ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Robopoly 5 nov 2018

Machines d'état et déverminage

Quelques notions importantes
pour programmer un robot

Jean-Daniel Nicoud

Test du matériel

Contrôle visuel
Consommation de la carte

Programme de test simple pour chaque fonction.

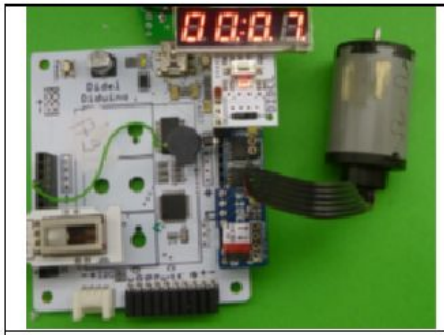
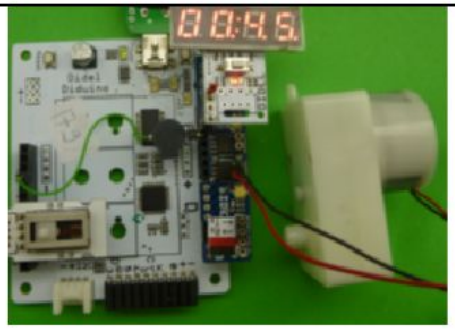
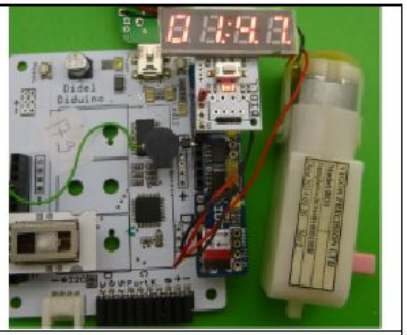
Si KO, contrôle visuel, contrôle de continuité, contrôle des états logiques (0, 1 flottant).

Signaux pulsés: crayon logique, oscilloscope.

Comportement critique ? → modifier la tension (3 à 5.5V).

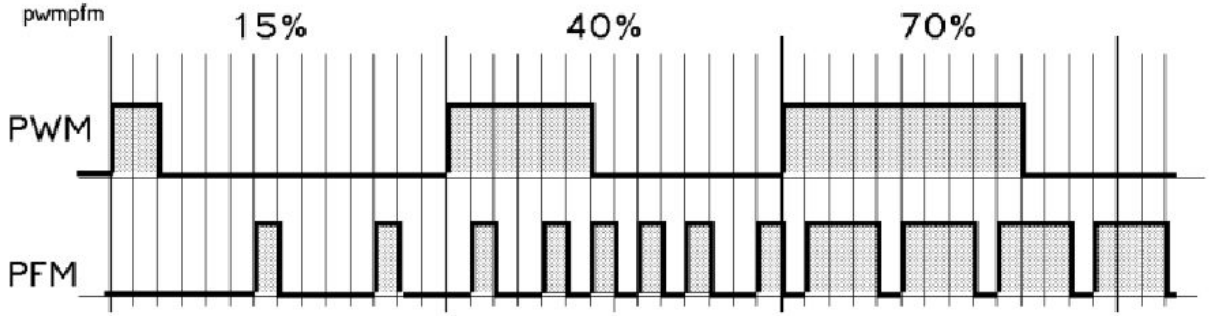
PFM vs PWM for Robots

Let us compare ! What is the minimum PWM value to start a motor?

		
Maxon A-max 22mm needs 7/256 PWM (3%) (no gearbox)	Vigor BO-10 needs 45/256 PWM (18%) (1:192 gearbox)	Vigor B0-1 needs 147/256 PWM (57%) (1:120 gearbox)

PWM and PFM

PWM is fix frequency, pulses widen to bring more power.
PFM send constant duration pulses that get closer to give more power.



For getting very low motor speed, PFM pulses must be long enough to bring enough energy to make the motor win against frictions. Hence, there is no limit to minimal speed, but movement is cogged. For speed of few %, the grearbox will smoothe it.

Pr. Jean-Daniel Nicoud

Syntaxe: résumé sous didel.com/C/Resume.pdf
(3 slides suivantes)
Doc MOOC goo.gl/7EfBsm

Messages d'erreur

Souvent cabalistiques, signalés à un numéro de ligne souvent postérieur.
Relire le code depuis le début

Erreurs non signalée par le compilateur

Comparaisons if (aa = 0)
0x oublié devant un nombre hexa 78
Parenthèses manquantes a+1<<2
Table qui déborde ou erreur de no - 0 1 2 si 3 éléments
Types incorrects, oubli volatile, static

Résumé Arduino/C

Vous savez, mais vous n'êtes plus très sûr

Définitions, déclarations
utiliser des mots qui parlent
	`#define Led 13 // ! pas de ;`
	`int Duree = 100 ;`
	`const int Led=13 ; // constante`

variables char codeA = 'A'; 8 bits signé unsigned int nom; 16 bits	`byte` (non signé 0..255) `int` (-32768..+32767)
tableau taVal [5]; taVal []={18,5,3,7,5};	
void set-up () { configuration, initialisations }	`pinmode (LedTop, OUTPUT);` `DDRC = 0b00000111; // 0 = in 1 = out`

void loop () {
les instructions se terminent par un ;
}
	`loop() {`
	`digitalWrite (LedTop,LOW ;`
	`}` ! LOW ne veut pas dire inactif !

Calcul aa=aa+2 ; ou aa += 2 ; + - * / % j++ ; ajoute 1 j--; soustrait 1 **bitwise** & (and) , | (or), ^ (xor) , ~ (not) >> , << décale	**Comparaison** == (égalité), !=(différent), < , > **boolean** &&, || , ! (not) valeur =0 faux ou différent de 0 vrai

if (condition) { bloc d'instructions; }
si vrai on fait
On teste et on passe plus loin
	`if (condition) {`
	`instructions ;`
	`}`

if (condition) { } else { } si vrai on fait, autrement on fait autre chose	`if (condition) uneInstruction ;` `else uneInstruction ;`
while (condition) { } on fait et refait tant que la condition est vraie	`while (condition) {` `// instructions ; }`
while (1) { bloc d'instructions; } on fait en boucle les instruction (comme loop),	`while (1){ instructions; }`
while (1) { } on ne fait plus rien	
do { bloc d'instructions; } **while (condition) ;**	`do { a++ ;} // a déclaré avant` `while (a < 5) ;`
for (init ; condition ; modif) { bloc d'instructions; }	`for (byte i; i<5; i++)` `{ Cligno (); }`

byte etat;
switch (etat) {
case 0: instructions; **break;**
case 1: instructions; **break;**
default: instructions; // optionnel
}
	enum { Avance, Recule,..} **etat;**
	switch (etat) {
	case avance: instructions;
	etat = recule; break;
	case recule: . . . .
	}

Commentaires // commentaire jusqu'à la fin de ligne	/* Commentaire sur plusieurs lignes */

Constantes: HIGH (maj) Variables: temp (minuscules) Fonctions: FaireCeci (maj-min)

Fonctions Arduino

pinMode(pin,b); digitalWrite (pin,b); digitalRead (pin);	boolean boolean boolean	0 = LOW 1 = HIGH
analogWrite (pin,val) analogRead (pin);	byte unsigned int	0..255 pins 5,6, 3,11, 9,10 0..1023 pins 14,15,16,17,18,19 A0 .. A5
delay (ms); delayMicroseconds ();	unsigned long unsigned int	0..~10^{10} millisecondes (~50 jours) 0.. 65535 us (~0.06 sec)
millis (temps); micros(temps);	unsigned long unsigned long	0..~10^{10} millisecondes (~50 jours) depuis reset 0.. ~10^{10} us (~60 sec)
pulseIn (pin,b);	unsigned long	b=HIGH mesure imp à 1 HIGH attend l'impulsion et mesure sa durée
min(a,b) max(a,b)		choisit le min ou max

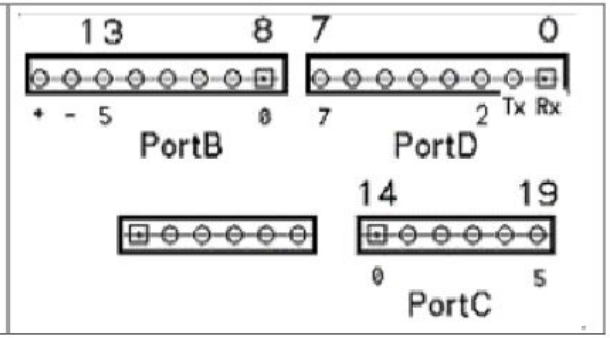
constrain (x, a, b) ;		garde la vaeur x entre a et b
map (value, fromLow, fromHigh, toLow, toHigh) ;		applique une régle de 3
bitSet(var,n); bitClear(var,n);	boolean boolean	bits numérotés depuis la gauche. s'applique aussi aux PORTs, DDRs
bitRead(var,n); bitWrite(var,n,b);	boolean boolean	if (!bitRead(PORTC,bMousD)) - vrai si MousD pressée PORTC&(1<<bMousD) préférable
lowByte(var); highByte(var);	int, long int, long	prend les 8 bits de poids faible var&0xFF plus rapide prend les 8 bits de poids fort var&0xFF00 var&0xFF000000

		val&0xFF000000
Serial.begin(9600); Serial.end();		setup canal série du terminal désactive, rarement utilisé
Serial.print (); Serial.println();		(75) → 75 (75,BIN) → 100101 (75,HEX) → 4B ("Texte 1""2") → Texte 1"2 '1' code Ascii de 1 = 0x31=49
Serial.write();		(65) → A ("abcd") → abcd
Serial.available() Serial.read(); Serial.flush():		if (Serial.available() > 0) { octetRecu = Serial.read(); } vide le tampon
tone (pin, fréqHz); tone (pin, fréqHz,durée); notone ();		démarre un son démarre un son pour la durée spécifiée en ms stoppe le son
shiftOut(D,Ck,dir,val8);		décale 8 bits – très lent

randomSeed(valeur);
random(max);
random(min,max);
	int
	long
	long
		randomseed (analogRead(A0);) A0 flottant
		valeur rendue entre 0 et max-1
		valeur rendue entre min et max-1

attachInterrupt(pin,fct,m); detachInterrupt();		Appelle la fonction s'il y a transition selon mode sur la pin 2 ou 3
interrupt();nointerrupt();		active/désactive toutes les interruptions

Correspondance pins Arduino – bits AVR 168/328
Les pins 0 à 7 vont sur les bits 0 à 7 du portD. Les pins 0 et 1 ne sont pas utilisées pour ne pas interférer avec USB.
Les pins 8 à 13 vont sur les bits 0 à 5 du portB
Les pins 14 à 19 vont sur les bits 0 à 5 du portC; elles acceptent l'ordre analogRead (pin); (valeur 10 bits)
Les pins 3, 5, 6, 9, 10, and 11 acceptent l'ordre analogWrite (pin,pwm8bits);



Interrupt **Timer0** 8 bits (delay(), millis(), PWM 5 and 6) coupe 8us toutes les ~700 us
Timer1 16 bits (PWM 9 and 10, servos)
Timer2 8 bits (PWM 3 and 11) xBot Interruption 100us

Syntaxe: résumé sous didel.com/C/Resume.pdf
(3 slides suivantes)
Doc MOOC goo.gl/7EfBsm

Messages d'erreur

Souvent cabalistiques, signalés à un numéro de ligne souvent postérieur.
Relire le code depuis le début

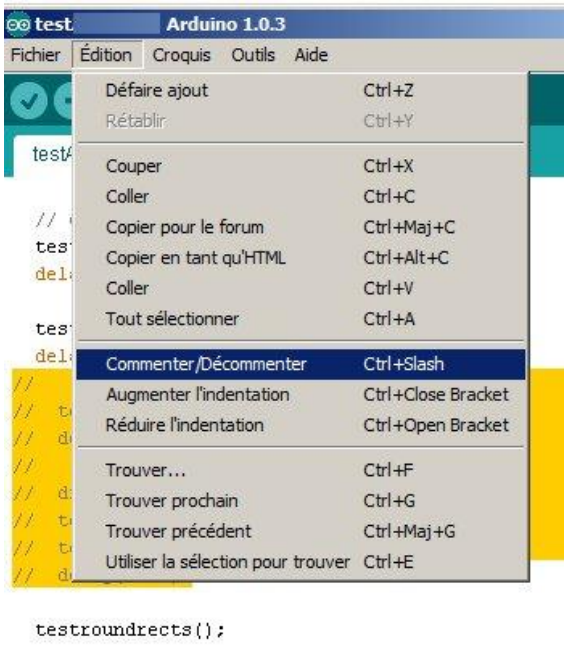
Erreurs non signalée par le compilateur

Comparaisons if (aa = 0)
0x oublié devant un nombre hexa 78
Parenthèses manquantes a+1<<2
Table qui déborde ou erreur de no - 0 1 2 si 3 éléments
Types incorrects, oubli volatile, static

Mise au point

Ajouter une fonctionnalité à la fois

Commenter/décommenter



On cache temporairement des instructions pour en essayer d'autres

Insertion conditionnelle

```
#define Debug
   void setup()  {
      pinMode (Led, OUTPUT);
   }
   void loop()  {
      #ifdef Debug
      LedOn; delay (200);
      LedOff; delay (200);
      #endif
   }
```

Si Debug n'est pas déclaré,les blocs `ifdef` ne sont pas exécutés

Signaler si on passe par un point du programme

Terminal (lent et l'info se déroule)

Led On/Off ou clignote (lent)

Bloquer, observer et continuer:

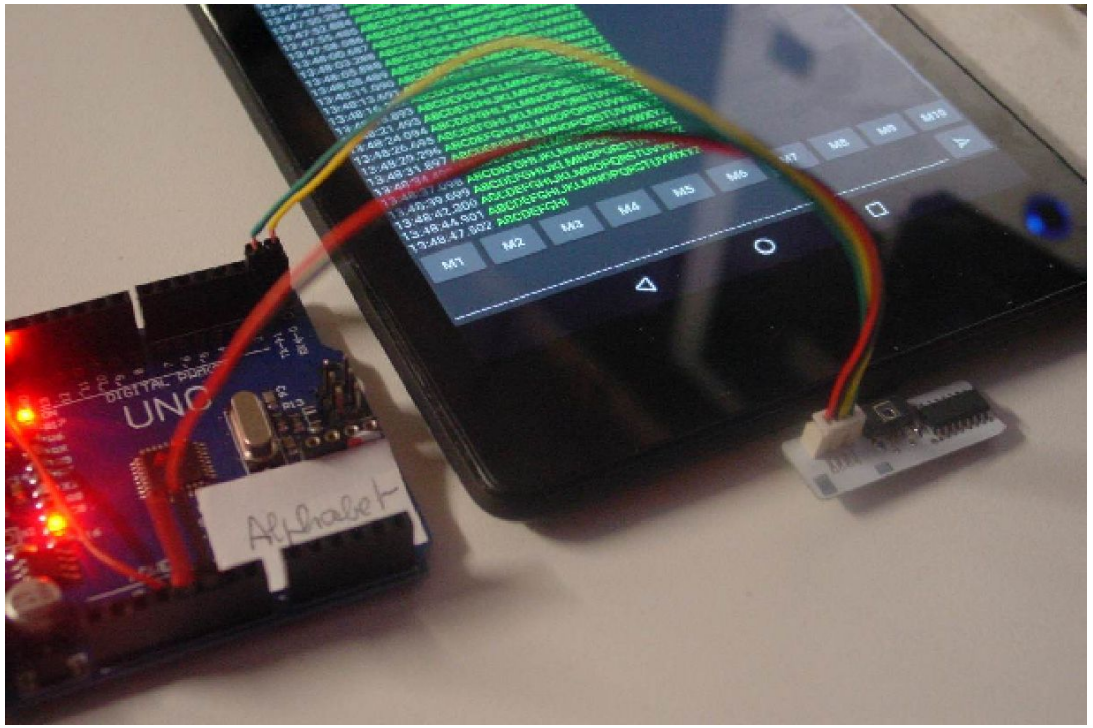
```
void  AffiVar (byte nn,int vv) {
        Serial.print (nn);
        Serial.print (" ") ;
        Serial.println (vv) ;  // décimal

        while (!PousOn)  {delay (20); }
        while  (PousOn)  {delay (20); }
    }
```

Appel par AffiVar (3,nomVar)

Terminal sur tablette

Il faut intercepter les signaux Rx Tx sur la carte et les envoyer sur un adaptateur série-USB/OTG.
Il y a un grand choix d' émulateur série pour Android, Apple, etc

didel.com/USBtoSerial.pdf



Concepts importants

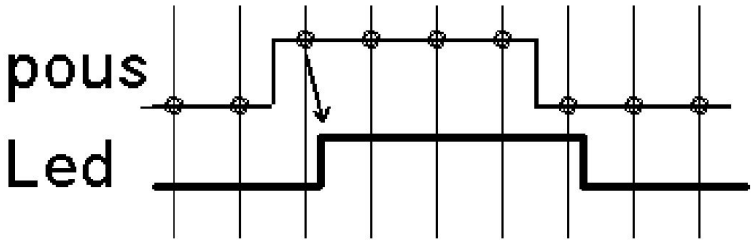
Suivi des actions

- **Bloquant: on attend l'action**
- **Non bloquant: on teste si l'action est faite**

Gestion de l'application, organisation du travail

- **Balayage (synchrone)**
- **Interruptions**

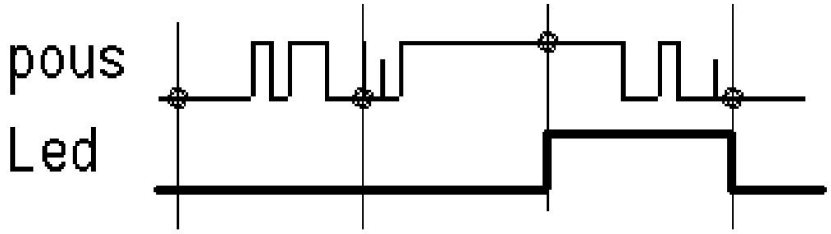
lire un signal




```
if (pous) {LedOn;}
else      {LedOff;}
```

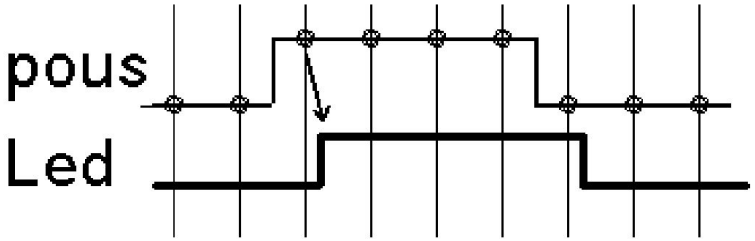
```
. . .
void setup
. . .
void loop () {
if (pous) {LedOn;}   // échantillonne toutes les ~1us
  else {LedOff;}
}
```

S'il y a des rebonds, il suffit d'échantillonner toutes les 20ms




```
delay(20);
if (pous)  { LedOn; }
else       { LedOff ; }
```

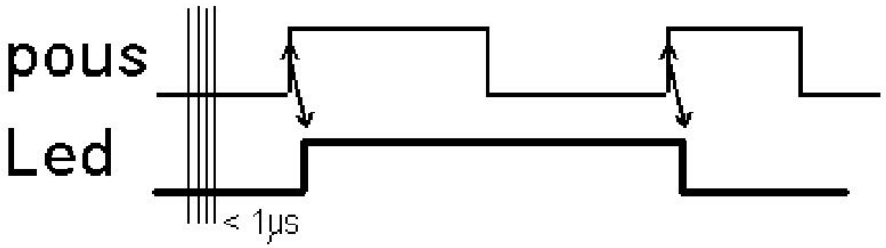
lire un signal



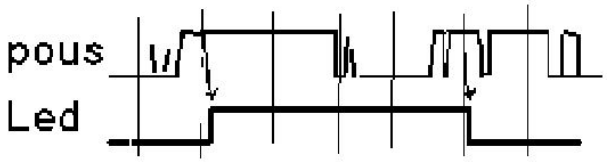

```
if (pous) {LedOn;}
else      {LedOff;}
```

```
#define pinP 2
#define pous digitalRead(pinP)
#define pinL 13
#define LedOn digitalWrite (pinL,HIGH)
#define LedOff digitalWrite (pinL,LOW)
#define LedToggle digitalWrite (pinL,!digitalRead(Pled))
void setup () {
   pinMode (pinP,INPUT_PULLUP);
   pinMode (pinL,OUTPUT);
}
```

attendre une transition (bloquant)

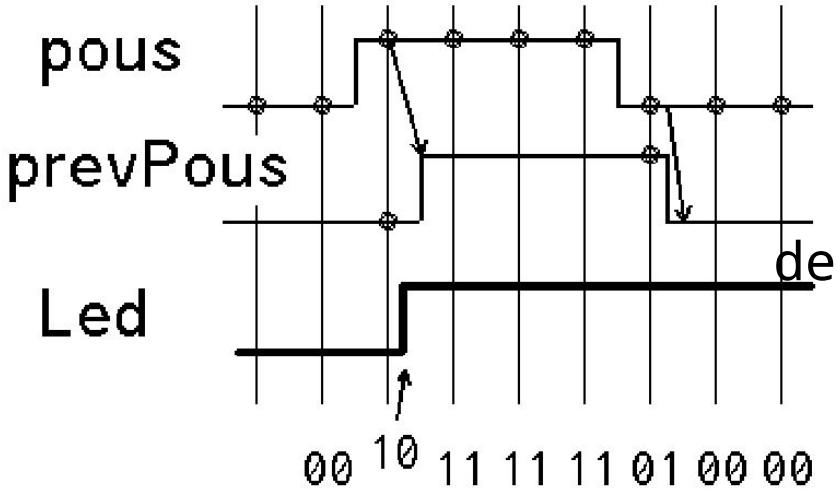



```
while (!pous) {}
LedToggle;
while (pous) {}
```




```
while (!pous) {delay(20);}
LedToggle;
while (pous) {delay(20);}
```

attendre une transition (non bloquant)




```
defin
  if (pous && (!prevPous))
    { LedToggle; }
  prevPous = pous;
  ...
  delay(20);
```

Entre deux test du poussoir, on a presque 20 ms
Pour faire autre chose

Extrêmes:

Faire les choses selon une séquence prédéfinies, même si souvent il n’y a rien à faire pour la tâche testée.

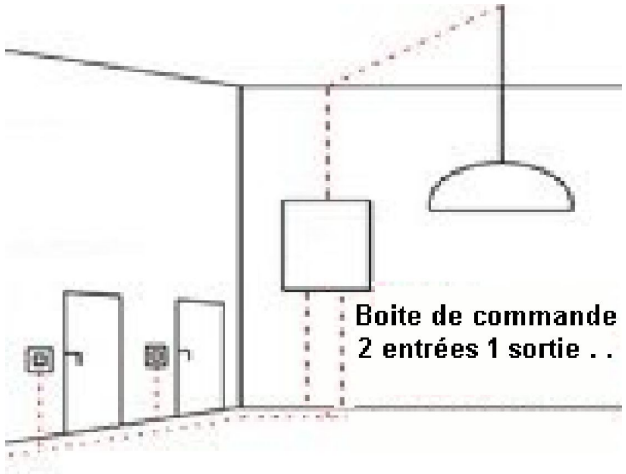
Attendre d’être interrompu pour faire ce qu’il faut.

```
//minuterie d'escalier simple
#include « Def.h"
void setup () { SetupDef(); }

#define TempsOn 5*60*1000  // 5minutes de 60 seconde
                           // 1 seconde = 1000ms

void loop() {
   while (!PousG) {}
   LampeOn;      // inutile d'attendre si relâché
   delay(TempsOn);   // pour tester mettre DelMs(1000);
   LampeOff;
}
```

```
//Minuterie d'escalier avec redémarrage
#include « Def.h"
void setup () { SetupEduC(); }

#define TempsOn 5*60*10  // cycles de 100ms
int cnt;
void loop() {
   while (!PousG)
   LampeOn; cnt = 0;
   while (cnt++ < TempsOn) {
     DelMs(100);
     if (PousG) {cnt=0;}  // on réinitialise
   }
   LampeOff;
}
```

Questions

```
//DeuxPousUneLampe.ino
#include "EduC.h"
#include "Oled.h"
void setup(){  SetupEduC(); SetupOled(); }

byte AnyOn () {
  return ( PousG||PousD);
}
byte BothOff () {
  return (!PousG&&!PousD);
}

void loop() {
    while (BothOff()) {delay (20);}  // attend que qqn presse
    LedGToggle;
    while (AnyOn()) {delay (20);}  // attend relâchement
}
```



Le "switch case"

```
if(cond==4) {
    RougeOn;
} else if(cond==5) {
    VertOn;
} else if (cond==8){
    BleuOn;
} else {
    Erreur(3);
}
```

```
switch (cond) {
    case 4:  RougeOn; break;
    case 5:  VertOn;  break;
    case 8:  BleuOn;  break;
    default: Erreur(3); //
autre cas
}
```

```
switch (etat) {
   . . .
    case 4:
       DoCeci();
       if (cela) {x = 43; etat=7;}
       if (truc) {DoCela(tt,vv); etat=2;}
       break;
    case 5:
         . . .
        break;
    default: Erreur(3); // autre cas
}
```

Aucun cas ne doit être bloquant. Le switch est appelé tous les 20ms par exemple.

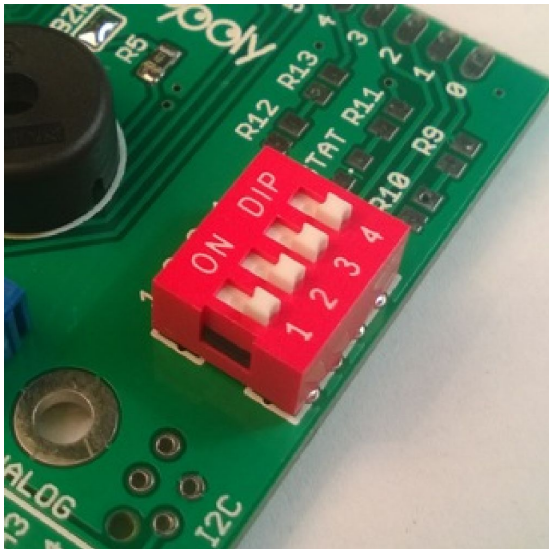
A chaque passage, on définit ce qu'il faut faire au prochain passage par le switch: même cas ou un nouveau décidé dans le cas appelé.

-Compter le nombre de passages et aller ailleurs quand la limite est atteinte

-Tester les capteurs et décider où continuer au prochain passage

-Prévoir les conditions anormales et les signaler (clignotement, etc) pour en tenir compte plus tard.

```
// Fichier inclus Prisme.h
//#include "Prisme.h" //SetupPrisme(); dans le PP
   #define PinDip1 2 // PD2
   #define PinDip2 3 // PD3
#define Dip1On !digitalRead(!PinDip1)
#define Dip2On !digitalRead(!PinDip2)

byte EtatDip() { // etat selon pos sw
   return ((Dip2<<1)+Dip1);
   // Dip2 Dip1 = 00 01 10 11
}
void SetupPrisme () {
   pinMode (Dip1,0);
   pinMode (Dip2,0);
}
```

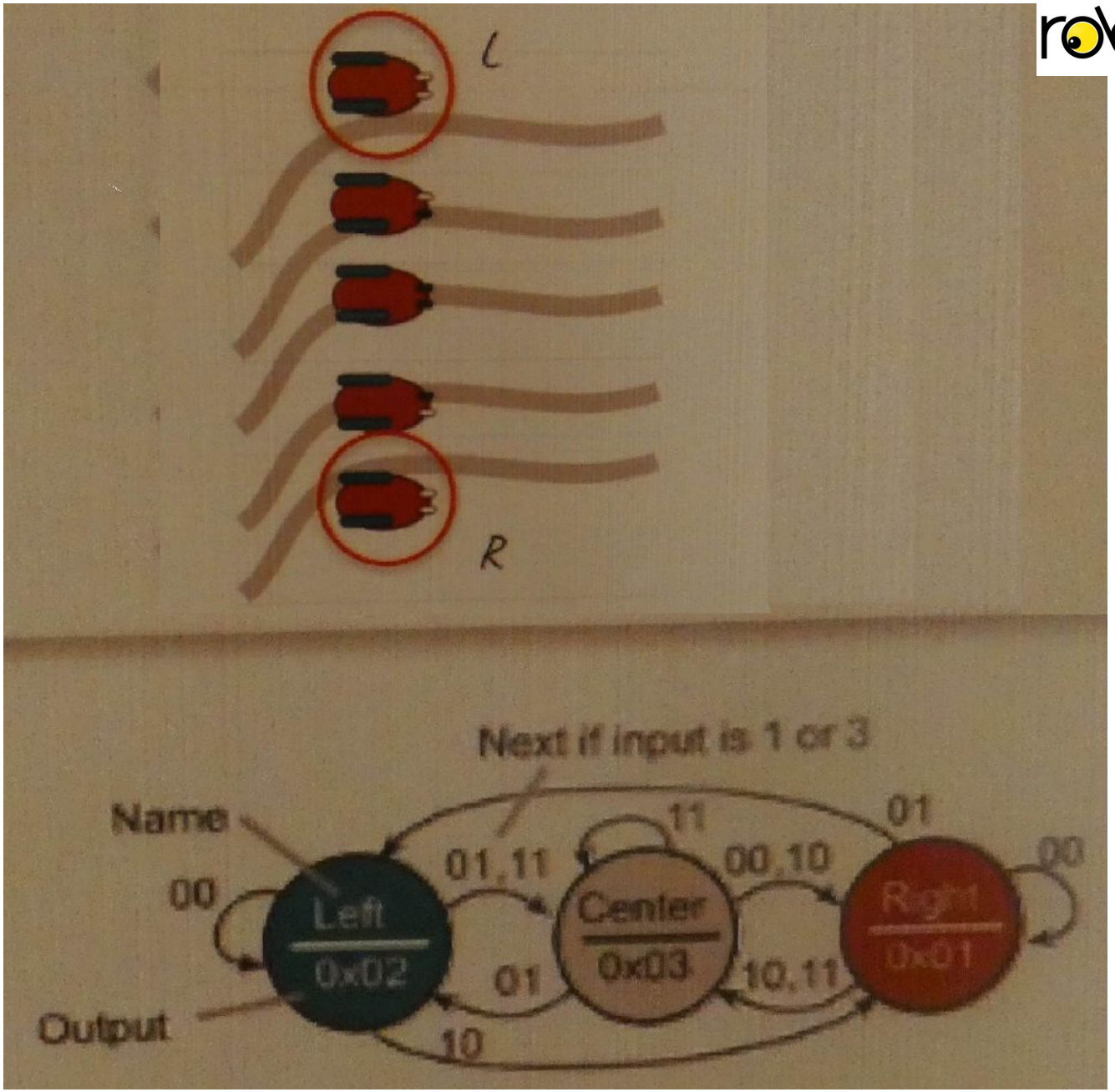


Autres définitions (pous, Led) à ajouter
On peut voir 4 contacts séparés testés avec if (sw0) {} etc

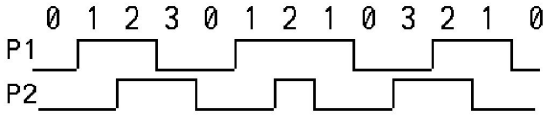
```
//TestDipswitch.ino
#include "Prisme.h" //definit Dip1 Dip2
void setup(){
  SetupPrisme();
  Serial.begin(9600);
}
byte posDip;
int val;
void loop() {
  delay(50);  // supprime rebonds
  posDip = EtatDip();
      // etat logique lu. Inverser év selon indications 0-1
  switch (posDip) {
    case 0:  // case 0b00: plus clair
        Serial.println("00");
      break;
    case 1:   // case 0b01:
      ...
      break;
    case 2:   // case 0b10:
     ...
      break;
    case 3:   // case 0b11:
     ...
      break;
    default:
      // clignotement spécial
      break;
   } // fin switch
} //fin loop
```

```
//Le robot doit éviter les obstacles
#include «Prisme.h"  // définit ObsG; Avance();, etc.
void setup () { SetupPrisme();}

enum {ObsNo,ObsG,ObsD,ObsDev} etatObs;
//     0    G 1  D 2  D+G 3
// Si on ajoute un état, tout se renumérote
void loop() {
  delay(50);
  etatObs=0;  // on doit coder 4 états 00 01 10 11
  if (ObsG()) {etatObs +=1;}
  if (ObsD()) {etatObs +=2;}
  switch (etatObs) {
    case ObsNo:  Avance(); break;
    case ObsD:   TourneG(); break;
    case ObsG:   TourneD(); break;
    case ObsDev: Stop(); break;
  } // fin switch
}
```

EtatMous n’est pas une variable (nombres entieurs seulement)


L
R
Next if input is 1 or 3
Name
00
Left
0x02
01,11
11
Center
0x03
00,10
01
Right
0x01
00
01
10,11
10
Output

Encodeur



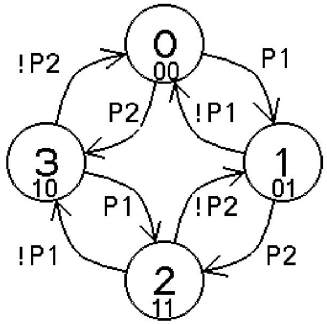
In state 0 for example, there are 3 possibilities because one should not have P1 and P2 which are activated simultaneously :

P1 = 1 we go to state 1 and count

P2 = 1 we go to state 3 and decount

P1 = 0 and P2 = 0 one remain in state 0

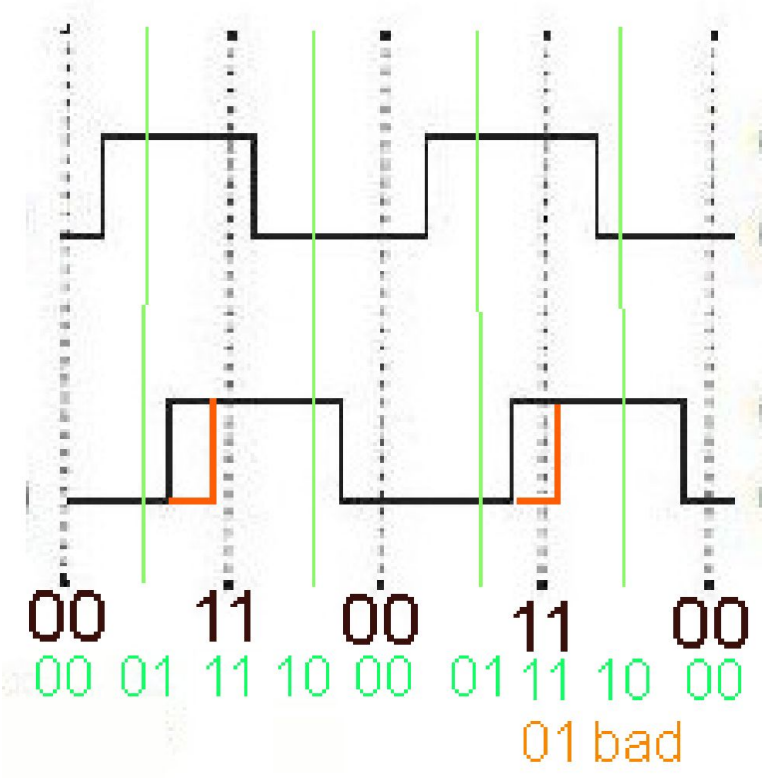
P1 = 1 and P2 = 1 erreur de timing




```
   . . .
 delay (10);// sample delay (debounce)
  switch (etat) {
   case 0 : // P1=0 P2=0
    if (digitalRead (P1)) { etat = 1 ; cnt++ ; }
    if (digitalRead (P2)) { etat = 3 ; cnt-- ; }
    break;
   case 1 : // P1=1 P2=0
    if (!digitalRead (P1)) { etat = 0 ; cnt-- ; }
    if (digitalRead  (P2)) { etat = 2 ; cnt++ ; }
    break;
  . . . .
```

La période d'échantillonnage doit être inférieure au temps minimum entre 2 transitions.

Les encodeur manuels on des rebonds de contact important; la période d'échantillonnage doit être supérieure à la durée des rebonds.
(En fait les rebonds font compter/décompter, mais les niveaux logiques sont parfois ambigu).



(Pour information, éventuellement utile)

RomeEnco -- Relation between encoder, wheel diameter and distance

The encoder software gives 12 pulses per turn
The geartrain has 3 possible factors :

Original	8 – 36 / 9 – 37 / 9 – 37 / 8 – 24	factor **1 : 228**
Option 1	8 – 36 / 16 – 30 / 9 – 37 / 8 – 24	factor **1 :104**
Option 2	8 – 36 / 24 – 22 / 9 – 37 / 8 – 24	factor **1 :51**

1:228 means 12 x 228 = **2736 pulses** per shaft turn
If the wheel is 50mm dia, 157mm circumference, this means 0.0578 mm per pulse
If you want 0.05 mm per pulse (20 pulses per mm) use a wheel dia of 43.5mm
If you want 0.0625 mm per pulse (16 pulses per mm) use a wheel dia of 54.46mm

1:104 means 12 x 104 = **1248 pulses** per shaft turn
If the wheel is 50mm dia, 157mm circumference, this means 0.126 mm per pulse
If you want **0.125 mm** per pulse (8 pulses per mm) use a wheel dia of **49.6mm**
note : for a processor, the best is to divide by 2, 4, 8, 16..

1:51 means 12 x 51 = **612 pulses** per shaft turn
If the wheel is 50mm dia, 157mm circumference, this means 0.256 mm per pulse
If you want **0.25 mm** per pulse (4 pulses per mm) use a wheel dia of **48.70mm**

Pr. Jean-Daniel Nicoud