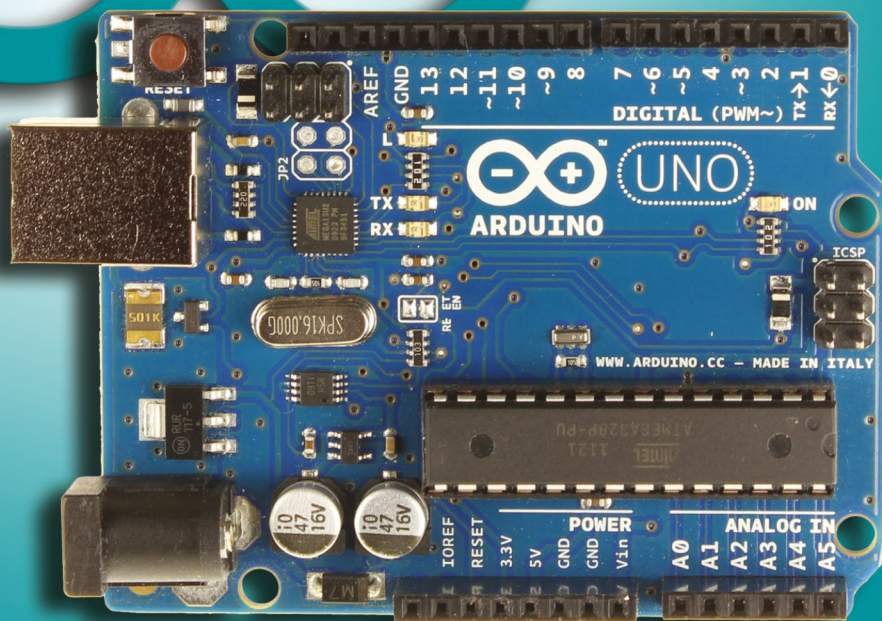




Bert van Dam

Traduction : Pascal Duchesnes

TUTO ARDUINO UNO

45 projets électroniques originaux

Téléchargement du code présenté dans ce livre :

www.elektor.fr/livres-arduino



Copyright © 2016 – Publitronic - Elektor International Media

Conformément au droit d'auteur, ce copyright ne s'applique pas à certains schémas reproduits dans ce livre à titre de citation et d'illustration des propos et de la démarche intellectuelle de l'auteur, avec l'aimable autorisation des ayants-droit.

Toute reproduction ou copie, même partielle, de ce livre, sans l'accord écrit de l'éditeur, est interdite.

No part of this book may be reproduced, in any form or means whatsoever, without permission in writing from the publisher. While every precaution has been taken in the preparation of this book, the publisher and author assume no responsibility for errors or omissions. Neither is any liability assumed for damages resulting from the use of the information contained herein.

La protection du droit d'auteur s'étend non seulement au contenu mais également aux illustrations, y compris aux circuits imprimés et aux projets y relatifs. En conformité avec l'article 30 de la Loi sur les brevets, les circuits mentionnés ne peuvent être exécutés qu'à des fins particulières ou scientifiques et non pas dans ou pour une entreprise ; **ces exécutions et/ou applications se font en dehors de toute responsabilité de l'éditeur.**

En application de la loi du 11 mars 1957, toute reproduction ou copie de ce livre, même partielle et sur quelque support que ce soit, sans l'accord écrit de l'éditeur, est interdite.

Le code de la propriété intellectuelle du 1^{er} juillet 1992 interdit expressément la photocopie à usage collectif sans autorisation des ayants droit.

L'éditeur remercie d'avance le lecteur qui prendra la peine de lui signaler les erreurs éventuelles qui auront échappé à sa vigilance (écrire à webmaster@elektor.fr).

1^{ère} édition, 1^{er} tirage

Mise en page : Mariline THIEBAUT-BRODIER
Coordination : Denis MEYER
Imprimé aux Pays-Bas par Wilco (04/2016)

ISBN : 978-2-86661-202-3

Sommaire

Introduction	7
1. Qu'est-ce qu'un Arduino ?	9
2. Éléments requis	11
2.1 Arduino	11
2.2 Kit	12
2.3 Logiciel gratuit	13
2.3.1 Téléchargement gratuit des codes sources	13
2.3.2 Environnement de développement intégré (Arduino IDE)	13
2.3.3 HyperTerminal	13
2.3.4 Émulateur d'oscilloscope	14
3. Un projet pour s'entraîner	19
Logiciels	19
Pilotes de la carte	19
Votre premier programme	21
4. Premiers pas (LED, boutons-poussoirs, liaison série)	23
4.1 LED clignotante	23
4.2 LED à clignotement alterné	31
4.3 Compteur sur port série	33
4.3.1 Moniteur série de l'Arduino	33
4.3.2 Comment recevoir une belle image sur le PC sans programmer	35
4.4 Débogage par liaison série	39
4.4.1 Le <i>sketch</i> ne fonctionne pas	39
4.4.2 Le résultat est incorrect	39
4.4.3 Plusieurs variables sont incorrectes	40
4.5 Dresser son propre tableau ASCII	45
4.6 Bouton-poussoir	48
4.7 Minuterie redéclenchable	54

4.8	Commutateur (basculer)	56
4.9	Dé	57
4.10	Sonnerie avec code d'autorisation	59

5. Conversion analogique-numérique (potentiomètre, photorésistance, tensions, modulation de largeur d'impulsion, capteurs) 63

5.1	LED à fréquence de clignotement réglable	63
5.2	Voltmètre	66
5.3	Éclairage de nuit (interrupteur crépusculaire)	69
5.4	Luxmètre	72
5.5	Détecteur de lumière pour chambre d'enfant	76
5.6	LED à luminosité réglable (MLI)	77
5.7	Wattmètre	81
5.8	Alarme silencieuse	86
5.9	Analyse d'un transistor	91
5.10	Capteur d'humidité pour plante	96

6. Régler et mesurer avec des tensions et des intensités plus fortes (moteurs, capteurs, son) 99

6.1	Alimentation	99
6.1.1	Alimentation via port USB	99
6.1.2	Alimentation externe	101
	Récapitulatif	103
6.2	Piloter un moteur électrique	104
6.3	Moteur électrique à vitesse variable	110
6.4	Compte-tours	115
6.5	Régulateur de vitesse (régime constant avec boucle de réglage)	121
6.6	Surveillance d'objet par infrarouge	122
6.7	Capteur à ultrasons (télémètre)	124
6.8	Capteur d'inclinaison ou capteur de mouvement	129
6.9	Mémoire	132
6.9.1	EEPROM : des souvenirs inoubliables	132
6.9.2	Mémoire flash (mémoire de programmes)	136
	Chiffres	138
	Textes (chaînes de caractères)	139

6.10	Commuter une tension alternative à l'aide d'un relais	140
6.11	Sirène de police	144

7. **Nous naissons tous fous. Quelques-uns le demeurent. (intelligence artificielle, Arduino et internet) 149**

7.1	Ma couleur préférée (intelligence artificielle)	149
	En option	152
7.2	Détecteur de présence humaine	152
7.3	Bougie électrique	155
7.4	Testeur de fleuret et d'épée	158
7.5	Capteur de toc-toc (qui frappe à ma porte ?)	162
7.6	Casse-pieds	165
7.7	Tetris avec 126 LED en <i>charlieplexing</i> (multiplexage)	166
7.7.1	<i>Charlieplexing</i>	166
	Mémoire flash	173
7.7.2	Tetris	177
7.8	Internet	179
7.8.1	Interroger un interrupteur depuis l'internet	179
7.8.2	Commuter une LED depuis l'internet	192
7.8.3	Internet et intranet	197
7.9	Commande sans fil d'un relais (liaison radio)	198
7.10	Fabriquer son propre Arduino	204
7.10.1	Microcontrôleur autonome	204
7.10.2	ATmega328P avec prise USB	211
7.10.3	Démonstration avec l'Arduino fait maison : LED à clignotement réglable	214

8. **Annexe 217**

8.1	Alimentation réglable (entre 1,2 et 13 V)	217
8.2	<i>Shield</i> tout équipé	219

Index 221

Votre premier programme

Pour notre premier test, nous allons faire clignoter la LED jaune de votre Arduino Uno. Pour cela, nous allons nous servir du programme de démonstration qui se trouve dans l'Arduino IDE sous : *Fichier, Exemples, 01.Basics, Blink* (en français « clignotant »).

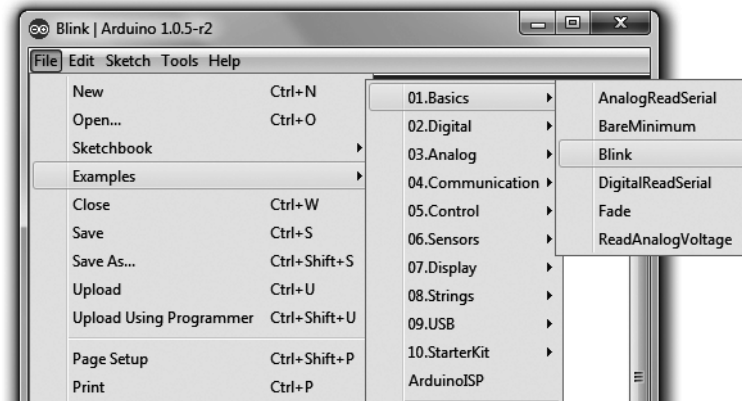


Figure 9 - Programme de démonstration Blink (clignotant)

Le programme est chargé automatiquement. Ne vous précipitez pas une fois que vous avez lancé Arduino IDE, car l'affichage de tous les programmes de la rubrique « Exemples » prend du temps. Si vous êtes allé trop vite, fermez le menu, attendez un instant puis réessayez ; la structure et le mode de fonctionnement du programme ne nous intéressent pas, nous y reviendrons dans le chapitre 4, page 23. Maintenant, le programme doit être traduit dans le langage Arduino avant d'être envoyé à la carte. Votre Arduino Uno étant relié à votre ordinateur, cliquez sur le bouton avec la flèche pointant vers la droite (voir numéro 1 dans la figure 11). Apparaît alors le message « Compilation du croquis en cours... » dans la barre de progression située au-dessous du programme.

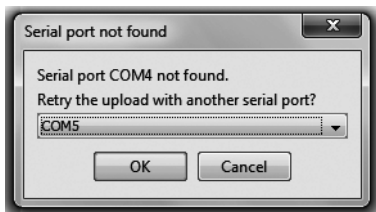


Figure 10 - L'Arduino IDE cherche un autre port COM.

```
b=t*5;
c=3*b-a;
Serial.write(27);
Serial.print("[H");
Serial.print("t= ");
Serial.println(t);
Serial.print("a= ");
Serial.println(a);
Serial.print("b= ");
Serial.println(b);
Serial.print("c= ");
Serial.println(c);
while(!Serial.available());
dummy=Serial.read();
t++;
}
```

À l'aide des données que nous avons dans la fenêtre du terminal, nous voyons ce qui ne colle pas avec les calculs. Un problème fréquent est que les calculs sont exécutés avec des valeurs qui ne conviennent pas au type de variable prévu à cet effet, c'est-à-dire lorsque « word » est utilisé alors qu'on aurait besoin de « int » ou lorsque les valeurs dépassent celles qui sont escomptées pour ce type de variable. Faites le calcul en vous servant d'une calculatrice pour dépister l'erreur.

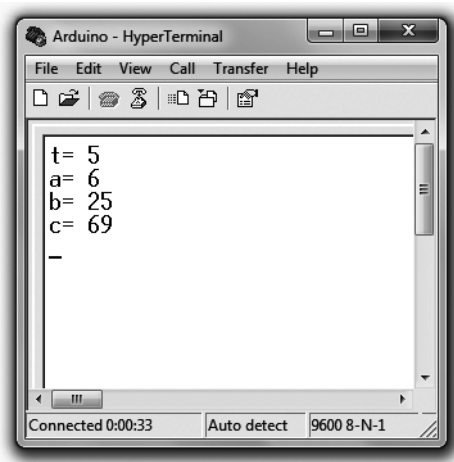


Figure 19 - Toutes les valeurs figurent les unes après les autres ; dans l'attente qu'on appuie sur *Entrée*.

Cette méthode permet également d'engendrer plusieurs colonnes à l'écran. Pour cela, on laisse le compteur « t » déterminer les colonnes. On voit bien que les différentes variables modifient leur valeur respective. Dans le prochain *sketch*, nous allons décaler

Nous utilisons le même matériel que dans le projet précédent. Vous pouvez tester le circuit en dirigeant la lumière d'une lampe de poche vers la photorésistance.

5.6 LED à luminosité réglable (MLI)

Je vous propose ici de régler la luminosité d'une LED en nous servant d'un potentiomètre. Nous allons utiliser pour cela la modulation de largeur d'impulsion (MLI ou *PWM* en anglais).

En principe, pour régler la luminosité d'une lampe ou la vitesse d'un moteur, il suffirait de faire varier sa tension d'alimentation. En pratique, ce n'est pas aussi simple et une commande par variation de tension ne fonctionne pas bien. Pour qu'ils commencent à tourner, la plupart des moteurs ont besoin d'une tension minimale. D'autre part, ils cessent de tourner en-dessous d'une certaine vitesse de rotation. Un deuxième inconvénient est leur faible couple, c'est-à-dire la force du moteur, à basse vitesse. Il en va de même pour les LED, qui elles aussi ne s'allument qu'au-dessus d'un seuil de tension minimale. En résumé, pour commander la luminosité d'une LED ou la vitesse d'un moteur, il ne suffit pas d'en faire varier la tension.

Une meilleure solution consiste à maintenir la tension constamment au maximum (dans notre cas, +5 V) et d'interrompre cette tension plus ou moins souvent ou longtemps. Plus la tension reste longtemps présente, plus le moteur tourne vite et plus la lumière de la LED est forte. Nous maintenons constante la fréquence d'interruption de la tension d'alimentation du moteur ou de la LED, mais nous modifions la durée (ou plutôt la largeur) des impulsions. Cette technique est appelée « modulation de largeur d'impulsion ».

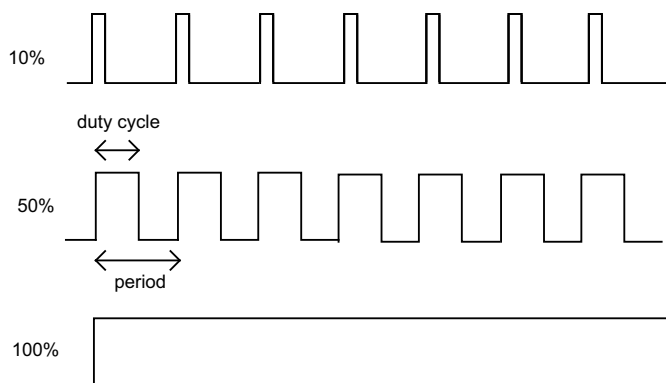


Figure 35 - Signal à MLI avec trois rapports cycliques différents



Figure 51 - Contacts et leur placement dans le pot de fleurs

Dès que le terreau est trop sec et donc que la valeur de mesure analogique est trop élevée, la LED s'allume pour signaler que la plante doit être arrosée. Les valeurs exactes dépendent naturellement de vos capteurs et de leur écartement. J'ai obtenu les résultats suivants :

humidité	ma mesure
insuffisante	env. 500
suffisante	env. 300
excessive	env. 200

Tableau 7 - Mesures de l'humidité

Vous pouvez dresser ce tableau en vous servant du *sketch* présenté au paragraphe 5.2, page 66. Suite à mes mesures, j'ai noté une valeur de seuil de 400 dans mon *sketch*.

```
int plant = A3;
int led=2;
int button = 8;
int threshold=400, value;

void setup() {
  pinMode(led, OUTPUT);
  pinMode(button, INPUT);
}

void loop(){
  value = analogRead(plant);
  // la configuration de la LED est requise
  if (value>threshold+20){
    digitalWrite(led, HIGH);
  }
}
```

Suite à cela, mon microcontrôleur s'est mis en grève et mon robot refusait de bouger.

En me servant d'un émulateur d'oscilloscope, j'ai constaté que ces capteurs provoquaient des perturbations dans le bloc d'alimentation. J'ai résolu le problème en me servant d'un condensateur de 100 μF connecté aux broches d'alimentation de chaque capteur. Ici, vous n'avez pas besoin d'un tel condensateur. Ce dernier n'est requis que lorsqu'on utilise des fils de connexion longs dans un environnement perturbateur (moteurs).

6.7 Capteur à ultrasons (télémètre)

Dans ce projet, nous connectons le capteur à ultrasons (en anglais *Ultrasonic Range Finder*) SRF05 à l'Arduino pour mesurer la distance qui le sépare des objets qui l'entourent.



Figure 72 - Capteur à ultrasons, modèle *Ultrasonic Range Finder SRF05*

Le SRF05 de Daventech est un capteur à ultrasons économique qui permet de mesurer la distance qui le sépare d'un objet qui émet des ondes acoustiques.

La figure 73 illustre que le cycle de détection de l'Arduino requiert une impulsion d'une longueur d'au moins 10 μs sur l'entrée impulsion de déclenchement (en anglais *pulse trigger*) pour pouvoir démarrer. Dès que cette borne repasse à l'état bas, le module envoie une série de huit impulsions ultrasoniques d'une fréquence de 40 kHz. Pour parer à une rétroaction entre l'émetteur et le récepteur, le module attend un instant et met la sortie impulsion d'écho (en anglais *echo pulse*) à l'état haut (HIGH). C'est le

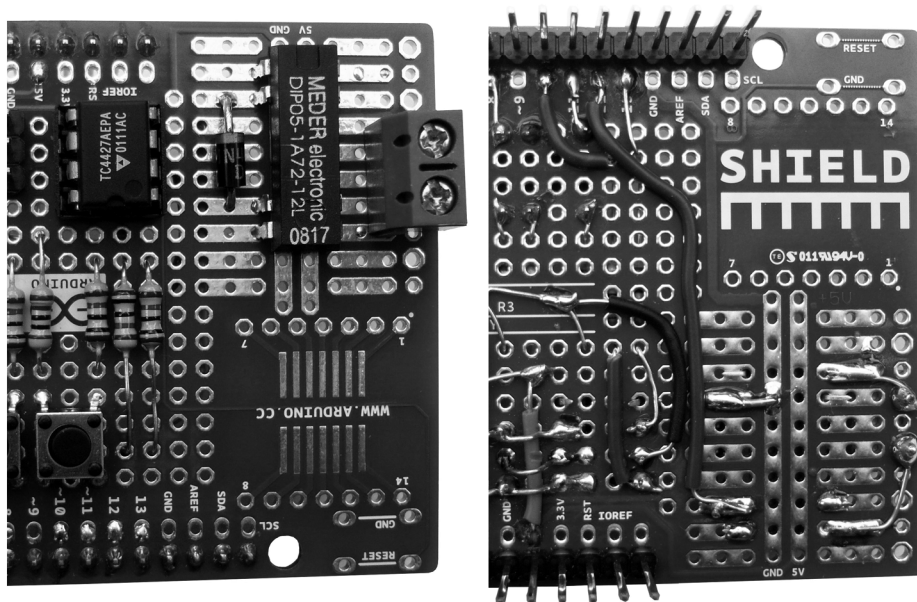


Figure 88 - Relais et diode sur le *shield*

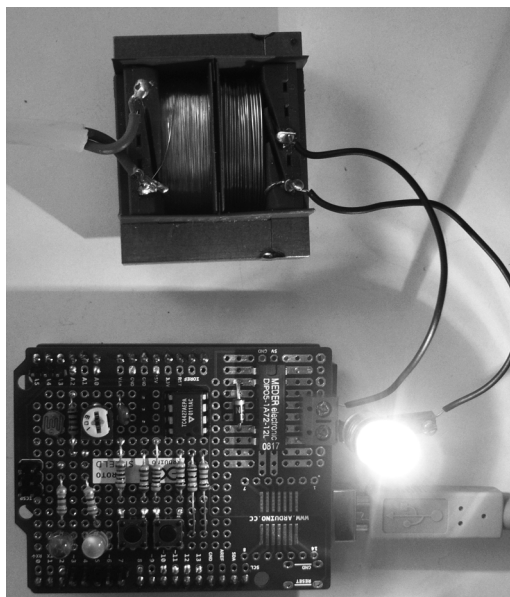


Figure 89 - Lampe de 10 V clignotante

change d'état (elle s'allume si elle était éteinte et vice versa). Et on attend une nouvelle fois 100 ms. Si la deuxième LED s'allume, c'est que le temps d'allumage est faussé ou qu'il ne correspond plus à ce qui avait été prévu. Si cela vous dérange, vous pouvez insérer après le *delay*, l'instruction $t=t+100$. Le compteur fait alors un bond en avant de 100 ms.

Et voilà comment nous allons nous en servir :

1. Tapotez l'élément piézoélectrique pour activer la LED connectée à la broche 2.
2. Si la LED s'allume, tapotez l'élément piézoélectrique pour allumer et éteindre la LED connectée à la broche 5.

Il serait facile de réaliser une gâche électrique à combinaison secrète en incorporant un compteur de tocs dans la boucle de comptage et en commutant un relais lorsque le compteur a atteint la valeur de consigne.

7.6 Casse-pieds

Sur l'internet j'ai découvert un montage proposé par le bien nommé Zach le terrible.⁸ En effet, son projet a pour objet de déranger les gens, la nuit, durant leur sommeil.

Il s'agit d'un petit circuit que l'on cache dans la chambre de sa victime. Dès qu'il fait noir, le circuit se met en route et produit un bip irritant. Ce bip s'arrête dès que la victime allume la lumière. La personne croit avoir rêvé, elle éteint donc la lumière pour se recoucher. Mais le cirque recommence.

Dès qu'il est sous tension, le circuit mesure la luminosité. Si celle-ci se trouve au-dessous d'une certaine valeur de seuil (c'est-à-dire dès qu'il fait sombre), le bip se fait entendre. Dès que la luminosité dépasse la valeur de seuil, le circuit reste silencieux, il attend entre 30 et 90 s, puis il vérifie si la lumière a été éteinte. Vous pouvez bien sûr allonger le temps d'attente, mais pas trop longtemps pour que la victime ne se rendorme pas.

```
int speaker = A1;
int ldr = A2;
int pot = A0;
int ldrn, potm;

void setup(){
  randomSeed(analogRead(5));
  Serial.begin(9600);
}
```

8. Ce montage est fait de manière traditionnelle sans Arduino et sans microcontrôleur.

!

.....	53, 175	analogRead	65
-	53	analogReference	65
--	34	Arduino IDE	13
!	32	array	59
!=	53	ASCII	36, 45
*	53	ATmega328P	11, 205
**	53	avr/pgmspace	138
*=	34		
/	53	B	
/=	34	BC547	156
&	111	BC547C	91
%	53	BJT	91
+	53	bobine	141
++	34	byte	30
+=	34		
-=	34	C	
=	53	capteur d'inclinaison	129
==	53	char()	200
>	53	<i>Charlieplexing.h</i>	173
>=	53	clé USB SRF	199
>>	53	connecteur mâle	25
.....	111	convertir	66
		CS12 à CS10	111

Numérique

1N4007	142
71427	104

A

accélération	132
alimentation	99
alimentation réglable	217

D

débit en bauds	33
delay	29
digitalRead	52
digitalWrite	28
DIP05-1A72-12L	141
disque d'impulsion	116
duty cycle	78

E

EEPROM	132
EEPROM.read	133
EEPROM.write	133
else	52
émulateur d'oscilloscope	14
ET, opération logique	111

F

F ()	140
<i>favicon</i>	183
flash, mémoire	136
flotter	48
fréquence MLI	111

G

GP1S036HEZ	129
GP2Y0D340K	122

I

imp	81
-----------	----

L

LM317	218
loi d'Ohm	24

M

mémoire flash	136
MLI	77
modulation de largeur d'impulsion ..	77
moniteur série	35

N

<i>NOT</i>	32
noTone	147

O

OptiLoader	207
OU, opération logique	111
OUTPUT	28

P

<i>pgm_read_word_near</i>	138
piézo	144
pilote	19
pinMode	28
pointeur	139
<i>port forwarding</i>	197
potentiomètre	63
prédiviseur	111
prog_	139
PROGMEM	138
<i>PWM</i>	77
PySerial	180
Python	180

Q

QRB1134 116

R

radio, liaison 198
 random 57
 randomSeed 57
 rapport cyclique 78
 rapport impulsion-pause 78
 relais Reed 141
 résistance
 de rappel vers le bas 48
 de rappel vers le haut 48

S

Serial.available 33
 Serial.begin 33
 Serial.Event 33
 Serial.print 33, 37
 Serial.println 33
 Serial.read 33
 Serial.write 33, 37
 setup() 27
 sleep 188
 SRF05 124
 static lease 184
 substring 202

T

TC4427A 104
 TCCR1B 111
 temporisateur 126
 TI CC1111 200
 timer 126
 tone 147
 tristate 167
 try/except 188
 TTL-232R USB -> TTL série 212

V

void 28
 VT100 36

W

Watanabe 149
 while 56
 while(1) 45
 WinOscillo 14
 word 30

X

XinoRF 199