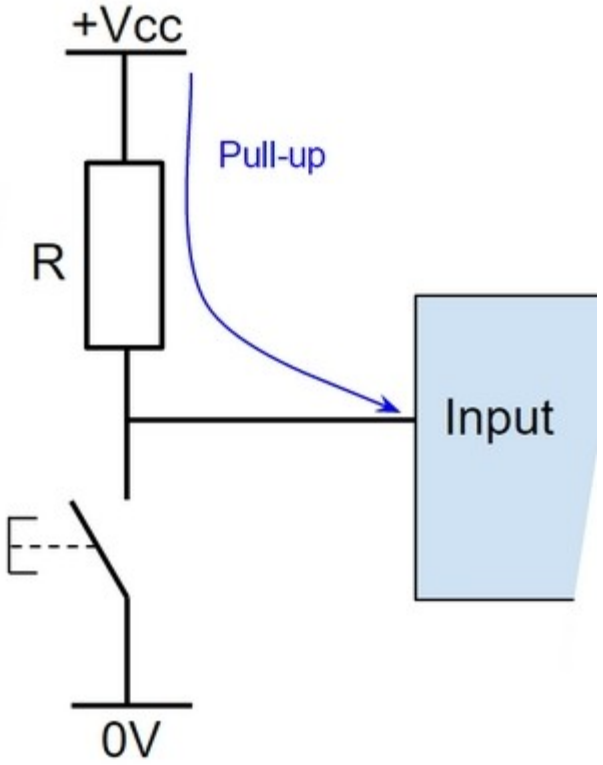
Le µC – Résistances de tirage

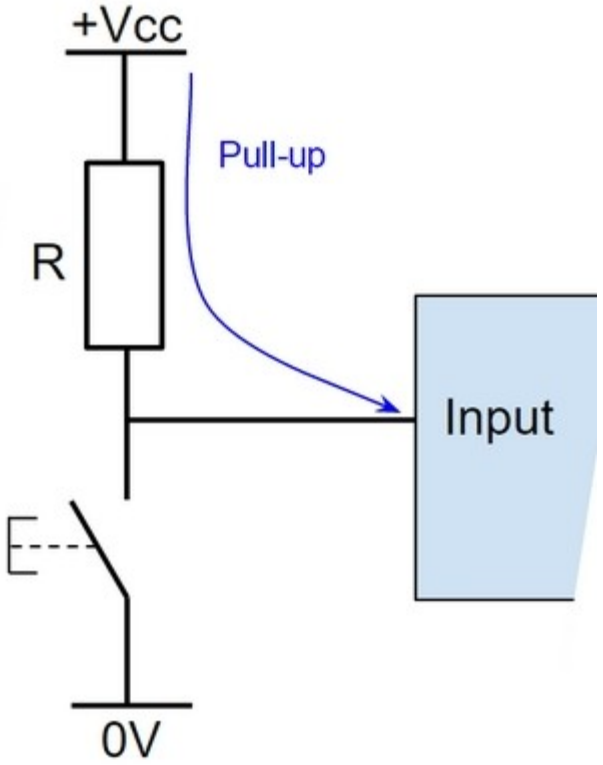
Pull-Up Resistor → Résistance de tirage vers le haut (c'est à dire le +Vcc)



BP	Input
0	NL1
1	NL0

Le µC – Résistances de tirage

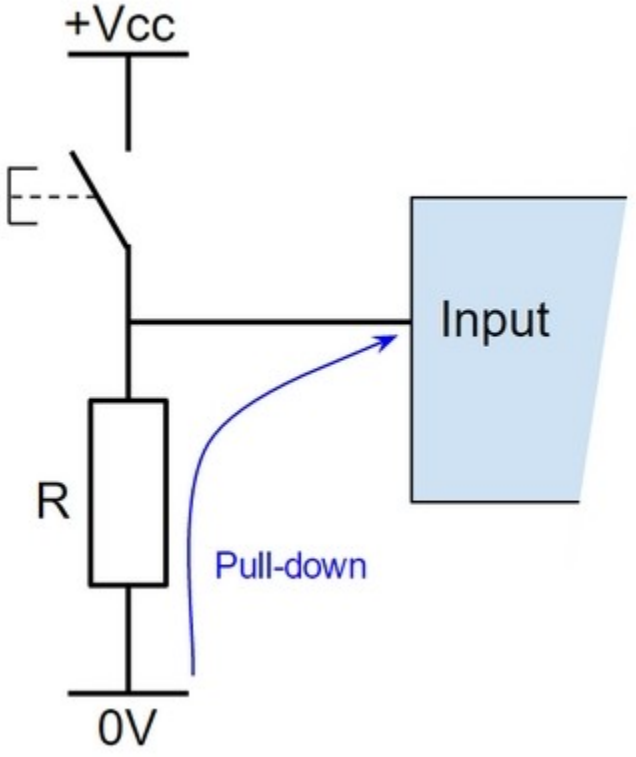
Pull-Up Resistor → Résistance de tirage vers le haut (c'est à dire le +Vcc)




```
const int numero_broche = xxx;

void setup() {
  pinMode(numero_broche,INPUT);
}
```

Le µC – Résistances de tirage

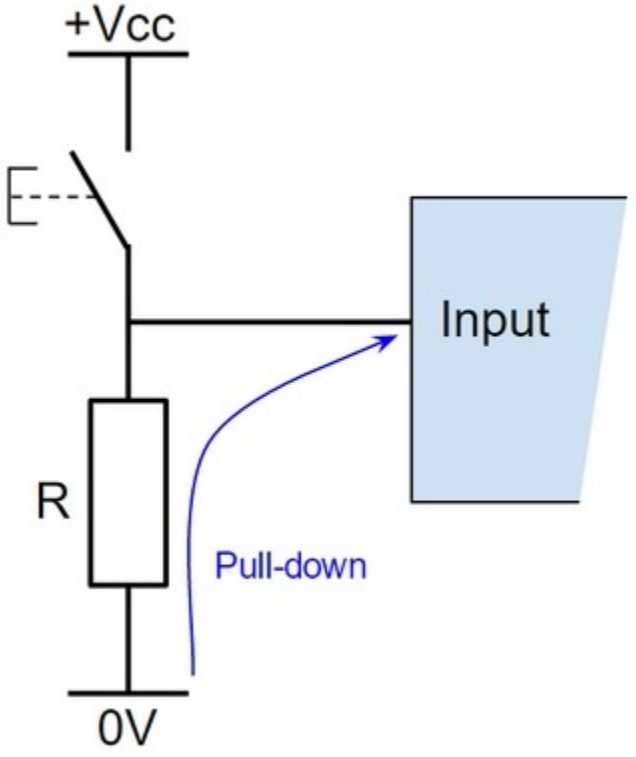
Pull-Down Resistor → Résistance de tirage vers le bas (c'est à dire le GND)



BP	Input
0	NL0
1	NL1

Le µC – Résistances de tirage

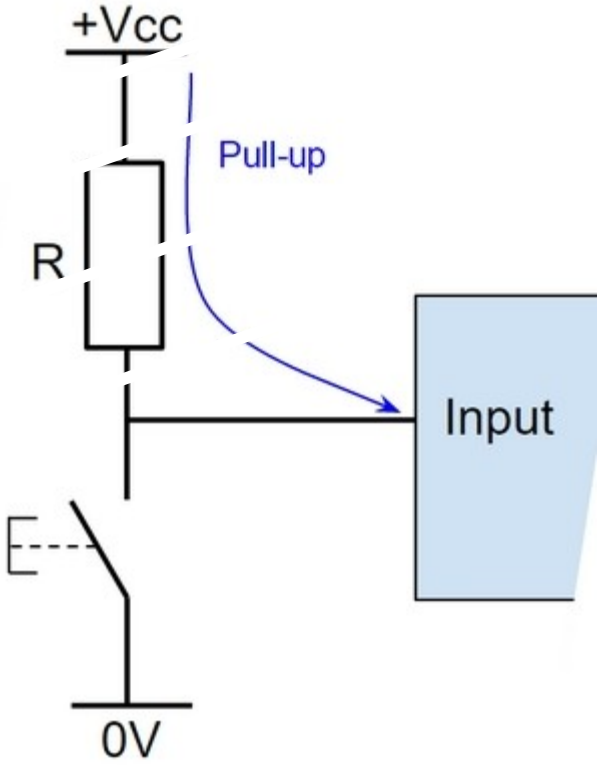
Pull-Down Resistor → Résistance de tirage vers le bas (c'est à dire le GND)




```
const int numero_broche = xxx;

void setup() {
  pinMode(numero_broche,INPUT);
}
```

Le µC – Résistances de tirage

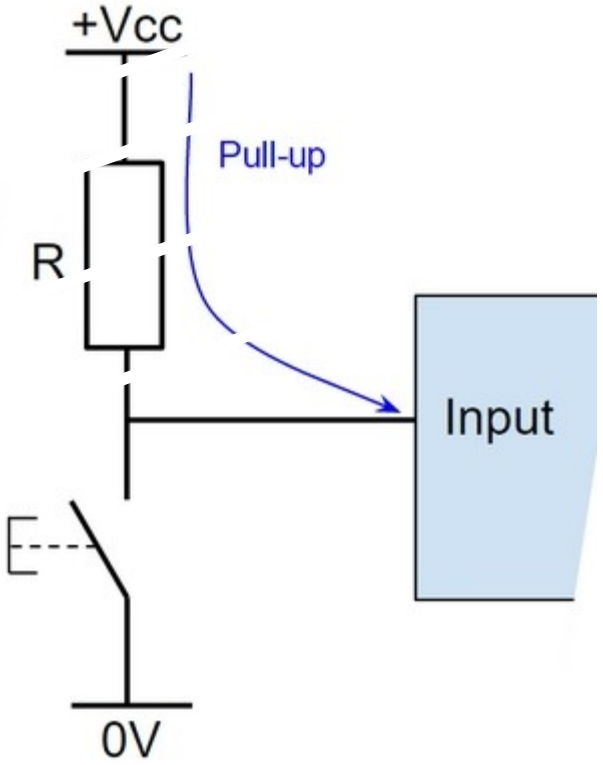
Internal Pull-Up Resistor → Résistance de tirage vers le haut (c'est à dire le +Vcc) mais en interne au µC



BP	Input
0	NL1
1	NL0

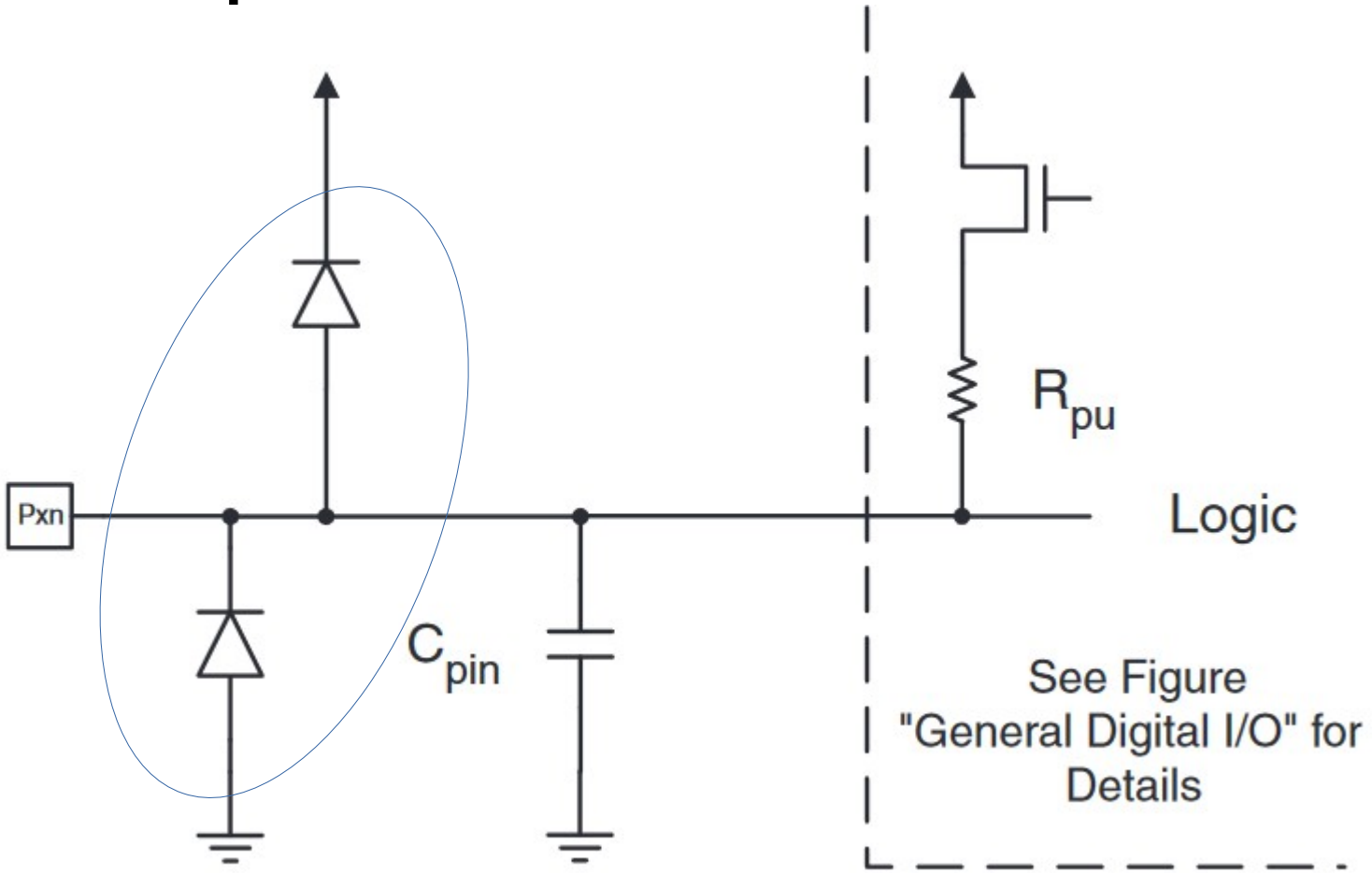
Le µC – Résistances de tirage

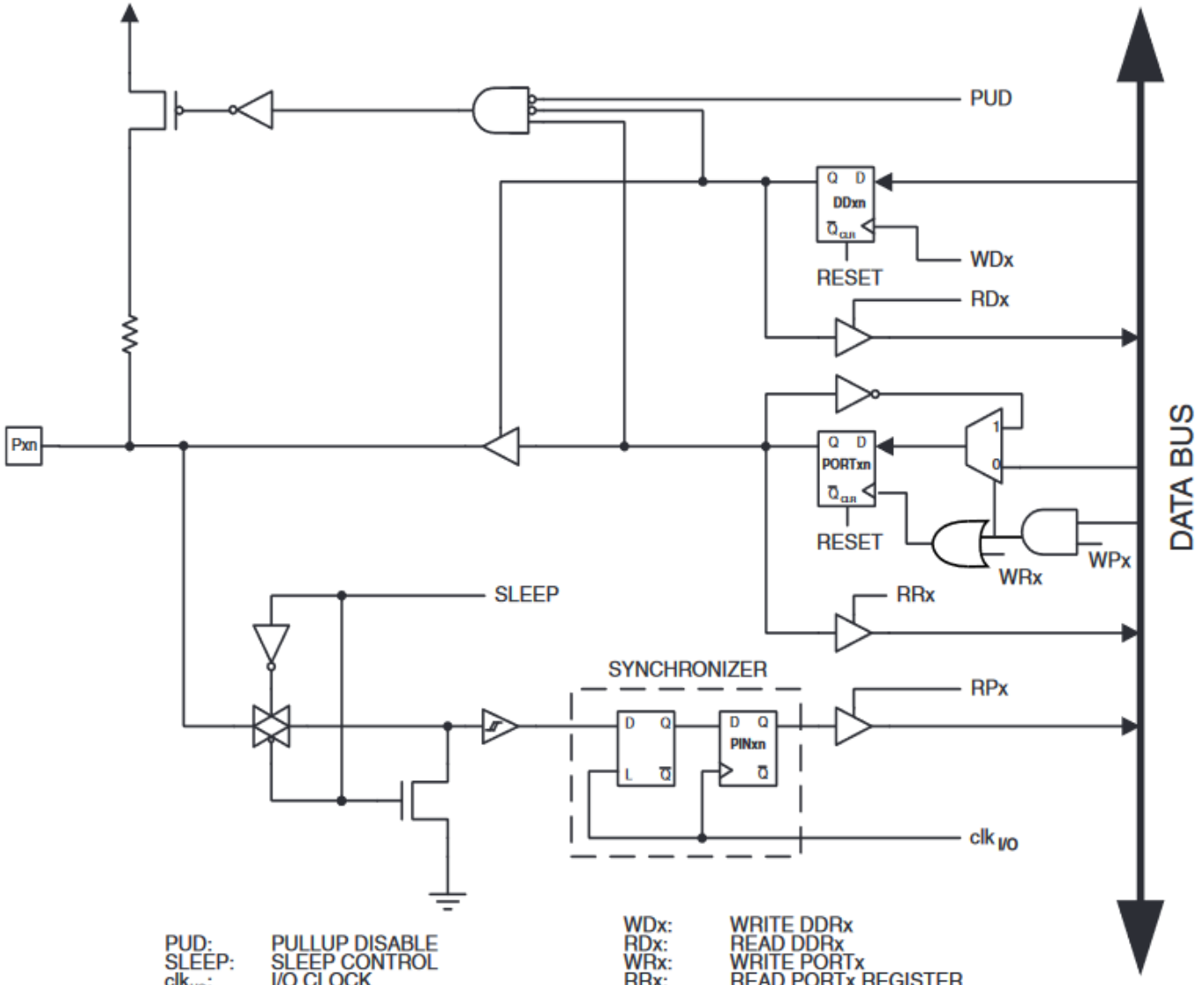
Internal Pull-Up Resistor → Résistance de tirage vers le haut (c'est à dire le +Vcc) mais en **interne** au µC




```
const int numero_broche = xxx;

void setup() {
  pinMode(numero_broche,INPUT_PULLUP);
}
```

Le µC – Protection des entrées





PUD: PULLUP DISABLE
SLEEP: SLEEP CONTROL
$clk_{I/O}$: I/O CLOCK

WDx: WRITE DDRx
RDx: READ DDRx
WRx: WRITE PORTx
RRx: READ PORTx REGISTER
RPx: READ PORTx PIN
WPx: WRITE PINx REGISTER

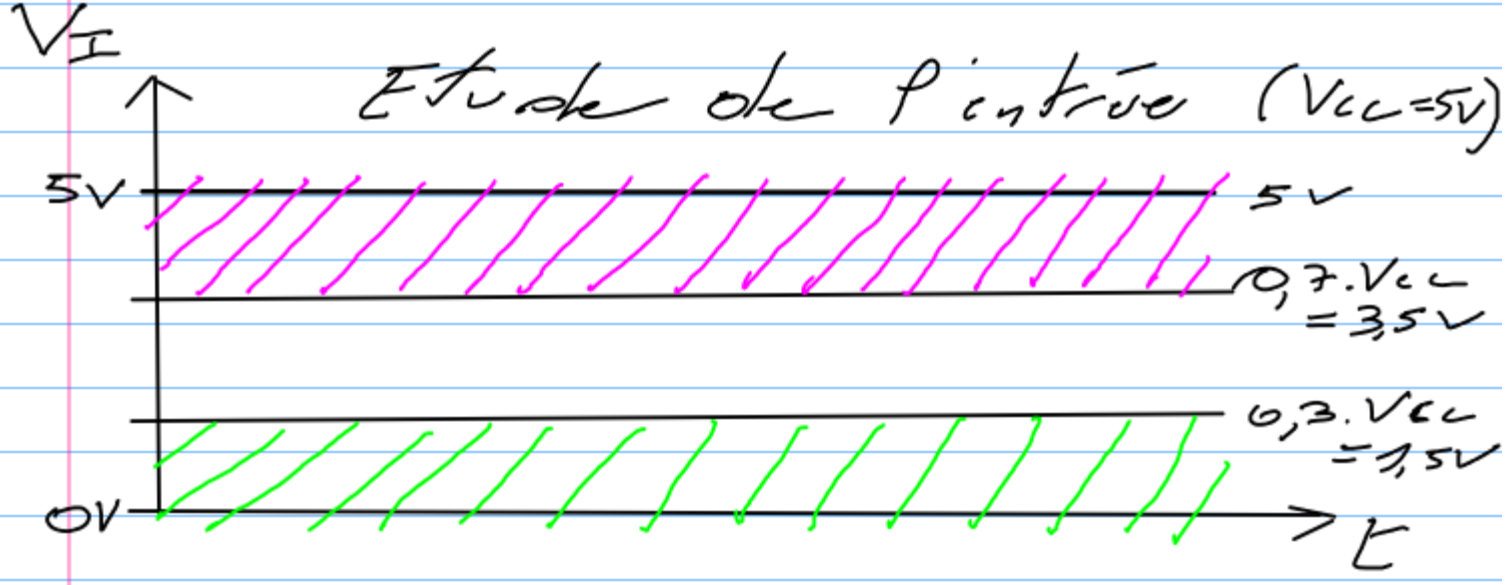
Symbol	Parameter	Condition		Min.	Typ.	Max.	Units
V_{IL}	Input Low Voltage, except XTAL1 and RESET pin	V_{CC} = 1.8V - 2.4V V_{CC} = 2.4V - 5.5V		-0.5 -0.5		$0.2V_{CC}$[1] $0.3V_{CC}$[1]	V
V_{IH}	Input High Voltage, except XTAL1 and RESET pins	V_{CC} = 1.8V - 2.4V V_{CC} = 2.4V - 5.5V		$0.7V_{CC}$[2] $0.6V_{CC}$[2]		V_{CC} + 0.5 V_{CC} + 0.5	V
V_{IL1}	Input Low Voltage, XTAL1 pin	V_{CC} = 1.8V - 5.5V		-0.5		$0.1V_{CC}$[1]	V
V_{IH1}	Input High Voltage, XTAL1 pin	V_{CC} = 1.8V - 2.4V V_{CC} = 2.4V - 5.5V		$0.8V_{CC}$[2] $0.7V_{CC}$[2]		V_{CC} + 0.5 V_{CC} + 0.5	V
V_{IL2}	Input Low Voltage, RESET pin	V_{CC} = 1.8V - 5.5V		-0.5		$0.1V_{CC}$[1]	V
V_{IH2}	Input High Voltage, RESET pin	V_{CC} = 1.8V - 5.5V		$0.9V_{CC}$[2]		V_{CC} + 0.5	V
V_{IL3}	Input Low Voltage, RESET pin as I/O	V_{CC} = 1.8V - 2.4V V_{CC} = 2.4V - 5.5V		-0.5 -0.5		$0.2V_{CC}$[1] $0.3V_{CC}$[1]	V
V_{IH3}	Input High Voltage, RESET pin as I/O	V_{CC} = 1.8V - 2.4V V_{CC} = 2.4V - 5.5V		$0.7V_{CC}$[2] $0.6V_{CC}$[2]		V_{CC} + 0.5 V_{CC} + 0.5	V
V_{OL}	Output Low Voltage[4] except RESET pin	I_{OL} = 20mA, V_{CC} = 5V	T_A=85°C			0.9	
			T_A=105°C			1.0	
		I_{OL} = 10mA, V_{CC} = 3V	T_A=85°C			0.6	
			T_A=105°C			0.7	V

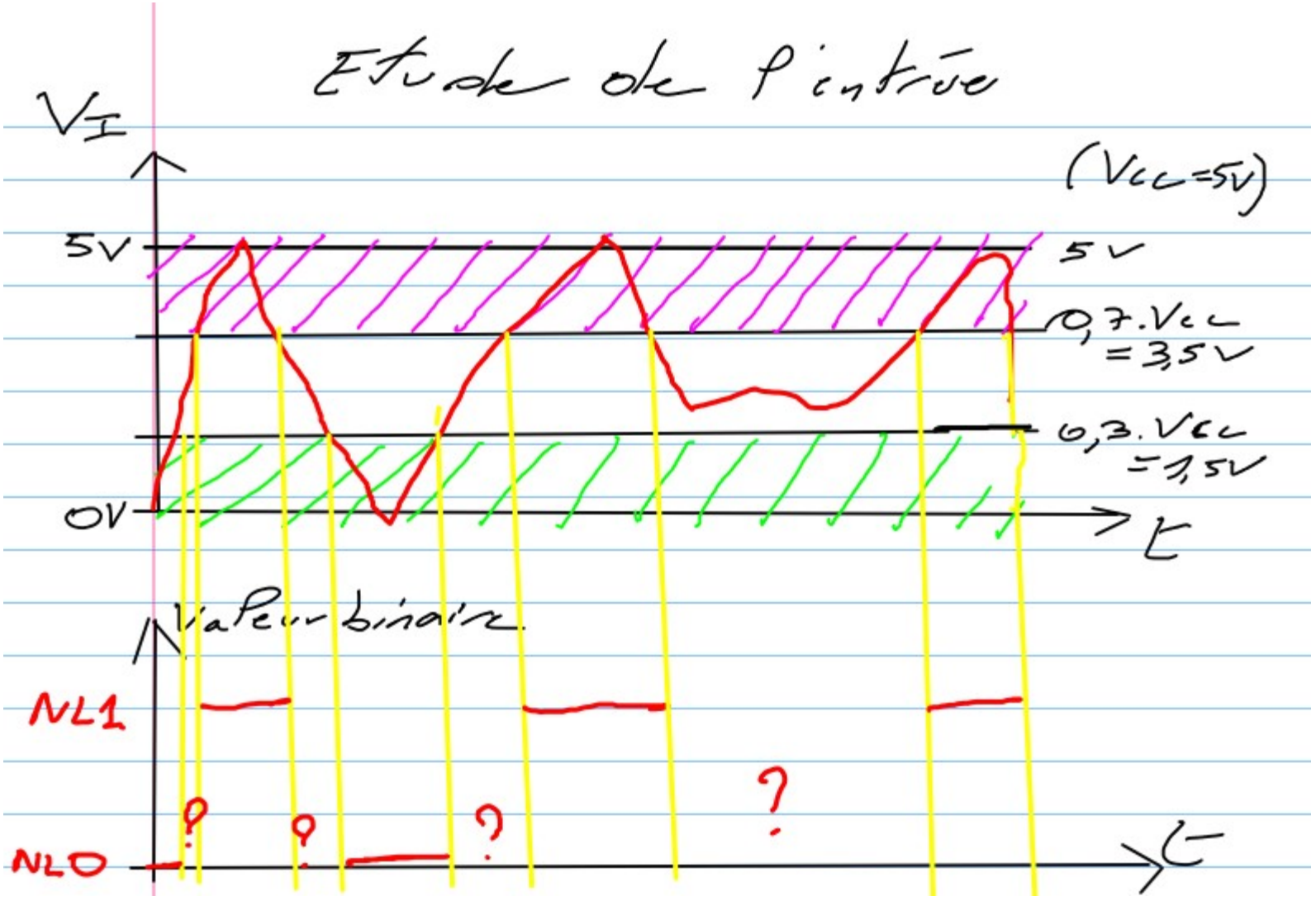
HIGH

V_{IH} ⇒ Niveau logique 1 en entrée

INPUT

V_{IL} ← Low ⇒ " " 0



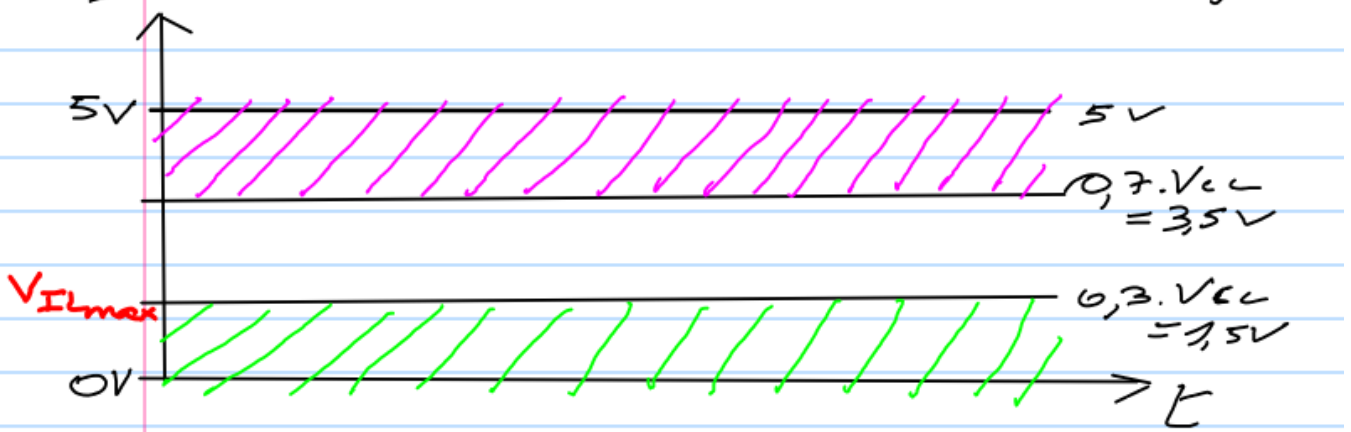

Etude de l'entrée
V_I
(V_{cc}=5V)
5V
0V
5 V
0,7.Vcc
=3,5V
0,3.Vcc
=1,5V
t
Valeur binaire
NL1
NL0
?
?
?
?
t

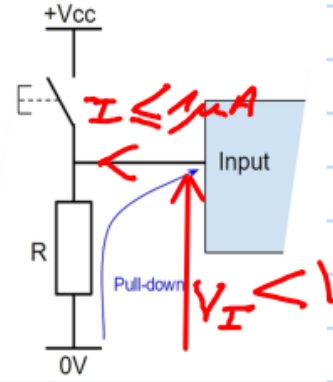
Symbol	Parameter	Condition		Min.	Typ.	Max.	Units
V_{IL}	Input Low Voltage, except XTAL1 and RESET pin	V_{CC} = 1.8V - 2.4V V_{CC} = 2.4V - 5.5V		-0.5 -0.5		$0.2V_{CC}$[1] $0.3V_{CC}$[1]	V
V_{IH}	Input High Voltage, except XTAL1 and RESET pins	V_{CC} = 1.8V - 2.4V V_{CC} = 2.4V - 5.5V		$0.7V_{CC}$[2] $0.6V_{CC}$[2]		V_{CC} + 0.5 V_{CC} + 0.5	V
V_{IL1}	Input Low Voltage, XTAL1 pin	V_{CC} = 1.8V - 5.5V		-0.5		$0.1V_{CC}$[1]	V
V_{IH1}	Input High Voltage, XTAL1 pin	V_{CC} = 1.8V - 2.4V V_{CC} = 2.4V - 5.5V		$0.8V_{CC}$[2] $0.7V_{CC}$[2]		V_{CC} + 0.5 V_{CC} + 0.5	V
V_{IL2}	Input Low Voltage, RESET pin	V_{CC} = 1.8V - 5.5V		-0.5		$0.1V_{CC}$[1]	V
V_{IH2}	Input High Voltage, RESET pin	V_{CC} = 1.8V - 5.5V		$0.9V_{CC}$[2]		V_{CC} + 0.5	V
V_{IL3}	Input Low Voltage, RESET pin as I/O	V_{CC} = 1.8V - 2.4V V_{CC} = 2.4V - 5.5V		-0.5 -0.5		$0.2V_{CC}$[1] $0.3V_{CC}$[1]	V
V_{IH3}	Input High Voltage, RESET pin as I/O	V_{CC} = 1.8V - 2.4V V_{CC} = 2.4V - 5.5V		$0.7V_{CC}$[2] $0.6V_{CC}$[2]		V_{CC} + 0.5 V_{CC} + 0.5	V
V_{OL}	Output Low Voltage[4] except RESET pin	I_{OL} = 20mA, V_{CC} = 5V	T_A=85°C			0.9	
			T_A=105°C			1.0	
		I_{OL} = 10mA, V_{CC} = 3V	T_A=85°C			0.6	
			T_A=105°C			0.7	V

Symbol	Parameter	Condition		Min.	Typ.	Max.	Units
V_{IL}	Input Low Voltage, except XTAL1 and RESET pin	V_{CC} = 1.8V - 2.4V V_{CC} = 2.4V - 5.5V		-0.5 -0.5		$0.2V_{CC}$[1] $0.3V_{CC}$[1]	V
V_{IH}	Input High Voltage, except XTAL1 and RESET pins	V_{CC} = 1.8V - 2.4V V_{CC} = 2.4V - 5.5V		$0.7V_{CC}$[2] $0.6V_{CC}$[2]		V_{CC} + 0.5 V_{CC} + 0.5	V
V_{IL1}	Input Low Voltage, XTAL1 pin	V_{CC} = 1.8V - 5.5V		-0.5		0.1V[1]	V
V_{IH1}	Input High Voltage, XTAL1 pin	V_{CC} = 1.8V - 2.4V V_{CC} = 2.4V - 5.5V		0.8V[2]		V_{CC} + 0.5	V
V_{IL2}	Input Low Voltage, RESET pin	V					
V_{IH2}	Input H	V					V
V_{IL3}	Input L			-0.5		$0.2V_{CC}$[1] $0.3V_{CC}$[1]	V
		2.4V		$0.7V_{CC}$[2] $0.6V_{CC}$[2]		V_{CC} + 0.5 V_{CC} + 0.5	V
V_{CC} = 2.4V - 5.5V							
V_{OL}	Output Low Voltage[4] except RESET pin	I_{OL} = 20mA, V_{CC} = 5V	T_A=85°C			0.9	
			T_A=105°C			1.0	
		I_{OL} = 10mA, V_{CC} = 3V	T_A=85°C			0.6	
			T_A=105°C			0.7	V

C_i[1]	Capacitance for each I/O Pin		–	10	pF
I_{IL}	Input Leakage Current I/O Pin	V_{CC} = 5.5V, pin low (absolute value)		1	µA
I_{IH}	Input Leakage Current I/O Pin	V_{CC} = 5.5V, pin high (absolute value)		1	µA

Etude de l'entrée (V_{CC}=5V)





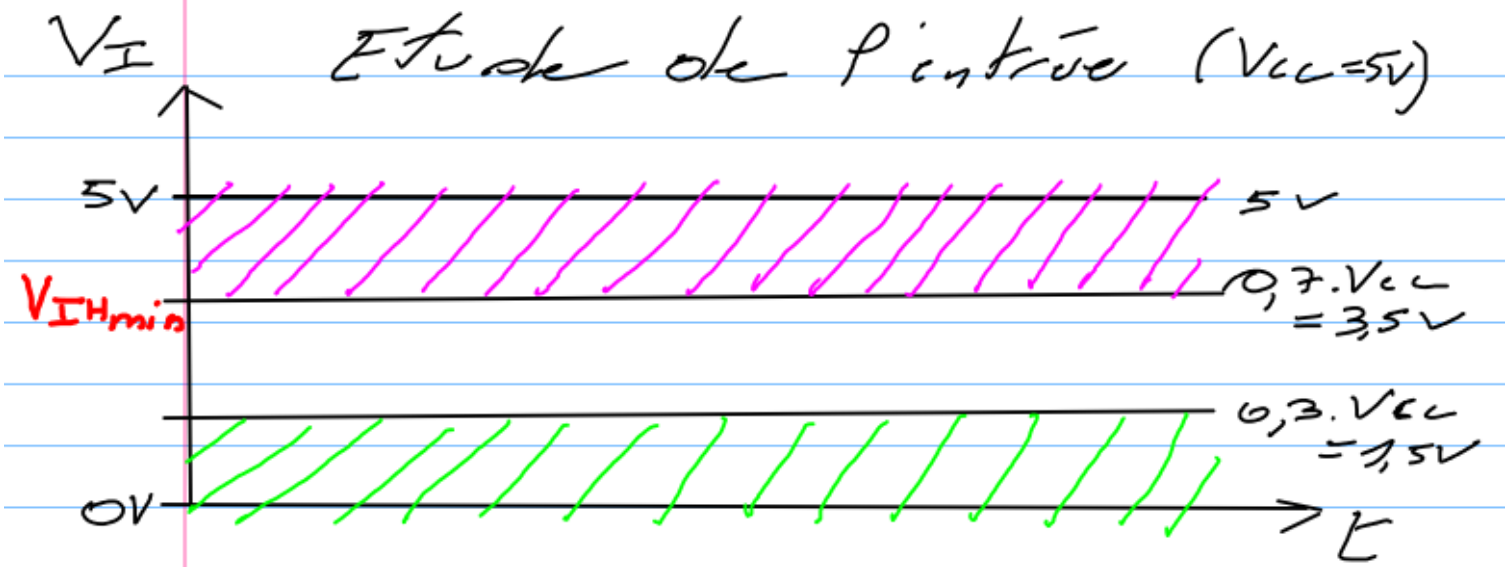
Pull . Down

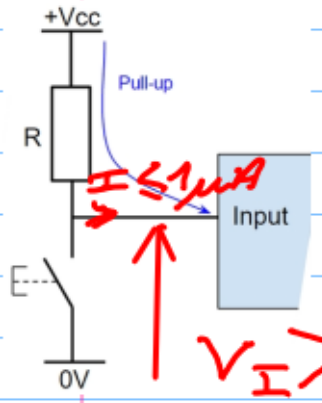
$V_I < V_{ILmax} = 0,3.Vcc = 1,5\,V$

$$R < \frac{V_{ILmax}}{I} = \frac{1,5}{1.10^{-6}} = 1,5\,M\Omega$$

$R < 1,5\,M\Omega$

Etude de l'entrée (V_{CC}=5V)





Pull-up

$V_I > V_{IH_{min}} = 0,7.V_{CC} = 3,5V$

$$R < \frac{5-3,5}{1.10^{-6}} = \frac{1,5}{1.10^{-6}} = 1,5\,M\Omega$$

$R < 1{,}5\,M\Omega$

Rapport de 10 $\Rightarrow$ $R < 150\,k\Omega$

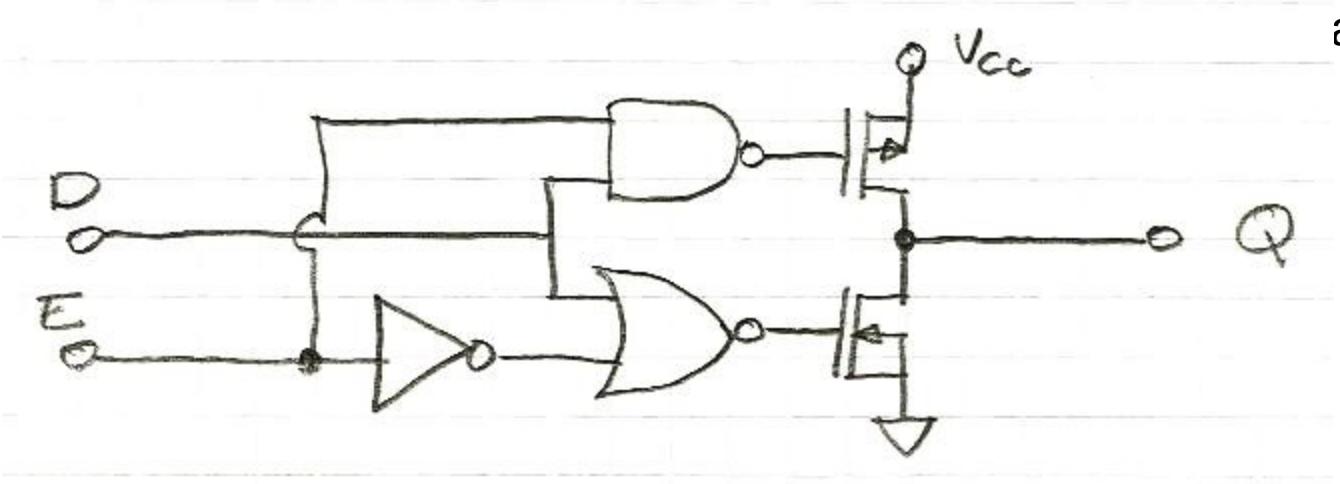
$\Rightarrow$ Technicien : $1\,k\Omega < R < 100\,k\Omega$

Les sorties

Sortie 3 états : NL0 ; NL1 et l'état Z, c'est à dire état « Haute Impédance »
ou « Collecteur Ouvert »

(si transistors à jonctions)

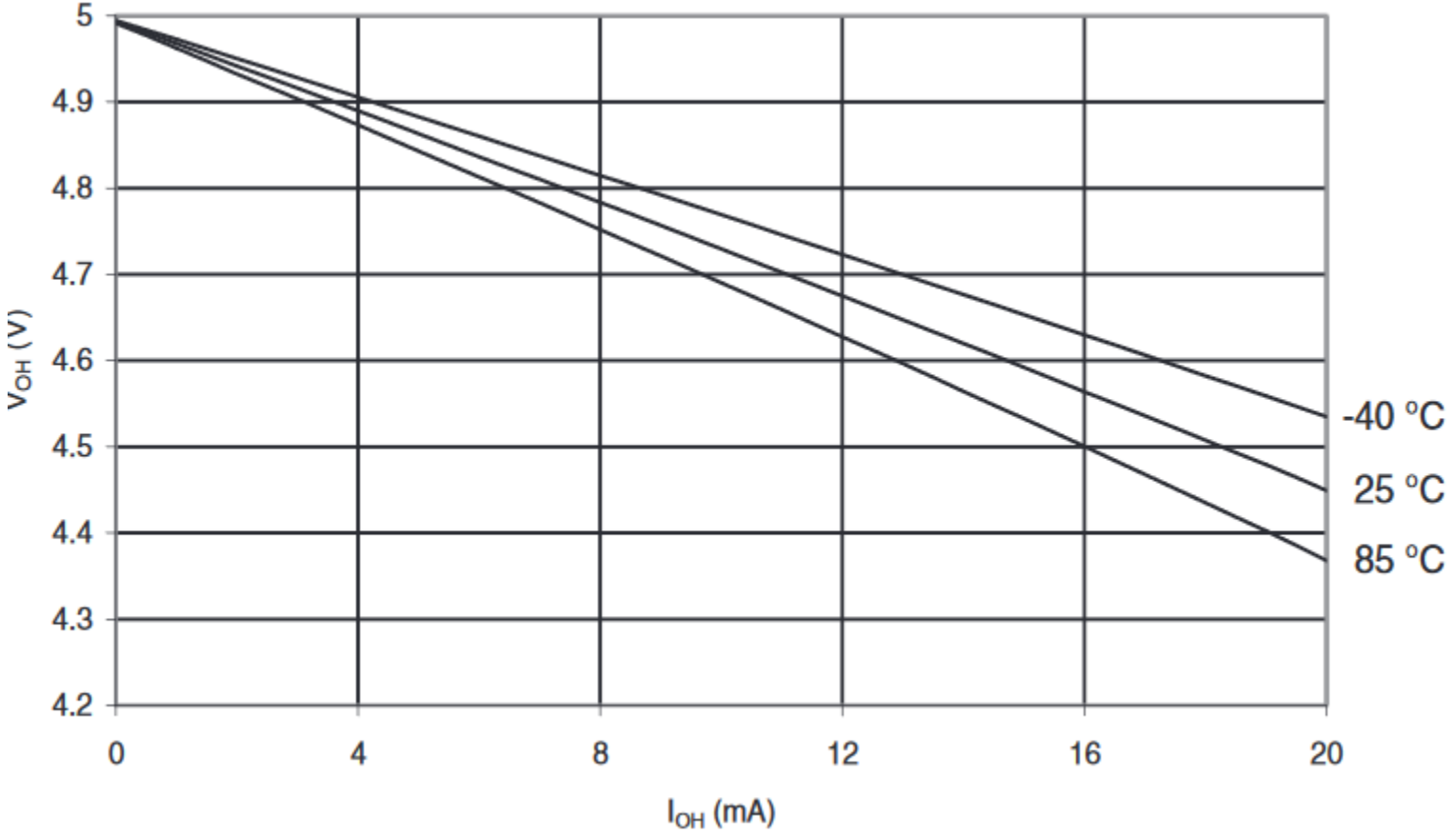
ain Ouvert » (si

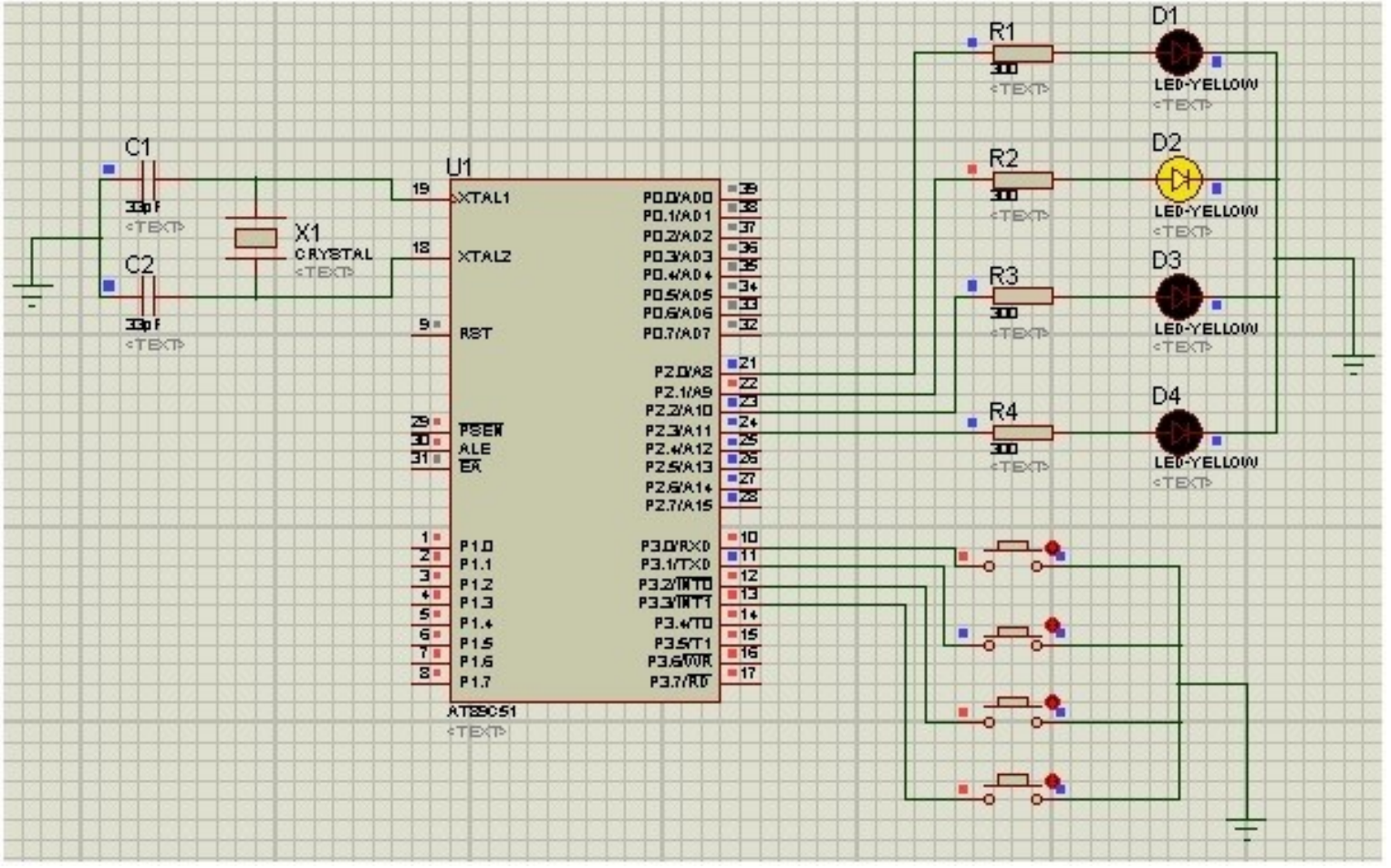


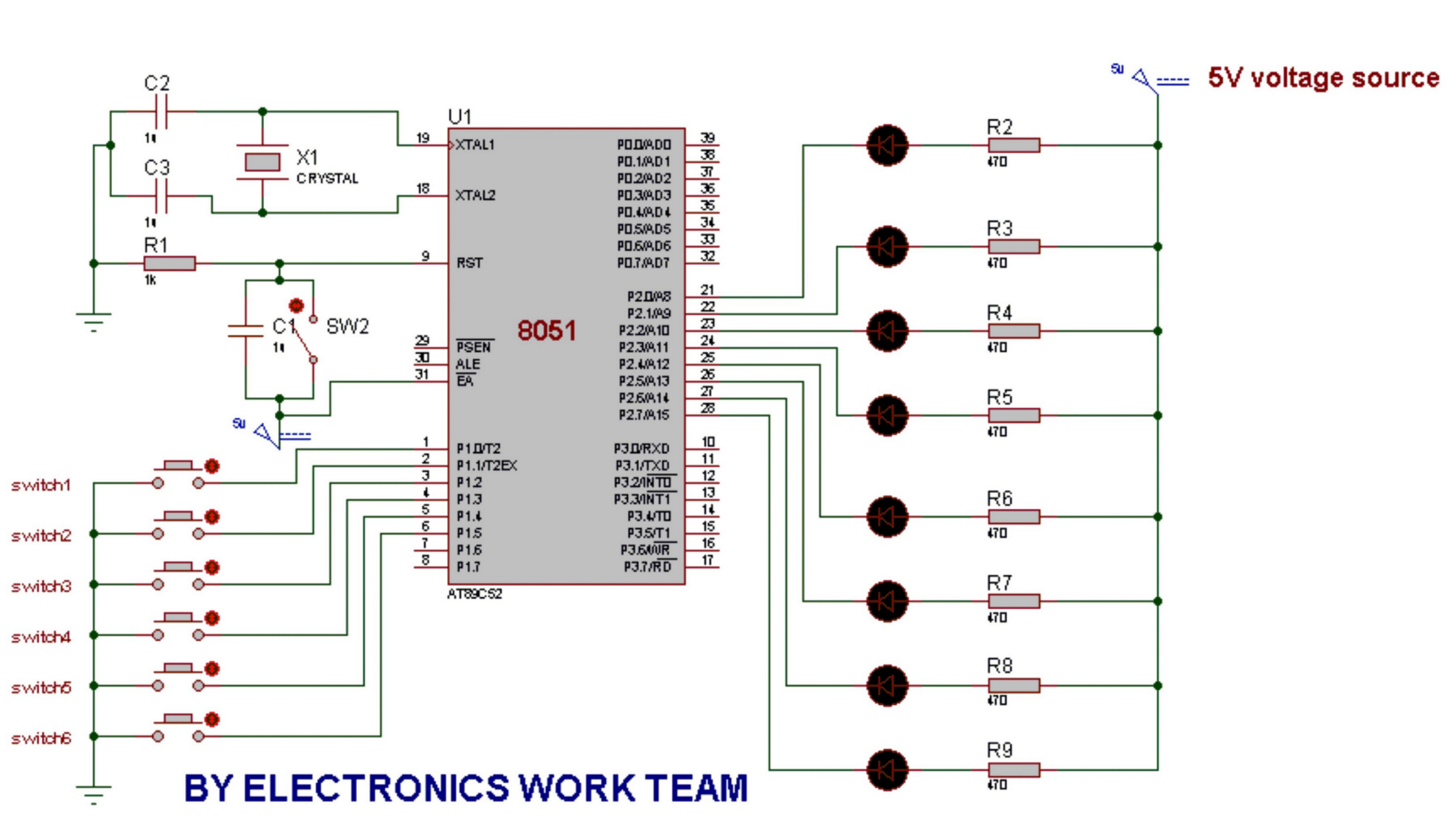
Les sorties

Symbol	Parameter	Condition		Min.	Typ.	Max.	Units
V_{OH}	Output High Voltage[3] except Reset pin	I_{OH} = -20mA, V_{CC} = 5V	T_A=85°C	4.2			
			T_A=105°C	4.1			
		I_{OH} = -10mA, V_{CC} = 3V	T_A=85°C	2.3			
			T_A=105°C	2.1			V

Les sorties

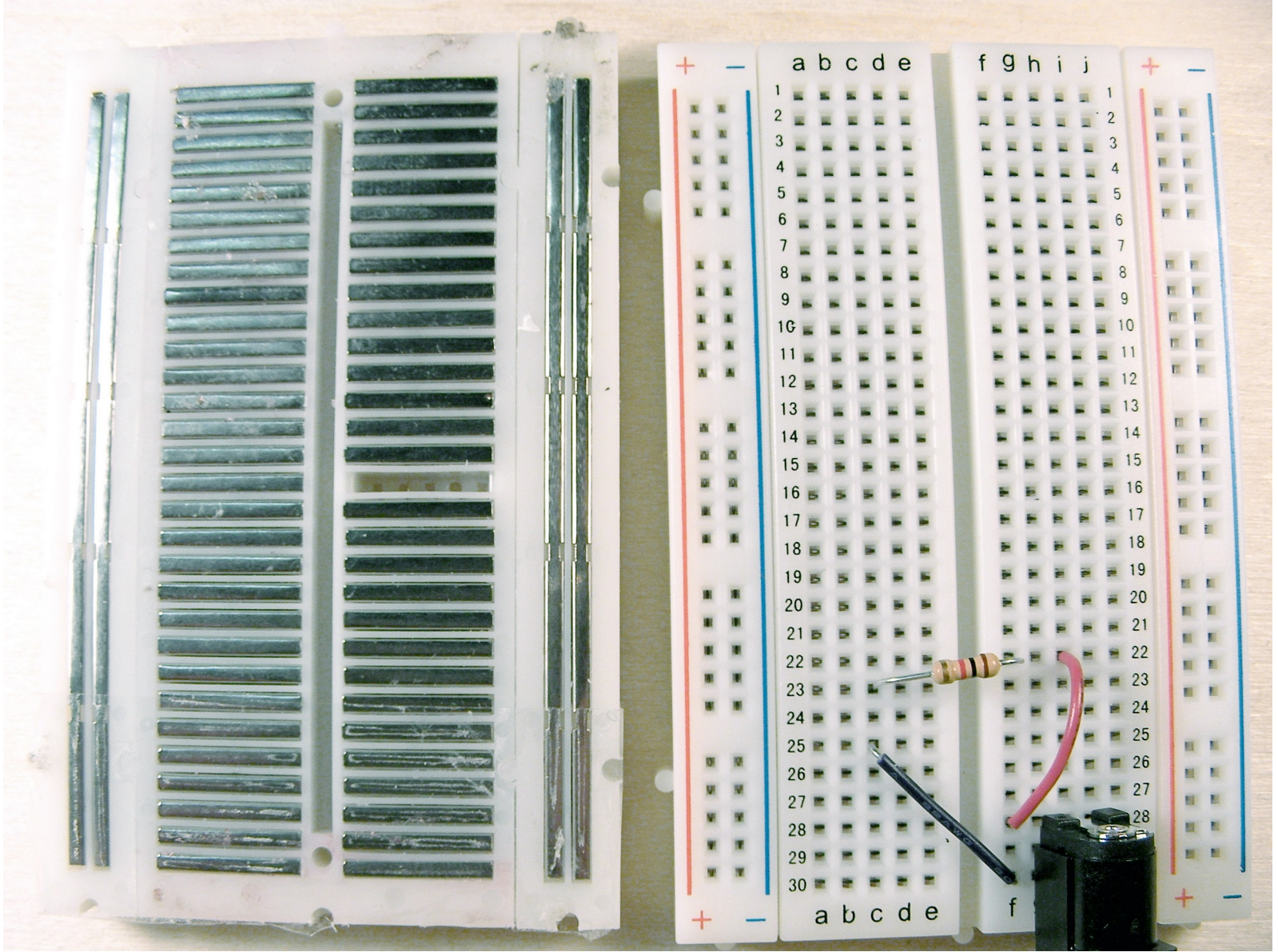


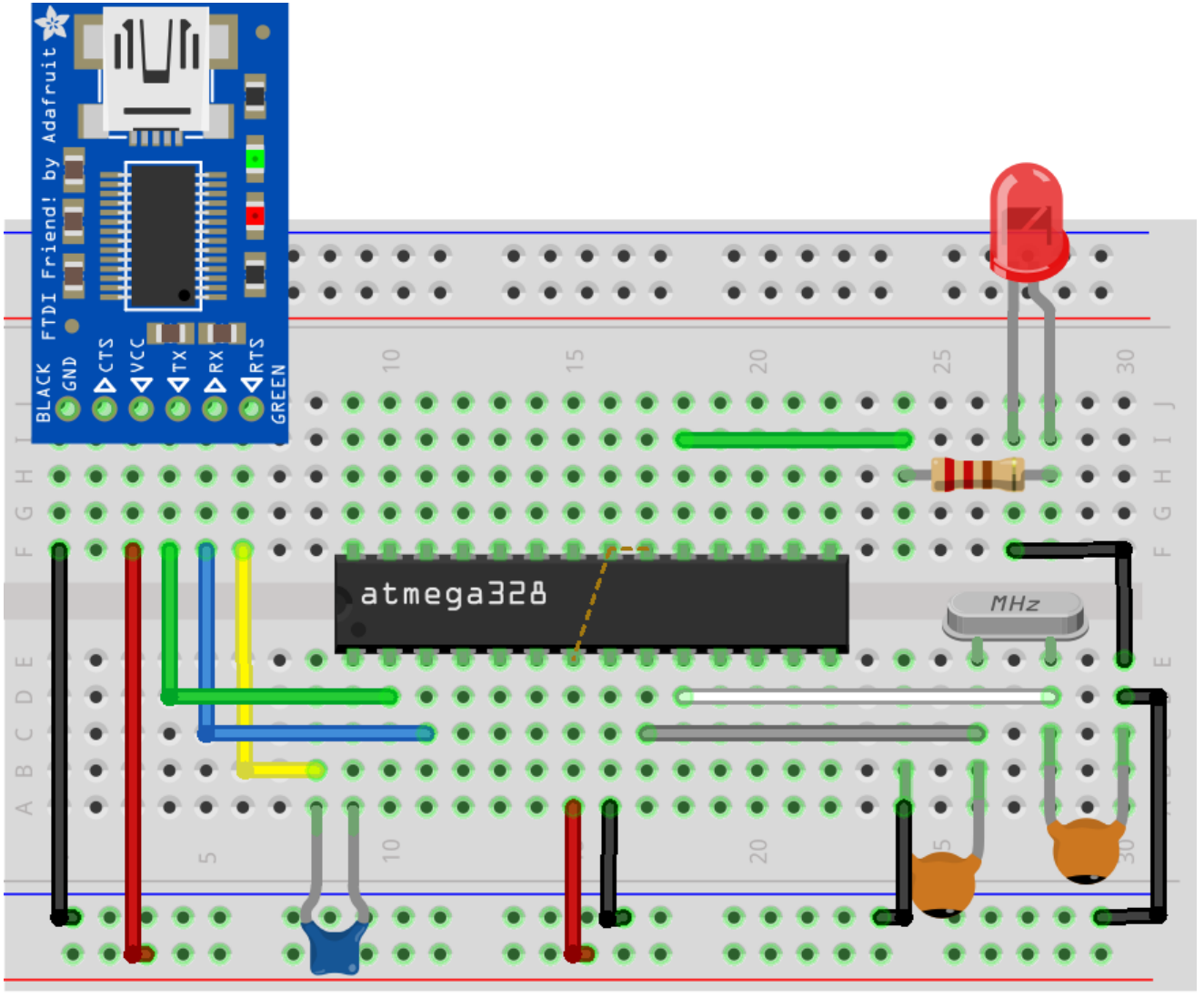

C1
33pF
<TEXT>
C2
33pF
<TEXT>
X1
CRYSTAL
<TEXT>
U1
XTAL1
XTAL2
RST
PSEN
ALE
EA
P1.0
P1.1
P1.2
P1.3
P1.4
P1.5
P1.6
P1.7
P0.0/AD0
P0.1/AD1
P0.2/AD2
P0.3/AD3
P0.4/AD4
P0.5/AD5
P0.6/AD6
P0.7/AD7
P2.0/A8
P2.1/A9
P2.2/A10
P2.3/A11
P2.4/A12
P2.5/A13
P2.6/A14
P2.7/A15
P3.0/RXD
P3.1/TXD
P3.2/INT0
P3.3/INT1
P3.4/T0
P3.5/T1
P3.6/WR
P3.7/RD
AT89C51
<TEXT>
R1
300
<TEXT>
R2
300
<TEXT>
R3
300
<TEXT>
R4
300
<TEXT>
D1
LED-YELLOW
<TEXT>
D2
LED-YELLOW
<TEXT>
D3
LED-YELLOW
<TEXT>
D4
LED-YELLOW
<TEXT>

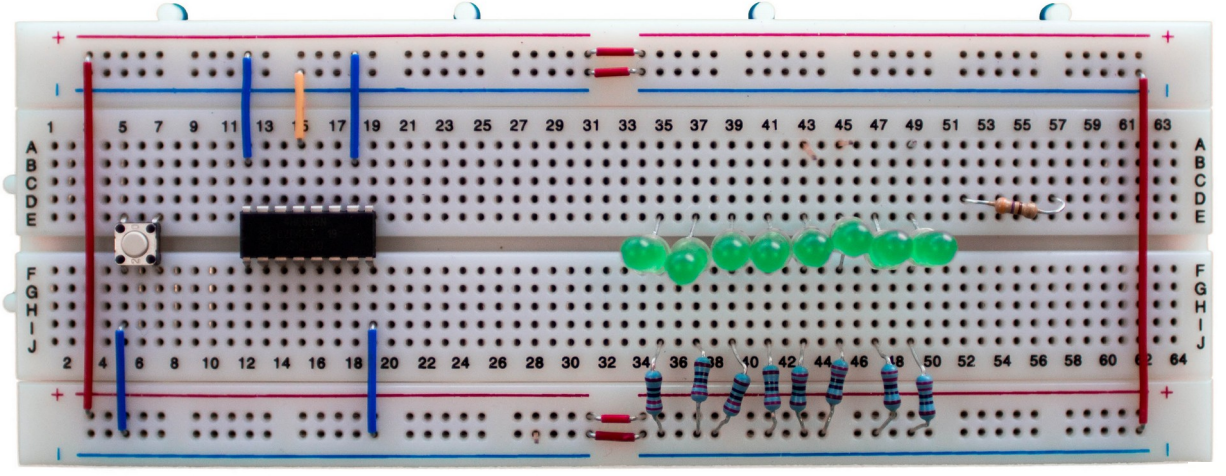

5V voltage source
C2
1u
C3
1u
X1
CRYSTAL
R1
1k
C1
1u
SW2
U1
XTAL1
XTAL2
RST
PSEN
ALE
EA
8051
AT89C52
P1.0/T2
P1.1/T2EX
P1.2
P1.3
P1.4
P1.5
P1.6
P1.7
P2.0/A8
P2.1/A9
P2.2/A10
P2.3/A11
P2.4/A12
P2.5/A13
P2.6/A14
P2.7/A15
P3.0/RXD
P3.1/TXD
P3.2/INT0
P3.3/INT1
P3.4/T0
P3.5/T1
P3.6/WR
P3.7/RD
switch1
switch2
switch3
switch4
switch5
switch6
R2
R3
R4
R5
R6
R7
R8
R9
470
BY ELECTRONICS WORK TEAM

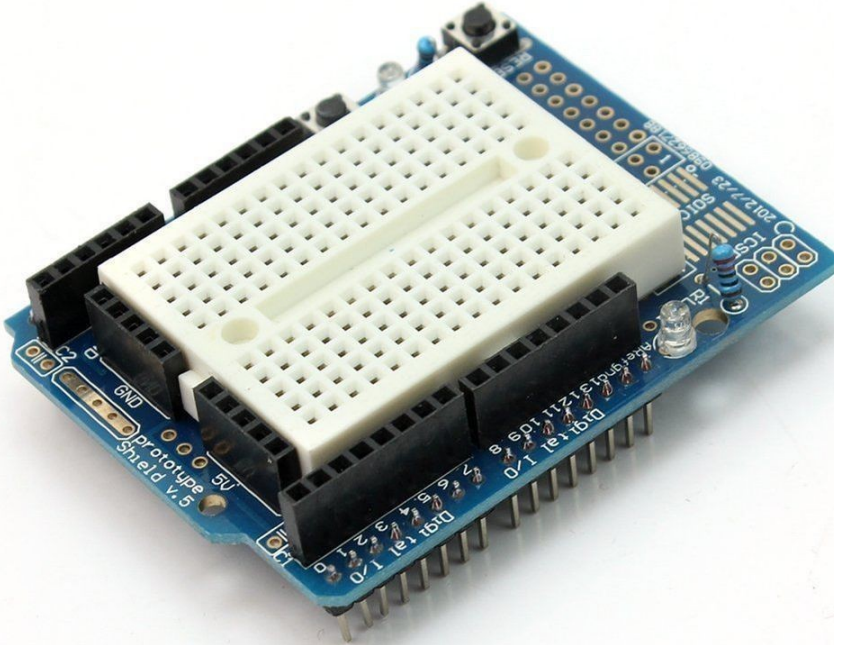
La plaque d’essais

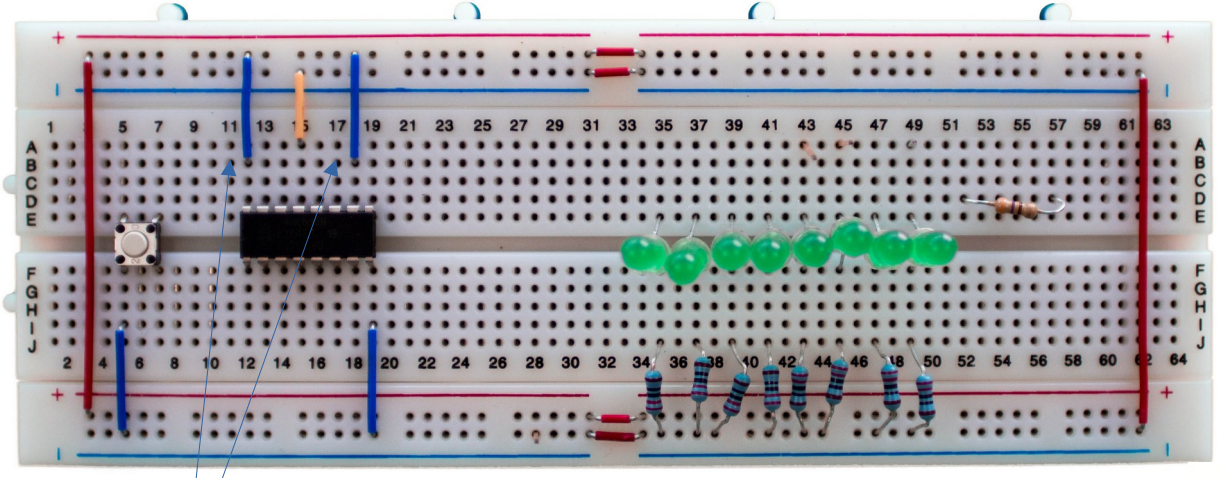
- Plaque Labdec
- Breadboard


a b c d e
f g h i j
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30
+ −

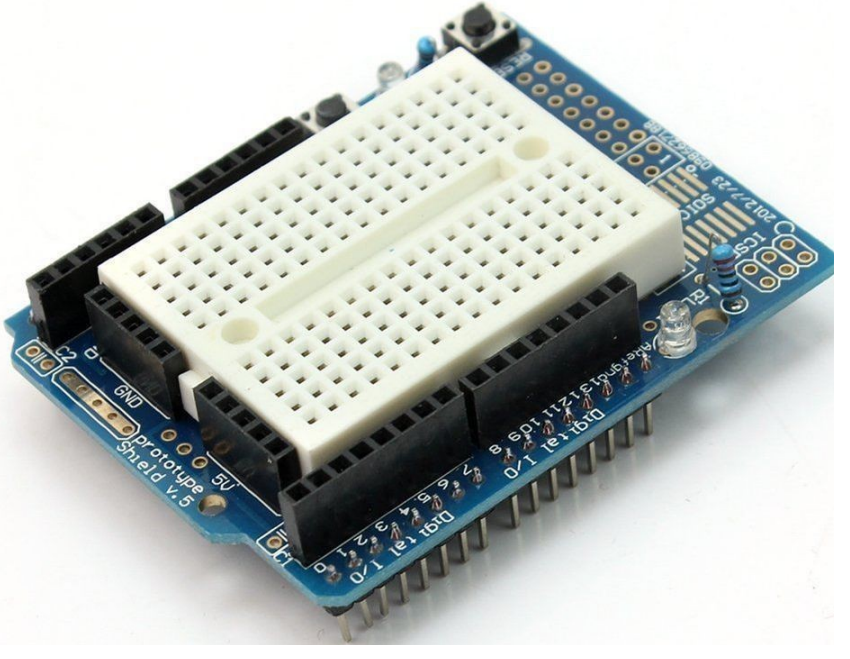

FTDI Friend! by Adafruit
BLACK
GND
CTS
VCC
TX
RX
RTS
GREEN
atmega328
MHz


1 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39 41 43 45 47 49 51 53 55 57 59 61 63
A B C D E
F G H I J
2 4 6 8 10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40 42 44 46 48 50 52 54 56 58 60 62 64


GND
prototype
Shield v.5
5V
Digital I/O
Digital I/O

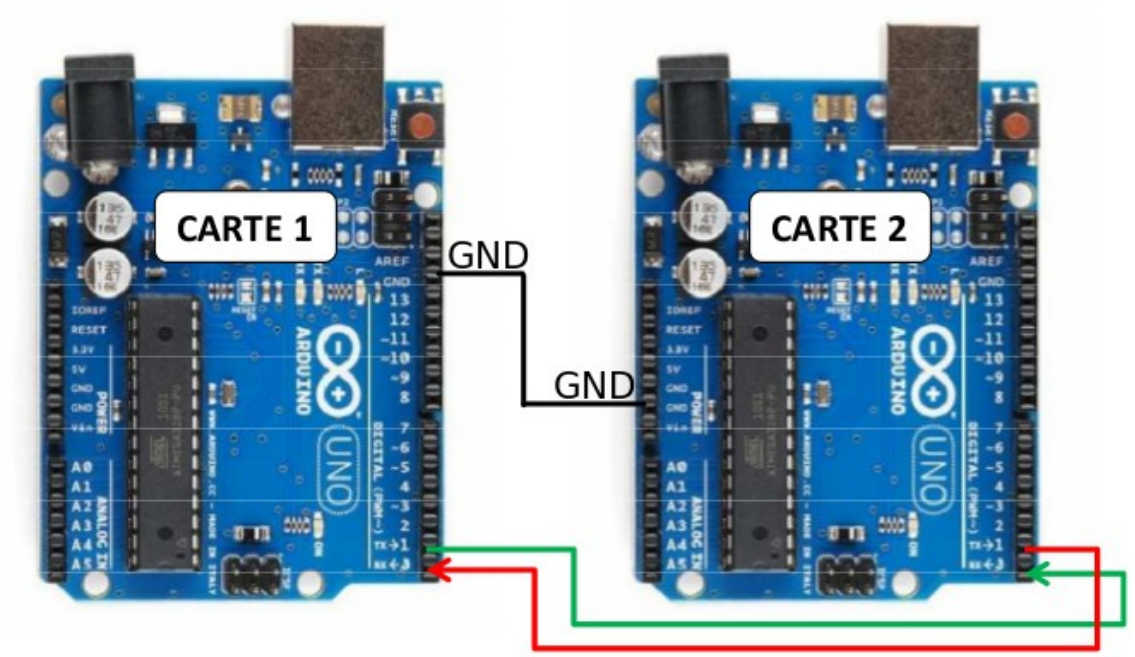


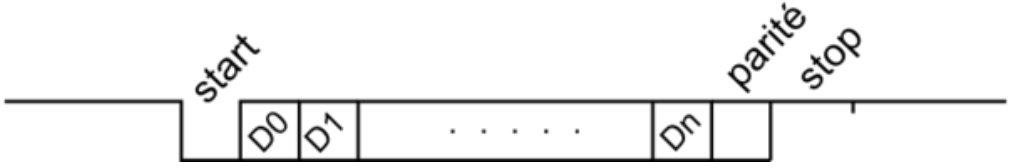
Sources d’erreurs
Difficultés de dépannage



La liaison série

Transmission de données





Liaison série

```
void setup() {
Serial.begin(9600); // opens serial port, sets data rate to 9600 bps
}
void loop() {
    Serial.println("Hello Computer")
Delay(1000);
}
```

Print(« message »);
Print(variable);

Functions

if(Serial)
available()
availableForWrite()
begin()
end()
find()
findUntil()
flush()
parseFloat()
parseInt()
peek()
print()
println()
read()
readBytes()
readBytesUntil()
readString()
readStringUntil()
setTimeout()
write()
serialEvent()

Serial.print()

Prints data to the serial port as human-readable ASCII text.
This command can take many forms.
Numbers are printed using an ASCII character for each digit.
Floats are similarly printed as ASCII digits, defaulting to two decimal places.
Bytes are sent as a single character. Characters and strings are sent as is.
For example:

Serial.print(78) gives "78"
Serial.print(1.23456) gives "1.23"
Serial.print('N') gives "N"
Serial.print("Hello world.") gives "Hello world."

An optional second parameter specifies the base (format) to use; permitted values are
BIN(binary, or base 2),
OCT(octal, or base 8),
DEC(decimal, or base 10),
HEX(hexadecimal, or base 16). For floating point numbers, this parameter specifies the number of decimal places to use. For example

Serial.print(78, BIN) gives "1001110"

Serial.print()

Note :
\n retour à la ligne suivante
\r CR
\t tab
\o caractère nul
\\ pour imprimer \

```
int val,i;
void setup()
{
Serial.begin(9600);
}

void loop()
{

    val = 45;
    Serial.println(val);  // ln si saut de ligne
    Serial.println(val,DEC);  // ln si saut de ligne
    Serial.print(val,HEX);
    Serial.println("texte");
    for (i=0;i<7;i++)
        {
        Serial.print(i,DEC);
        }
    Serial.print ("Diduino \\\tSimple \tRobot \r\n");
    Serial.print ("----------\r\n");
```

ASCII

Dec	Hex	Oct	Chr	Dec	Hex	Oct	HTML	Chr	Dec	Hex	Oct	HTML	Chr	Dec	Hex	Oct	HTML	Chr
0	0	000	NULL	32	20	040	&#032;	Space	64	40	100	&#064;	@	96	60	140	&#096;	`
1	1	001	SoH	33	21	041	&#033;	!	65	41	101	&#065;	A	97	61	141	&#097;	a
2	2	002	SoTxt	34	22	042	&#034;	"	66	42	102	&#066;	B	98	62	142	&#098;	b
3	3	003	EoTxt	35	23	043	&#035;	#	67	43	103	&#067;	C	99	63	143	&#099;	c
4	4	004	EoT	36	24	044	&#036;	$	68	44	104	&#068;	D	100	64	144	&#100;	d
5	5	005	Enq	37	25	045	&#037;	%	69	45	105	&#069;	E	101	65	145	&#101;	e
6	6	006	Ack	38	26	046	&#038;	&	70	46	106	&#070;	F	102	66	146	&#102;	f
7	7	007	Bell	39	27	047	&#039;	'	71	47	107	&#071;	G	103	67	147	&#103;	g
8	8	010	Bsp	40	28	050	&#040;	(	72	48	110	&#072;	H	104	68	150	&#104;	h
9	9	011	HTab	41	29	051	&#041;	)	73	49	111	&#073;	I	105	69	151	&#105;	i
10	A	012	LFeed	42	2A	052	&#042;	*	74	4A	112	&#074;	J	106	6A	152	&#106;	j
11	B	013	VTab	43	2B	053	&#043;	+	75	4B	113	&#075;	K	107	6B	153	&#107;	k
12	C	014	FFeed	44	2C	054	&#044;	,	76	4C	114	&#076;	L	108	6C	154	&#108;	l
13	D	015	CR	45	2D	055	&#045;	-	77	4D	115	&#077;	M	109	6D	155	&#109;	m
14	E	016	SOut	46	2E	056	&#046;	.	78	4E	116	&#078;	N	110	6E	156	&#110;	n
15	F	017	SIn	47	2F	057	&#047;	/	79	4F	117	&#079;	O	111	6F	157	&#111;	o
16	10	020	DLE	48	30	060	&#048;	0	80	50	120	&#080;	P	112	70	160	&#112;	p
17	11	021	DC1	49	31	061	&#049;	1	81	51	121	&#081;	Q	113	71	161	&#113;	q
18	12	022	DC2	50	32	062	&#050;	2	82	52	122	&#082;	R	114	72	162	&#114;	r
19	13	023	DC3	51	33	063	&#051;	3	83	53	123	&#083;	S	115	73	163	&#115;	s
20	14	024	DC4	52	34	064	&#052;	4	84	54	124	&#084;	T	116	74	164	&#116;	t
21	15	025	NAck	53	35	065	&#053;	5	85	55	125	&#085;	U	117	75	165	&#117;	u
22	16	026	Syn	54	36	066	&#054;	6	86	56	126	&#086;	V	118	76	166	&#118;	v
23	17	027	EoTB	55	37	067	&#055;	7	87	57	127	&#087;	W	119	77	167	&#119;	w
24	18	030	Can	56	38	070	&#056;	8	88	58	130	&#088;	X	120	78	170	&#120;	x
25	19	031	EoM	57	39	071	&#057;	9	89	59	131	&#089;	Y	121	79	171	&#121;	y
26	1A	032	Sub	58	3A	072	&#058;	:	90	5A	132	&#090;	Z	122	7A	172	&#122;	z
27	1B	033	Esc	59	3B	073	&#059;	;	91	5B	133	&#091;	[	123	7B	173	&#123;	{
28	1C	034	FSep	60	3C	074	&#060;	<	92	5C	134	&#092;	\	124	7C	174	&#124;	|
29	1D	035	GSep	61	3D	075	&#061;	=	93	5D	135	&#093;	]	125	7D	175	&#125;	}
30	1E	036	RSep	62	3E	076	&#062;	>	94	5E	136	&#094;	^	126	7E	176	&#126;	~
31	1F	037	USep	63	3F	077	&#063;	?	95	5F	137	&#095;	_	127	7F	177	&#127;	Del

charstable.com

Serial.write()

Writes binary data to the serial port. This data is sent as a byte or series of bytes; to send the characters representing the digits of a number use the print() function instead.

Syntax

Serial.write(val)
Serial.write(str)
Serial.write(buf, len)

Parameters

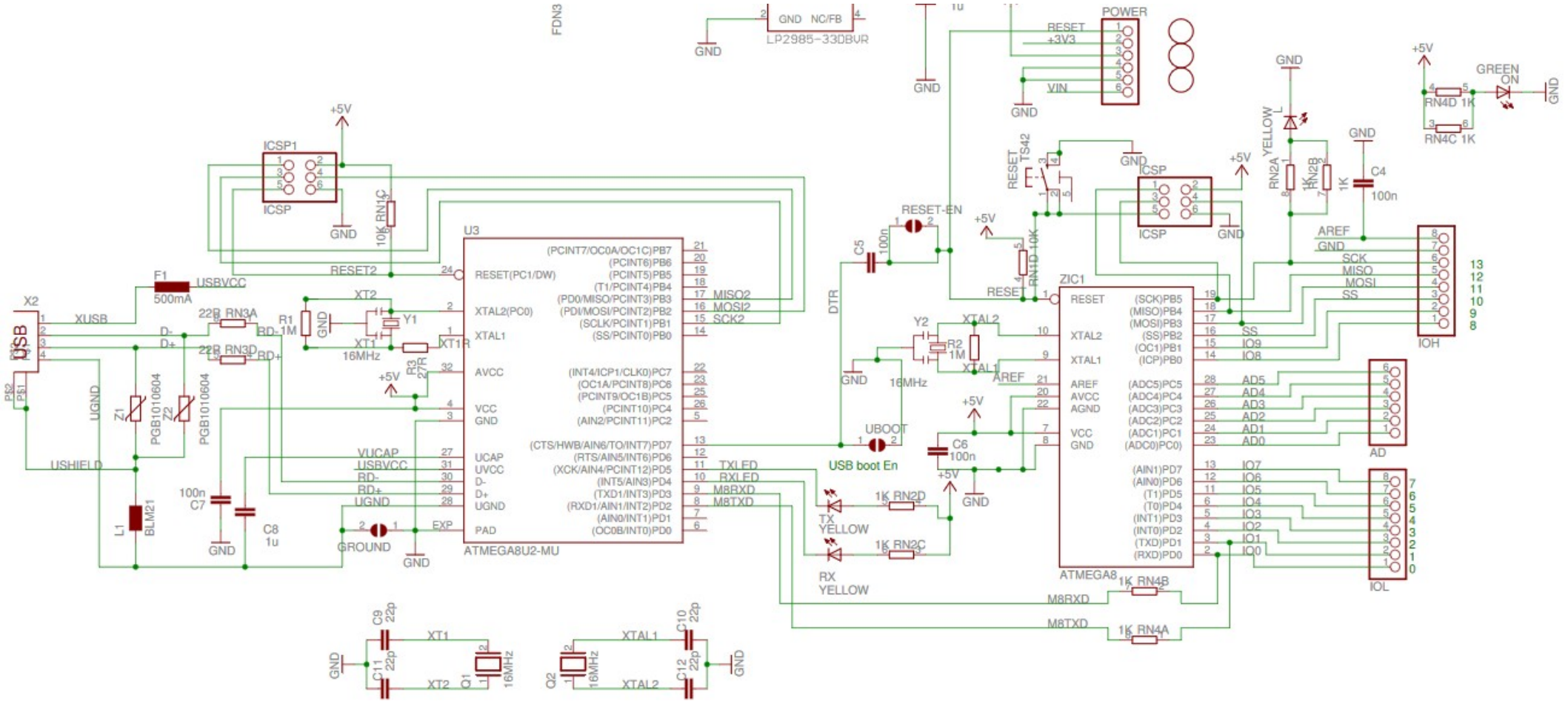
Serial: serial port object. val: a value to send as a single byte.
str: a string to send as a series of bytes.
buf: an array to send as a series of bytes.
len: the number of bytes to be sent from the arra

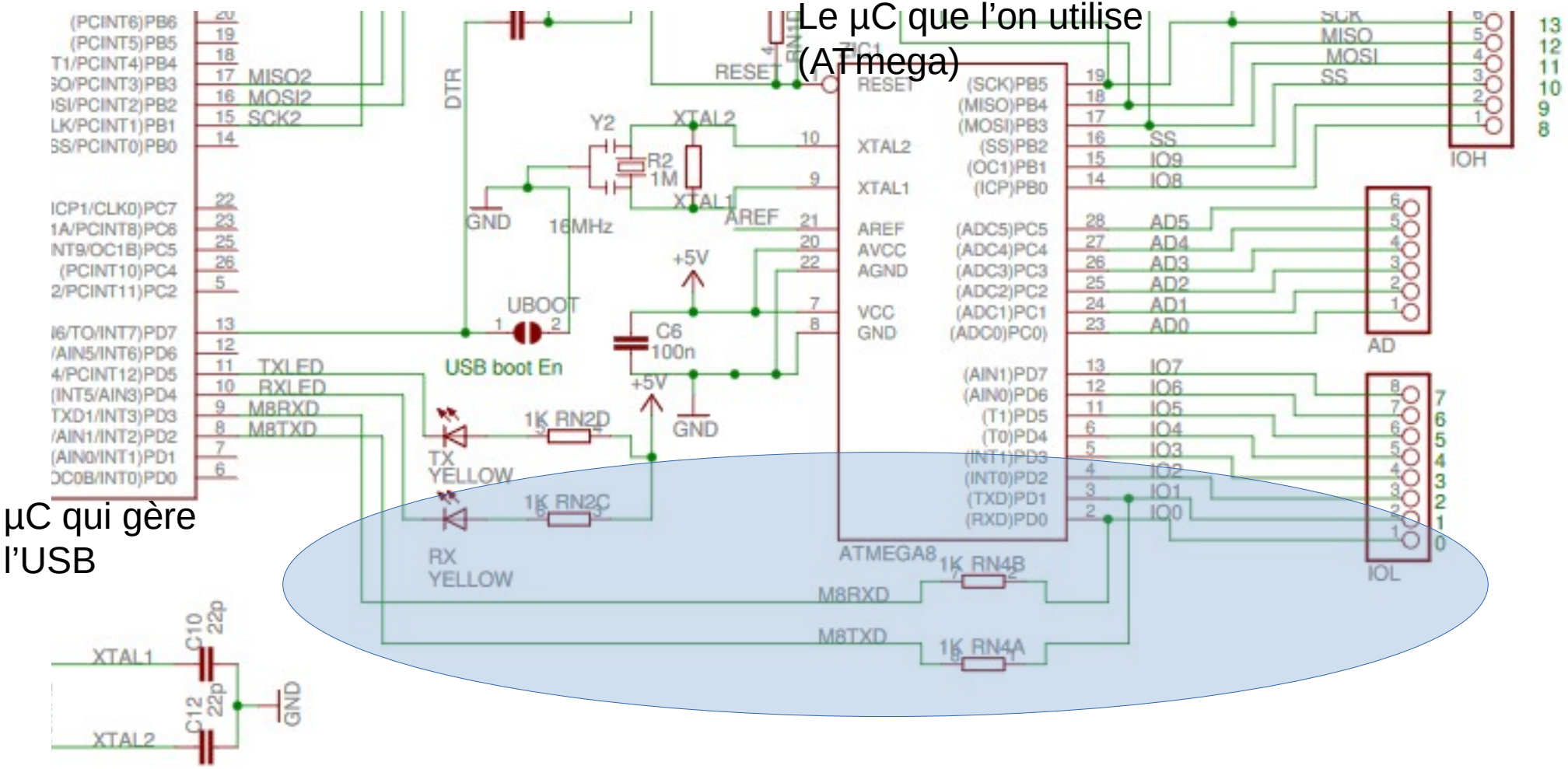
Serial.write()

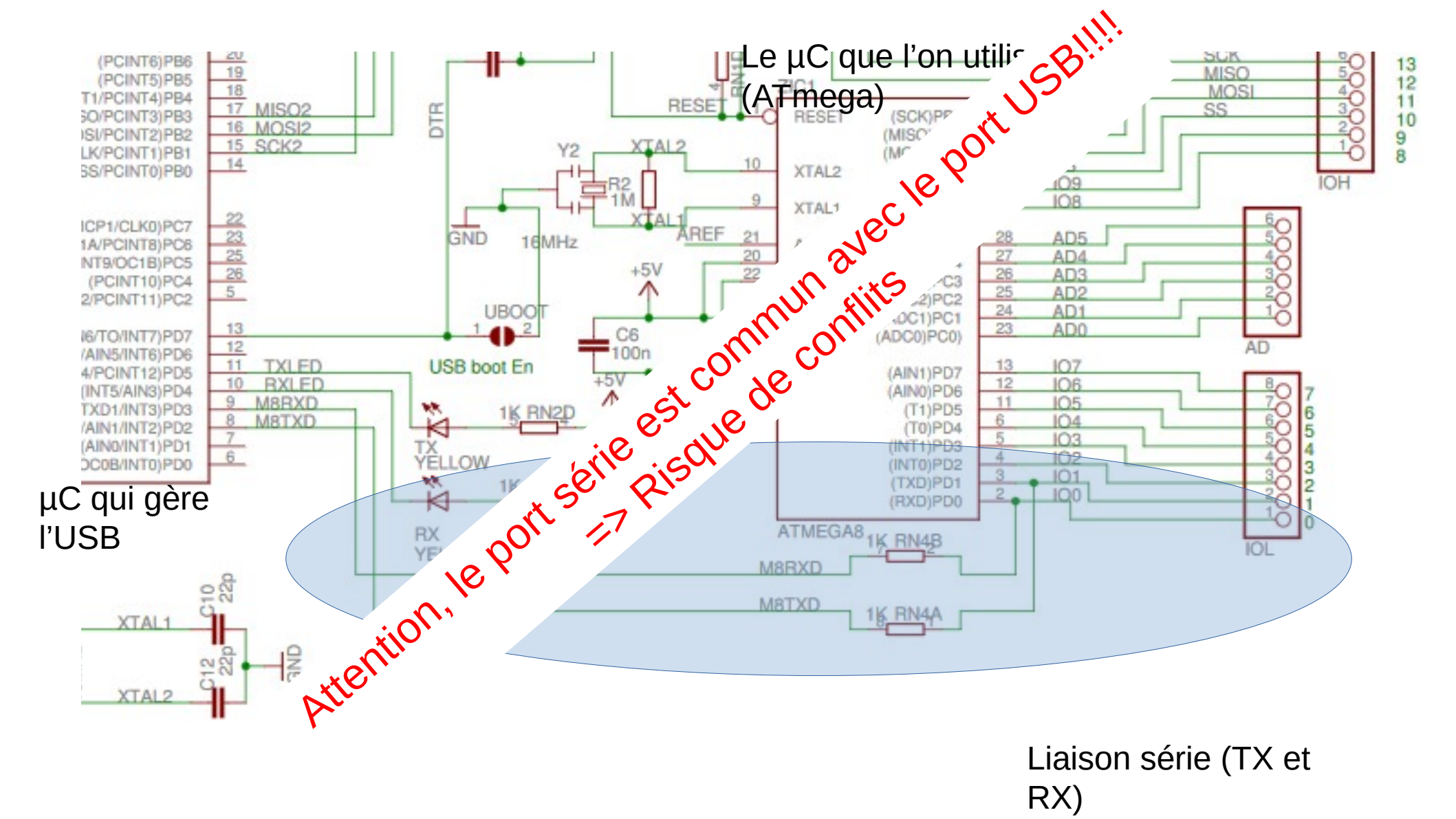
```
void setup(){
  Serial.begin(9600);
}

void loop(){
  Serial.write(45); // send a byte with the value 45 (ASCII)

  int bytesSent = Serial.write("hello"); //send the string "hello" and return the length of the string.

  Serial.println(bytesSent);

  Serial.println("hello"); //send the string "hello"

}
```

Le port série !!

Attention, le port série est commun avec le port USB!!!!


FDN3
GND NC/FB
LP2985-33DBVR
POWER
RESET
+3V3
VIN
ICSP1
ICSP
RESET2
10K RN1C
F1
USBVCC
500mA
X2
USB
XUSB
D-
D+
UGND
USHIELD
22R RN3A
22R RN3D
R1
1M
XT2
XT1
16MHz
Y1
XT1R
R3
27R
Z1
Z2
PGB1010604
L1
BLM21
100n
C7
C8
1u
GROUND
U3
ATMEGA8U2-MU
RESET(PC1/DW)
XTAL2(PC0)
XTAL1
AVCC
VCC
GND
UCAP
UVCC
D-
D+
UGND
PAD
VUCAP
USBVCC
RD-
RD+
EXP
(PCINT7/OC0A/OC1C)PB7
(PCINT6)PB6
(PCINT5)PB5
(T1/PCINT4)PB4
(PDO/MISO/PCINT3)PB3
(PDI/MOSI/PCINT2)PB2
(SCLK/PCINT1)PB1
(SS/PCINT0)PB0
(INT4/ICP1/CLK0)PC7
(OC1A/PCINT8)PC6
(PCINT9/OC1B)PC5
(PCINT10)PC4
(AIN2/PCINT11)PC2
(CTS/HWB/AIN6/TO/INT7)PD7
(RTS/AIN5/INT6)PD6
(XCK/AIN4/PCINT12)PD5
(INT5/AIN3)PD4
(TXD1/INT3)PD3
(RXD1/AIN1/INT2)PD2
(AIN0/INT1)PD1
(OC0B/INT0)PD0
MISO2
MOSI2
SCK2
TXLED
RXLED
M8RXD
M8TXD
C9 22p
C10 22p
C11 22p
C12 22p
XT1
XT2
Q1 16MHz
Q2 16MHz
XTAL1
XTAL2
C5 100n
RESET-EN
DTR
UBOOT
USB boot En
TX YELLOW
RX YELLOW
1K RN2D
1K RN2C
Y2
R2 1M
16MHz
C6 100n
ZIC1
ATMEGA8
AREF
AVCC
AGND
VCC
GND
(SCK)PB5
(MISO)PB4
(MOSI)PB3
(SS)PB2
(OC1)PB1
(ICP)PB0
(ADC5)PC5
(ADC4)PC4
(ADC3)PC3
(ADC2)PC2
(ADC1)PC1
(ADC0)PC0
(AIN1)PD7
(AIN0)PD6
(T1)PD5
(T0)PD4
(INT1)PD3
(INT0)PD2
(TXD)PD1
(RXD)PD0
1K RN4B
1K RN4A
TS42
RESET
YELLOW
RN2A 1K
RN2B 1K
C4 100n
GREEN ON
RN4D 1K
RN4C 1K
AREF
SCK
MISO
MOSI
SS
IOH
IOL
AD
IO9
IO8
IO7
IO6
IO5
IO4
IO3
IO2
IO1
IO0
AD5
AD4
AD3
AD2
AD1
AD0

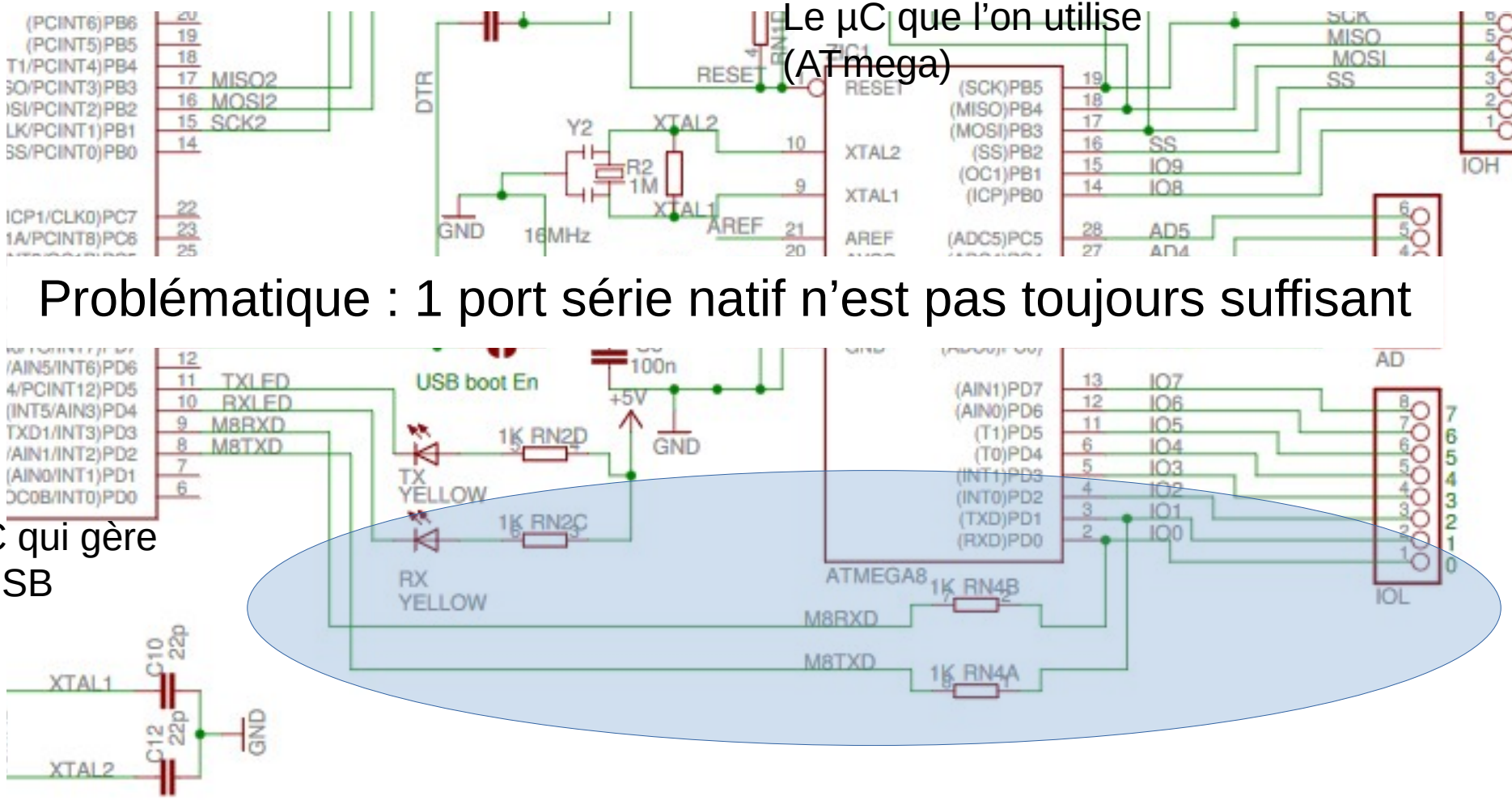

Le µC que l'on utilise (ATmega)
µC qui gère l'USB
Liaison série (TX et RX)
ATMEGA8
RESET
XTAL2
XTAL1
AREF
AVCC
AGND
VCC
GND
Y2
R2
1M
16MHz
C6
100n
+5V
UBOOT
USB boot En
DTR
TXLED
RXLED
M8RXD
M8TXD
TX YELLOW
RX YELLOW
1K RN2D
1K RN2C
1K RN4B
1K RN4A
C10 22p
C12 22p
MISO2
MOSI2
SCK2
SCK
MISO
MOSI
SS
IOH
IOL
AD
AD5
AD4
AD3
AD2
AD1
AD0
IO9
IO8
IO7
IO6
IO5
IO4
IO3
IO2
IO1
IO0


Le µC que l'on utilise (ATmega)
µC qui gère l'USB
Attention, le port série est commun avec le port USB!!!!
=> Risque de conflits
Liaison série (TX et RX)
ATMEGA8
RESET
XTAL1
XTAL2
AREF
GND
16MHz
Y2
R2
1M
C6
100n
C10
22p
C12
22p
UBOOT
USB boot En
DTR
+5V
TX YELLOW
RX
TXLED
RXLED
M8RXD
M8TXD
1K RN2D
1K RN4B
1K RN4A
MISO2
MOSI2
SCK2
AD5
AD4
AD3
AD2
AD1
AD0
IO7
IO6
IO5
IO4
IO3
IO2
IO1
IO0
IOH
IOL
AD
MISO
MOSI
SS

Le µC que l'on utilise (ATmega)

Problématique : 1 port série natif n'est pas toujours suffisant

µC qui gère l'USB



Liaison série (TX et RX)

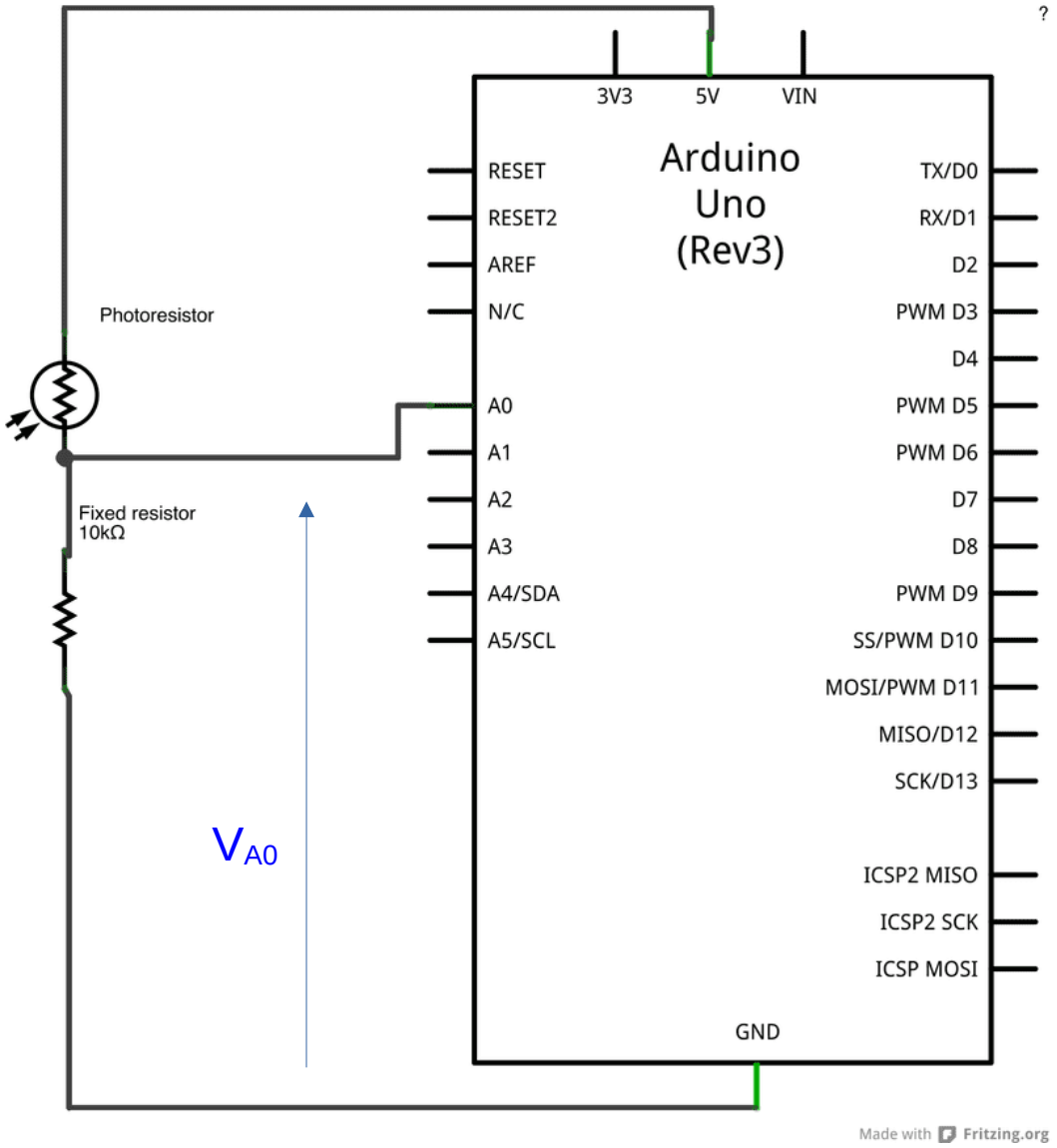
Librairie SoftwareSerial

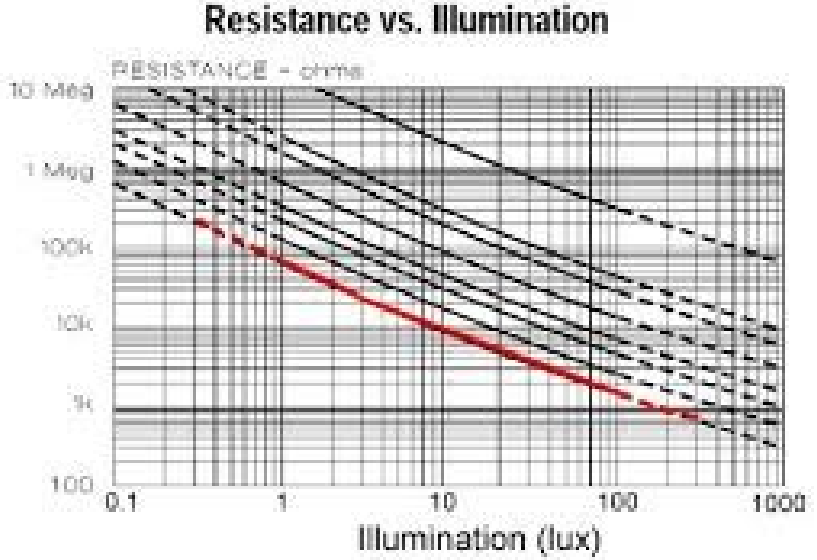
SoftwareSerial is used to create an instance of a SoftwareSerial object, whose name you need to provide as in the example below. The inverse_logic argument is optional and defaults to false. See below for more details about what it does. Multiple SoftwareSerial objects may be created, however only one can be active at a given moment.

```
#include <SoftwareSerial.h>

const byte rxPin = 2;
const byte txPin = 3;

// set up a new serial object
SoftwareSerial mySerial (rxPin, txPin);
```

```
#include <SoftwareSerial.h>

SoftwareSerial mySerial(2, 3); // RX, TX

void setup()
{
  // Open serial communications and wait for port to open:
  Serial.begin(115200);
  while (!Serial) {
    ; // wait for serial port to connect. Needed for Native USB only
  }


  Serial.println("Goodnight moon!");

  // set the data rate for the SoftwareSerial port
  mySerial.begin(38400);
  mySerial.println("Hello, world?");
}


void loop() // run over and over
{
  if (mySerial.available())
    Serial.write(mySerial.read());
  if (Serial.available())
    mySerial.write(Serial.read());
}
```

Le convertisseur Analogique → Numérique

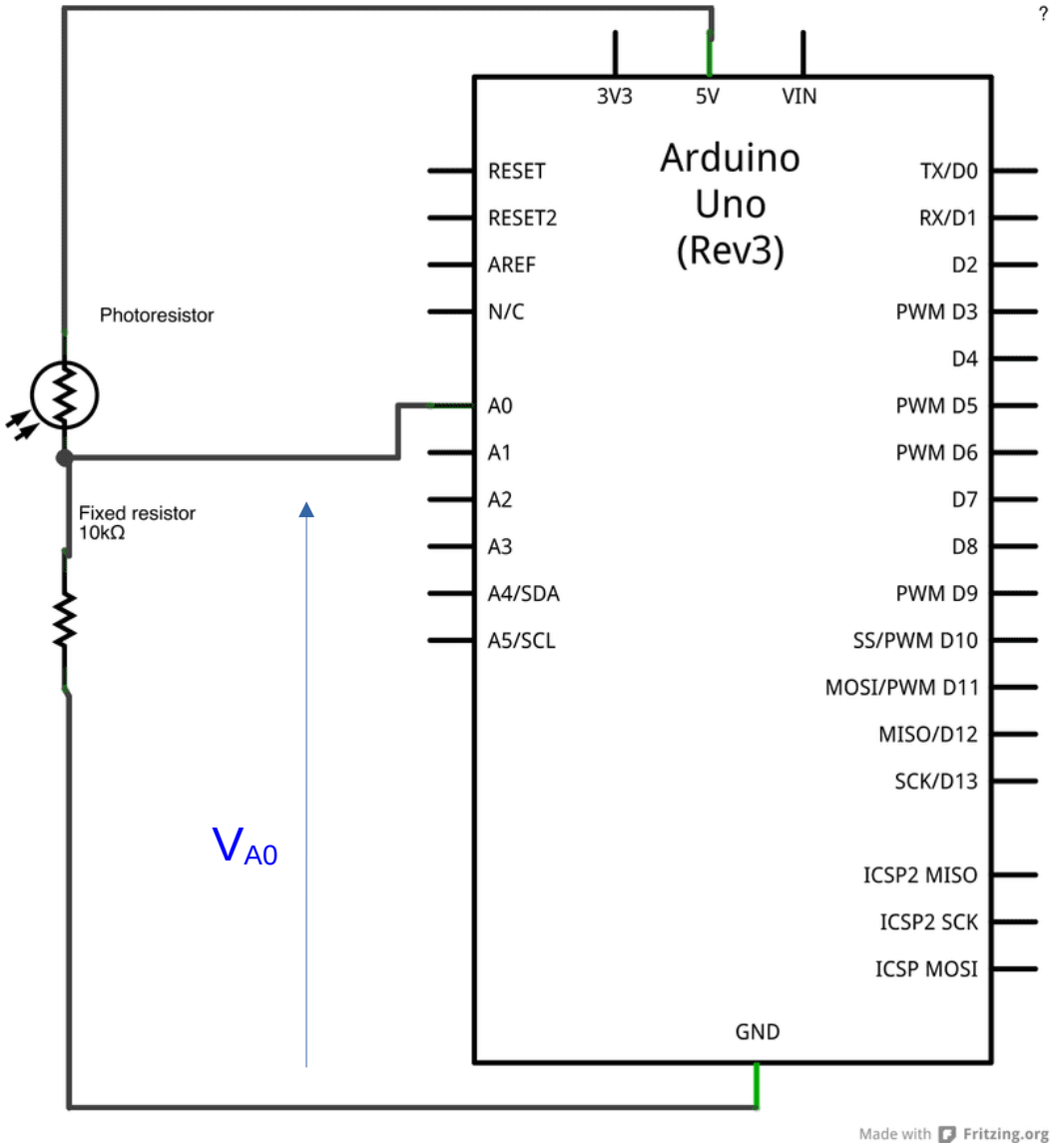


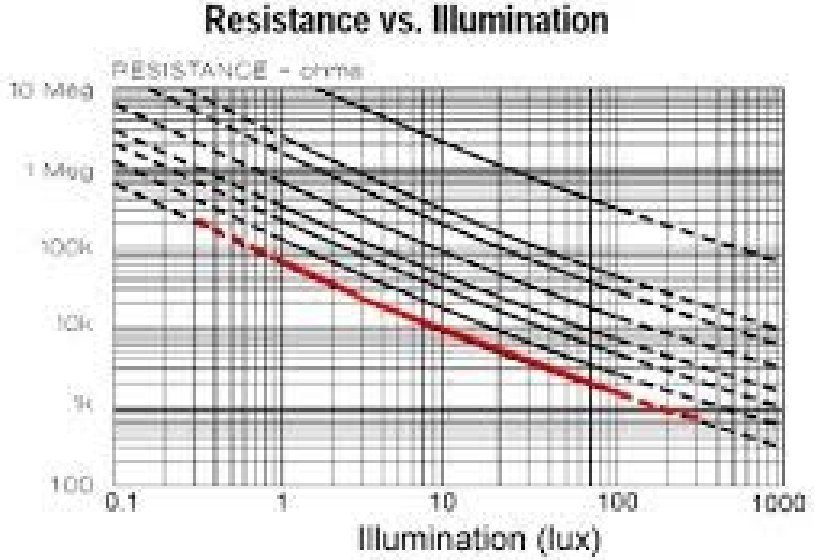


$$V_{A0} = \frac{R_{10k}}{R_{10k} + R_{Photoresistor}} x\, 5\,V$$

Éclairement = 100 lux → 2kΩ

Éclairement = 1 lux → 100kΩ

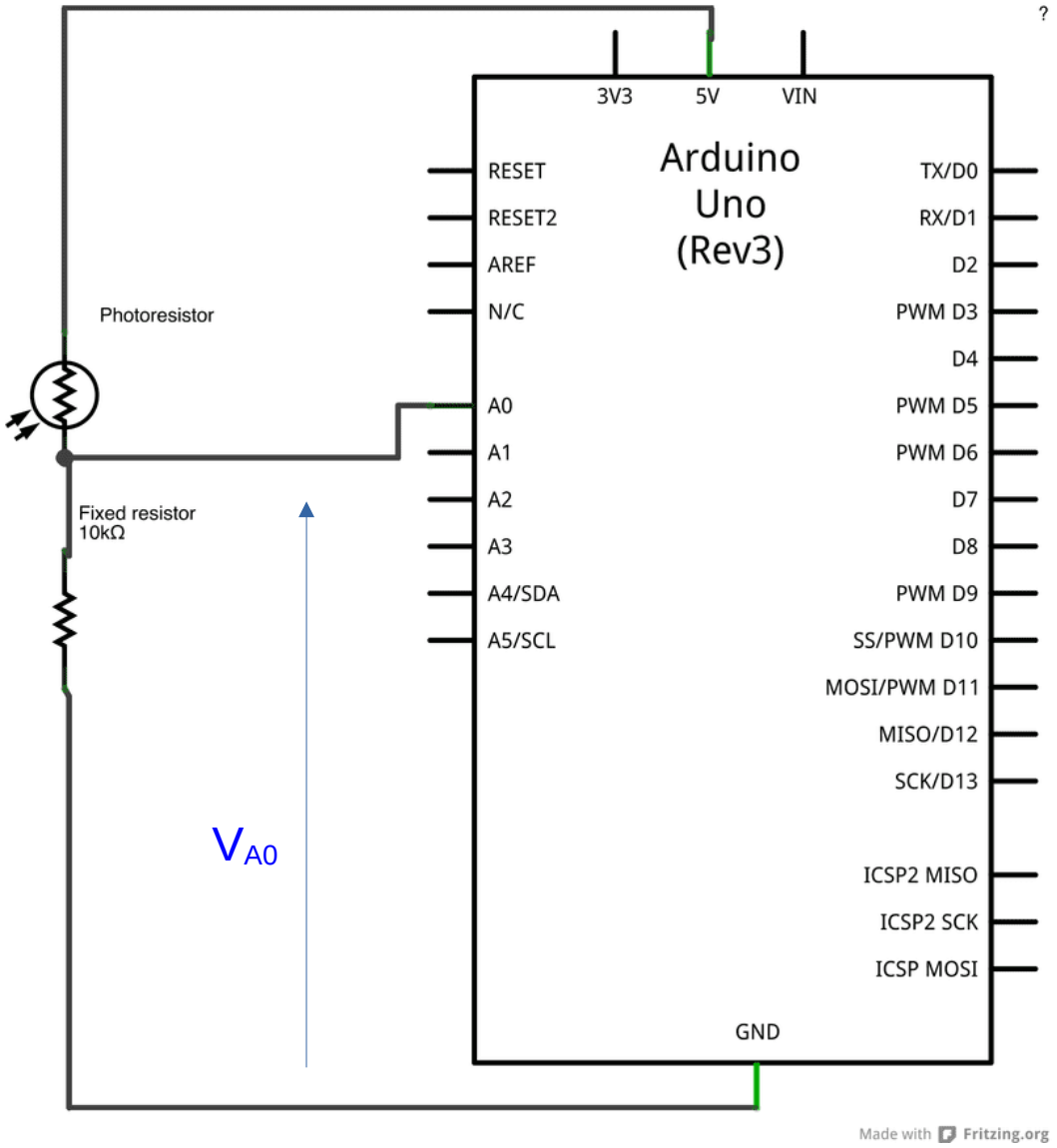




$$V_{A0} = \frac{R_{10k}}{R_{10k} + R_{Photoresistor}} x\, 5\,V$$

Éclairement = 100 lux → 2kΩ → **4,2V**

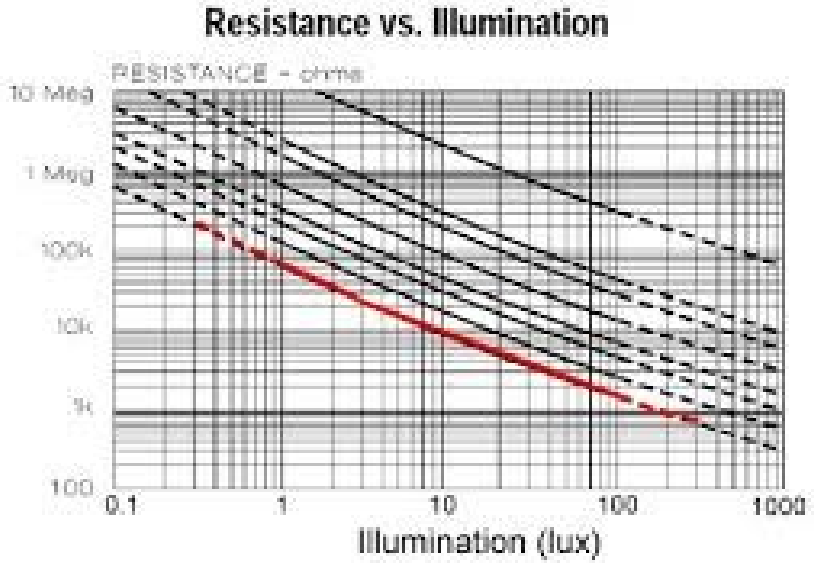
Éclairement = 1 lux → 100kΩ → **0,45V**



Principe du CAN

Si V_{Ax} = 5V → 1023 (2^{10}-1)

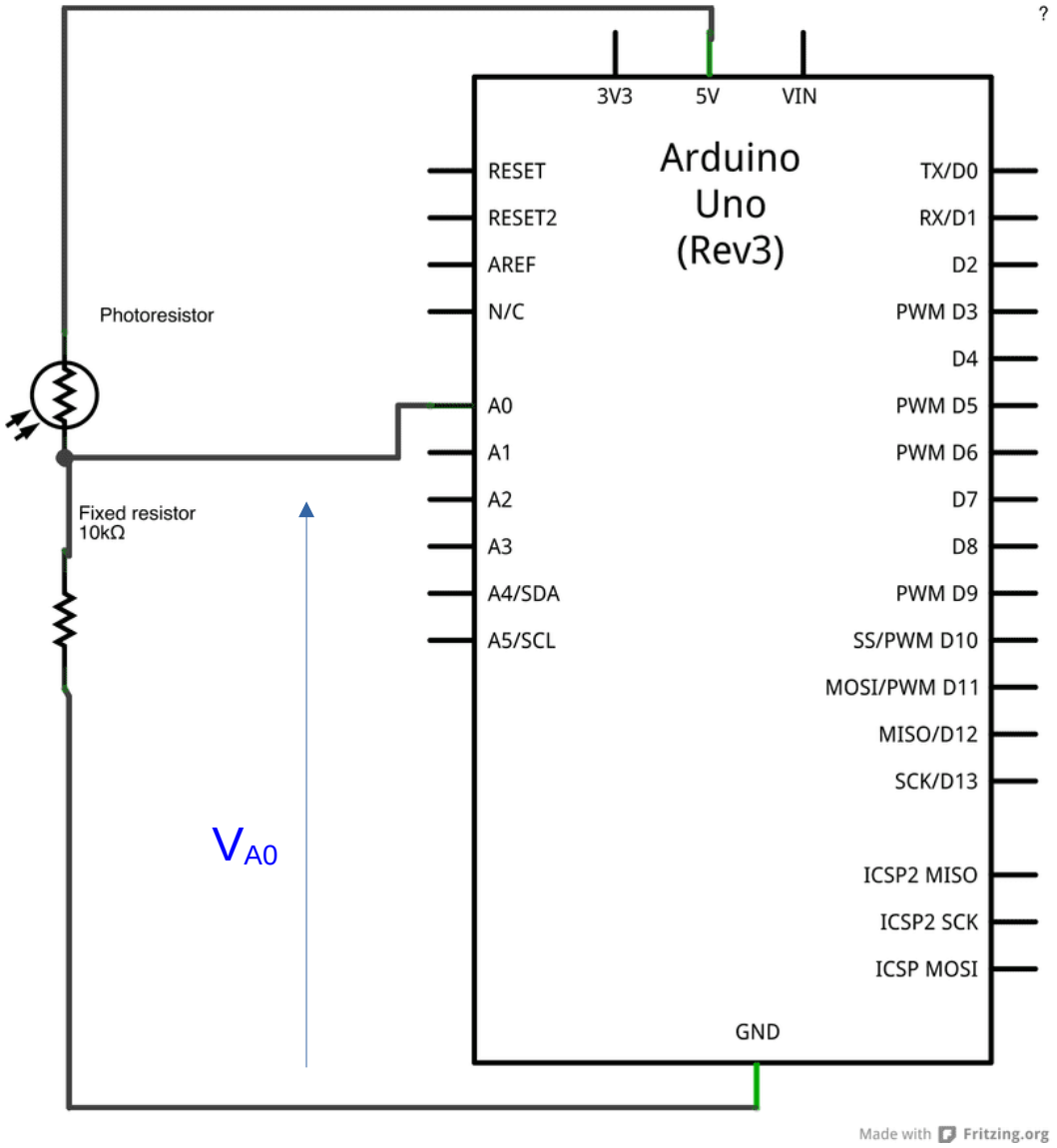
Si V_{Ax} = 0V → 0



$$V_{A0} = \frac{R_{10k}}{R_{10k} + R_{Photoresistor}} x\, 5\,V$$

Éclairement = 100 lux → 2kΩ → **4,2V**

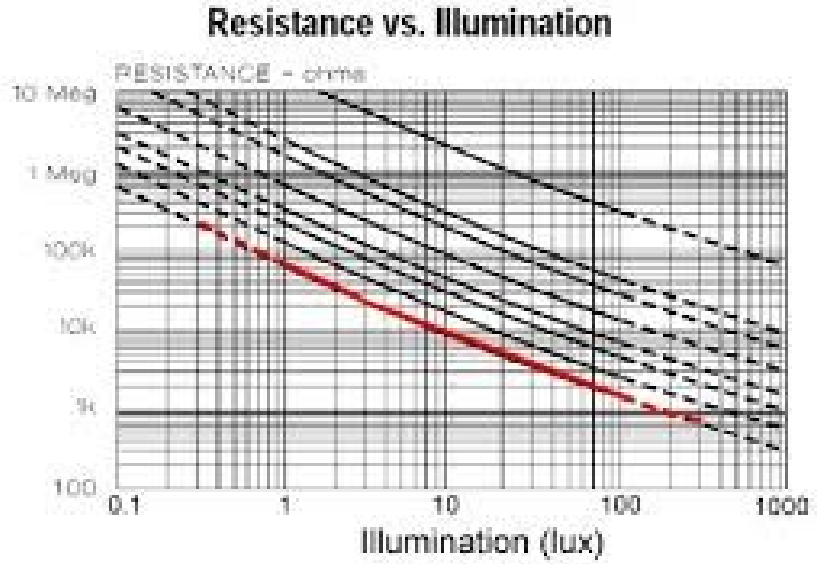
Éclairement = 1 lux → 100kΩ → **0,45V**



Principe du CAN

Si V_{Ax} = 5V → 1023 (2^{10}-1)

Si V_{Ax} = 0V → 0



92,07

859,32

$$V_{A0} = \frac{R_{10k}}{R_{10k} + R_{Photoresistor}} x\, 5\,V$$

Éclairement = 100 lux → 2kΩ → **4,2V**

Éclairement = 1 lux → 100kΩ → **0,45V**

Le CAN (ADC): *analogRead()*

Reads the value from the specified analog pin. Arduino boards contain a multichannel, 10-bit analog to digital converter. This means that it will map input voltages between 0 and the operating voltage(5V or 3.3V) into integer values between 0 and 1023. On an Arduino UNO, for example, this yields a resolution between readings of: 5 volts / 1024 units or, 0.0049 volts (4.9 mV) per unit.

Lien

The code reads the voltage on analogPin and displays it.

```
int analogPin = A3; // potentiometer wiper (middle terminal) connected to analog pin 3
                    // outside leads to ground and +5V
int val = 0;  // variable to store the value read

void setup() {
  Serial.begin(9600);           //  setup serial
}

void loop() {
  val = analogRead(analogPin);  // read the input pin
  Serial.println(val);          // debug value
}
```

Le CAN (ADC): *analogReference()*

Configures the reference voltage used for analog input (i.e. the value used as the top of the input range). The options are:

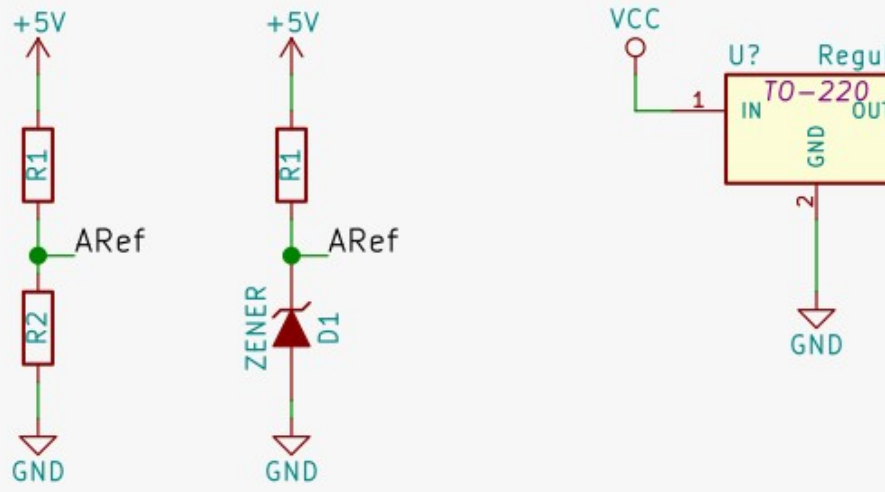
Arduino AVR Boards (Uno, Mega, Leonardo, etc.)

- DEFAULT: the default analog reference of 5 volts (on 5V Arduino boards) or 3.3 volts (on 3.3V Arduino boards)
- INTERNAL: a built-in reference, equal to 1.1 volts on the ATmega168 or ATmega328P and 2.56 volts on the ATmega32U4 and ATmega8 (not available on the Arduino Mega)
- INTERNAL1V1: a built-in 1.1V reference (Arduino Mega only)
- INTERNAL2V56: a built-in 2.56V reference (Arduino Mega only)
- EXTERNAL: the voltage applied to the AREF pin (0 to 5V only) is used as the reference.

Lien

Le CAN (ADC): *analogReference()*

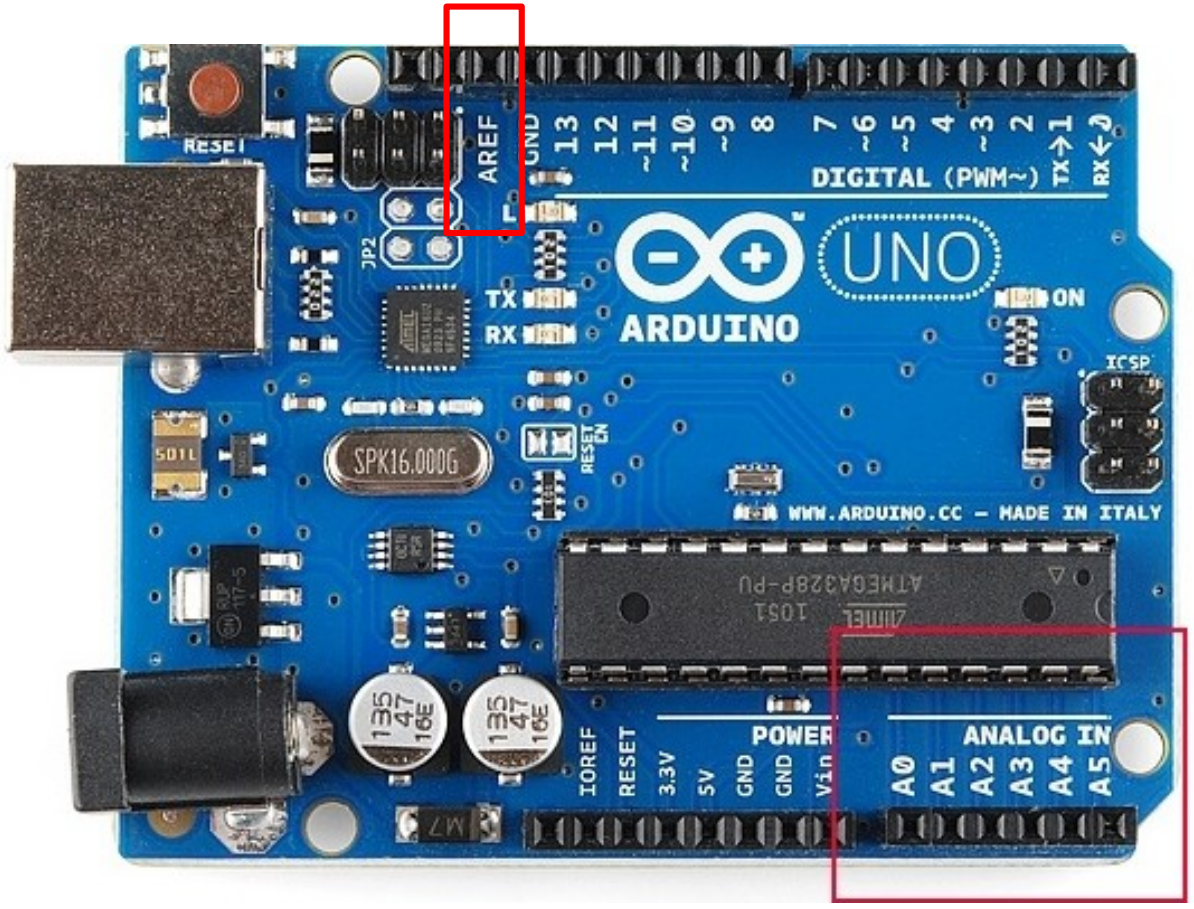
Voyons donc trois moyens de faire cette référence de tension :



Diverses références de tension

L7803 → +3V

Le CAN (ADC):

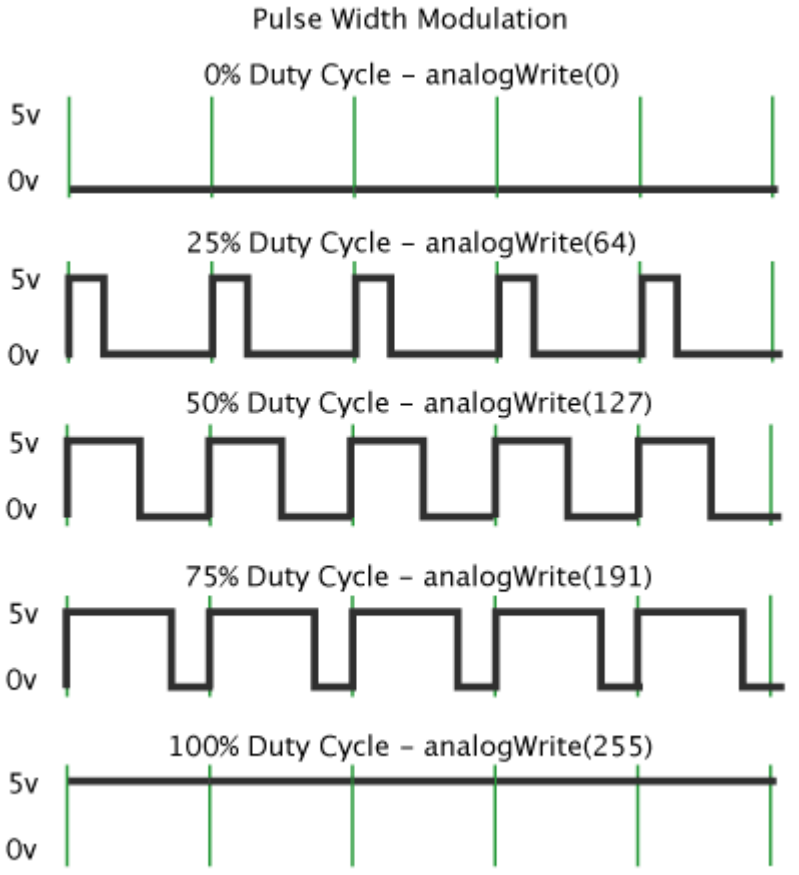

AREF
GND
13
12
~11
~10
~9
8
7
~6
~5
4
~3
2
TX→1
RX←0
DIGITAL (PWM~)
RESET
JP2
TX
RX
ARDUINO
UNO
ON
ICSP
SPK16.000G
RESET EN
WWW.ARDUINO.CC – MADE IN ITALY
POWER
ANALOG IN
IOREF
RESET
3.3V
5V
GND
GND
Vin
A0
A1
A2
A3
A4
A5

La MLI (PWM)

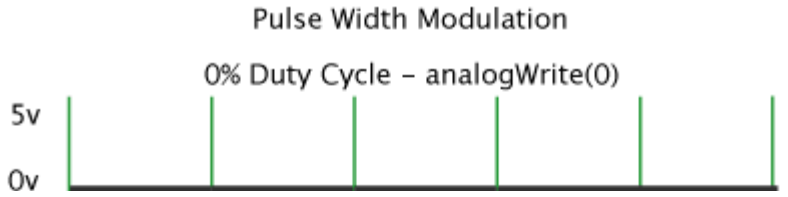
Pulse Width Modulation, or PWM, is a technique for getting analog results with digital means. Digital control is used to create a square wave, a signal switched between on and off. This on-off pattern can simulate voltages in between full on (5 Volts) and off (0 Volts) by changing the portion of the time the signal spends on versus the time that the signal spends off. The duration of "on time" is called the pulse width. To get varying analog values, you change, or modulate, that pulse width. If you repeat this on-off pattern fast enough with an LED for example, the result is as if the signal is a steady voltage between 0 and 5v controlling the brightness of the LED.

A call to analogWrite() is on a scale of 0 - 255, such that analogWrite(255) requests a 100% duty cycle (always on), and analogWrite(127) is a 50% duty cycle (on half the time) for example.

La MLI (PWM)


Pulse Width Modulation
0% Duty Cycle – analogWrite(0)
5v
0v
25% Duty Cycle – analogWrite(64)
5v
0v
50% Duty Cycle – analogWrite(127)
5v
0v
75% Duty Cycle – analogWrite(191)
5v
0v
100% Duty Cycle – analogWrite(255)
5v
0v

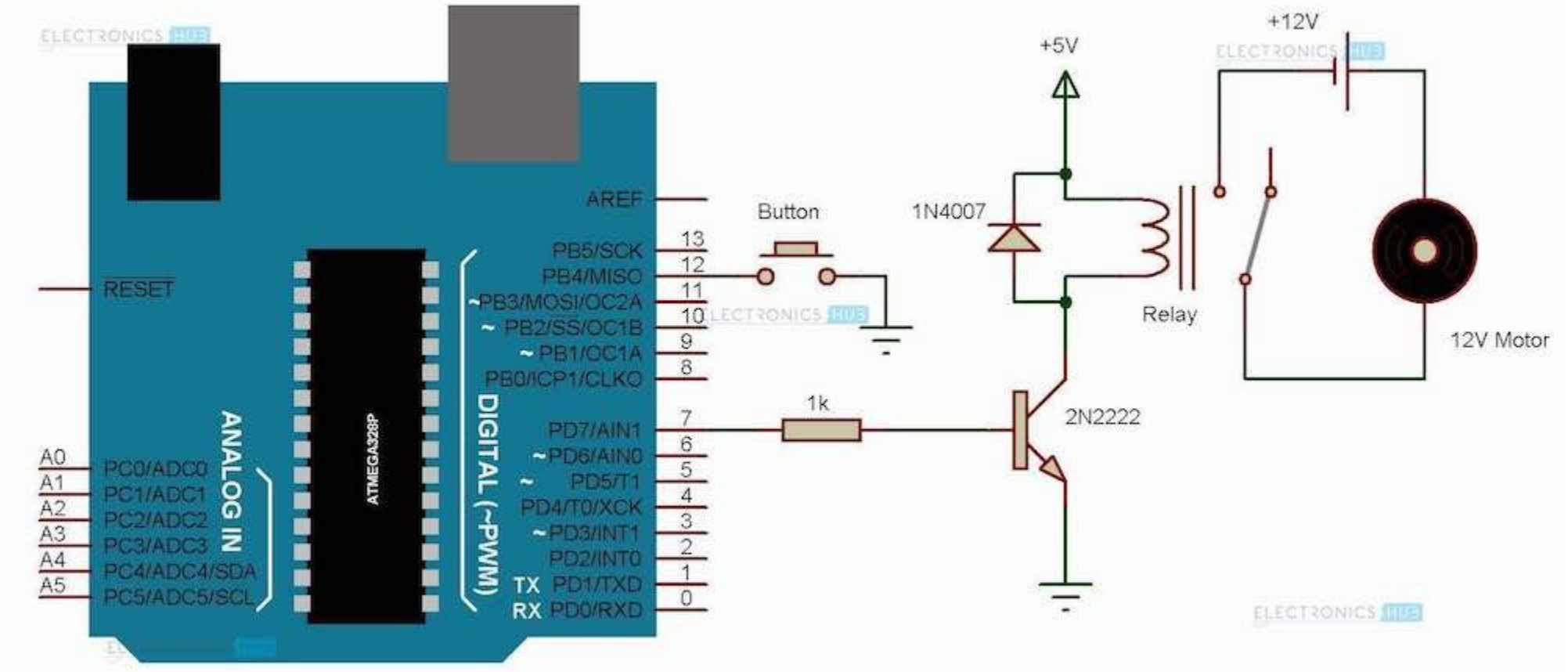
La MLI (PWM)

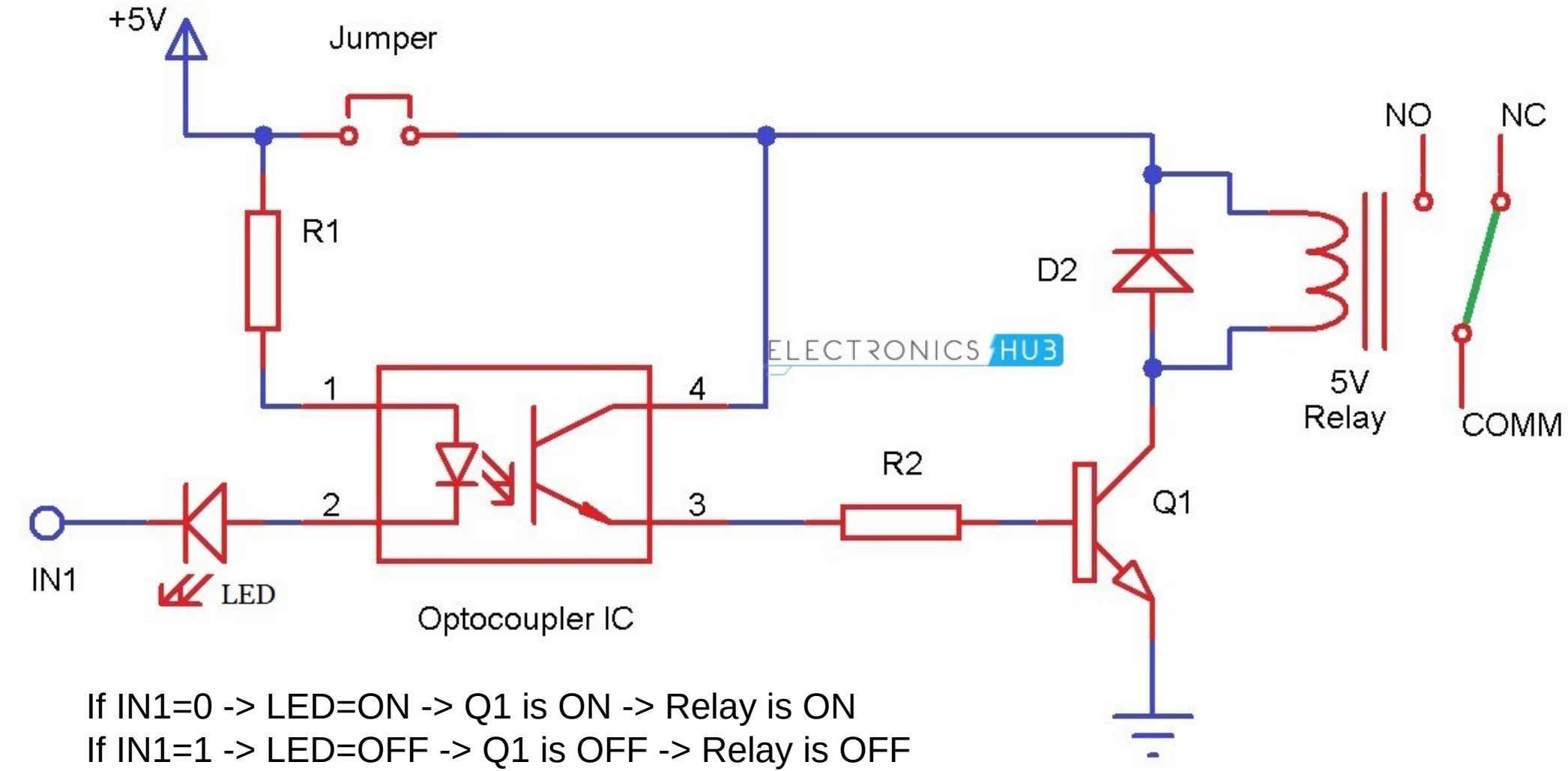


PWM has several uses:

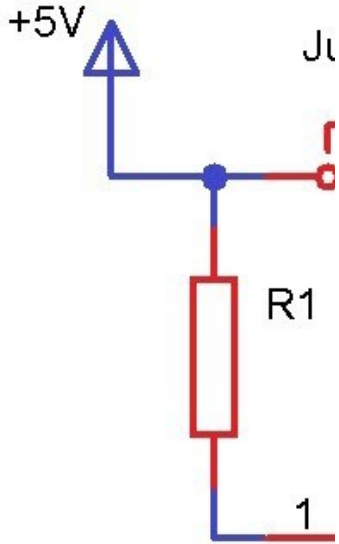
- Dimming an LED
- Providing an analog output; if the digital output is filtered, it will provide an analog voltage between 0% and 100% .
- Generating audio signals.
- Providing variable speed control for motors.
- Generating a modulated signal, for example to drive an infrared LED for a remote control.

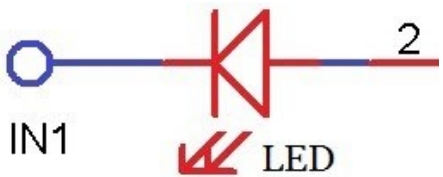
La commande de puissance

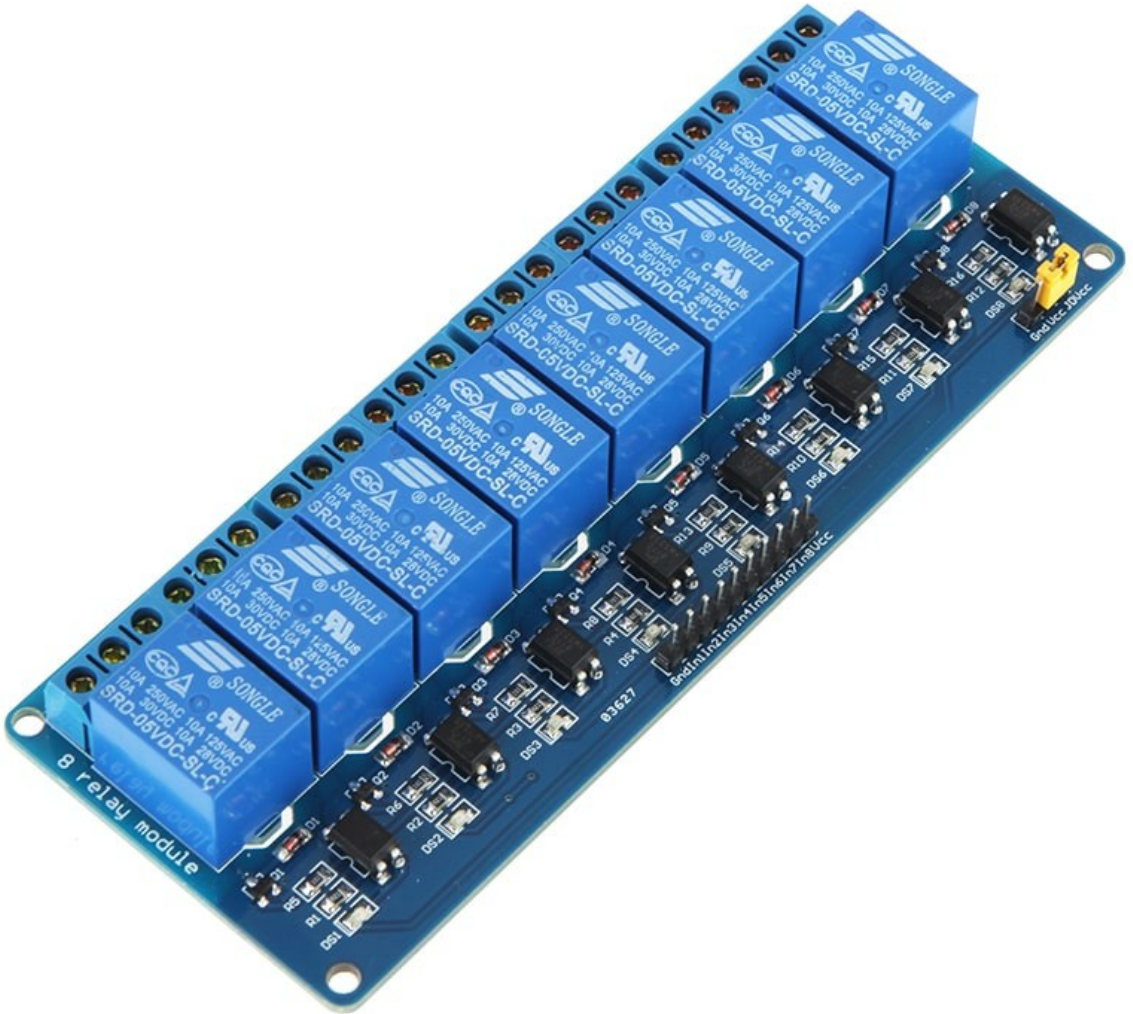


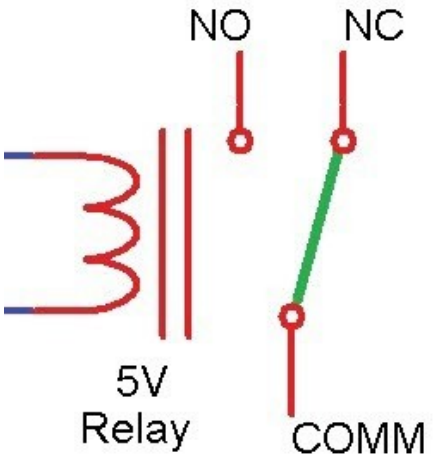


If IN1=0 -> LED=ON -> Q1 is ON -> Relay is ON
If IN1=1 -> LED=OFF -> Q1 is OFF -> Relay is OFF





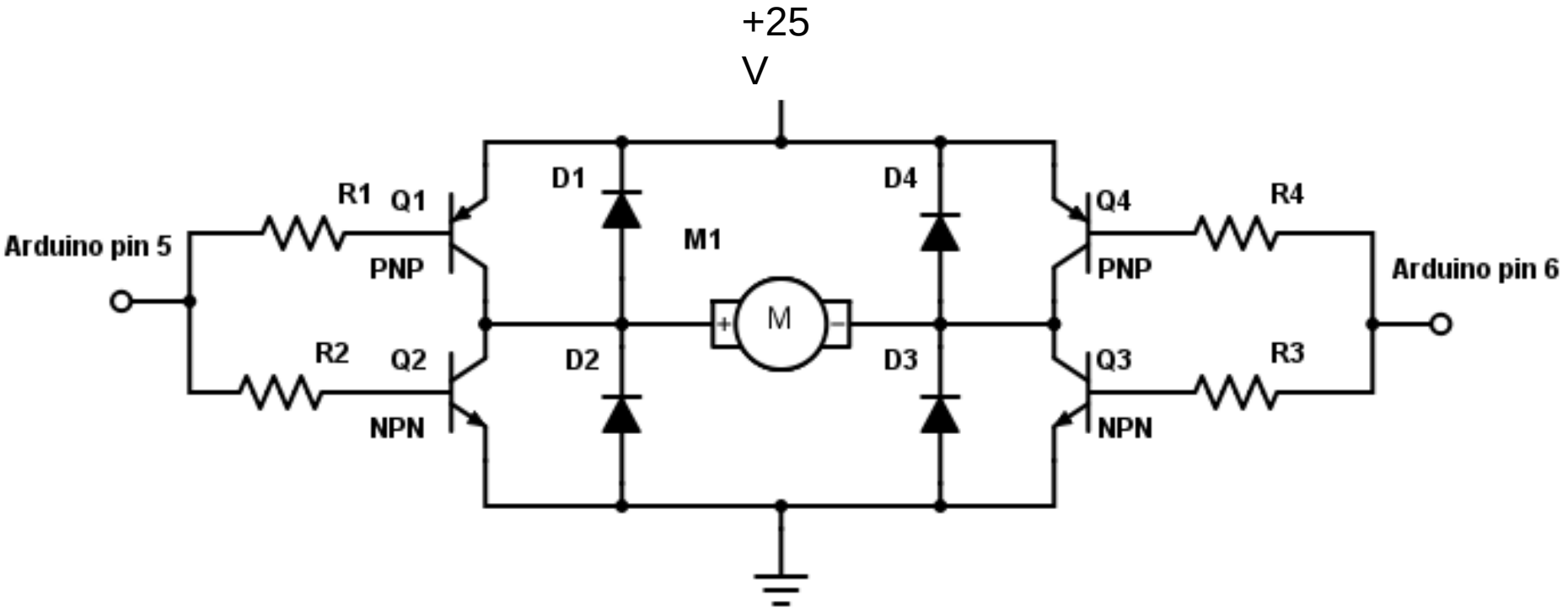




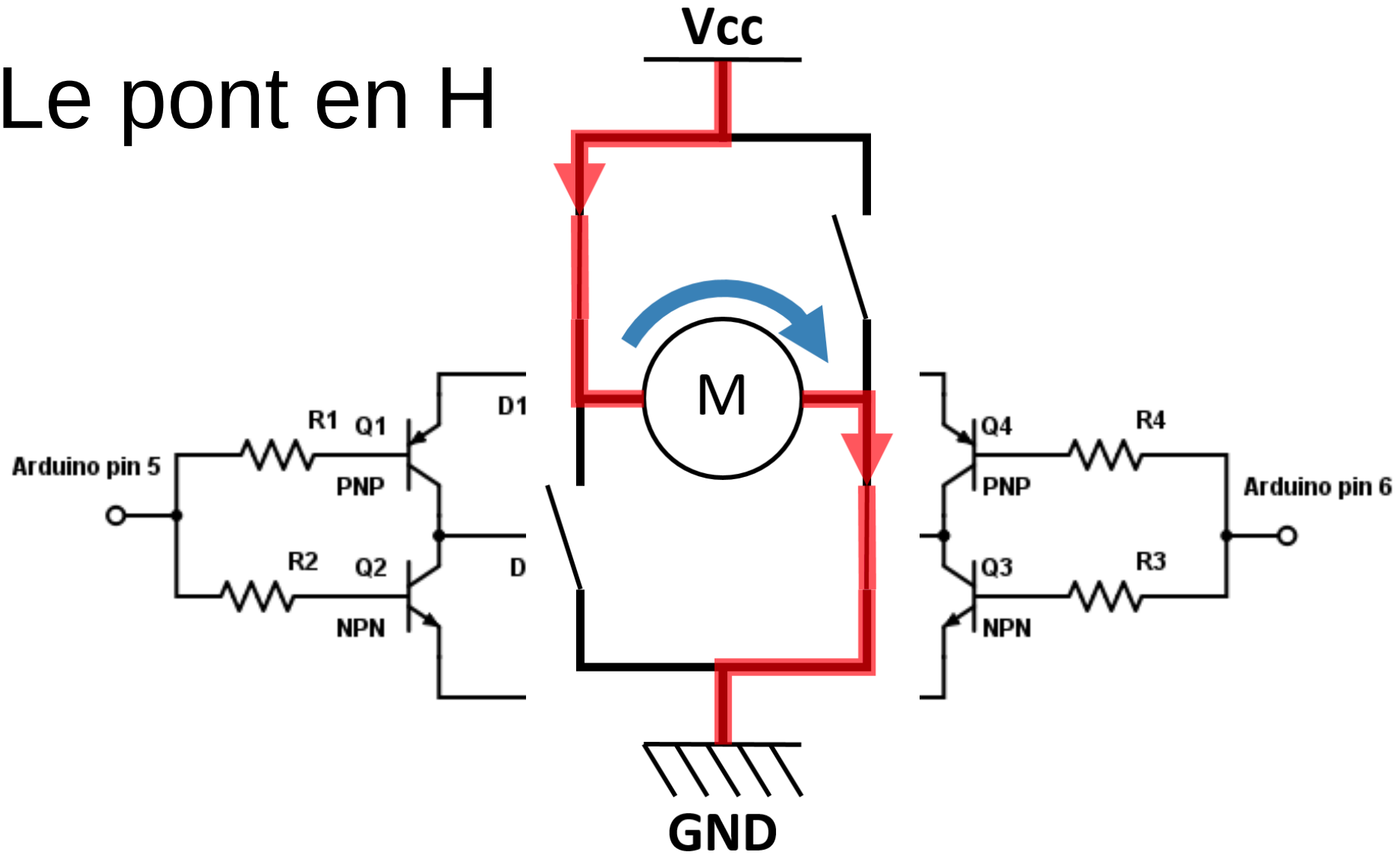
If IN1=0 -> LED=

If IN1=1 -> LED=

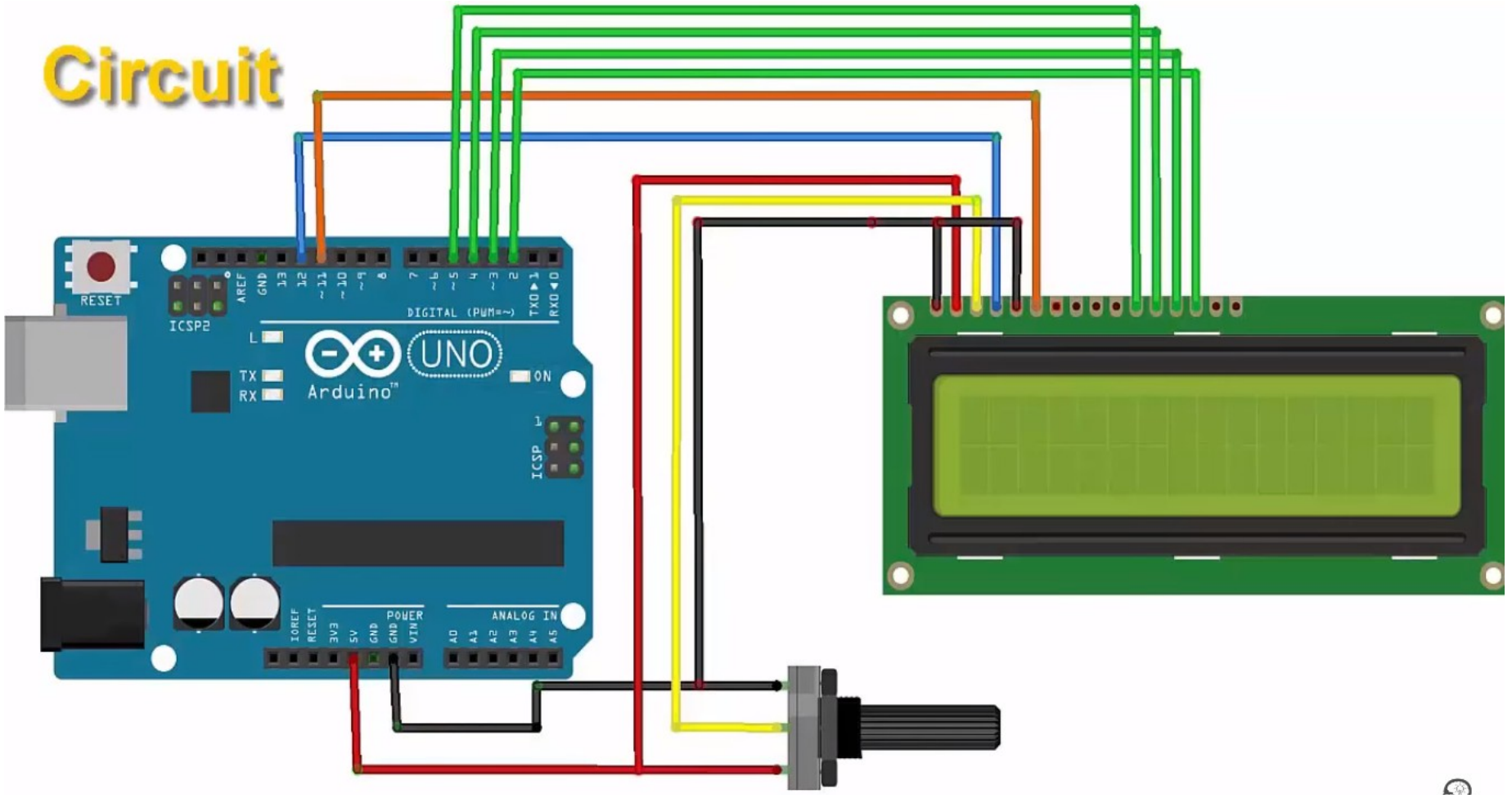
Le pont en H


+25
V
D1
D4
R1
Q1
Arduino pin 5
PNP
M1
Q4
R4
PNP
Arduino pin 6
M
R2
Q2
D2
D3
Q3
R3
NPN
NPN

Le pont en H


Vcc
M
GND
Arduino pin 5
R1
Q1
PNP
D1
R2
Q2
NPN
D
Q4
R4
PNP
Arduino pin 6
Q3
R3
NPN

Le LCD

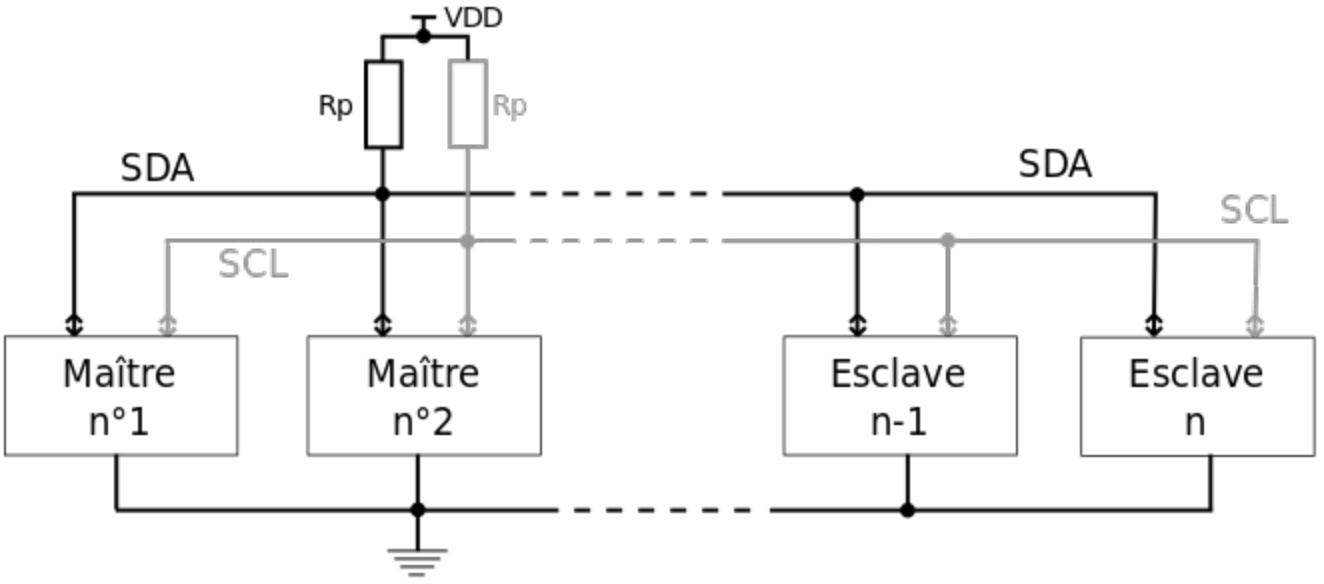

Circuit
RESET
ICSP2
AREF
GND
DIGITAL (PWM=~)
TX
RX
Arduino
UNO
ON
ICSP
POWER
ANALOG IN

Le LCD

```
#include <LiquidCrystal.h>
// initialize the library by associating any needed LCD interface pin
// with the arduino pin number it is connected to

const int rs = 12, en = 11, d4 = 5, d5 = 4, d6 = 3, d7 = 2; //mode 4 bits
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);

void setup() {
        // set up the LCD's number of columns and rows:
        lcd.begin(16, 2);  // Print a message to the LCD.
        lcd.print("hello, world!");
}
```

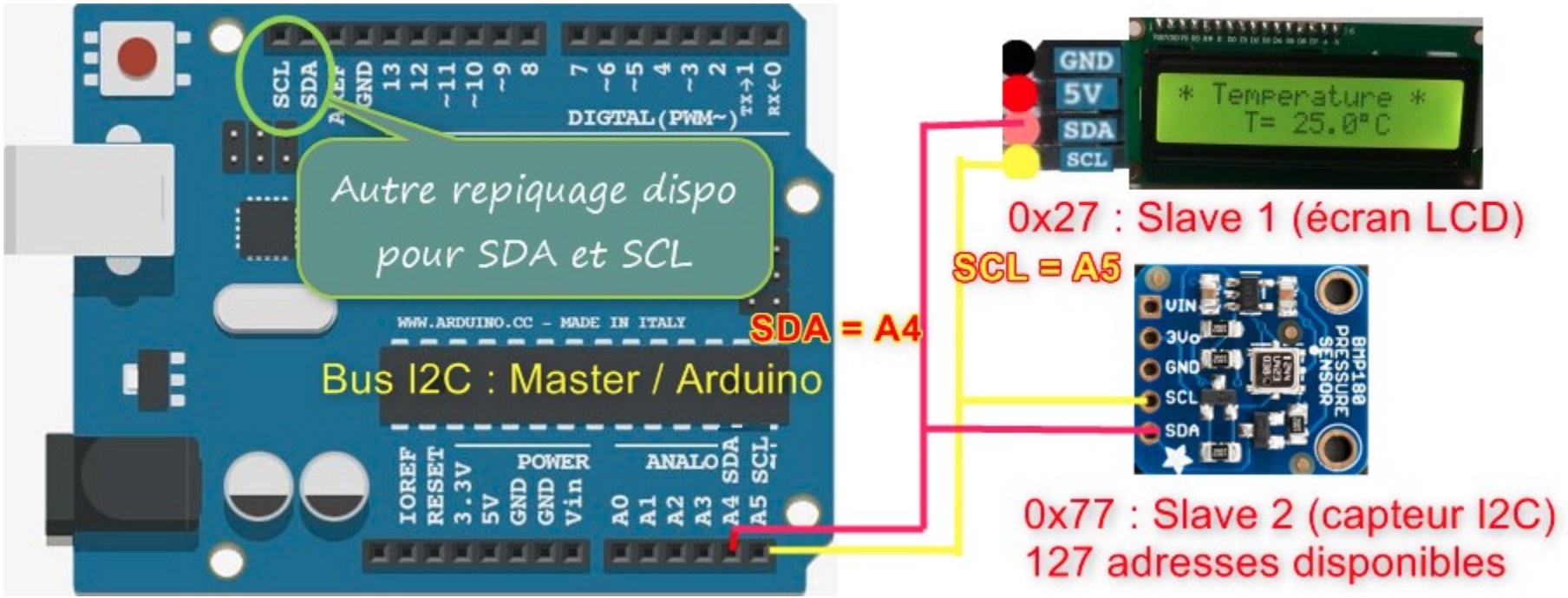
```
lcd.setCursor(0, 0); // top left
lcd.setCursor(0, 1); // bottom
left
```

Le bus I2C

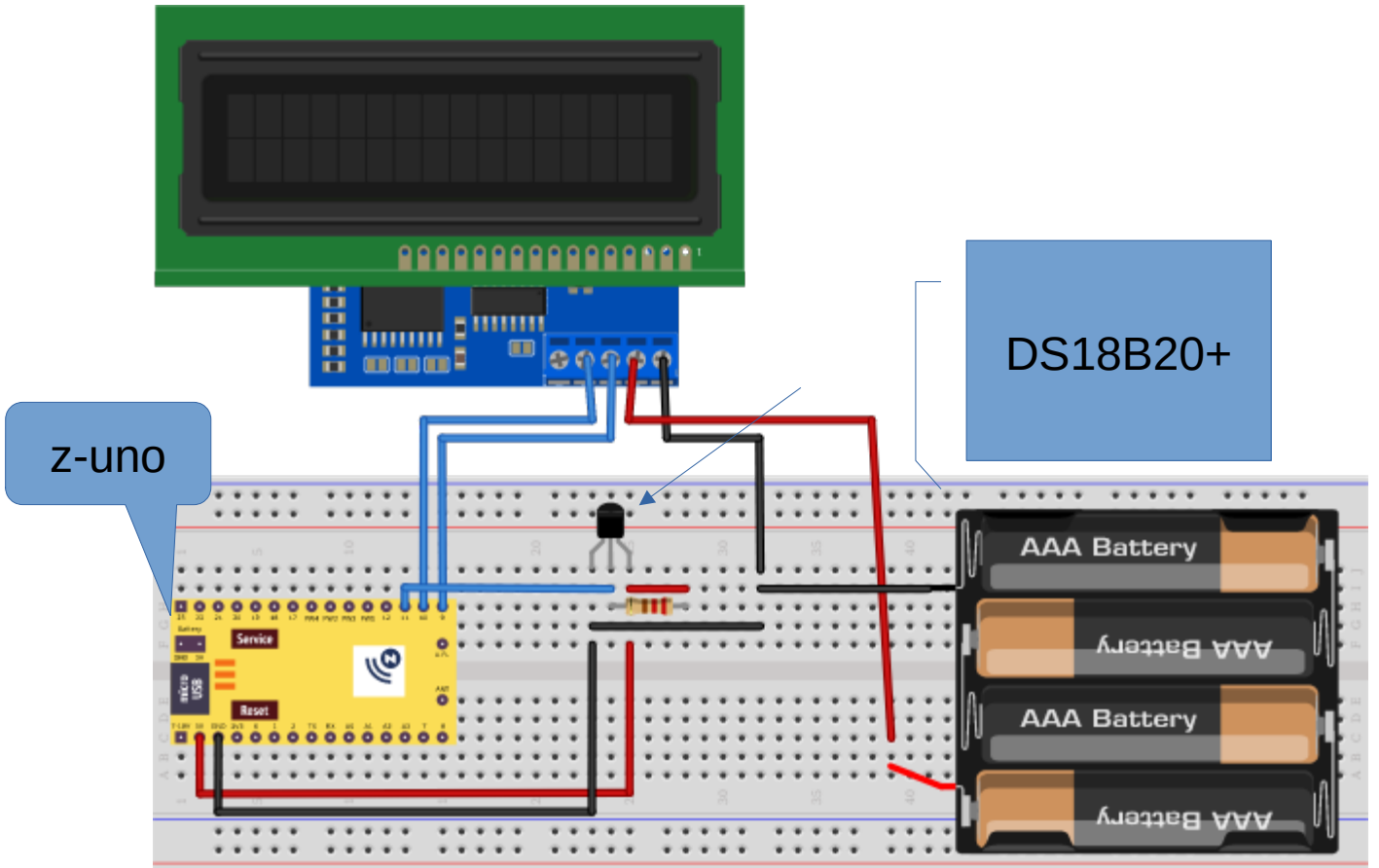


Chaque appareil a une adresse (paramétrable) sur le bus I2C

Le bus I2C


SCL
SDA
DIGITAL(PWM~)
Autre repiquage dispo
pour SDA et SCL
WWW.ARDUINO.CC – MADE IN ITALY
Bus I2C : Master / Arduino
SDA = A4
SCL = A5
IOREF
RESET
3.3V
5V
GND
GND
Vin
POWER
A0
A1
A2
A3
A4
A5
ANALO
GND
5V
SDA
SCL
* Temperature *
T= 25.0°C
0x27 : Slave 1 (écran LCD)
VIN
3Vo
GND
SCL
SDA
BMP180
PRESSURE
SENSOR
0x77 : Slave 2 (capteur I2C)
127 adresses disponibles

Le bus I2C


DS18B20+
z-uno
Service
Reset
micro USB
AAA Battery
AAA Battery
AAA Battery
AAA Battery

Le bus I2C