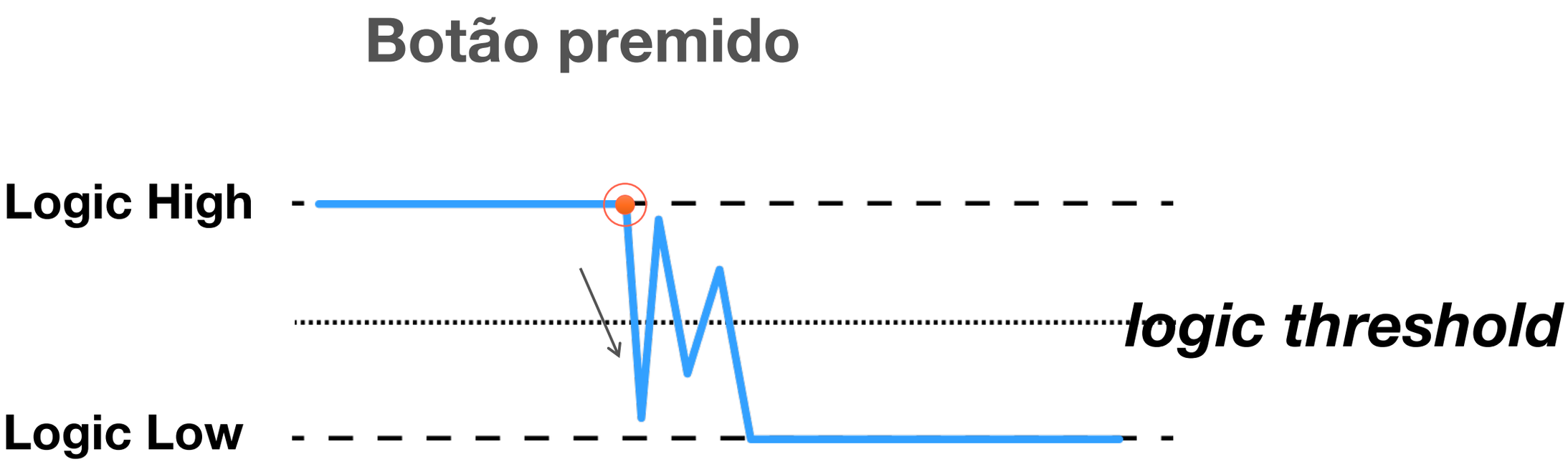
Uma solução

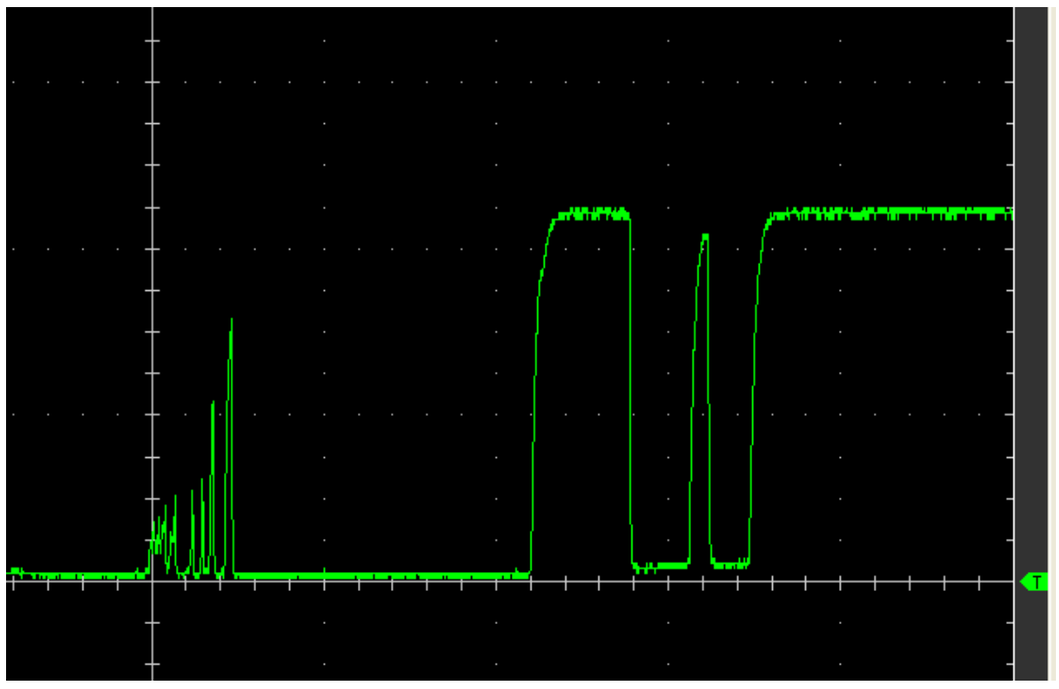
```
int presscount=0;
int lastreading=LOW;

void setup() {
  pinMode(7, INPUT_PULLUP);
  Serial.begin(9600);
}
```

```
void loop() {
 int reading=!digitalRead(7);

  if ((reading==true) && (reading!=lastreading))
    presscount= presscount+1;
  Serial.println(presscount);
  lastreading=reading;
}
```

Está melhor, mas a solução não é perfeita…

...por causa do bouncing





Source: https://www.allaboutcircuits.com/technical-articles/switch-bounce-how-to-deal-with-it/

Como evitar o bouncing ?
Fazendo um *debouncing* por software!!!

1. Ler o estado do botão
2. Se o estado do botão mudou em relação ao *loop* anterior, fazer *reset* a um temporizador
3. Se o intervalo decorrido desde o último *reset* for maior que um patamar pré-definido, assumir que a leitura mais recente corresponde ao estado do botão a assumir
4. Guardar a última leitura do botão para comparação no *loop* seguinte

Debounce, parte I

```
// constants won't change. They're used here to  set pin numbers:
const int buttonPin = 2;      // the number of the pushbutton pin
const int ledPin =  13;       // the number of the LED pin

// Variables will change:
int buttonState = LOW;        // the current reading from the input pin
int lastButtonState = LOW;    // the previous reading from the input pin

// the following variables are longs because the time, measured in ms,
// will quickly become a bigger number than can be stored in an int.
long lastDebounceTime = 0;  // the last time the output pin was toggled
long debounceDelay = 50;    // the debounce time; increase if the output
flickers

void setup() {
  pinMode(buttonPin, INPUT);
  pinMode(ledPin, OUTPUT);
}
```

Debounce, parte II

```
void loop() {
  // read the state of the switch into a local variable:
 int reading = digitalRead(buttonPin);

  // If the switch changed, due to noise or pressing:
  if (reading != lastButtonState) {
    lastDebounceTime = millis(); // reset the debouncing timer
  }

  if ((millis() - lastDebounceTime) > debounceDelay) {
    // whatever the reading is at, it's been there for longer
    // than the debounce delay, so take it as the actual current state:
    if (buttonState!=reading)  buttonState = reading;
  }

  digitalWrite(ledPin, buttonState); //set the LED using the state of the button:

  // save the reading.  Next time through the loop,
  // it'll be the lastButtonState:
  lastButtonState = reading;
}
```

Nota: existe uma diferença entre esta versão e a do Arduino IDE. Esta versão assume que o LED estará sempre no mesmo estado do botão (fazendo o *debounce)*, enquanto que a do Arduino IDE liga e desliga o LED conforme o botão é sucessivamente pressionado.

Estruturas de controle - ciclos

Ciclos

- Permitem repetir um determinado número de vezes um conjunto de instruções.

- Existem vários tipos de ciclos

- O seu uso deve ter em conta o funcionamento da função *loop()*

while

- Executa instruções enquanto for verdade uma determinada condição.

- Sintaxe:

```
while(condição){
   // instruções
   // alterar valor condição
}
```

Exemplo:

```
int n=0;
while(n< 100){
   Serial.println(n);
   n=n+1;
}
```

for

- Executa instruções enquanto for verdade uma determinada condição.

- Sintaxe:

```
for(expressão1; condição; altera_valor_condição){
   // instruções
}
```

Exemplo:

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

for (cont’d)

- É um dos ciclos mais usados, pois permite controlar o inicio e o fim do ciclo.

- As expressões de inicialização e alteração de valores podem ser omitidas (*use at your own risk*).

Exemplo:

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

Exemplos de ciclos

1

```
void setup()
{
    Serial.begin(9600);
}

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

2

```
int i=0;
void setup()
{
    Serial.begin(9600);
}

void loop()
{
   while(i<100){
    Serial.println(i);
    i++;
  }
}
```

- Quantas vezes é executado o ciclo em cada uma das abordagens ?

Exemplos de ciclos (cont'd)

1

```
void setup()
{
    Serial.begin(9600);
}

void loop()
  int n_par=0;
  for (int n=0  ;n < 100 && n_par < 10;){

    if (n%2==0){
       n_par = n_par +1;
       Serial.println(n);
    }
    n++;
   }
}
```

- O que fez este programa ?

Operadores *bitwise*

Números em binário

- Pode-se usar o prefixo 0b
    - Exemplo: 0b10 --> número 2 (decimal)
      (Nota: o prefixo 'B' é também suportado, mas limitado a 8 bits)
- Os inteiros são representados com 16 bits
- Pode-se usar a opção BIN para apresentar o número em formato binário em `Serial.println`().
    - Exemplo: `Serial.println`(3,BIN);

Operadores *bitwise* ou binários

Designação	Operador	Descrição	Exemplo
AND (E)	&	Ambos os bits quando são 1 resultam em 1, caso contrário é 0.	(dec) 5 & 6 = 4 (bin) b0101 & b0110 = b0100
OR (OU)	|	Se um dos bits é um o resultado é 1	(dec) 5 | 6 = 7 (bin) b0101 | b0110 = b0111
XOR (OU EXCLUSIVO)	^	Se os bits são diferentes o resultado é 1, se iguais o valor é 0 (usado para inverter os valores)	(dec) 5 ^ 6 = 3 (bin) b0101 ^ b0110 = b0011
NOT	~	Usado para inverter os valores bit a bit. Equivale a ter –x-1	~5 = -6 ~b0101 = 11111111111111111111111111111010

- No exemplo, para ~5 o resultado é -6. Porquê ?

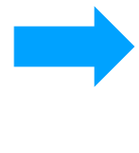
Exemplos

```
void setup() {
  Serial.begin(9600);
  int n1=0b0101; //5, em decimal
  int n2=0b0110; //6, em decimal

  Serial.println("Operador binário E ");
  Serial.println(n1 & n2,DEC);
  Serial.println(n1 & n2,BIN);

  Serial.println("Operador binário OU ");
  Serial.println(n1 | n2,DEC);
  Serial.println(n1 | n2,BIN);

  Serial.println("Operador binário XOR ");
  Serial.println(n1 ^ n2,DEC);
  Serial.println(n1 ^ n2,BIN);

  Serial.println("Operador binário NOT ");
  Serial.println(n1,BIN );
  Serial.println(~n1,DEC );
  Serial.println(~n1,BIN );

}
void loop() { }
```

```
Operador binário E
4
100
Operador binário OU
7
111
Operador binário XOR
3
11
Operador binário NOT
101
-6
11111111111111111111111111111010
```

Exemplos (Exercício)

```
void setup() {
  Serial.begin(9600);
  int n1=20; //b10100
  int n2=19; //b10011
  int n3=8; //b1000

  Serial.println("Numeros ");
  Serial.println(n1, BIN);
  Serial.println(n2, BIN);
  Serial.println(n3, BIN);

  //Que operações binárias pode(m) ser feita com os 3 numeros
  // para o resultado ser 31 ? (b11111)
  ???

  //Que operações binárias pode(m) ser feita com os 3 numeros
  // para o resultado ser 0 ? (b0)
   ???

}
void loop() { }
```



```
Numeros
10100
10011
1000
```

Operadores binários

Designação	Operador	Descrição	Exemplo
Deslocamento à direita	>>	Deslocar um n bits para a direita Deslocar do MSB para o LSB Sintaxe: *numero >> n_bits_deslocar*	5 >> 2 = 1 0b0101 >> 2 = 0b0001 * Os bits 0 e 1 são descartados
Deslocamento à esquerda	<<	Deslocar um n bits para a esquerda Deslocar do LSB para o MSB Sintaxe: *numero << n_bits_deslocar*	5 << 2 = 20 b0101 << 2 = 0b10100

- O deslocamento à esquerda é útil para implementar potências de 2^n

- Mais informação disponível em: https://playground.arduino.cc/Code/BitMath/

Funções para números binários

- Função bitRead(num, i) → ler o bit de um número na posição i.

- Função bitWrite(num, i, valor) → define o valor de do bit i do número num com o valor.

- Função bitSet(num, i) → permite definir o valor 1 no bit i do número num.

- Função bitClear(num, i) → permite definir o valor 0 no bit i do número num.

Exemplos

```
void setup() {
  Serial.begin(9600);
  int deslocamento=1;
  int n1=5;
  Serial.println(n1, BIN);
  Serial.println("deslocando para a direita");
  Serial.println(n1>>deslocamento, BIN);

  Serial.println("deslocando para a esquerda");
  Serial.println(n1<<deslocamento, BIN);

  int n2=0b01010; //10
  int mask=0b00100;

  Serial.println();
  Serial.println(n2,BIN);

  Serial.println("Verifica valor do 3.bit ");
  Serial.println(n2 & mask, BIN);
  Serial.println(bitRead(n2, 2), BIN); // LSB para o MSB

  Serial.println("Atribui valor do 3.bit para 1");
  int n3 = n2 | (1 << 2);
  Serial.println(n3, BIN);
  Serial.println(bitWrite(n2,2,1) , BIN); // ou bitSet
}

void loop() {}
```

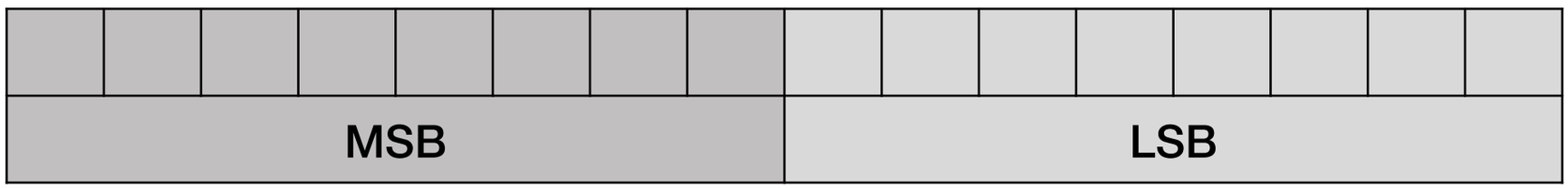
Exemplos

```
void setup() {
  Serial.begin(9600);
  int deslocamento=1;
  int n1=5;
  Serial.println(n1, BIN);
  Serial.println("deslocando para a direita");
  Serial.println(n1>>deslocamento, BIN);

  Serial.println("deslocando para a esquerda");
  Serial.println(n1<<deslocamento, BIN);

  int n2=0b01010; //10
  int mask=0b00100;

  Serial.println();
  Serial.println(n2,BIN);

  Serial.println("Verifica valor do 3.bit ");
  Serial.println(n2 & mask, BIN);
  Serial.println(bitRead(n2, 2), BIN); // LSB para o MS

  Serial.println("Atribui valor do 3.bit para 1");
  int n3 = n2 | (1 << 2);
  Serial.println(n3, BIN);
  Serial.println(bitWrite(n2,2,1) , BIN); // ou bitSet
}

void loop() {}
```

```
deslocando para a direita
10
deslocando para a esquerda
1010

1010
Verifica valor do 3.bit
0
0
Atribui valor do 3.bit para 1
1110
1110
```

Um exemplo diferente

- Considere uma variável *var*, inteira de 16 bits, sem sinal (`unsigned int`).



- Como é que podemos extrair o *Least Significant Byte* ?

  Resposta: (`var & 0x00FF`)

- Como é que podemos extrair o *Most Significant Byte* ?

  Resposta: (`(var >> 8) & 0x00FF`)

Leitura da porta série (caratere a caratere)

Ler da porta série

Leitura caratere a caratere:

```
char incomingByte = 0; // variável para o dado recebido

void setup() {
  Serial.begin(9600);
  Serial.print("START");
}
```

```
void loop() {
  // quando for >0 significa que há bytes para ser lidos
  if (Serial.available() > 0) {
    // lê do buffer um byte de dados:
    incomingByte = Serial.read();

    // imprime com o que foi recebido:
    Serial.print("I received (value/code ASCII): ");
    Serial.println(incomingByte, DEC);
    Serial.print("I received (CHAR): ");
    Serial.println(incomingByte);
  }
}
```

ASCII

American Standard Code for Information Interchange

Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char
0	0	0		32	20	40	[space]	64	40	100	@	96	60	140	`
1	1	1		33	21	41	!	65	41	101	A	97	61	141	a
2	2	2		34	22	42	"	66	42	102	B	98	62	142	b
3	3	3		35	23	43	#	67	43	103	C	99	63	143	c
4	4	4		36	24	44	$	68	44	104	D	100	64	144	d
5	5	5		37	25	45	%	69	45	105	E	101	65	145	e
6	6	6		38	26	46	&	70	46	106	F	102	66	146	f
7	7	7		39	27	47	'	71	47	107	G	103	67	147	g
8	8	10		40	28	50	(	72	48	110	H	104	68	150	h
9	9	11		41	29	51	)	73	49	111	I	105	69	151	i
10	A	12		42	2A	52	*	74	4A	112	J	106	6A	152	j
11	B	13		43	2B	53	+	75	4B	113	K	107	6B	153	k
12	C	14		44	2C	54	,	76	4C	114	L	108	6C	154	l
13	D	15		45	2D	55	-	77	4D	115	M	109	6D	155	m
14	E	16		46	2E	56	.	78	4E	116	N	110	6E	156	n
15	F	17		47	2F	57	/	79	4F	117	O	111	6F	157	o
16	10	20		48	30	60	0	80	50	120	P	112	70	160	p
17	11	21		49	31	61	1	81	51	121	Q	113	71	161	q
18	12	22		50	32	62	2	82	52	122	R	114	72	162	r
19	13	23		51	33	63	3	83	53	123	S	115	73	163	s
20	14	24		52	34	64	4	84	54	124	T	116	74	164	t
21	15	25		53	35	65	5	85	55	125	U	117	75	165	u
22	16	26		54	36	66	6	86	56	126	V	118	76	166	v
23	17	27		55	37	67	7	87	57	127	W	119	77	167	w
24	18	30		56	38	70	8	88	58	130	X	120	78	170	x
25	19	31		57	39	71	9	89	59	131	Y	121	79	171	y
26	1A	32		58	3A	72	:	90	5A	132	Z	122	7A	172	z
27	1B	33		59	3B	73	;	91	5B	133	[	123	7B	173	{
28	1C	34		60	3C	74	<	92	5C	134	\	124	7C	174	|
29	1D	35		61	3D	75	=	93	5D	135	]	125	7D	175	}
30	1E	36		62	3E	76	>	94	5E	136	^	126	7E	176	~
31	1F	37		63	3F	77	?	95	5F	137	_	127	7F	177	

Para pensar/fazer fora da aula

Exercícios recomendados (da ficha de exercícios)

Exercício 2.1 – Escreva um programa que implemente a potência de 2^n usando o operador de deslocamento binário. O programa deve iniciar quando for premido o botão de pressão e deve fazer as seguintes potências 2^0, 2^1, 2^2, 2^3, 2^4. Utilize a função `Serial.println`() para mostrar o resultado em decimal e em binário. Não precisa de fazer *debounce*.

Exercício 2.2 - Escreva um programa que permita verificar se um dado número é uma potência de dois usando os operadores binários. Use variáveis com o valor 5, 8 e 10 para validar o seu programa. Sugestão: utilize o AND binário entre o próprio número e o número menos uma unidade. Utilize a função `Serial.println`() para mostrar o resultado.

Exercício 2.3 – Resolva o exercício 2.1 utilizando ciclos. Altere-o de forma a imprimir as potências de 2 inferiores a 100.

Exercício 2.11 - Como sabe, a leitura da porta série é feita sob a forma de carateres em código ASCII. Faça um programa que leia um dígito entre 0 e 9 e o converta para o respetivo valor numérico (para armazenar numa variável do tipo byte). Acrescente validações para que outros carateres não sejam aceites.