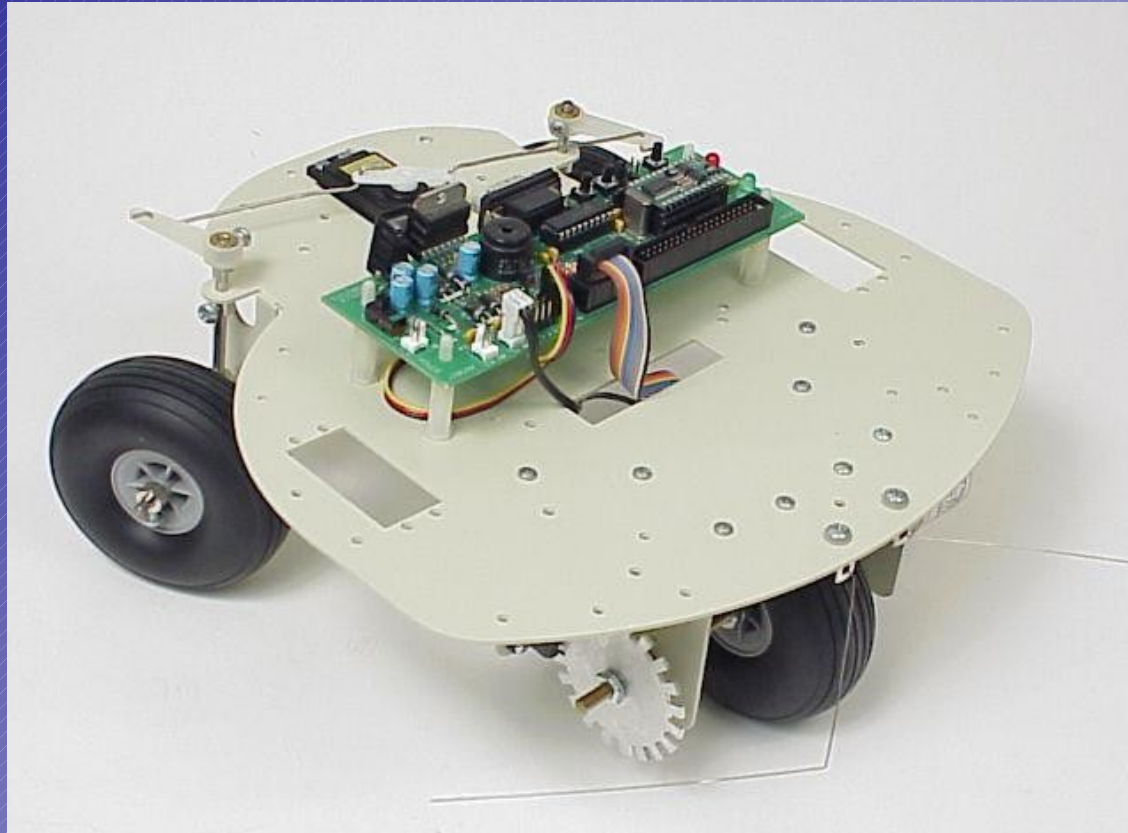


# Robot explorateur



Source : <http://www.robotshop.ca/robot-mobile-arobot-arrick-bs2-2.html>

**Hugo KARPINSKI – Sébastien SIROT**

Promotion 2008-2010

IUT de Tours – Département Génie Electrique et Informatique Industrielle

# Plan

- Présentation générale du projet
  1. Cahier des charges
  2. Structure
  3. Coût du projet
  4. Planning prévisionnel et planning réel
- Présentation du PICKIT 3
- Les différentes cartes et la programmation en C

# Présentation générale du projet

## 1. Cahier des charges

**But :** Fabrication d'un petit robot autonome

**Matériel à notre disposition au début du projet :**

- 2 moteurs Mabuchi
- 1 programmeur PIC et sa carte d'essai équipée d'un PIC18F45K20
- Des photo-résistances, des DELs, ...
- Des transistors, des résistances et des résistances variables, ...
- Des coupleurs de piles

## 2. Structure du système

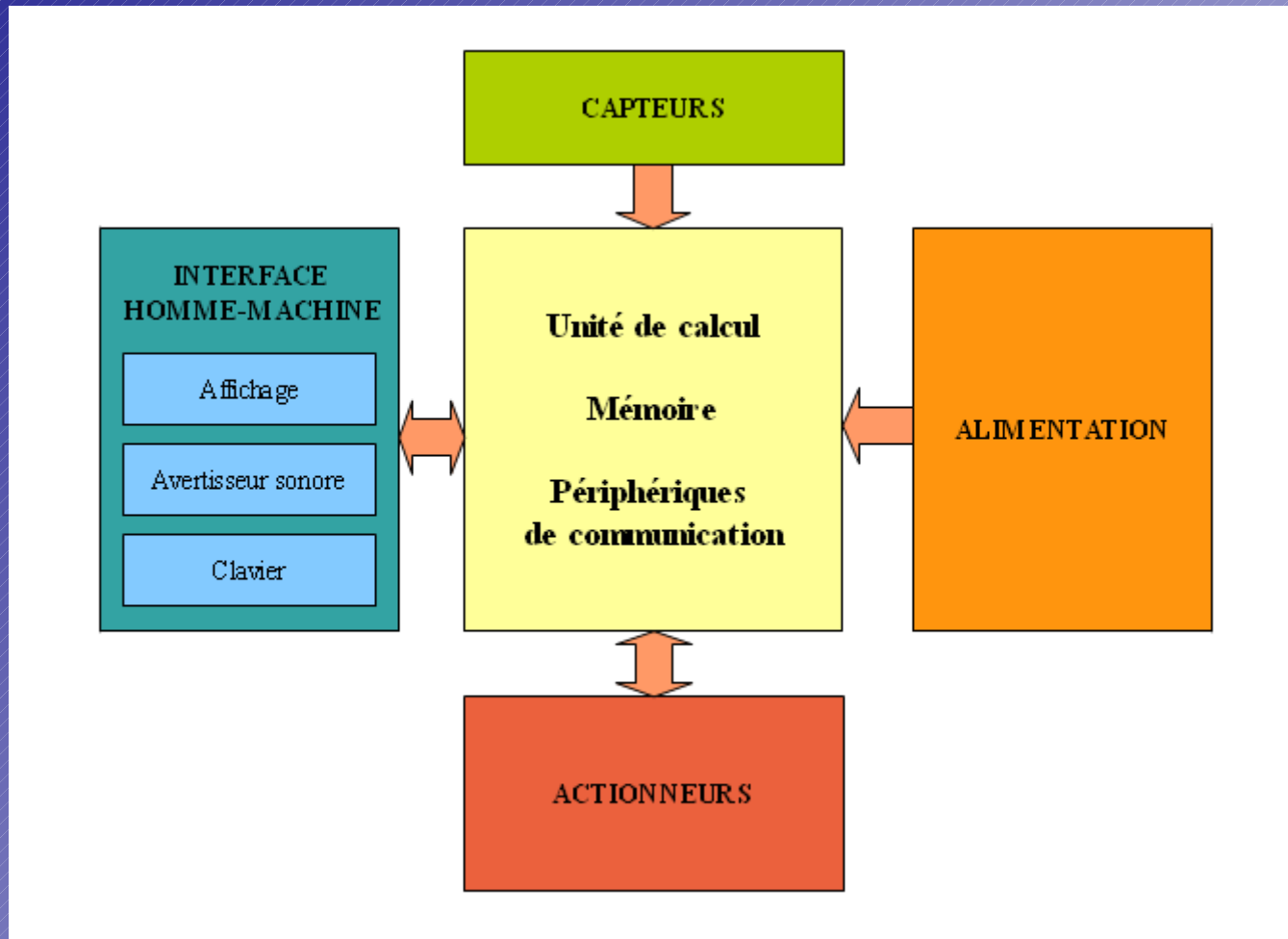


Schéma général du projet

### 3. Coût du projet

Désignation	Prix unitaire TTC	Quantité	Prix TTC
LED blanches	0,2	5	1
LED rouge	0,1	2	0,1
LED vertes	0,2	1	0,2
Résistance ¼ W	0,04	25	1
Mini bouton poussoir métal	0,4	6	2,4
Bouton poussoir rouge/noir	0,1	2	0,2
Résistances variables	0,1	5	0,5
AOP LM324	0,2	4	0,8
Régulateur LM 317	0,4	3	1,2
LCD 2x16	4	1	4
Moteur Mabuchi	4	2	8
Radiateur	0,4	3	1,2
Pont en H TLE-5206S	4,5	2	9
Support 40 pins	0,3	1	0,3

+ PIC 18F46K20  
 + 2 transistors  
 + 7 photo-résistances  
 + ...

Budget total :



**environ 40 euros**

(sans les frais de port, le prix des câbles et des plaques d'époxy)

# 4. Planning prévisionnel et planning réel

N° de semaine	3	4	5	6	7	8	9	10	11	12	13
Définition du projet : cahiers des charges et planning	Prévisionnel			Réel	Réel						
	Réel			Prévisionnel	Prévisionnel						
Conception de la partie électronique				Prévisionnel	Prévisionnel	Réel	Réel		Réel	Réel	Réel
			Réel	Réel	Réel						
Étude du micro-contrôleur PIC et du PICKIT 3			Réel	Réel	Réel	Réel	Réel	Réel	Réel	Réel	
		Réel	Réel	Prévisionnel	Prévisionnel						
Programmation du micro-contrôleur				Prévisionnel	Prévisionnel			Réel	Réel	Réel	Réel
				Prévisionnel	Prévisionnel						
Assemblage				Prévisionnel	Prévisionnel					Réel	
				Prévisionnel	Prévisionnel						
Rédaction du rapport		Réel	Réel	Réel	Réel					Réel	Réel
		Réel	Réel	Prévisionnel	Prévisionnel	Réel	Réel	Réel	Réel	Réel	
Remise du rapport				Prévisionnel	Prévisionnel						Réel
				Prévisionnel	Prévisionnel						Réel

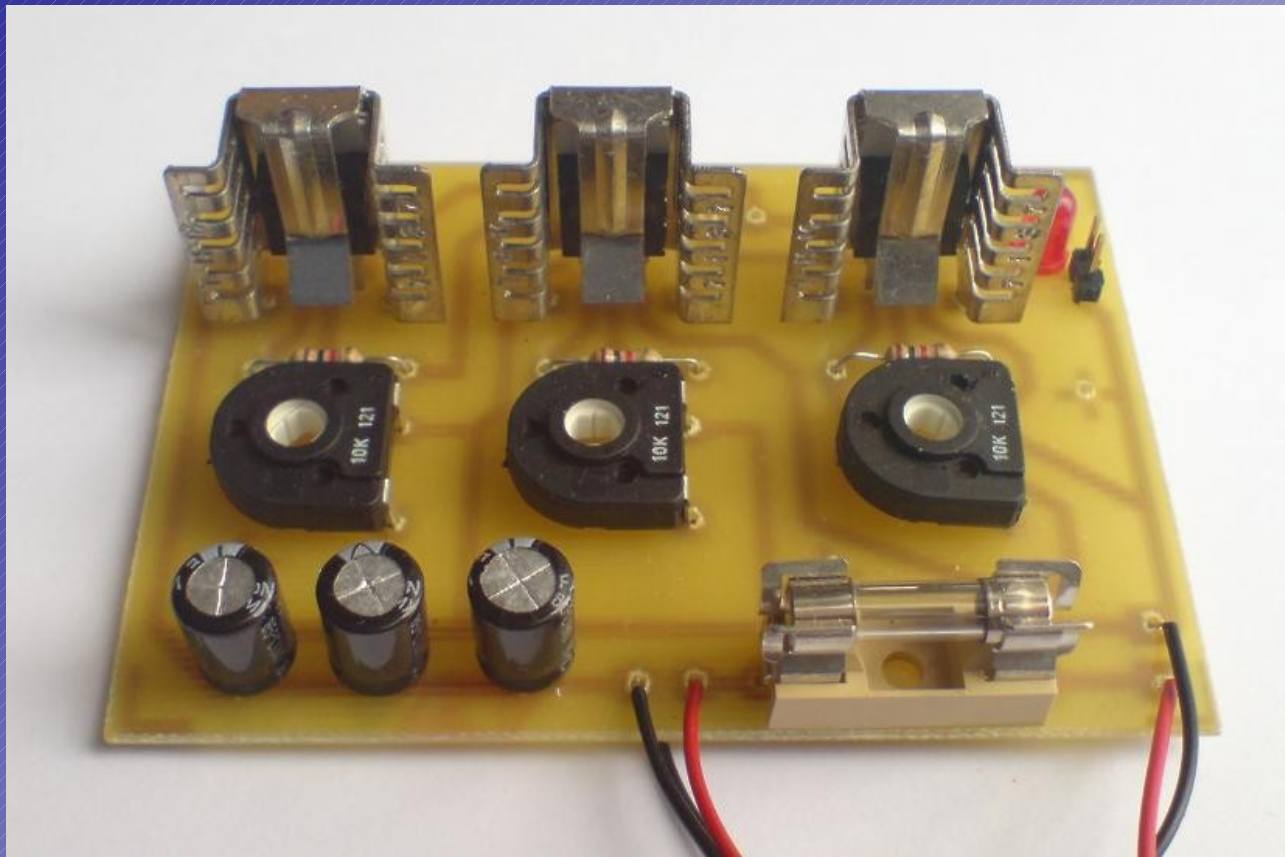
	Prévisionnel
	Réel

# Présentation du PICkit 3



Source : [http://www.napier.co.uk/client\\_news.php?nid=761](http://www.napier.co.uk/client_news.php?nid=761)

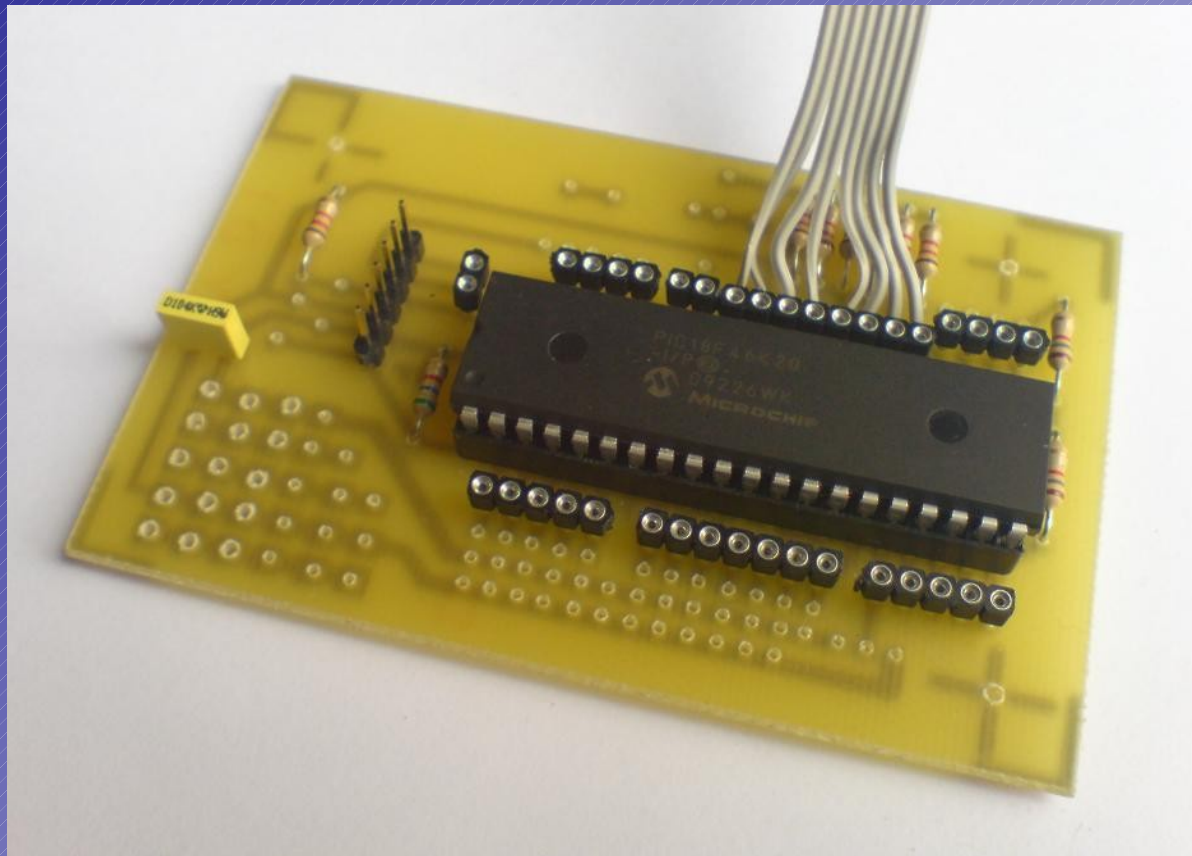
# Les différentes cartes



Carte d'alimentation

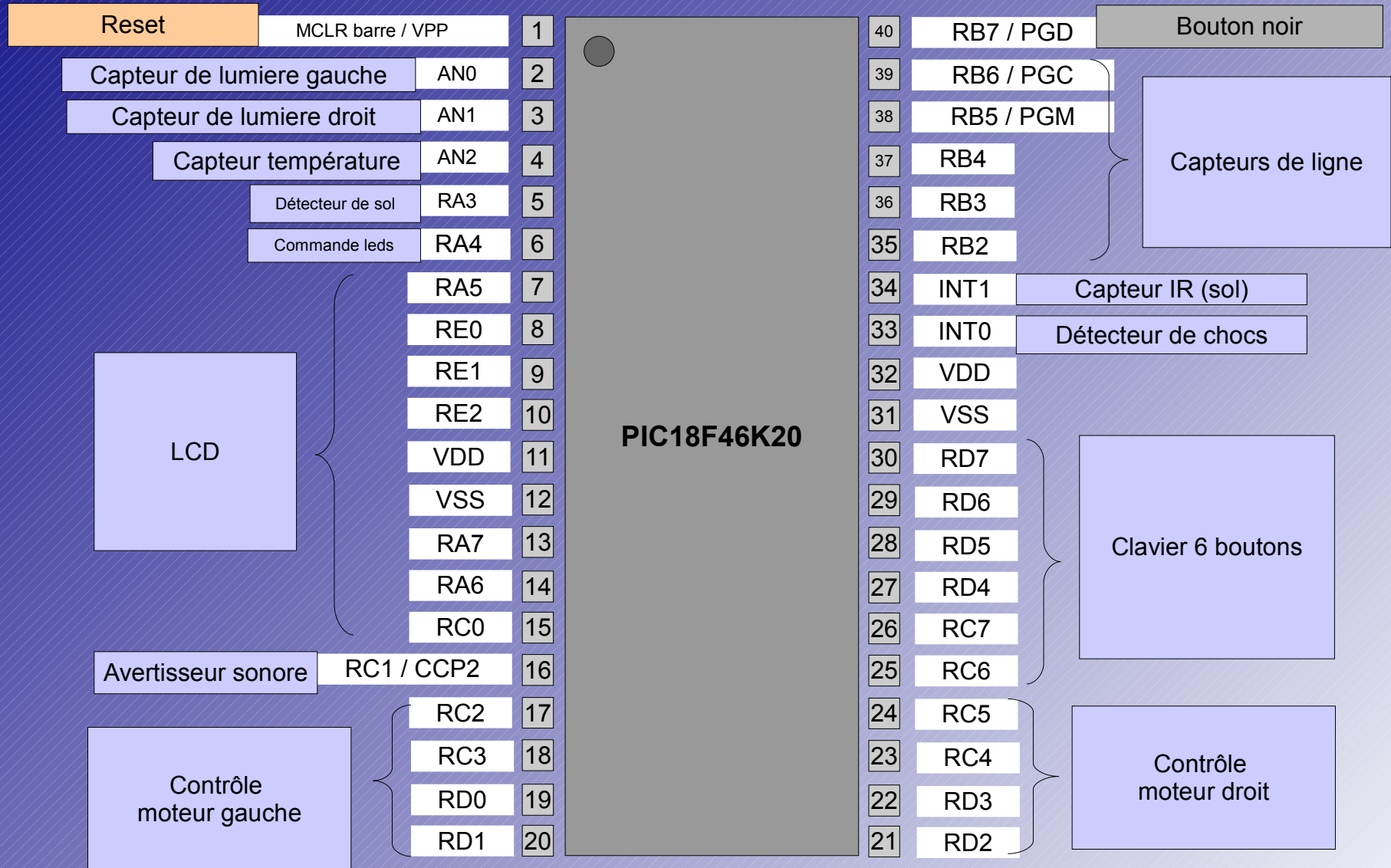


# Les différentes cartes

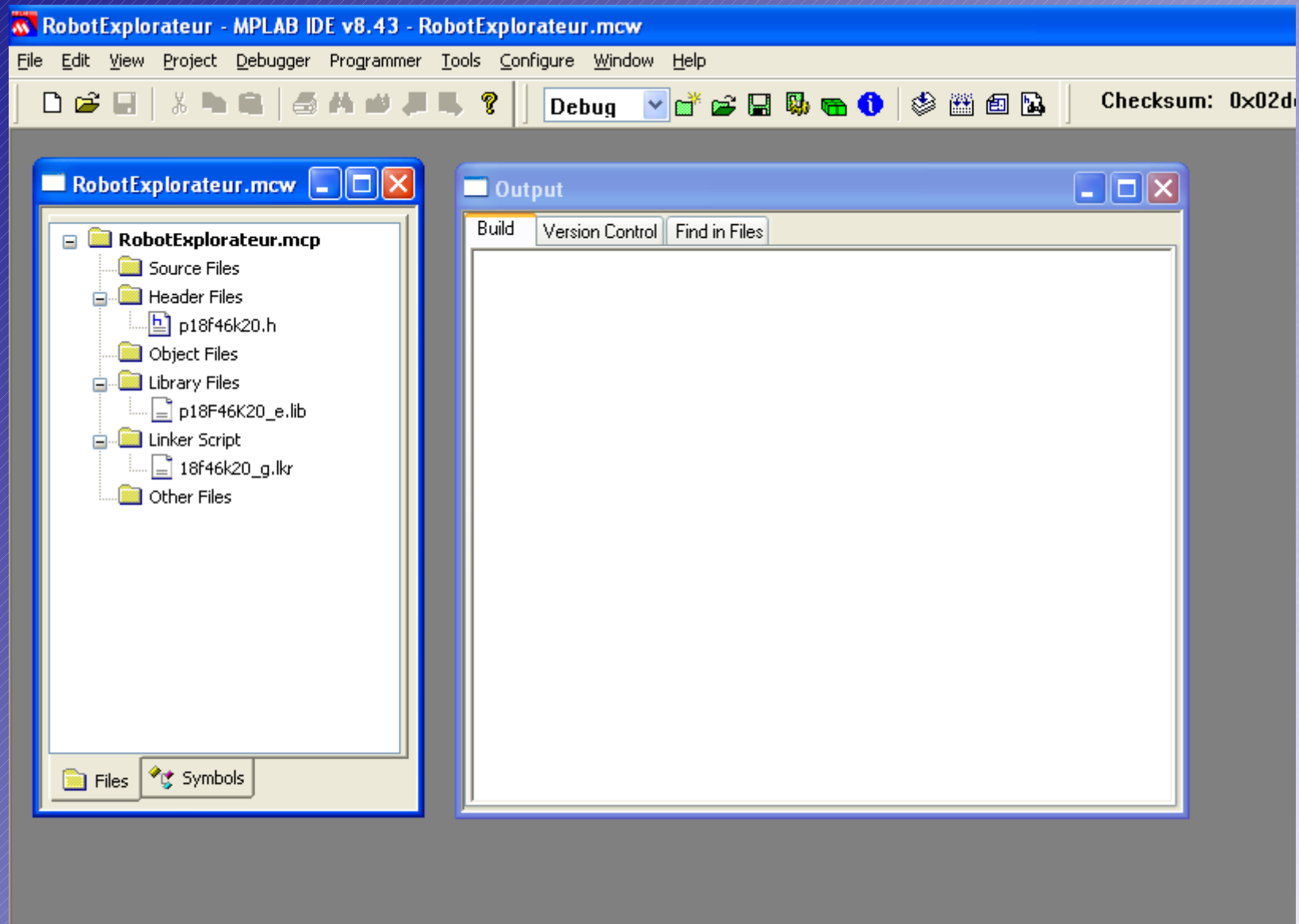


Carte micro-contrôleur

# Avant de débuter la programmation...



# MPLAB IDE



# Activation des périphériques

```
// Nous choisissons l'oscillateur interne comme horloge
// et nous définissons RA6 et RA7 comme des ports I/O.
// Ceux-ci sont utilisés pour le contrôle du LCD.
#pragma config FOSC = INTIO67

// Le RESET externe est mis à disposition (RE3 inutilisable)
// Ainsi, on pourra forcer un redémarrage du programme, en appuyant
// sur un bouton rouge par exemple.
#pragma config MCLRE = ON

// Nous désactivons la fonction "entrée analogique" sur les bits PORTB<4:0>
#pragma config PBAEN = OFF
|
// Nous multiplexons le signal MLI de CCP2 avec la broche RC1
// (et non avec la broche RB2). La broche RC1 servira peut-être de
// commande au module "Avertisseur sonore", car il est très facile
// de faire varier la fréquence de CCP2 via un registre. Toutefois,
// peut-être utiliserons nous une autre méthode pour obtenir
// un signal de fréquence variable sur la sortie RC1...
#pragma config CCP2MX = PORTC

// Nous désactivons la plupart des périphériques avancés
#pragma config FCMEN = OFF, IESO = OFF, PWRT = OFF, BOREN = OFF
#pragma config HFOFST = OFF, LPT1OSC = OFF, WDTCN = OFF

// Nous désactivons la protection du code et permettons l'accès à l'EEPROM
#pragma config CPO = OFF, CP1 = OFF, CP2 = OFF, CP3 = OFF
#pragma config CPB = OFF, CPD = OFF
#pragma config WRTO = OFF, WRT1 = OFF, WRT2 = OFF, WRT3 = OFF
#pragma config WRTE = OFF, WRTC = OFF, WRTD = OFF
#pragma config EBTR0 = OFF, EBTR1 = OFF, EBTR2 = OFF, EBTR3 = OFF, EBTRB = OFF
```

# Déclaration de symboles dans différents fichiers

```
// Capteurs de lumière et de température
#define CAPT_LUM_G  LATAbits.LATA0
#define CAPT_LUM_D  LATAbits.LATA1
#define CAPT_TEMP   LATAbits.LATA2

// Détecteur IR - Couple émetteur et récepteur infrarouge IR
#define EMET_IR     LATAbits.LATA3
#define RECEPT_IR LATBbits.LATB1

// Commande des leds blanches du détecteur de ligne
#define CMD_LEDS    LATAbits.LATA4

// Les 5 détecteurs du détecteur de ligne
#define DETECT_LB1  LATBbits.LATB2
#define DETECT_LB2  LATBbits.LATB3
#define DETECT_LB3  LATBbits.LATB4
#define DETECT_LB4  LATBbits.LATB5
#define DETECT_LB5  LATBbits.LATB6

// Le détecteurs de chocs
#define DETECT_CHOC LATBbits.LATB0
```

## Utilisation de *#define*

```
// Moteur droit
#define CMD1_MOT_D  LATDbits.LATD3 // Commande n°1 moteur droit CMD1_MOT_D
#define CMD2_MOT_D  LATCbits.LATC4 // Commande n°2 moteur droit CMD2_MOT_D
#define COMPT_MOT_D LATCbits.LATC5 // Compteur moteur droit   COMPT_MOT_D
#define ERR_MOT_D   LATDbits.LATD2 // Erreur moteur droit   ERR_MOT_D

// Moteur gauche
#define CMD1_MOT_G  LATCbits.LATC3 // Commande n°1 moteur gauche CMD1_MOT_G
#define CMD2_MOT_G  LATDbits.LATD0 // Commande n°2 moteur gauche CMD2_MOT_G
#define COMPT_MOT_G LATCbits.LATC2 // Compteur moteur gauche   COMPT_MOT_G
#define ERR_MOT_G   LATDbits.LATD1 // Erreur moteur gauche   ERR_MOT_G
```

```
#define CMD_SON LATCbits.LATC1
```

```
#define LCD_RS  LATAbits.LATA5
#define LCD_RW  LATEbits.LATE0
#define LCD_E   LATEbits.LATE1
#define LCD_DB4 LATEbits.LATE2
#define LCD_DB5 LATAbits.LATA7
#define LCD_DB6 LATAbits.LATA6
#define LCD_DB7 LATCbits.LATC0
```

```

/*****
*****
REGLAGE DES ENTREES NUMERIQUES
*****
*****/
TRISBbits.TRISB7=1; // Bouton noir
TRISCbits.TRISC6=1; // Bouton 1 BT1
TRISCbits.TRISC7=1; // Bouton 2 BT2
TRISDbits.TRISD4=1; // Bouton 3 BT3
TRISDbits.TRISD5=1; // Bouton 4 BT4
TRISDbits.TRISD6=1; // Bouton 5 BT5
TRISDbits.TRISD7=1; // Bouton 6 BT6
TRISBbits.TRISB0=1; // Détecteur de choc DETECT_CHOC
TRISBbits.TRISB1=1; // Détecteur de sol (Récepteur) RECEPT_IR
TRISBbits.TRISB2=1; // Détecteur n°1 Ligne Blanche DETECT_LB1
TRISBbits.TRISB3=1; // Détecteur n°2 Ligne Blanche DETECT_LB2
TRISBbits.TRISB4=1; // Détecteur n°3 Ligne Blanche DETECT_LB3
TRISBbits.TRISB5=1; // Détecteur n°4 Ligne Blanche DETECT_LB4
TRISBbits.TRISB6=1; // Détecteur n°5 Ligne Blanche DETECT_LB5
TRISCbits.TRISC2=1; // Compteur moteur gauche COMPT_MOT_G
TRISDbits.TRISD1=1; // Erreur moteur gauche ERR_MOT_G
TRISCbits.TRISC5=1; // Compteur moteur droit COMPT_MOT_D
TRISDbits.TRISD2=1; // Erreur moteur droit ERR_MOT_D

```

```

/*****
*****
REGLAGE DES SORTIES NUMERIQUES
*****
*****/
TRISAbits.TRISA5=0; // LCD_RS
TRISEbits.TRISE0=0; // LCD_RW
TRISEbits.TRISE1=0; // LCD_E
TRISEbits.TRISE2=0; // LCD_DB4
TRISAbits.TRISA7=0; // LCD_DB5
TRISAbits.TRISA6=0; // LCD_DB6
TRISCbits.TRISCO=0; // LCD_DB7
TRISAbits.TRISA3=0; // Détecteur de sol (Emetteur) EMET_IR
TRISAbits.TRISA4=0; // CMD_LEDS
TRISCbits.TRISC1=0; // CMD_SON
TRISCbits.TRISC3=0; // CMD1_MOT_G
TRISDbits.TRISD0=0; // CMD2_MOT_G
TRISCbits.TRISC4=0; // CMD1_MOT_D
TRISDbits.TRISD3=0; // CMD2_MOT_D

```

```

/*****
*****
REGLAGE DES ENTREES ANALOGIQUES
*****
*****/
TRISAbits.TRISA0=1; //capteur lumière gauche CAPT_LUM_G
TRISAbits.TRISA1=1; //capteur lumière droite CAPT_LUM_D
TRISAbits.TRISA2=1; //capteur température CAPT_TEMP
/* NB : au démarrage/redémarrage, les broches PORTA<3:0>
sont des entrées analogiques par défaut. Il n'est donc
pas nécessaire de les configurer.*/

```

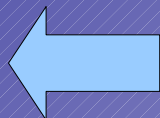
```

OSCCONbits.IRCF0=0; // Nous réglons les bits du Registre IRCF
OSCCONbits.IRCF1=1; // pour imposer une fréquence d'horloge
OSCCONbits.IRCF2=1; // de 16 MHz.
OSCTUNEbits.PLEN=0; // Nous désactivons la PLL (qui multiplierait
// sinon par 4 l'horloge interne)

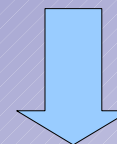
```

## Paramétrage des périphériques

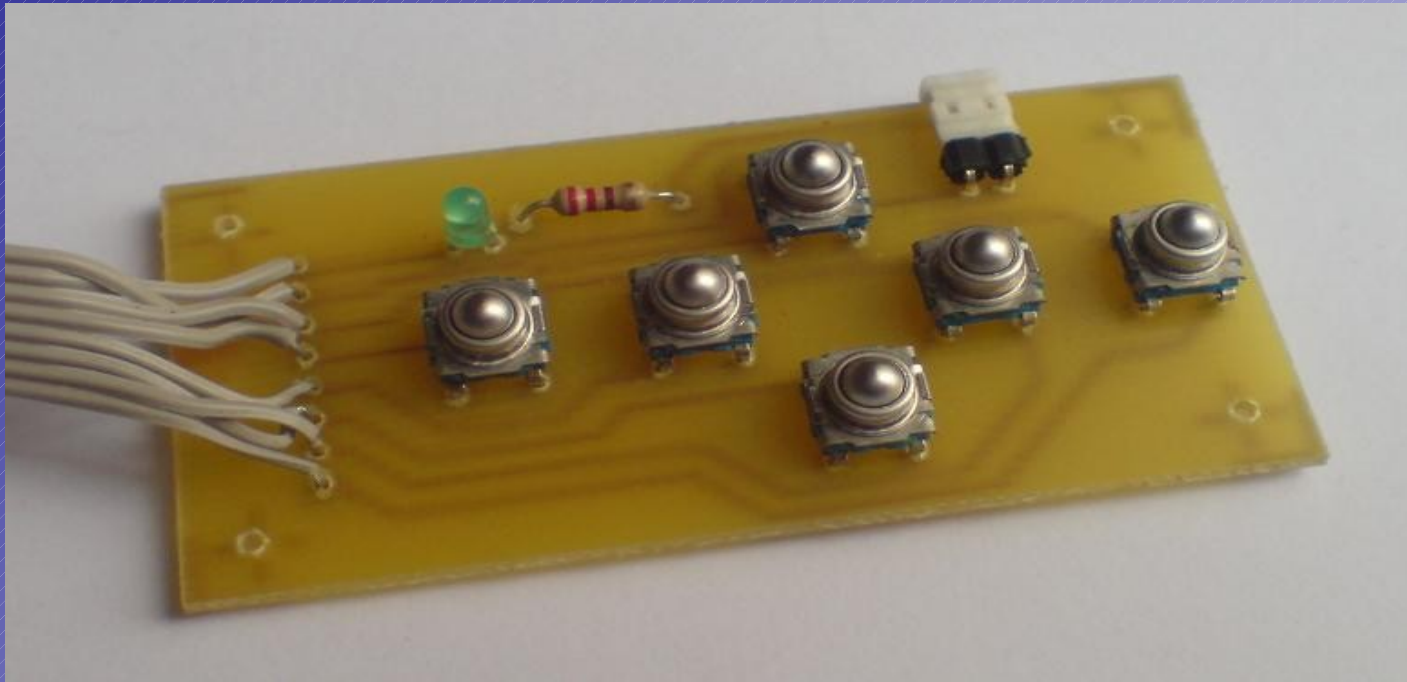
Des entrées/sorties...



...mais aussi de l'horloge interne



# Les différentes cartes



Carte 6 boutons

## Programmation liée aux boutons

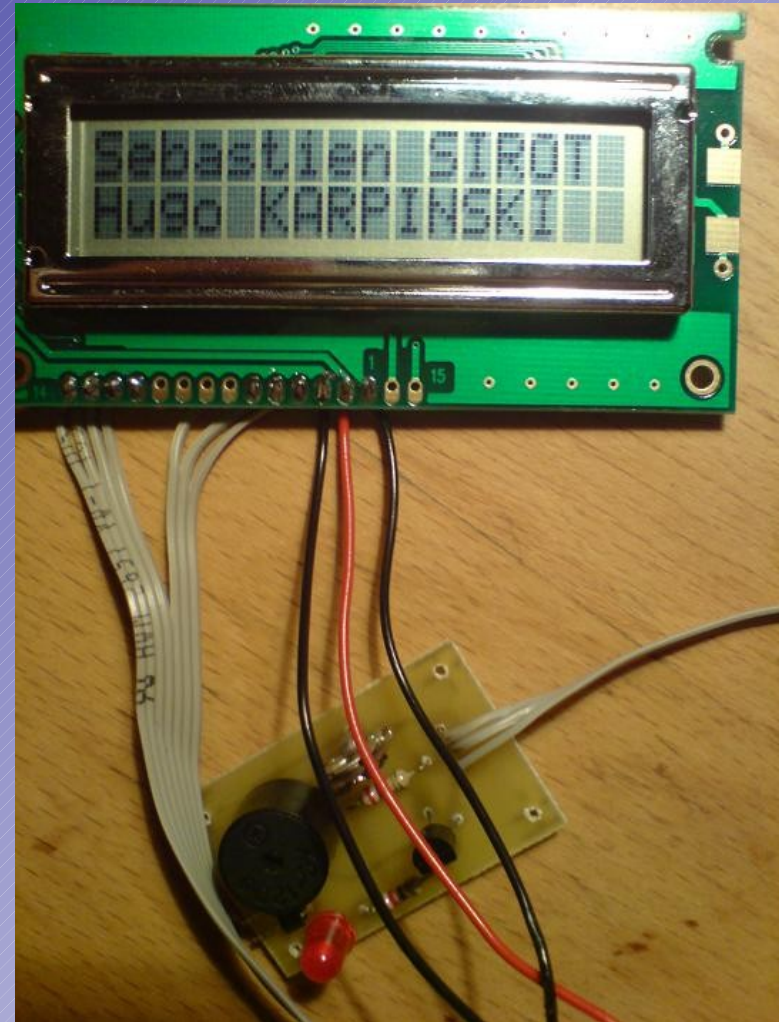
Une fonction simple (pas de prise en compte des rebonds, ni de l'appui sur plusieurs touches en même temps)

```
unsigned char BoutonPresse(void)
{
    char bt;
    bt=0;
    if ((PORTD&0x80)==0x80) bt=1;
    if ((PORTD&0x40)==0x40) bt=2;
    if ((PORTD&0x20)==0x20) bt=3;
    if ((PORTD&0x10)==0x10) bt=4;
    if ((PORTC&0x80)==0x80) bt=5;
    if ((PORTC&0x40)==0x40) bt=6;
    return bt;
}
```



# Les différentes cartes

- L'interface Homme-machine
  - Module à interfacer en mode 8 ou 4 bits, au choix
  - Espace mémoire pour des caractères spéciaux



L'afficheur LCD

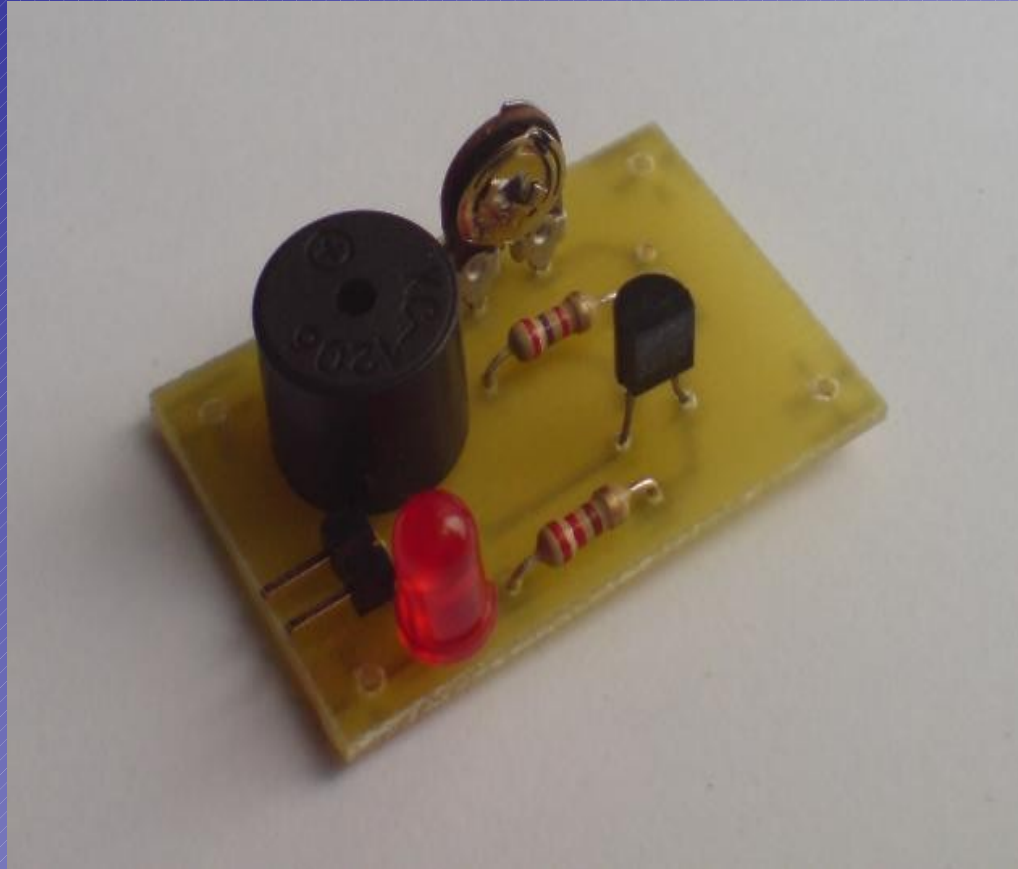
# Programmation du LCD

- Plusieurs mémoires
- 2 types d'instruction :  
« commande » et  
« écriture d'un caractère »
- Temps d'attente à respecter
- Différence entre les modes 8 et 4 bits

```
void InitialiserLCD(void)
{
    Delay10KTCYx(50);
    LCD_E=0;
    LCD_RS = 0;
    LCD_RW = 0;
    LCD_DB7 = 0;
    LCD_DB6 = 0;
    LCD_DB5 = 1;
    LCD_DB4 = 0;
    Delay10KTCYx(50);
    LCD_E=1;
    Delay10KTCYx(50);
    EnvoyerCommandeLCD(0,0,1,0,1,0,0,0);
    Delay10KTCYx(50);
    EnvoyerCommandeLCD(0,0,0,0,1,1,0,0); // curseur / blink
    Delay10KTCYx(50);
    EnvoyerCommandeLCD(0,0,0,0,0,0,0,1);
    Delay10KTCYx(50);
    EnvoyerCommandeLCD(0,0,0,0,0,0,1,1);
    Delay10KTCYx(50);
}
```

... d'où l'écriture d'une bibliothèque pour faciliter son utilisation dans notre projet

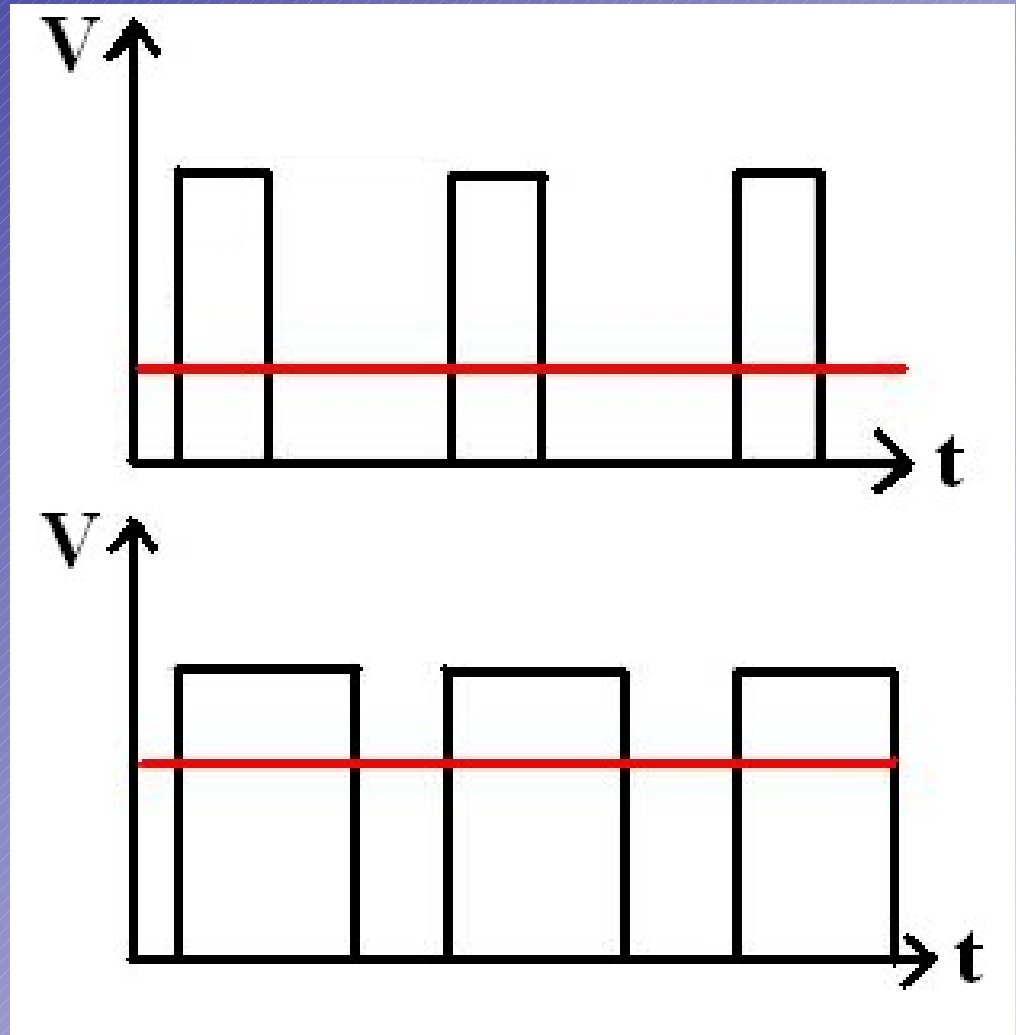
# Les différentes cartes



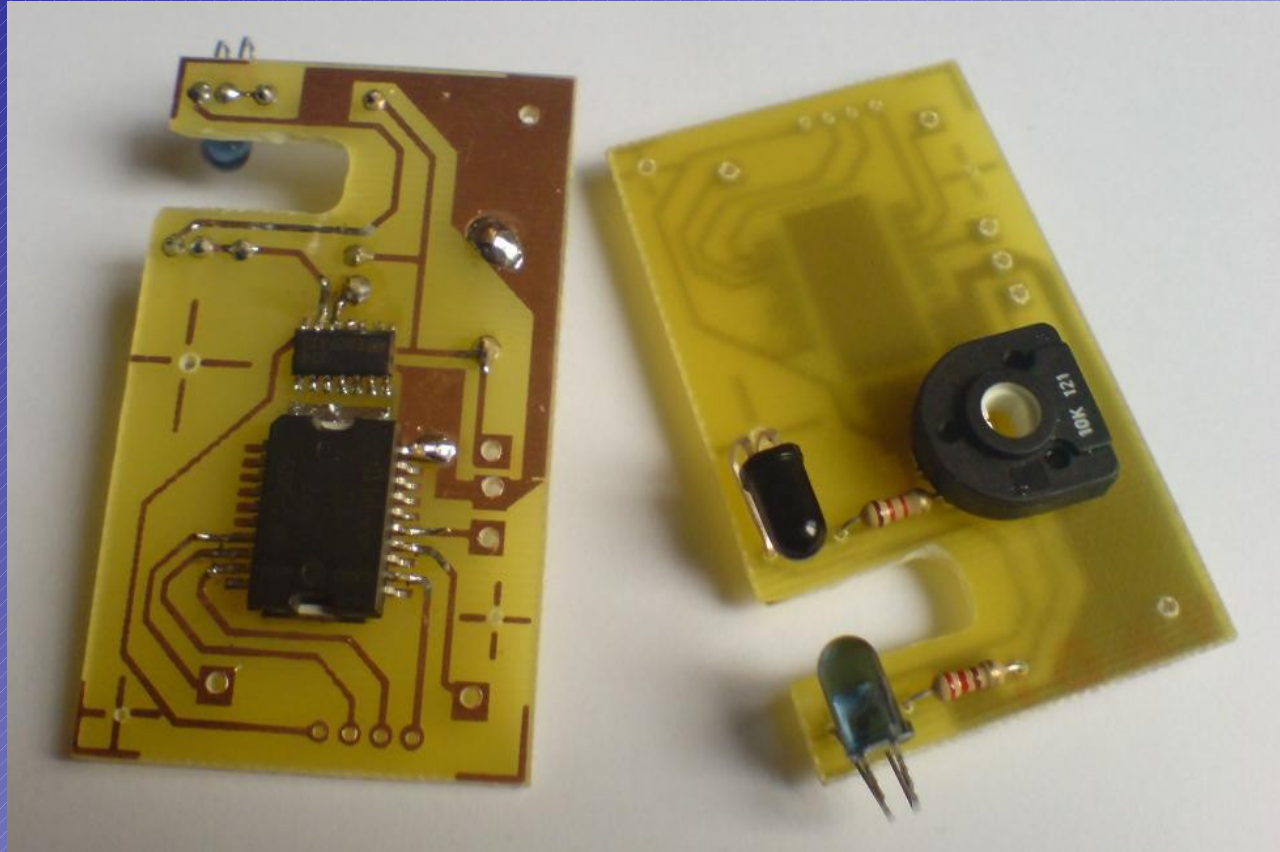
Avertisseur sonore et témoin lumineux

# Modulation

- Port avec PWM
- Plusieurs registres à configurer



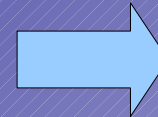
# Les différentes cartes



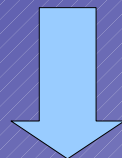
Carte contrôleur

# Programmation liée aux cartes moteurs DC

Contrôle des moteurs



Compteur de distance et de vitesse



```
void MettreAJourCompteurTotal (void)
{
    if ((COMPT_MOT_D==1) & etat==0)
    {
        compt=compt+1;
        etat=1;
    }
    else if ((COMPT_MOT_D==0) & etat==1) etat=0;
}
```

```
void Avancer (void)
{
    CMD1_MOT_G=1;
    CMD2_MOT_G=0;
    CMD1_MOT_D=1;
    CMD2_MOT_D=0;
}

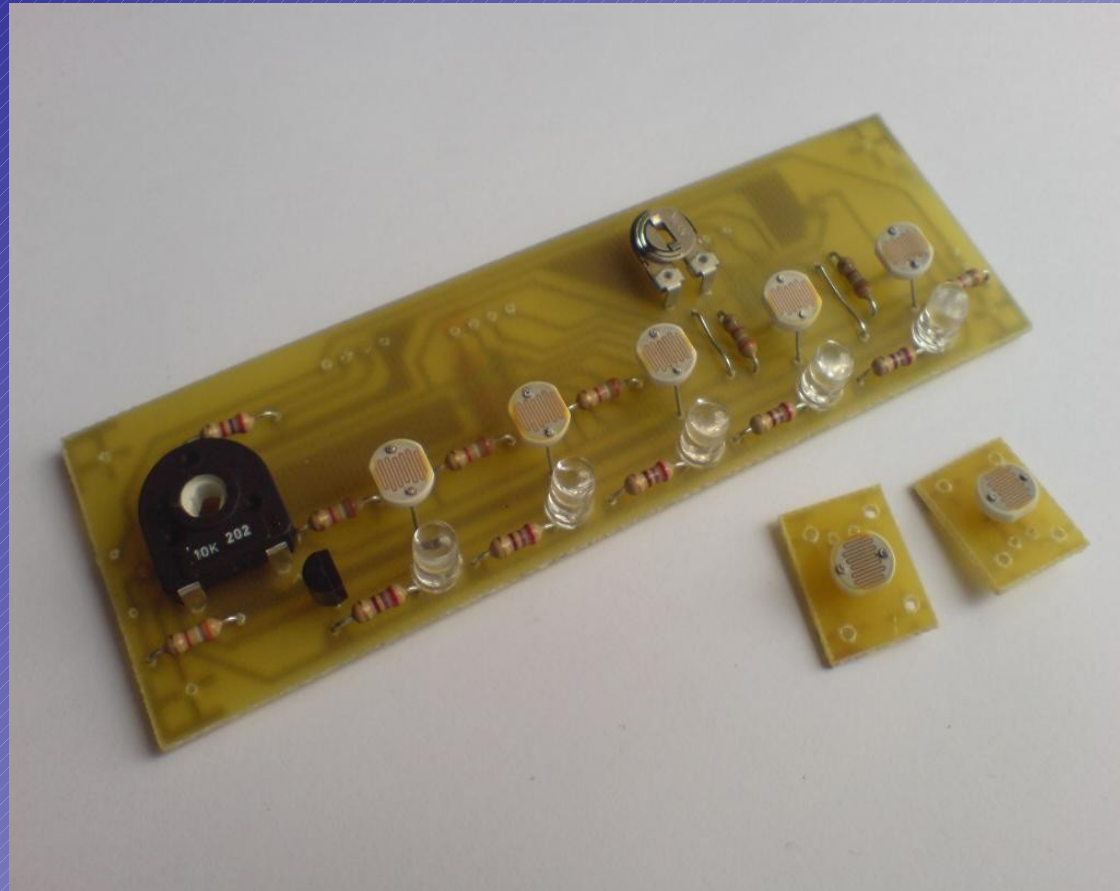
void Reculer (void)
{
    CMD1_MOT_G=0;
    CMD2_MOT_G=1;
    CMD1_MOT_D=0;
    CMD2_MOT_D=1;
}

void Stopper (void)
{
    CMD1_MOT_G=0;
    CMD2_MOT_G=0;
    CMD1_MOT_D=0;
    CMD2_MOT_D=0;
}

void TournerG (int angle)
{
    CMD1_MOT_G=0;
    CMD2_MOT_G=1;
    CMD1_MOT_D=1;
    CMD2_MOT_D=0;
    // Stopper();
}

void TournerD (int angle)
{
    CMD1_MOT_G=1;
    CMD2_MOT_G=0;
    CMD1_MOT_D=0;
    CMD2_MOT_D=1;
    // Stopper();
}
```

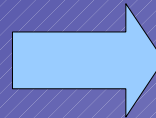
# Les différentes cartes



Carte capteur

# Programmation liée au capteur – Usage du CAN

Initialisation

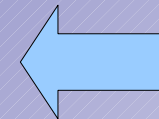


```
void InitialiserCAN(void)
{
    ANSEL=0b00000111;
    ANSELH=0b00000000; // sur les ports 2,3,4 uniquement
    ADCON1bits.VCFG1=0; // Voltage de ref : VSS
    ADCON1bits.VCFG0=0; // Voltage de ref : VDD
    ADCON2=0b10011101; // Nous réglons le TAD, l'horloge de
                        // conversion à Fosc/16 et le résultat
                        // de la conv est justifié à droite
                        // (2 bits présent dans ADRESH et
                        // 8 bits présents dans ADRESL)
    ADCON0bits.ADON=1; // Nous activons le CAN
}
```

```
unsigned long LireCAN (unsigned char voie)
{
    unsigned int res;
    switch (voie)
    {
        case (1):    ADCON0=0b00000000;
                    break;

        case (2):    ADCON0=0b00000100;
                    break;

        case (3):    ADCON0=0b00001000;
                    break;
        default :
                    return 0;
    }
    ADCON0bits.ADON=1; // Nous activons le CAN
    ADCON0bits.GO_DONE=1;
    while (ADCON0bits.GO_DONE!=0);
    ADCON0bits.ADON=0; // Nous désactivons le CAN
    res=ADRESH&0x03; // recupère poids fort de res et mets sur poids faible de res
    res=res<<8; // poids faible devient poids fort
    res=res+ADRESL; // compte avec le pids faible
    return res;
}
```



Utilisation



# Conclusion

